

Machine learning applications in fuels research



Sergey Anufriev

Department of Mechanical Engineering
University College London

This dissertation is submitted for the degree of
Doctor of Philosophy

November 2023

Declaration

I, Sergey Anufriev confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis

Sergey Anufriev
November 2023

Abstract

Energy and climate crises both require engineering solutions to the dependence on fossil fuels. Bio-fuels can be a good alternative, because these fuels do not require significant change in the automotive design and can be used in poorer countries, where electricity production is fossil fuels based and electrical vehicles are not yet affordable.

This research was done to demonstrate applications of machine learning and chemical informatics in fuel research in order to accelerate the development of the future renewable fuels. At the moment these applications are limited to regression models of fuels physical and chemical properties, which are tested and compared on different sets of test molecules. This is contradictory to chemical informatics research, where models compared use the same train and test sets of molecules.

Three major research directions were undertaken including interpretation of octane quantitative structure activity model, establishing fuel properties applicability domain and the inverse structure generative model. The aim of the first direction is to show that more insights apart from predictions can be taken from the fuel structure activity models. Literature review was taken to find suitable interpretation method. It was found that model and input agnostic method is best suited since it does not introduce bias from particular algorithm type and is chemically intuitive. The developed interpretation showed agreement with 6 known structure knock ignition relationships.

The goal of the second direction goal was to find limitations on what fuel structure properties can be predicted given the available data-sets. This was achieved by finding a relationship between test compounds properties prediction errors and four similarity measures to the training set of compounds. The outlier detection algorithms successfully found such a trend, while the distance based approaches did not.

Lastly, the generative modelling goal was to suggest new fuel like structures, which can not be found by intuition. Two models were investigated, a general adversarial (GAN) network and auto-regressive LSTM model. The later model generated structures which were not present in either of the generative modelling training set or the octane database, and showed some traits of highly knock resistant compounds.

Impact Statement

This research focused on exploring applications of machine learning and chemical informatics in the future renewable fuels for spark ignition engines. Throughout the research, valuable scientific insights were gained. These included what fuel chemical structures can be predicted given their properties datasets, interpretations behind fuels quantitative structure activity models by methods used in drug discovery and the prototype fuels structure generation by generative machine learning models. These results can benefit the academic community and industrial companies as follows.

In academic research, the common methods in studying fuels properties are experiments and molecular dynamics. Experimental work requires expertise, funding and time to conduct a study. Molecular dynamics methods despite being rigorous in terms of physics are computationally expensive and require the knowledge of intermediate combustion reactions. Future renewable fuels might come from sources not previously known. Consequently there could be too many potential structures to make experiments. This research helps to evaluate compounds before an experiment takes place, by finding their physical properties and the research octane number. In addition this allows confidence in predictions to be found and suggest new structures.

From an economic perspective, this research helps to reduce development and the cost of exploring new fuels. The developed methodology can be applied to fuel development in other sectors, such as aviation and shipping. Economic impact can be obtained with the new fuels in areas where there is a lot of agricultural waste. From a societal perspective development of new fuels can make environment-friendly transport affordable, since old cars can be potentially retrofitted to use new bio-fuels. Additionally new bio-fuels do not require substantial upgrade in the existing infrastructure, such as storage and transportation for existing cars. In addition, retrofitting is positive for the environment since reduces the need to spend energy and materials on new cars. On the other hand developed methodology in this research can be applied to finding other biochemical materials. Lastly new bio-fuels will reduce carbon dioxide emissions, which will help environment.

The research outcomes can be used as a basis of subsequent academic studies. For example modification of current fuel structures by graph based generative models. Further practical experiments can validate developed models and evaluate new structures by an experiment.

Outputs of this work are also achieve impact in the fuels industry. In this regard, a presentation of the results was made to expert technologies involved in developing new

sustainable fuels for the transport industry at BP with follow-up meetings. Academic dissemination will be achieved by journal publications for example a manuscript based on chapter 5 and 6, which are in preparation.

Acknowledgements

I would like to thank my Ph.D. supervisors Dr. Paul Hellier and Prof. Nicos Ladommatos for directing and managing my research. Their feedback helped to complete this long journey. In addition I would like to thank my parents Irina Anufrieva and Sergey Anufriev for supporting me.

Table of contents

| | |
|--|-------------|
| List of figures | ix |
| List of tables | xiii |
| 1 Introduction | 1 |
| 1.1 Motivation | 1 |
| 1.2 Thesis aim | 2 |
| 1.3 Thesis plan | 3 |
| 2 Literature review | 5 |
| 2.1 The chemical structure and octane number | 5 |
| 2.2 Current fuel methodology | 7 |
| 2.3 QSAR in pharmaceutical industry | 13 |
| 2.4 Algorithms | 15 |
| 2.4.1 Support vector machine | 15 |
| 2.4.2 Tree based models | 18 |
| 2.4.3 Neural Networks | 22 |
| 2.4.3.1 Feed forward functions | 22 |
| 2.4.3.2 Recurrent functions | 23 |
| 2.4.3.3 Graph convolution functions | 24 |
| 2.4.3.4 Back-propagation | 28 |
| 2.4.3.5 Optimizers | 29 |
| 2.4.3.6 Regularization | 30 |
| 2.4.3.7 Convergence acceleration | 31 |
| 2.4.4 Comparison of algorithms | 34 |
| 2.5 Variable selection | 36 |
| 2.6 Validation strategies | 38 |
| 2.7 QSAR modelling practices | 39 |

| | | |
|----------|--|-----------|
| 2.8 | Modeling fuel properties | 39 |
| 2.9 | Knowledge gaps | 42 |
| 2.9.1 | Applicability domain | 42 |
| 2.9.2 | Inverse QSAR | 44 |
| 2.9.3 | Model interpretations | 45 |
| 3 | Methodology | 47 |
| 3.1 | Modelling overview | 47 |
| 3.2 | Molecular representations | 48 |
| 3.3 | Interpretation | 53 |
| 3.3.1 | Introduction | 53 |
| 3.3.2 | Data source and description | 54 |
| 3.3.3 | Modelling | 54 |
| 3.3.4 | Interpretation method | 55 |
| 3.4 | Applicability domain | 56 |
| 3.4.1 | Introduction | 56 |
| 3.4.2 | Fuel properties selection | 57 |
| 3.4.3 | Data collection | 58 |
| 3.4.4 | QSAR modelling | 59 |
| 3.4.5 | Applicability domain | 60 |
| 3.5 | Generative modelling | 62 |
| 3.5.1 | Introduction | 62 |
| 3.5.2 | Generative adversarial network | 63 |
| 3.5.3 | Auto-regressive model | 64 |
| 3.5.4 | Generated molecules assessment | 65 |
| 3.5.5 | Experiment strategy | 66 |
| 4 | Interpreting octane number predictions | 69 |
| 4.1 | Classification models validation | 69 |
| 4.2 | Interpretation results | 72 |
| 4.3 | Conclusion | 73 |
| 5 | Fuel properties models applicability domain | 77 |
| 5.1 | Fuel properties modelling result | 77 |
| 5.2 | Applicability domain results | 81 |
| 5.3 | Conclusion | 82 |

| | | |
|----------|--|------------|
| 6 | Generative modelling of high octane fuel molecules | 91 |
| 6.1 | Introduction | 91 |
| 6.2 | Experiment set up | 92 |
| 6.3 | Valid organic molecules test | 94 |
| 6.4 | Generating hydrocarbons with high octane | 100 |
| 6.5 | Generating oxygenated molecules with high octane | 104 |
| 6.6 | Conclusion | 107 |
| 7 | Conclusions and Future work | 109 |
| 7.0.1 | Literature review | 109 |
| 7.0.2 | Results | 110 |
| 7.0.3 | Future work | 111 |
| | References | 113 |

List of figures

| | | |
|------|---|----|
| 2.1 | Impact of the main carbon chain (highlighted in red) length on RON | 6 |
| 2.2 | Relationship between the number of methyl groups and RON | 6 |
| 2.3 | Impact of carbon number and double bond site on olefin reactivity [19] | 7 |
| 2.4 | Cylinder Pressure versus Crank Angle (CA) [10] | 8 |
| 2.5 | Pressure and band-pass filtered signal for knock detection and knock intensity definition. [121] | 9 |
| 2.6 | MAPO values vs engine speed and air to fuel ratio [92] | 10 |
| 2.7 | Singular value decomposition of in cylinder pressure trace [122] | 11 |
| 2.8 | General schematic mechanism for fuel oxidation [38] | 12 |
| 2.9 | Schematic of the shock tube apparatus [155] | 12 |
| 2.10 | The role of computational modelling in the early-stage drug-discovery process [149]. | 14 |
| 2.11 | SVM algorithm diagram | 16 |
| 2.12 | Decision tree prediction | 19 |
| 2.13 | Decision tree with 4 leafs fitted to sinusoidal data | 20 |
| 2.14 | Neural network with 2 layers with 3 neurons each fitted to synthetic data with Gaussian noise | 23 |
| 2.15 | Graph classification [162] | 25 |
| 2.16 | Example of relational GCN applied to 2-methylpentane-1,3 diene | 26 |
| 2.17 | Forward pass in multi-layer network | 28 |
| 2.18 | Deep learning normalisation methods adopted from [152] | 34 |
| 2.19 | Architecture of the ANN for predicting CN, RON, and MON from the group structure of a fuel molecule [90] | 40 |
| 2.20 | Plot of the absolute error in prediction vs. the binned distance to model for the global plasma protein binding | 43 |
| 2.21 | Prediction accuracy vs novelty score [108] | 44 |
| 3.1 | Methodological workflow | 48 |

| | | |
|------|---|-----|
| 3.2 | Morgan Fingerprint generation | 50 |
| 3.3 | Toluene | 51 |
| 3.4 | 10-fold cross validation | 55 |
| 3.5 | Universal interpretation adapted from [127] | 56 |
| 3.6 | Impact of contamination factor ν on data points classification | 61 |
| 3.7 | Auto-regressive model set up | 65 |
| 3.8 | Hill Climb algorithm | 66 |
| 4.1 | ROC-AUC curve for the four classification models | 70 |
| 4.2 | Reliability diagram for the four classification models | 71 |
| 4.3 | Six interpretation cases by the four classification models | 74 |
| 5.1 | Predicted and actual fuel properties, where RON(no units), AIT(K), HC(KJ/kg), BP(K), DEN(mg/cm ³), VIS(mPa s) | 80 |
| 5.2 | Mean average absolute octane number prediction error vs mean Tanimoto distance | 82 |
| 5.3 | Octane predictions errors distribution at different mean Tanimoto distance bins | 83 |
| 5.4 | Octane models mean average absolute error distribution vs contamination hyper-parameter ν | 84 |
| 5.5 | Octane models mean average absolute error distribution vs contamination hyper-parameter ν with outliers in previous figure excluded | 85 |
| 5.6 | Impact of contamination factor ν on predictions mean absolute error for different fuel properties | 86 |
| 5.7 | Impact of contamination factor ν on predictions mean absolute error for different fuel properties | 87 |
| 5.8 | Octane models worst predictions | 88 |
| 6.1 | Autoregressive model diagram | 93 |
| 6.2 | Generator architecture | 95 |
| 6.3 | Discriminator architecture | 96 |
| 6.4 | Auto-regressive model vs GAN training | 97 |
| 6.5 | Vector field comparison | 97 |
| 6.6 | Valid, Unique and Novel molecules vs sampling temperature T | 98 |
| 6.7 | Generated molecules $\log P$ values at each climb hill step | 99 |
| 6.8 | Ratio of generated compounds with ketone group at each climb hill step | 99 |
| 6.9 | Sample of generated molecules with ketone group | 100 |
| 6.10 | Octane number distributions at different hill climb steps | 101 |
| 6.11 | Generated molecular structures with the highest octane values | 102 |

| | | |
|------|--|-----|
| 6.12 | Molecules not present in either QM9 and octane data-sets | 103 |
| 6.13 | Generated structures starting with oxygen symbol, which were not present in QM9 and octane datasets | 105 |

List of tables

| | | |
|-----|---|-----|
| 2.1 | Knock detection methods based on in-cylinder pressure measurement [92] | 9 |
| 2.2 | Random forest and Gradient boosting machine parameters | 21 |
| 2.3 | Summary of variable selection methods | 38 |
| 2.4 | List of Molecular Descriptors Selected by Multi-objective Feature Selection | 41 |
| 2.5 | RON Predicting Features by Weight and Type | 41 |
| 3.1 | Example of smiles molecular representation in RON data | 49 |
| 3.2 | Python libraries for molecular representations | 53 |
| 3.3 | Datasets sizes | 59 |
| 4.1 | Classification models test result | 70 |
| 5.1 | Models parameters grid | 77 |
| 5.2 | Selected hyper-parameters in best performing models | 78 |
| 5.3 | QSAR fuel properties models performance | 79 |
| 6.1 | Auto-regressive model parameter search | 94 |
| 6.2 | Generated compounds not present in QM9 data-set | 103 |
| 6.3 | Physical properties of generated molecules not present in QM9 and octane datasets | 104 |
| 6.4 | Generated compounds not present in QM9 and octane datasets | 106 |

Chapter 1

Introduction

1.1 Motivation

The use of fossil fuels has been acknowledged by the wider academic community to contribute to global warming [68]. The effects of global warming have become evident not only to scientists but also the general public who have noticed weather pattern change over a 10 to 20-year period. In addition, fossil fuel sources are not evenly distributed around the world. Such distribution leads to conflicts and economic dependence on fossil fuel suppliers.

Bio-fuels can be an alternative fuel source, which is any fuel derived from biomass. The most common types of fuels are ethanol and bio-diesel. Bio-fuels can be derived from vegetable oils, animal fat and even algae and other organic materials. The choice of materials from which to make bio-fuels and resultant fuels themselves is very large.

Despite the wide use of electric cars, these remain expensive and most parts of the world, especially poorer countries, can not afford them yet. So the impact to climate change by electric cars is very limited at the moment, since most parts of the world produce electricity not by renewable sources. Furthermore bio-fuels do not require drastic change in the automotive design process. That means poorer countries can use bio-fuels before the electric car technology becomes more economically sufficient and the production of electricity production becomes carbon neutral there. Additionally bio-fuels use was already trialled in the aircraft industry, where the weight power ratio is critical and flying with electric motors still requires substantial development.

Successfully designing and making effective use of the next generation of liquid fuels, which will be derived from a range of biomass and fossil sources, requires an understanding of the interactions between structurally similar and dissimilar fuel components when utilised in current engine technology [65]. Interactions between fuel components can influence the

release of energy and production of harmful emissions in compression ignition combustion through determination of the auto-ignition behavior of the fuel.

Such a relationship between fuel molecular structure and its properties is complex and there are no analytical methods, which can guide the development of the future fuels. As a result fuel development requires iterative experiments, which are costly and time consuming. In addition there are too many potential combinations of fuel structures to test.

So the question arises of how to maximise the utilisation of the available fuels structure property data. Machine learning algorithms become more widely used to model complex empirical data. Chemical informatics is a branch of science, which studies the application of machine learning methods in chemical structure property applications. Recent advances in chemical informatics can therefore potentially provide better utilisation of fuel structure property data. Therefore this research is focused on developing methodology to find better alternative fuels.

1.2 Thesis aim

The aim of this thesis is to introduce modern chemical informatics and machine learning applications to accelerate the development of future renewable fuels. At present applications of these techniques are limited to octane number and other fuel properties regression models, and where the accuracy models is compared on different sets of test compounds. This is contradictory to practices in chemical informatics research, where the train and test set of compounds are specified and model performance is compared on the same set of molecules. However there are many research problems remain to be addressed in application of these techniques to fuels research.

The first research aim is to get interpretations behind octane number model. Since such interpretations can help to deliver insights of the relationship between fuels chemical structure and ignition properties.

The second research aim is to find applicability domain of the fuel properties given available data. This will ensure confidence in predictions. A model with high accuracy on the test set does not guarantee that any compound can be predict confidently. Therefore achieving this aim can define a line between what can be predicted and what needs to be tested in experiment. Misleading predictions can result in experiment costs.

The last research aim is to generate new fuel like molecules, because the combustion process is very complex and future fuels might have unexpected structures and consequently biomass sources. Generative modelling of potential fuel structures can help to overcome limitation in intuition and current knowledge of highly knock resistant fuel structures.

1.3 Thesis plan

The overarching aim of this thesis plan includes a literature review, methodology, consequent results chapters, and conclusion. The literature review starts with evidence of relationship between octane number and fuels chemical structure. Subsequently, current experimental methods in fuel combustion studies are presented. From there on the literature review builds the link between limitations in experimental approach and modern practices in drug development, where experiments have limitations due to a combinatorial explosion of possible drugs to test. Later literature review finds the gaps in applications of drug design practices in the fuel research. These include models interpretations, applicability domains and inverse QSAR models.

The methodology chapter starts by summarising the modelling involved in this research, including property predictions and inverse QSAR models with molecular representations used. Later machine learning algorithms used in this research are explained. Thereafter the methodology explains how models variable selection and validation were undertaken.

The next four chapters show the research results. The interpreting octane number predictions chapter shows results of interpretation behind a classification model, which predicts if a given compound has octane number higher or lower than of standard gasoline, which is 95. The interpretations are later verified by known combustion mechanisms.

The fuel properties models applicability domain chapter shows the development of fuels properties regression models including octane number, auto-ignition temperature, density, viscosity, and heat of combustion. To establish these models applicability domain relationship was found between models predictions errors on the test set compounds and their similarity to the training set compounds.

The following chapter on generative modelling introduces concept of molecular structure generation. The chapter aim is to develop generative model, which can alter chemical structure for a given reward function. Here the model was tested by analytical chemical properties such as $\log P$ value and the presence of a ketone group. This was done to verify models structure targeted generation. Later the developed methodology will be used in the next chapter to generate highly knock resistant chemical structures.

The last results chapter targeted generation of fuel molecular structures, with high knock resistance, and demonstrates generated molecules with the octane number as the reward function. Consequently the molecules were assessed in terms of their similarity to data-sets used to train the generative model and the reward function. Lastly the unique molecules, which were not present in both data-sets are evaluated in terms of known structural characteristics of high octane value compounds.

The conclusion chapter makes a summary of the literature review, than gives conclusion to the main research findings. In addition it includes further work, which gives future research directions based on the results obtained and the limitations in methodology used.

Chapter 2

Literature review

2.1 The chemical structure and octane number

The primary fuel property is the octane number (ON), which is a rating scheme representing the resistance of a fuel to premature ignition when exposed to heat and pressure in the combustion chamber of an internal-combustion engine. Higher the octane number higher the resistance of a fuel to premature ignition, which is known as engine knock.

When engine knock takes place, there is an extremely rapid release of much of the chemical energy in the end-gas, causing very high local pressures and the propagation of pressure of substantial amplitude across the combustion chamber [171], which can result in engine damage. The efficiency of the ideal Otto cycle, which describes spark ignition engine thermodynamics is a function of the compression ratio (CR), but increasing compression ratio is limited by the onset of the engine knock, which can be prevented by increasing fuel octane number. Hence, in an ideal case, there is a direct connection between engine efficiency and fuel octane number. In the real world, other factors also contribute to the relationship and spark timing is the primary control variable that affects both knock and efficiency [44].

The octane number is measured against a prescribed binary mixture of isooctane (2,2,4-trimethylpentane, ON = 100) and n-heptane (ON = 0) under standard conditions [125]. Two octane numbers are routinely used to simulate engine performance: the RON simulates fuel performance under low severity (at 600 rpm and 49°C air temperature) , whereas the motor octane number (MON) reflects more severe conditions (at 900 rpm and 149°C air temperature). Figure 2.3 adopted from [19] shows RON and MON measurements for various paraffin's.

The first observation in relationship between fuels chemical structure and tendency to knock was done by [100]. Relationships were shown between different methods of engine testing, between structure and knock for various chemical classes of compounds, between

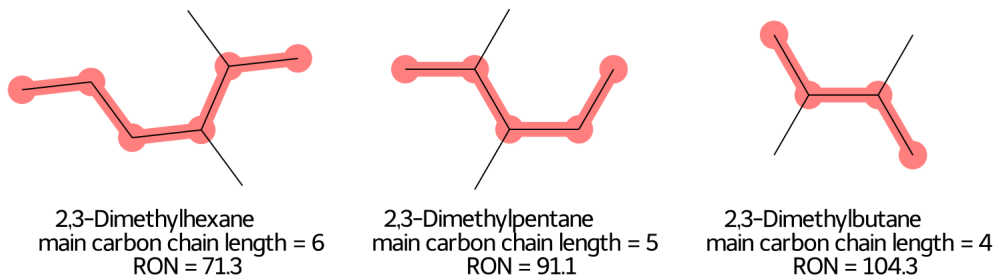


Fig. 2.1 Impact of the main carbon chain (highlighted in red) length on RON

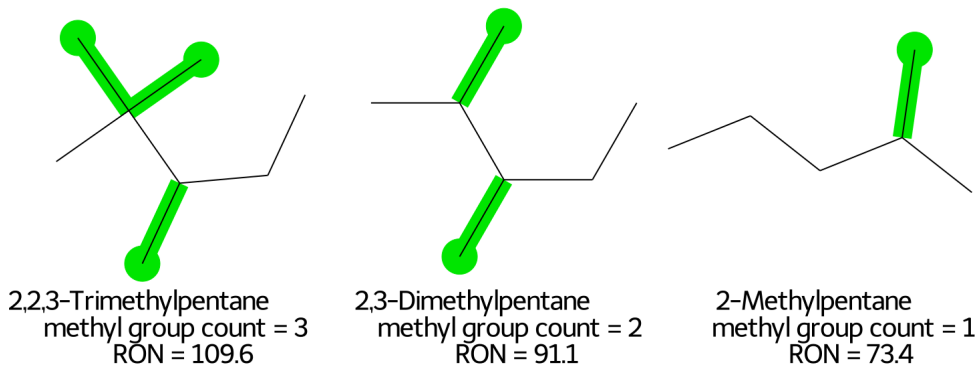


Fig. 2.2 Relationship between the number of methyl groups and RON

the knock behavior of hydrocarbons straight and in solution, and between the nature of the hydrocarbon and the extent to which tetraethyllead is effective in changing its antiknock value. The observation was based on data for 325 hydrocarbons. Hydrocarbons tendency to knock depends on main carbon chain length, branching and presence of cyclic groups. Figure 2.1 demonstrates that the longer the main carbon chain length the lower is fuel resistance to knock. Figure 2.2 shows that with increased branching higher the fuels resistance to knock. Rings are superior to unbranched chains, and unsaturation in a straight carbon atom chain is advantageous, especially if the double bond is near the centre of the molecule [100].

In addition the octane number increases, as the branching point moves toward the centre of a long chain in the alkane molecule [66]. The number of adjacent CH₂ groups has the highest but sigmoid influence, ON decreases with the separation between branches, ON increases with the bulkiness of the branched structure. [125]. Highly unsaturated cyclic compounds are the preferred octane boosters for modern SI engines [19]. Additional side chains of any variety will dilute this strong performance. Multi-branched paraffin's come in distant second place, owing to their negligible sensitivity. Depending on the type and location

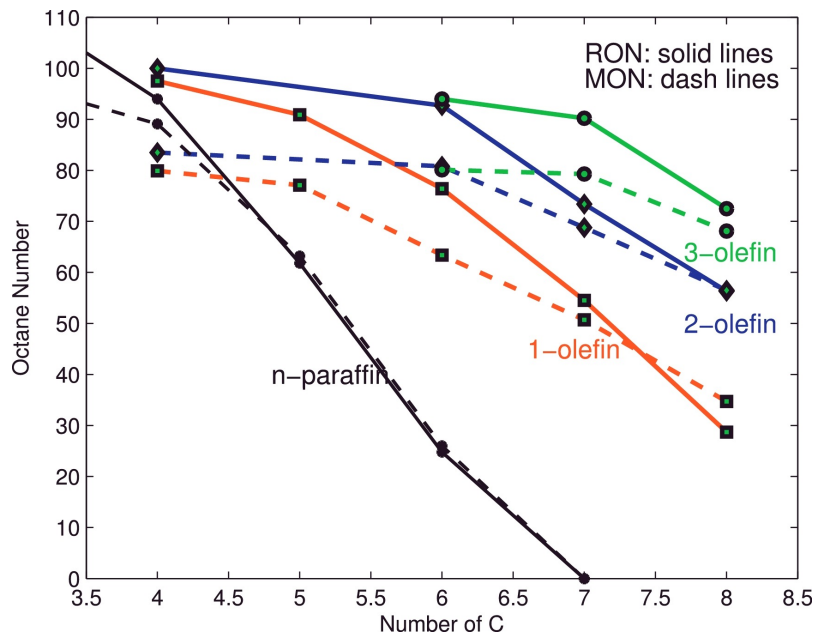


Fig. 2.3 Impact of carbon number and double bond site on olefin reactivity [19]

of functional oxygen groups, oxygenates can have a beneficial, neutral or detrimental impact on anti-knock quality [110].

These empirical observations indicate a complex relationship between fuels molecular structure and the octane number. The following parts of the literature review explore modelling techniques to establish an empirical relationship between fuels chemical structure a respective octane number.

2.2 Current fuel methodology

Fuel ignition properties are either investigated experimentally or by kinetics modelling. This chapter provides an introduction to these methods so their advantages and disadvantages can be deduced.

Experiment

To compare fuels tendencies to knock experimentally, in-cylinder pressure readings versus crank angle degrees are taken and analysed, because of the direct connection between knock and pressure oscillations [25]. Figure 2.4 shows in-cylinder pressure readings taken during normal combustion, with slight and strong knock.

These pressure readings are then processed by the pass-band filter utilising two cut of frequencies low and high to isolate knock specific pressure oscillations from other sources

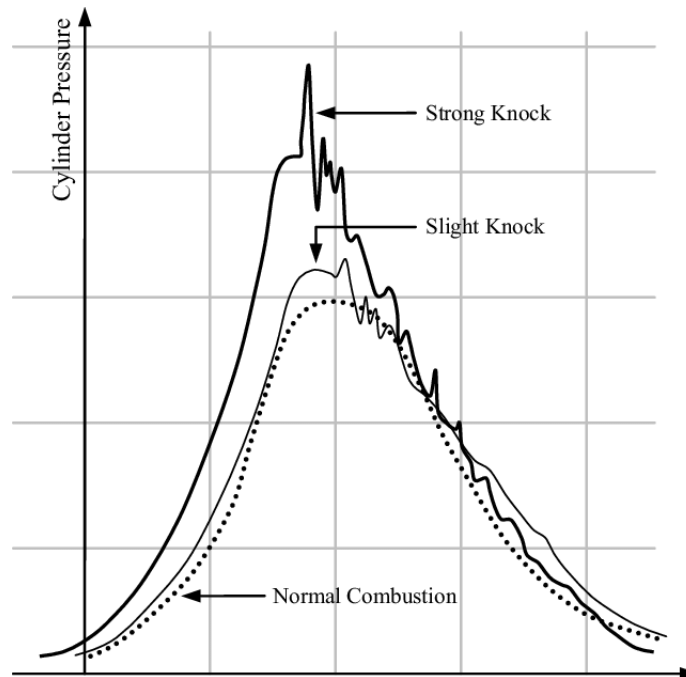


Fig. 2.4 Cylinder Pressure versus Crank Angle (CA) [10]

of oscillation in the pressure measurements obtained, for example from electrical noises [73]. The obtained pressure signal is then used to compute knocking index that is then compared to the knock threshold level [36]. Many kinds of indices exist. However there is no single index that can be used as standard knock detection index. This is because of complexity of the knock phenomena and difficulty setting predefined threshold. Maximum amplitude of pressure oscillation (MAPO), is the most widely used knock index [51], which is illustrated in Figure 2.5.

Integral of modulus of pressure oscillations (IMPO) is another knock index derived from the filtered pressure signal (Figure 2.5). It represents the energy contained in the high frequency oscillations of the cylinder pressure signal, which occurs due to knock [24]. It's worth noting that MAPO and IMPO values can be different for the same pressure cycle depending on pass-band filter low and high cut off frequencies. These frequencies can be found by applying fast Fourier transform (FFT) to the cylinder pressure signal. In addition, the value for the cut of frequencies depends on engine operating conditions. Other knock detection methods based on in-cylinder pressure signals are summarised in Table 2.1.

However knock indices strongly depend on engines operation conditions. Critical MAPO values for different engine operating conditions, including engine speed and air to fuel ratio

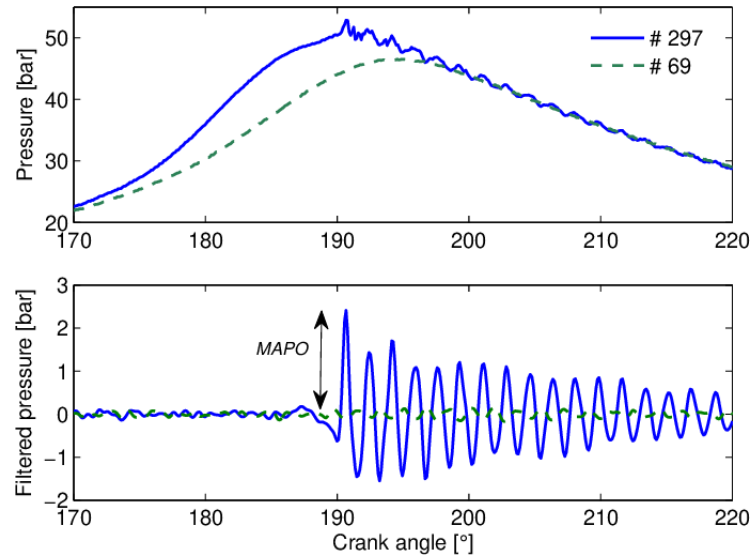


Fig. 2.5 Pressure and band-pass filtered signal for knock detection and knock intensity definition. [121]

| Object of analysis | Knock onset indicator |
|--|--|
| Raw pressure signal | Crank angle at max. amplitude |
| Band-pass filtered pressure signal | Crank angle at max. amplitude or amplitude exceeding a threshold value |
| 1 st derivative of filtered pressure signal | Crank angle at max. |
| 3 ^d derivative of filtered pressure signal | Crank angle at max. negative amplitude |
| Heat release rate (calculated from in-cylinder pressure) | Crank angle at abrupt rise of heat release or at change to high frequent signal components |

Table 2.1 Knock detection methods based on in-cylinder pressure measurement [92]

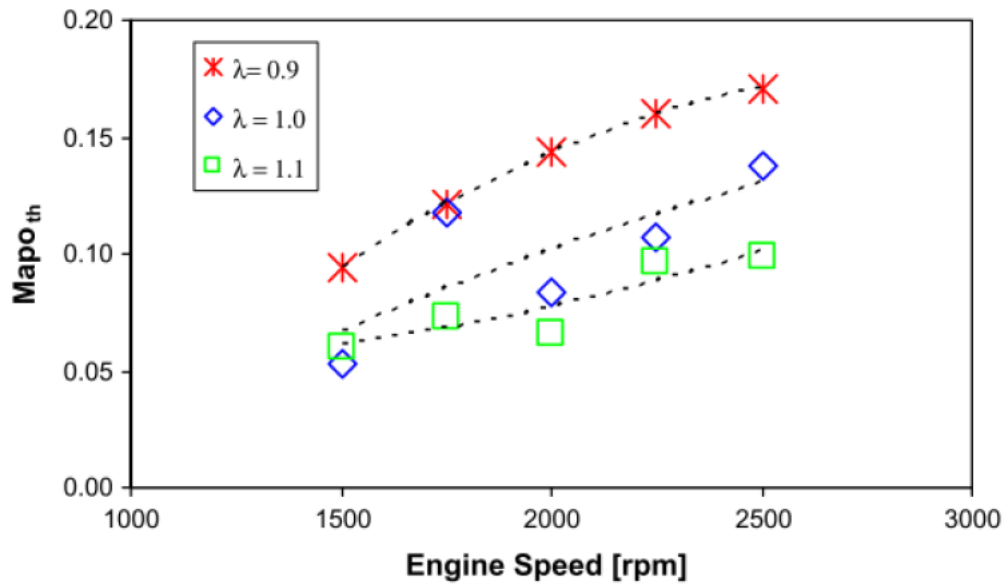


Fig. 2.6 MAPO values vs engine speed and air to fuel ratio [92]

are shown in Figure 2.6. Apparent in Figure 2.6 is an increase in MAPO with engine speed increase and λ mixture richness.

Since engine operating conditions influence knock indices, researches attempted to model this factors. The influence of exhaust gas recirculating system (EGR) on knock was modelled by [33]. Alternatively, to in-cylinder pressure trace band-pass filtering and computing knock index from it, principal component analysis was applied to raw in-cylinder pressure trace in [122]. The rational behind the proposed eigen decomposition of pressure signal lies in fact that a given pressure profile $p(\theta)$ can be described as a liner combination of basis functions in equation 2.1.

$$p(\theta) = \sum_{i=1}^l \gamma_i \bar{p}_i(\theta) \quad (2.1)$$

Where l is $(\theta_{max} - \theta_{min})/\Delta\theta = 720$ is the length of pressure trace vector and $\bar{p}_i(\theta)$ are basis functions found by singular value decomposition (SVD) on knock pressure data and γ_i are scalar coefficients. Figure 2.7 demonstrates first this decomposition of in cylinder pressure trace by first three basis functions with the largest singular values.

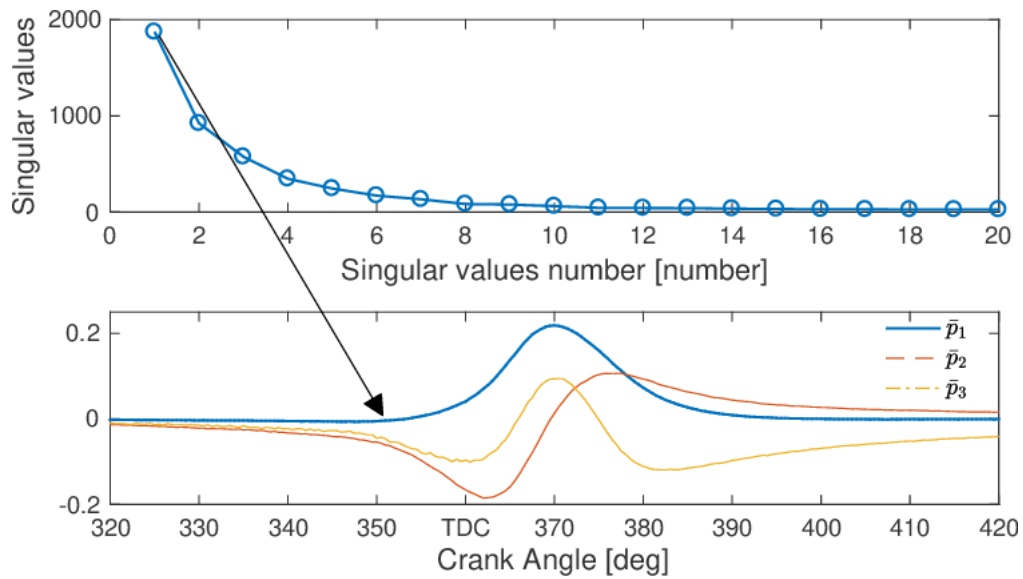


Fig. 2.7 Singular value decomposition of in cylinder pressure trace [122]

Kinetic modelling

Chemical kinetic modeling is an important tool in the analysis of many combustion systems. It is a branch of physical chemistry that is concerned with understanding the rates of chemical reactions. Each fuel molecule combustion has its own reaction mechanism, which is the step by step sequence of elementary reactions by which overall chemical change occurs [104]. The following Figure 2.8 demonstrates general scheme of low temperature fuel oxidation.

Chemical kinetic reaction mechanisms are strongly hierarchical in that mechanisms for the combustion of more complex fuels contain within them sub mechanisms for simpler fuel molecules [35]. These reaction mechanisms can be substantially complex. For example, the combustion mechanism for alkylbenzenes [11] at engine relevant conditions consist of 150 species and 759 reactions. Additionally combustion mechanism for small molecule as ethanol at engine relevant conditions includes 67 species and 1016 reactions [138]. To verify kinetic modelling experimentally shock tubes are used (Figure 2.9).

As shown in Fig. 2.9, a gas-driven shock tube consists of a long tube with a driver section and a driven section separated by a diaphragm. The driven section is filled with test gas, i.e. fuel air mixture, whereas the driver section is filled with inert gas. The pressure difference between the two chambers causes the diaphragm to move to cause compression in the test gas mixture. This makes the test gas mixture temperature rise. Therefore the pressure and temperature can be controlled for the test gas. Consequently, conditions similar to internal engines can be created.

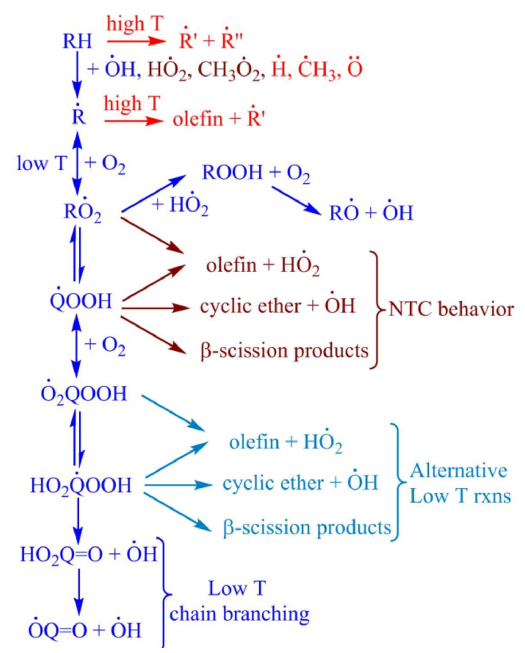


Fig. 2.8 General schematic mechanism for fuel oxidation [38]

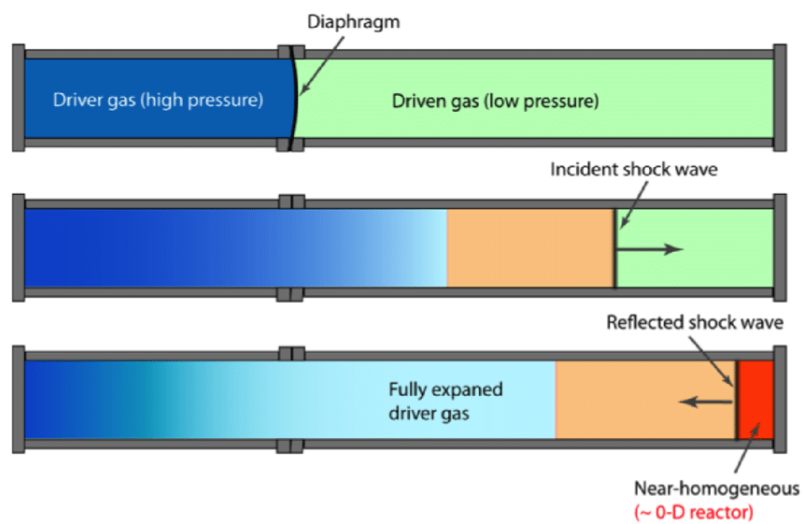


Fig. 2.9 Schematic of the shock tube apparatus [155]

To conclude chemical kinetic modelling involves finding complex reaction mechanisms, which are verified by the shock tube. However, each new fuel molecule requires finding its own unique reaction mechanism. In addition despite the shock tube giving flexibility in terms of pressure and temperature it still can't replicate engines conditions completely.

2.3 QSAR in pharmaceutical industry

This chapter will draw insights from pharmaceuticals to fuel research. The common theme is the chemical structure to property relationship, where the properties are bio-activity and ignition qualities respectively. In both fields, there is no analytical solution, which can provide an exact answer for these properties given chemical structure. Therefore trial and error process is common in both fields. However advances in modelling these relationships were made in the pharmaceutical industry, whereas fuel research lags with this respect.

The drug development process is a major challenge in the pharmaceutical industry since it takes a substantial amount of time and money to move through all the phases of developing a new drug. In the past, the discovery of new drugs was made through random screening and empirical observations of the effects of natural products for known diseases. This random screening process, although inefficient, led to the identification of several important compounds until the 1980s [102]. However the number of potential drug-like molecules was estimated to be in order 10^{33} , between 10^{18} and 10^{200} , 10^{60} by [128], [43], [132]. These results highlight theoretical limitations in the random screening process.

Currently, this process is improved by quantitative structure activity modelling (QSAR). The structure-activity models utilise a paradigm that similar compounds exhibit similar properties [81]. QSAR models are built by training machine learning models with chemical data-sets. The drug development process starts with the identification of bioactive compounds followed by lead optimization and then by clinical trials with regulatory approval. QSAR modeling helps to prioritize a large number of chemicals in terms of their desired biological activities as an *in silico* methodology and, as a result, significantly reduces the number of candidate chemicals to be tested with *in vivo* experiments [91]. Vast libraries of "drug-like" chemicals [116] were created by QSAR models. The lead optimization process involves altering selected bioactive compounds to improve their physico-chemical properties. The QSAR models help to guide lead optimization by predicting altered compound physico-chemical properties. Figure 2.10 illustrates the schematics of the computer aided drug development process.

Advances in application of structure activity modelling includes finding novel antimalarial compounds [169]. Schistosomiasis is another tropical disease, whose treatment relies on one drug at the moment. Discovery of potential candidates to treat this disease were found by

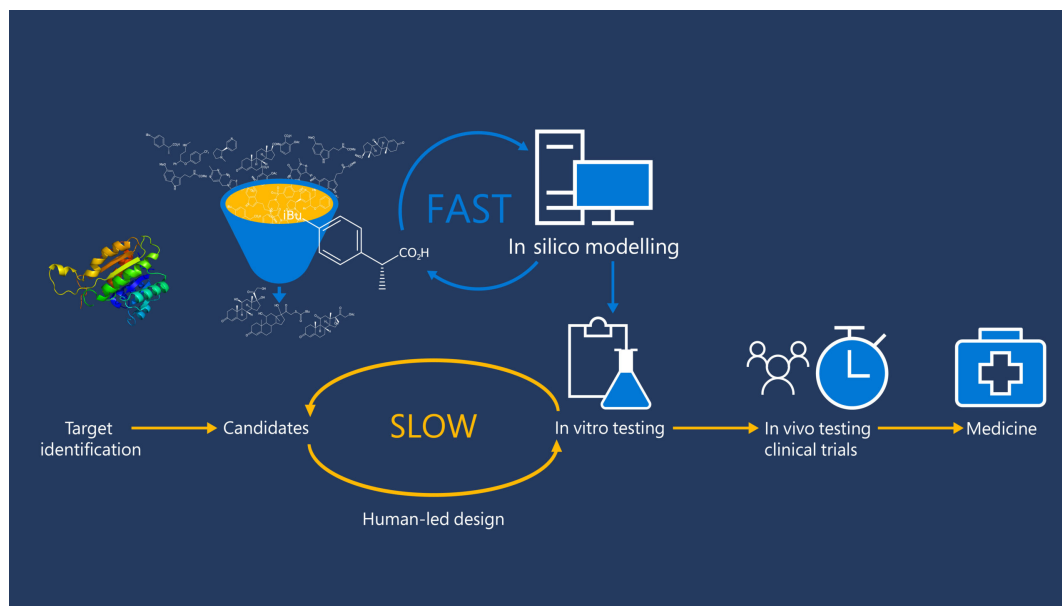


Fig. 2.10 The role of computational modelling in the early-stage drug-discovery process [149].

QSAR models [117]. Additionally model to find compounds suitable to treat *Mycobacterium tuberculosis* was developed by [3]. In response to the ongoing COVID-19 pandemic, there is a worldwide effort being made to identify potential anti-SARS-CoV-2 therapeutics. QSAR model was developed to identify novel drug candidates by [74]. QSAR models are regulated in pharmaceutical industry. OECD states the main principles in QSAR validation [2]:

1. a defined endpoint
2. an unambiguous algorithm
3. a defined domain of applicability
4. appropriate measures of goodness-of-fit, robustness and predictivity
5. a mechanistic interpretation, if possible

The intent of principle 1 (defined endpoint) is to ensure clarity of what is being predicted by a given model. If the training set endpoint values were obtained by experiment, the experiment protocol must be stated. The second principle insures transparency in the model development and that the model results are reproducible. Since QSAR modeling assumes that new molecules to be predicted will have some structural features in common with the training set that they are based on. If the new molecule is sufficiently different, one will obtain an unreliable (or even meaningless) prediction [60]. Therefore the third principle ensures

that model applicability domain is defined and molecules predicted fall within it. The fourth principle states robustness in model validation and the model productivity's is established by external validation with detailed guidance in validation procedure. The last principle recognises that is not always possible to give interpretations to QSAR model from scientific point view. However the intent of principle 5 is to give some consideration to the possibility of a mechanistic association between chemical structure descriptors used in model and the end point values being predicted and to ensure that this association is documented.

2.4 Algorithms

Algorithms with potential for use in fuel QSAR and molecular generative modelling are introduced in this sub-chapter. These include support vector machines, tree-based models and neural networks.

2.4.1 Support vector machine

Support vector machine models can be used for regression and classification tasks, such as predicting fuel properties and assigning probabilities of a given compound exceeding an octane number threshold. The description of SVM will start with the classification formulation and the regression modification will be given later. Such an order is chosen for convenience in describing algorithm principles.

Given a set of samples $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where $y \in (1, -1)$, SVM classification model aims to find a hyper-plane, which separates the two classes of points by maximum margin. Figure 2.11 shows blue and red data points, which correspond to two different classes. SVM finds a hyperplane, the green line shown in Figure 2.11, which separates the two classes of points by a maximum margin, shown by two black lines in Figure 2.11. The algorithm then assigns class y_i^{pred} to point x_i by signed distance between the point and the hyper-plane as shown in Equation 2.2

$$y_i^{pred} = \text{sign}(W^T x_i + b) \quad (2.2)$$

Where W is a vector normal to the hyper-plane and b is the distance between the origin and the hyper-plane. These parameters are found by solving constraint optimization problem 2.3. Where the first statement refers to maximizing the margin width equal to $\frac{2}{\|W\|_2}$, by minimizing its denominator. The constraints in 2.3 are attributed by the correct classification of points, which are checked by multiplying predicted class with the true label y_i .

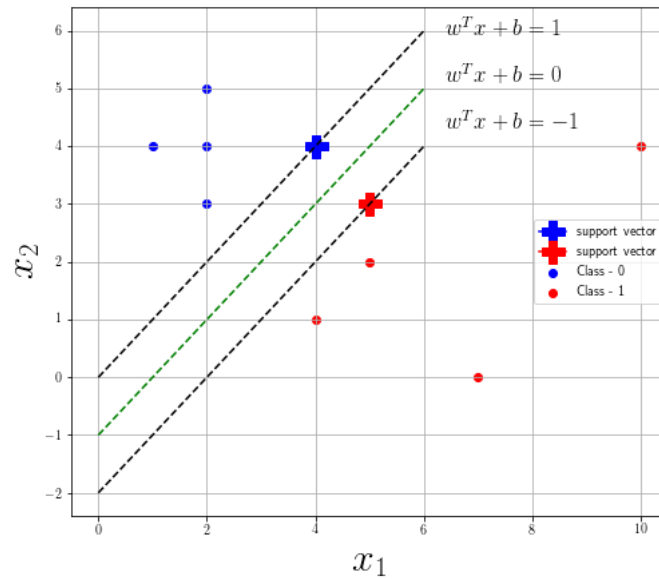


Fig. 2.11 SVM algorithm diagram

$$\begin{cases} \min_{W,b} W^T W \\ y_i(W^T x_i + b) \geq 1 \quad \forall i \end{cases} \quad (2.3)$$

This optimization problem is solved by quadratic programming [67], where solutions for W and b depend on finding the support vectors, shown by bold crosses in Figure 2.11. These support vectors are actually data points, which provide the maximum margin. Therefore it is important to emphasize that these data points determine the SVM algorithm predictions. For example, it becomes obvious that SVM is highly sensitive to outliers in data, which can happen to be the support vectors. On the other hand, SVM has the advantage of not assuming the data distribution form, since its predictions are based on few data points [105].

In addition the data points might not be linearly separable as in Figure 2.11. This leads to two options. The first option is to relax constraints in 2.3 to allow some miss classification illustrated by equation 2.4.

$$\begin{cases} \min_{W,b} W^T W + C \sum_{i=1}^n \varepsilon_i \\ y_i(W^T x_i + b) \geq 1 - \varepsilon_i \quad \forall i \end{cases} \quad (2.4)$$

Where ε_i is slack variable assigned to data point i , with value 0 if the data point was classified correctly and $|y_i^{pred} - y_i|$ otherwise. Parameter C trades off weight to put onto each

of two criteria, small C means more weight to large margin over small errors, while the effect of large C is opposite. While slack variables ε allow for overlapping class distributions, this framework is still sensitive to outliers because the penalty for misclassification increases linearly with ε .

The second option to deal with overlapping classes is to transform data points into higher dimensions, where the data points are linearly separable. Many techniques exist for such transformation, but they are common with the fact that SVM requires computation of the inner product only, as can be seen by $W^T W$ term in equations 2.3 and 2.4. Therefore instead of mapping data points into higher dimension and then computing the inner products in two steps, inner products can be modified directly, without explicitly finding data-points higher dimensional representations. Modified inner products are called kernels. Any symmetric function that is positive definite can be used as kernel. Equation 2.5 illustrates polynomial kernel as example.

$$K(x, y) = (1 + x^T y)^s \quad (2.5)$$

To apply SVM to regression problems only slight modifications in its formulation are needed, which result in the optimization problem described by equation 2.6

$$\begin{cases} \min_{W, b} \{W^T W + C \sum_{i=1}^n (\xi_i + \zeta_i)\} \\ y_i \leq y_i^{pred} + \varepsilon + \xi_i \quad \forall i \\ y_i \geq y_i^{pred} - \varepsilon - \zeta_i \quad \forall i \end{cases} \quad (2.6)$$

Where the objective in 2.6 is to maximize the margin width and minimize the sum of slack variables ξ_i and ζ_i , which are assigned to each data point, based on accuracy of predictions. The second and third lines in 2.6 correspond to slack variables conditions, which penalize predictions with absolute difference between predicted and real values $|y_i^{pred} - y_i| > \varepsilon$.

To conclude, tuning a support vector machine algorithm requires finding an appropriate kernel function, which acts as a parameterized feature transformation. For example, s variable in polynomial kernel (2.5) controls the complexity of feature transformation. Where one of the main goals of feature transformation is to make data linearly separable. Tuning parameter C (2.6) trades of between correct data points location with respect to the margin (Fig 2.11) and its width to prevent over-fitting, since only the support vectors close on the margin are used for predictions. In SVM regression problems parameter ε controls data points contribution to the total cost function (summation in 2.6). Therefore ε value affects the number of support vectors selected. Larger ε values will result in smaller number of support vectors and vice versa. Hence, both C and ε values affect model complexity, but in a different way.

2.4.2 Tree based models

Decision tree-based models, random forest Breiman [26] and gradient boosting machines [50] have a variety of potential use in fuel QSAR modelling. Both algorithms have decision tree subroutines, which differ in how they combine multiple decision trees.

At the beginning of this sub-chapter, the decision tree subroutine will be explained. Consequently, ensemble methods of the decision trees will be described. The sub-chapter will conclude with a random forest and gradient boosting algorithms summary of parameters, derived from the decision tree subroutine and the ensemble methods.

Decision tree

Given data samples $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, decision tree is a model, which partitions the input space in X domain into regions and assigns them a constant value in Y co domain. Fig 2.13 shows the example of a decision tree with the tree depth $td = 2$ applied to sinus data, where the decision tree model is shown by 4 red lines and the sinus data by blue points. Decision trees grow recursively by splitting each region in the X domain into left and right regions, where the final number of regions is equal to 2^{td} . Fig 2.12 shows example of a decision tree growth applied to the sinus data. The first split was at $x = 2.76$. The obtained left and right regions, which are $(x < 2.76)$ and $(x > 2.76)$ were partitioned again at x equal to 0.3 and 6.22 respectively. Each decision tree split location was found by solving the following optimization problem

$$\begin{cases} \min_{x_j} \{ \sum_{x_i \in \{x|x < x_j\}} (y_i - c_1)^2 + \sum_{x_i \in \{x|x > x_j\}} (y_i - c_2)^2 \} \\ c_1 = \text{ave}(y_i | x < x_j) \\ c_2 = \text{ave}(y_i | x > x_j) \end{cases} \quad (2.7)$$

Where the minimization objective is the sum of mean squared errors between data point values in right and left regions with constants c_1 and c_2 [77]. The second and third lines in optimization 2.7 show that c_1 and c_2 are means of the data points Y values in the left and right regions. In Figure 2.12 vertical dashed lines show the locations of the splits, while the horizontal lines show the regions means.

The question arises of how large should the tree grow. Clearly, a very large tree might over-fit the data, while a small tree might not capture the important structure. Therefore decision tree parameters are mostly concerned with when to stop the tree growth. The first parameter is the tree depth td . Alternatively, the tree growth can be stopped by the minimum number of samples n_{min} in decision tree leafs. The number of samples in decision tree leafs

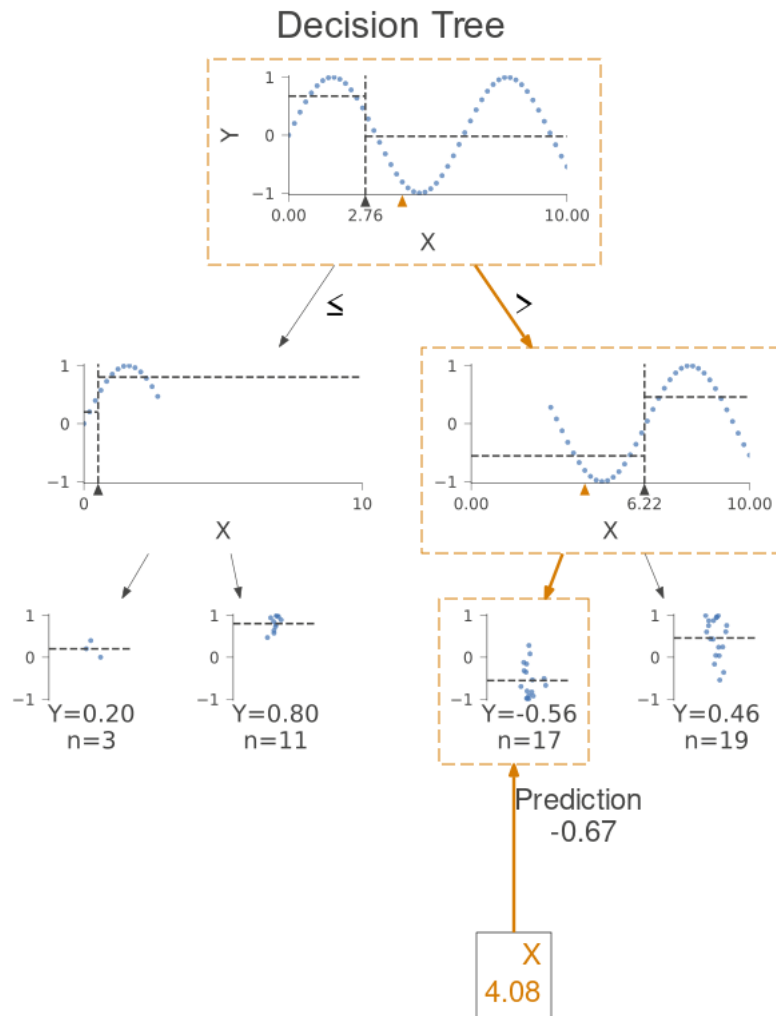


Fig. 2.12 Decision tree prediction

are 3,11,17 and 19 in Figure 2.13. In regression problems with more than one input variable, decision trees are controlled by the maximum number of parameters considered for split [77].

Combining decision trees

Random forest algorithm uses a bagging aggregation method to combine the decision trees. This technique is used to reduce variance of predictions made by decision trees. Given a standard data-set D of size n , bagging generates B new training sets of size m each, by sampling from D uniformly and with replacement. This sampling technique is known as bootstrapping. Thereafter B decision trees T_b are fitted into the generated data-sets and the

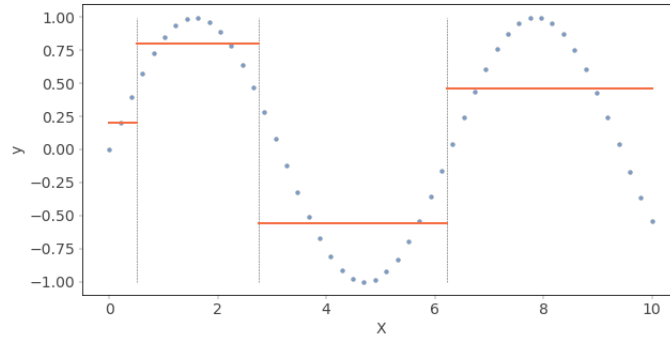


Fig. 2.13 Decision tree with 4 leaves fitted to sinusoidal data

final prediction of a random forest algorithm is given by the predictions average shown by equation 2.8

$$y^{pred}(x) = \frac{1}{B} \sum_{b=1}^B T_b(x) \quad (2.8)$$

A gradient boosting machine builds an additive model sequentially in forwarding stage manner with boosting method. The idea behind boosting is to combine many weak predictors to make a good one. Compared to the random forest (2.8), the gradient boosting machine uses a weighted average (2.9), whose terms are found by the Ada-boost algorithm [49].

$$y^{pred}(x) = \sum_{b=1}^B \gamma_b T_b(x) + \gamma_0 \quad (2.9)$$

Algorithm 1 Ada boost

Input: training data $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

- 1: $\gamma_0 = \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$
 - 2: $F_0(x) = \gamma_0$
 - 3: **for** $b = 1, 2, \dots, B$ **do**
 - 4: **for** $i = 1, 2, \dots, n$ **do**
 - 5: $r_{ib} = -\left[\frac{\partial L(y_i, F_{b-1}(x_i))}{\partial F_{b-1}(x_i)}\right]$
 - 6: **end for**
 - 7: Fit decision tree T_b on data $(x_1, r_{1b}), (x_2, r_{2b}), \dots, (x_n, r_{nb})$
 - 8: $\gamma_b = \min_{\gamma} \sum_{i=1}^n L(y_i, F_{b-1}(x_i) + \gamma T_b(x_i))$
 - 9: $F_b(x) = F_{b-1}(x) + lr \gamma_b T_b(x)$
 - 10: **end for**
-

The first line in Algorithm 1 computes the γ_0 constant in equation 2.9 by finding such γ that gives the minimum mean squared error between data point targets y_1, \dots, y_n and itself.

| Parameter | APIs input |
|---|-------------------|
| Tree based parameters | |
| Tree depth | max_depth |
| Minimum leaf samples | min_samples_leaf |
| Number parameters for split | min_samples_split |
| Number of trees | n_estimators |
| Random forest specific | |
| Number of data points in bootstrapped samples | max_samples |
| Gradient boosting machine specific | |
| Learning rate | learning_rate |

Table 2.2 Random forest and Gradient boosting machine parameters

The next line assigns first predictor F_0 constant value γ_0 . For loop 3-10 runs B times and updates predictor function F_b recursively by adding one decision tree in each loop [77]. B is a parameter to tune. Line 4 for starts inner loop which creates each decision tree its own targets y_{1b}, \dots, y_{nb} by transforming the data targets y_1, \dots, y_n by equation in Line 5. This equation finds gradient of mean squared error between the data targets and the predictions made by composition of already fitted decision trees F_{b-1} with respect to their predictions. Line 7 fits a decision tree to the created targets y_{1b}, \dots, y_{nb} . This tree is later added to composition of decision trees F_{b-1} in Line 9 with learning rate lr times the coefficient γ_b found by solving optimization task in Line 8. The task in line 8 means finding such γ that adding tree T_b to composition of previous $b - 1$ tree reduces the error prediction. The learning rate lr is regularization parameter, which prevents over fitting by partially minimising contribution of new tree.

It is important to note the differences between random forest and gradient boosting algorithms when increasing the number of decision trees B . With increased B a random forest will increase its accuracy on train data, however, after a certain tree number that accuracy will converge. This is because increasing the number of trees will make bootstrapped samples similar and hence the trees and therefore their average will not change much [99]. On the other hand, gradient boosting machines will try to extract as much accuracy as possible on each added decision tree and therefore can over-fit. However larger than a necessary number of trees in the random forest will lead to increased computational costs without accuracy gains [120]. These algorithms were implemented with scikit-learn library Buitinck et al. [29]. Parameters which were considered for tuning are listed in Table 2.2

2.4.3 Neural Networks

Three types of neural networks are described in this section, including feed forward, recurrent and graph neural networks, which were applied to molecules represented by descriptors, smiles strings and graphs respectively. The feed forward, recurrent and graph convolution functions sub chapters describe these networks forward propagation. These networks were trained iteratively with gradient-based optimizers. Back-propagation and optimizers subchapters describe the principles behind neural networks training and introduce parameters, which control the training process. A central problem in machine learning is how an algorithm performs on unseen data. The strategies used to reduce the test error in neural networks training are explained in the regularization sub-chapter. Heuristics used to speed up neural networks training are given in the convergence acceleration sub-chapter.

2.4.3.1 Feed forward functions

Feedforward neural networks can be described as a series of functional transformations (2.10). Where $f^{(n)}$ is called n^{th} layer and the total number of transformations n is the model depth. The inputs to the model are vectors, matrices and tensors, while the outputs can be the same type as inputs and scalars additionally.

$$f(x) = f^{(n)}(f^{(n-1)} \dots (f^0(x))) \quad (2.10)$$

Each functional transformation in a feed forward network consist of linear transformation parametrised by weight matrix $W_n \in R^{N \times M}$ and bias vector $b_n \in R^{M \times 1}$, followed by non linear activation function h (2.11). Where N is the input vector size and M is the output vector size. Commonly M refers to the number of layer neurons. The neural network learning goal is to find such weighs and biases $(W_n, b_n), (W_{n-1}, b_{n-1}), \dots, (W_0, b_0)$ that function 2.10 gives good approximation of the data [71].

$$f^{(n)}(x) = h_n(W_n^T x + b_n) \quad (2.11)$$

Figure 2.14 illustrates flexibility of neural network to approximate 4 data sets generated by parabola, sinus, heavy side and modules functions with noise added. The neural network had only 2 layers, with 3 neurons each, with hyperbolic tan activation function in the first layer and identity in the last. Mathematically such description means that the model depth $n = 2$, $W_0 \in R^{1 \times 3}$, $b_0 \in R^{3 \times 1}$ and $W_1 \in R^{3 \times 1}$, $b_1 \in R$, while $h_0(x) = \tanh(x)$, $h_1(x) = x$

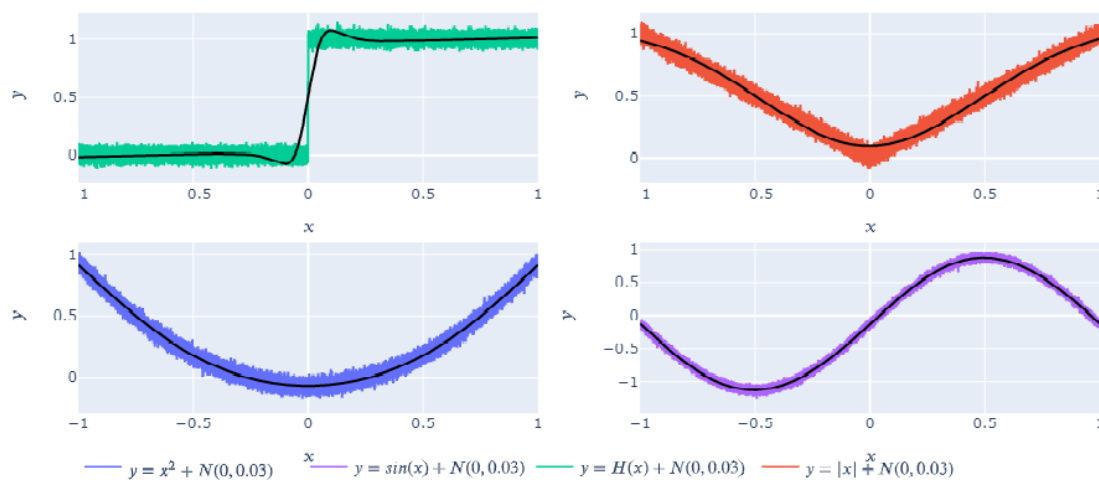


Fig. 2.14 Neural network with 2 layers with 3 neurons each fitted to synthetic data with Gaussian noise

2.4.3.2 Recurrent functions

Recurrent neural networks, or RNNs are a family of neural networks for processing sequential data, which includes smiles string molecular representation. RNNs keep hidden state compared to regular feedforward neural networks, which allows tracking on previously seen symbols. This feature leads to learning chemical rules in smiles strings by the model [13]. RNN take sequence of vectors $X = (x_1, \dots, x_n)$ and hidden state vector h_0 as input and outputs sequence of vectors $Y = (y_1, \dots, y_n)$. These computations are done recursively

$$\begin{cases} h_t = R(h_{t-1}, x_t) \\ y_t = O(h_t) \end{cases} \quad (2.12)$$

The recursive computation leads to parameters being shared between different steps in sequential input X and output Y . Parameter sharing allows for application of the model to sequences of different length and generalize across them. Consequently such property makes possible to model molecules with different number of atoms. Where each t^{th} smiles symbol represents input x_t . Recurrent neural networks configurations differ in terms of functions R and O .

RNNs with LSTM cells [69] can be used to compute hidden state h_t , where function R (2.12) is described by series of transformations (2.13) applied to current input x_t , previous hidden and cell states h_{t-1} , c_{t-1} . The memory cell c_t encodes the information about inputs that were observed before step t .

$$\left\{ \begin{array}{l} i_t = \sigma(W_{ii}^T x_t + b_{ii} + W_{hi}^T h_{t-1} + b_{hi}) \\ f_t = \sigma(W_{if}^T x_t + b_{if} + W_{hf}^T h_{t-1} + b_{hf}) \\ o_t = \sigma(W_{io}^T x_t + b_{io} + W_{ho}^T h_{t-1} + b_{ho}) \\ g_t = \sigma(W_{ig}^T x_t + b_{ig} + W_{hg}^T h_{t-1} + b_{hg}) \\ c_t = f_t \odot c_{t-1} + i_t \odot g_t \\ h_t = o_t \odot \tanh(c_t) \end{array} \right. \quad (2.13)$$

The symbols σ and \odot indicate sigmoid function and the Hadamard product. Where i_t, f_t, o_t are input, forget and output gates. The input gate controls the extent to which new input influences the cell state and the forget gate controls what to keep from previous cell state. While the output gate controls to what extend updated cell state influences computation of new hidden state.

2.4.3.3 Graph convolution functions

The advent of deep learning has led to extensive improvements in the technology used to recognize and utilize patterns in data, particularly convolutions neural networks (CNNs) [93] successfully leverage properties in the euclidean domain, such as text, images. These CNNs consists of convolution and pooling layers, which exploit the shift-invariance property and compositionality of grid structured data. As a result, CNN's perform well with a smaller number of parameters compared to dense only (2.11).

Graph neural networks (GNNs) were developed by replicating (CNNs) properties to graphs [140]. This section introduces graph neural networks (GNNs), which operate on graph-structured data, such as molecular graphs. Graph neural network configuration depends on analytical tasks, which are classified as node, edge and graph level [162]. Graph level tasks are those where the main goal is to find a function by utilising GNNs, which maps a graph to corresponding scalar or vectors. Where the first case refers to GNNs predicting fuel properties, while the second to the molecular generative model.

GNNs architecture for graph level tasks is similar to CNNs, where convolution and downsampling pooling layers pair is repeated several times, then followed by readout pooling layer, which in turn is followed by a sequence of dense layers (2.11) This subchapter will explain graph convolution and pooling layers. Figure 2.15 illustrates an architecture of a typical graph convolution neural network. Graph neural networks architectures are differ in terms of choice of graph convolution and pooling layers [172].

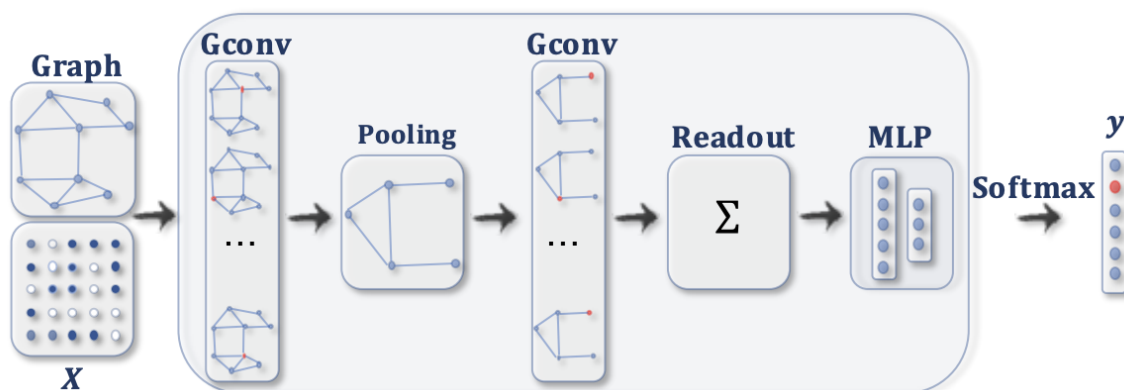


Fig. 2.15 Graph classification [162]

Convolution

Graph convolutions work by aggregating neighboring nodes information. Convolution layers take graphs adjacency matrix A , with node feature matrix X_m as input and output the graph with the same A and updated node feature matrix X_{m+1} . The goal is to find each node vector representation, i.e rows of X^{m+1} , which are derived from the each node's features in its neighborhood [170]. Given node i , its neighborhood $N(i)$ is defined as set of neighboring nodes, which can be reached by any edge coming from the node i . From chemistry perspective it is a set of atoms, which can be reached by single chemical bond from atom indexed i [131].

$$N(i) = \{j \in V | (i, j) \in E\} \quad (2.14)$$

Whereas V and E are sets of vertices and edges in graph, where node i is located. In this research V and E were atoms and edges of molecules. Let x_i be nodes i vector representation and e_{ij} is the edge between nodes i and j , then graph convolution can be defined as

$$x_i = \gamma(\square_{j \in N(i)}(\phi(x_i, x_j, e_{i,j}))) \quad (2.15)$$

Where \square denotes a differential and permutation invariant function, such as sum, mean or max, ϕ and γ are differential functions, usually feed forward dense neural network (2.10). Graph convolution functions are different in terms of \square , ϕ and γ formulations [47].

In addition convolutions layers are categorized into spectral and spatial approaches. Spectral approaches have mathematical foundation from graph signal processing. Node features are treated as signals over graph vertices. The goal of spectral approaches is to find spectral filter function over these signals in Fourier domain [89].

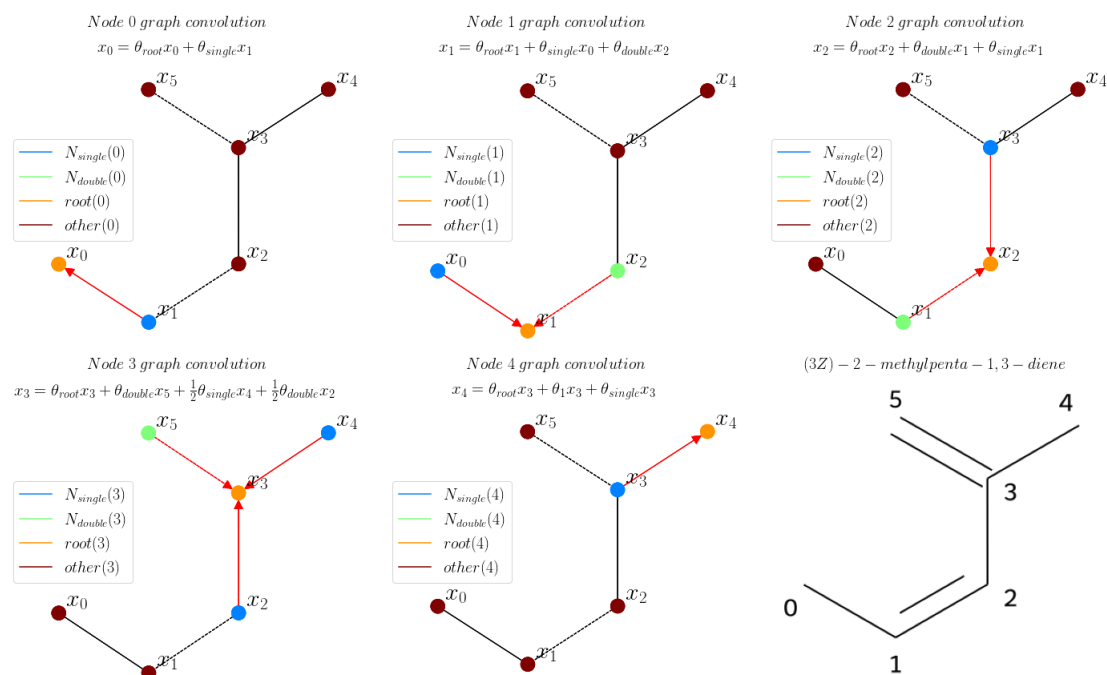


Fig. 2.16 Example of relational GCN applied to 2-methylpentane-1,3 diene

Learned spectral convolution filter depends on graph Laplacian eigenvalues, which make them inappropriate for inductive learning tasks since the laplacian graph matrix depends on graph topology. Therefore learned spectral convolution filter function may not be applicable to graphs with different typologies [28]. This can be a problem for processing molecular structure data since molecules have different structural topology. On the other hand spatial convolution filters do not utilise eigendecomposition of graph Laplacian matrix and operate directly in the spatial domain. These filters define convolutions based on nodes spatial relations. A relational graph convolution layer [143] can take into account different kinds of edge types, which resembles the valence model.

$$x_i = \theta_{root}x_i + \sum_{r \in R} \sum_{j \in N_r(i)} \frac{1}{|N_r(i)|} \theta_r x_j \quad (2.16)$$

Where $N_r(i)$ is subset of neighborhood $N(i)$, which includes i^{th} neighboring nodes, such that are connected to it by edges type r . The edge type set R consisted of single, double, triple and aromatic molecular bonds. Here the aggregation function \square is the weighted sum, where the weighting is given by the inverse number of points $|N_r(i)|$ in each neighbouring subset. The function ϕ is a linear transformation parameterised by weights θ_r and θ_{root} , where the index is attributed to the type of connection between the graph nodes. Figure 2.16 demonstrates the convolution given by equation 2.16 applied to 2 -methylpentane-1.3 diene.

Downsampling Pooling

Similarly to convolutions, pooling techniques were extended from CNNs to GNNs [170]. Downsampling pooling takes a graph as input and outputs a coarsened graph with fewer edges and nodes. It is used to reduce the number of hyperparameters and increase model generalization and achieve permutation invariance. The graph coarsening works by clustering graph nodes and then using these clusters as new nodes, where each coarsened graph nodes vector representation x is given by

$$x = P(\{f(x') : x' \in C(x)\}) \quad (2.17)$$

Where $C(x)$ is the set nodes in original graph which were clustered together to make the coarsened graph node x . f is a functional transformation applied to the original graph nodes vector representation and P is the permutation invariant function, such as mean/max/sum.

The main difference between various pooling techniques is the choice of graph coarsening process, and what graph attributes determine this process. A differential pooling layer [166] can learn how to coarsen graph instead of a predefined procedure. Equations 2.18 demonstrate coarsened graph adjacency $A^{(l+1)}$ and node feature $X^{(l+1)}$ matrices computations by the differential pooling applied to a graph represented by adjacency matrix $A^{(l)}$ and node feature matrix $X^{(l)}$.

$$\begin{cases} S = \sigma(GNN_{clust}(A^{(l)}, X^{(l)})) \\ A^{l+1} = S^T A^{(l)} S \\ X^{l+1} = S^T X \end{cases} \quad (2.18)$$

Where $S \in R^{N \times C}$ is an assignment matrix with number rows equal to the number of nodes N in original graph and number of clusters C . Each i^{th} row in S gives original graph node i probability being assigned to a particular cluster. GNN_{clust} is a graph convolution function (2.15), which computes logits used by softmax function σ to compute these probabilities.

Read-out pooling

Read-out or global pooling is used to find whole graph vector representation, which is usually followed by dense layers (2.11) [59]. In general global pooling is defined by

$$h_G = R(\{h_v : v \in G\}) \quad (2.19)$$

Where h_G is a graph G vector representation obtained by applying read-out function to graphs vertices v vectors representations h_v . The simplest case of read out function is a summation. Read-out layers are different in choice of aggregate function R .

2.4.3.4 Back-propagation

Forward propagation methods in neural networks can be applied to molecules represented by vectors, sequences and molecular graphs, I.e has given network f with parameters θ how predictions $f(x; \theta)$ are computed. During training forward propagation computes these predictions on data sampled from the training data set and then the average scalar cost L between these predictions and true targets is computed

$$L = \mathbb{E}_{(x,y) \sim p_{\text{data}}} L_{\text{func}}(f(x; \theta), y) \quad (2.20)$$

Where x is model input, i.e molecules represented by descriptors, smiles or molecular graphs and y the molecular properties. L_{func} is a loss function, which measures the quality of predictions.

The back-propagation algorithm is used to compute gradients of the scalar cost L with respect to neural network weights θ . The backpropagation algorithm is based on the differentiation chain rule. Consider abstract diagram of a typical neural network with model depth k (Fig. 2.17), where X is model input, θ_{i-1} and f_i are i^{th} layer parameters and output, while L is the scalar cost. The network layers are shown by rectangular boxes, with errors pointing at them illustrate inputs to each layer.

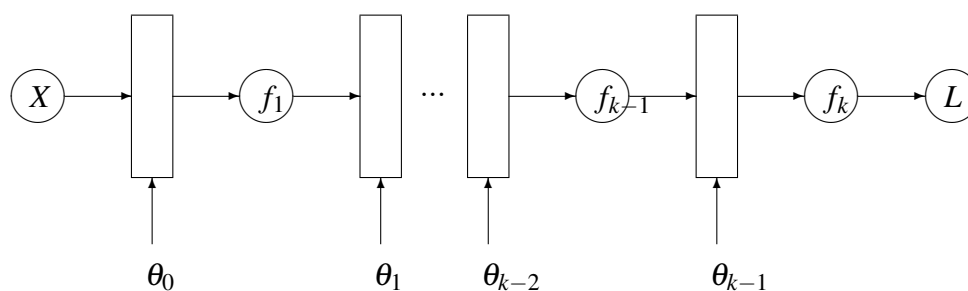


Fig. 2.17 Forward pass in multi-layer network

To get the gradients to the neural network weights $\theta_0, \theta_1, \dots, \theta_{k-1}$, partial derivatives of L with respect to each layer are needed. The chain rule allows to determine the partial derivatives as

$$\frac{\partial L}{\partial \theta_{k-1}} = \frac{\partial L}{\partial f_k} \frac{\partial f_k}{\partial \theta_{k-1}} \quad (2.21)$$

$$\frac{\partial L}{\partial \theta_{k-2}} = \frac{\partial L}{\partial f_k} \boxed{\frac{\partial f_k}{\partial f_{k-1}} \frac{\partial f_{k-1}}{\partial \theta_{k-2}}} \quad (2.22)$$

$$\frac{\partial L}{\partial \theta_{k-3}} = \frac{\partial L}{\partial f_k} \frac{\partial f_k}{\partial f_{k-1}} \boxed{\frac{\partial f_{k-1}}{\partial f_{k-2}} \frac{\partial f_{k-2}}{\partial \theta_{k-3}}} \quad (2.23)$$

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial f_k} \frac{\partial f_k}{\partial f_{k-1}} \cdots \boxed{\frac{\partial f_{i+2}}{\partial f_{i+1}} \frac{\partial f_{i+1}}{\partial \theta_i}} \quad (2.24)$$

The red terms are partial derivatives between layers outputs and inputs. These derivatives are a function of network weights and the layers functional form. This leads to observation that the product of these red terms can either amplify or vanish the gradients [123]. Whereas blue terms are partial derivatives of layer outputs with respect to layer parameters, which are functions of layers inputs. Assuming partial derivative $\frac{\partial L}{\partial \theta_{i-1}}$ is found, then to compute the next layer $\frac{\partial L}{\partial \theta_i}$ part of computation can be reused. Where the new terms to compute are highlighted by terms inside the boxes. Therefore gradients in later layers are partially products of the previous. For later convenience concatenated gradients of all layers, i.e left hand side of equation 2.24 will be written as $v(\theta)$.

2.4.3.5 Optimizers

Once the network weights gradients $v(\theta)$ are computed by the back-propagation algorithm, optimizers use these to update the weights θ to reduce scalar loss L . The simplest form of such update is stochastic gradient descent [21].

$$\theta_t = \theta_{t-1} - \eta v(\theta_{t-1}) \quad (2.25)$$

Where η is the learning rate, which is a positive scalar with value less than one. Too high η values will result in oscillating scalar loss (2.20) and too low value will result in too slow decrease of the scalar loss L . Modifications of equations 2.25, known as RMSprop and ADAM optimizers [139] commonly used to update the networks weights.

RMSprop

Often the loss function is highly sensitive to some directions in parameter space and insensitive to others [57]. Therefore it makes more sense to have different learning rates for

each parameter and automatically adapt these learning rates through the course of learning. Root mean square propagation optimizer (RMSprop) is an adaptive learning method [139]. RMSprop updates parameters less, which therefore have a higher weighted sum of past squared gradients. This weighted sum is calculated recursively by

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta)v(\theta_t)^2 \quad (2.26)$$

RMSprop introduces positive scalar parameter β , which controls the influence of past squared gradients on the weighted sum [57]. Larger β higher the influence of the past gradients for the current weight update. The network weights then get updated by RMSprop by

$$\theta_{t+1} = w_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}v(\theta_t) \quad (2.27)$$

Adam

Gradient descent (2.25) makes a monotonic improvement at every iteration, given appropriate learning rate. However, it can take a long time to converge. Different loss function sensitivity can make oscillations in a certain direction or no movement in the other. The method of momentum is designed to address these issues and accelerate training. The momentum algorithm accumulates an exponentially decaying moving average $E[g]_t$ of the past gradients and continues to move in their direction [87].

$$E[g]_t = \alpha E[g]_{t-1} + (1 - \alpha)v(\theta_t) \quad (2.28)$$

Parameter α is a positive scalar, which controls the influence of past gradients on accumulated momentum. Higher α value means larger impact of the past gradients to the current weight update. Adaptive moment estimation optimizer is a modification of RMSprop (2.27), where instead of gradients $v(\theta)$ accumulated gradients (2.28) are used [139].

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}E[g]_t \quad (2.29)$$

2.4.3.6 Regularization

Dropout [148], weight regularization and early stopping techniques can be used to regularize neural networks training. The primary aim of regularization is to reduce learning algorithm generalization error, but not its training error.

Dropout provides a computationally inexpensive but powerful method for regularizing deep learning models. The key idea is to randomly drop neurons from the network. This prevents neurons from co-adapting too much. Ensembles of neural networks with different model configurations are known to reduce over-fitting. By randomly dropping some neurons a single model can be used to simulate having a large number of different networks architectures, which is the goal of dropout regularization.

Drop out multiplies element-wise output of each layer by tensor with the same shape whose elements are either zero or one. These elements are sampled randomly from the Bernoulli distribution with probability p element equal to 0 and $p - 1$ equal to one. Parameter $p \in (0, 1)$ is another tun-able parameter, with $p = 1$ all the network neurons being dropped, while $p = 0$ means no drop out is applied [124].

Too large neural network weights can lead to a network being oversensitive to the inputs [119]. For example minor noise in inputs can result in completely different network outputs. The weight regularization was inspired by ridge regression, where it adds $\alpha \theta^T \theta$ to loss (2.20) [57]. This addition results in the following stochastic weight update by optimizer, where the change in the update rule to (2.30) is highlighted in red.

$$\theta_t = \theta_{t-1} - \eta v(\theta_{t-1}) - \eta \alpha \theta_{t-1} \quad (2.30)$$

Therefore the addition of the weight decay will multiplicatively shrink the weight vector by a constant factor on each step. Parameter α is tun-able, so its values are determined depending on the network architecture and a data-set.

When training a neural network model with sufficient capacity to over-fit the task, a steady decrease in training loss can be observed, but validation error begins to rise again. That means that the model can achieve better results by returning to the previous parameter settings. Therefore every time the model makes an improvement in validation error its weights are stored. The neural network training terminates when no improvement over validation error is recorded over the number of predefined training steps. This procedure is known as early stopping [14].

2.4.3.7 Convergence acceleration

Three common techniques used to accelerate the training of neural networks and reach convergence, which includes learning rate schedulers, weight initialization and normalization techniques. The first method addresses the problem of finding an appropriate learning rate η in (2.27) and (2.29). The weight initialisation tackles the problem of making an appropriate

neural network weight before the training begins. The normalisation techniques are used to improve neural networks architecture [72].

Learning rate schedulers

Learning rate schedules are helpful in training more adaptively, with the basic intuition of using a high learning rate at the beginning of training, and lowering the learning rate as the training progresses. Cyclic learning rates allow learning rates to increase and decrease over the course of training, which gives the model capability to escape undesired stationary points. In addition learning rate schedulers allow to find maximum practical learning rate [75].

Weight initialisation

Deep learning models are iterative and therefore require some parameters to be initialised. Training deep learning models is a sufficiently difficult task that most algorithms are strongly affected by the choice of initialization. Most commonly initialisation means the initial neural network weights values. The initial point can determine whether the algorithm converges at all, with some initial points being so unstable that the algorithm encounters numerical difficulties and fails altogether. When learning does converge, the initial point can determine how quickly learning converges and whether it converges to a point with high or low cost. There is no strict rule which determines, which weight initialisation method must be used [146].

On the other hand, one aim that is clear with respect to the weight initialisation is that it needs to break the symmetry problem [32]. The symmetry problem occurs when initialised weights have the same value. If all the weights start with equal values and if the solution requires that unequal weights be developed, the system can never learn, since the weights will receive the same update and remain equal. Therefore all weight initialisation methods include initialising weights by sampling from a particular distribution. The choice of the distribution is what makes these initialisation differ [57].

Another problem to consider in the weight initialisation includes the vanishing gradient problem, where the network gradients are close to zero, which results in network weights not being updated and hence the network won't learn anything (equation 2.27). Similarly, improper weights initialisation can cause gradient explosion. Improper weight initialisation causes the products of each layer input to output gradients (red terms in 2.24) to either vanish or explode. However many heuristic techniques were proposed to initialise neural networks weights, which depend mostly on the type of the networks activation functions including sigmoid, tanh and relu [22]. The main idea behind initialisation strategies for

different activations functions is to keep activation's (f_i in Fig 2.17) zero centred with uniform variance in order to prevent gradient vanishing or explosion.

The standard approach is to initialise neural network weights depending on the type of activation function used in each neural network layer. For neural network layers with weights $\theta \in \mathbb{R}^{m \times n}$ and Sigmoid or Tanh activation functions, glorot (2.31) initialisation [52] and Kaiming (2.32) initialisation He et al. [64] for layers with Relu activation's. Where U and N represent uniform and normal distributions in 2.31 and 2.32.

$$\theta_{ij} \sim U\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}\right) \quad (2.31)$$

$$\theta_{ij} \sim N\left(0, \frac{2}{(n+m)^2}\right) \quad (2.32)$$

Normalisation

To speed up training different normalisation techniques can be used including batch (BN), layer, layer (LN), instance (IN) and spectral normalization (SN) [72]. All except the last one transform neural network layers activation's (f_i Fig 2.17), while the last one transforms network layer weights θ . The difference between batch, layer and instance normalisation is the difference in how mean μ and variance σ for normalisation is calculated, were the normalisation is given by

$$x_{normalised} = \frac{x - \mu}{\sigma}$$

In order to make comparison between normalisation techniques assume a network activation is x and it has the following dimensions

$$x \in \mathbb{R}^{N \times C \times H \times W}$$

Where N, C are the mini-batch size, number of channels, and H, W are spatial activation's dimensions (Figure 2.18). In batch normalisation (BN), mean and variance are calculated for each individual channel across all samples and dimensions H, W . Mean and variance in instance normalisation are calculated in the same manner as in BN, but for each instance as well. In layer normalisation mean and variance are calculated for each instance across all channels and dimensions H, W .

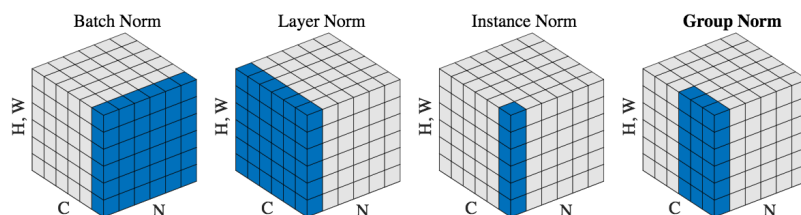


Fig. 2.18 Deep learning normalisation methods adopted from [152]

Spectral normalization (SN) was specifically designed for WGAN networks discriminator, in order to make discriminator a 1-Lipschitz function and to scale weights [113]. Spectral normalisation transforms each 2-dimensional weight by

$$W = \frac{W}{\sigma(W)}$$

Where $\sigma(\theta)$ is the largest singular value of weight matrix θ . Therefore spectral normalisation limits the maximum possible elongation of a vector to which the weight matrix θ is applied to.

2.4.4 Comparison of algorithms

This sub-chapter summarises the advantages and disadvantages of the algorithms described in the literature review. This discussion aims to understand why a family of different algorithms must be applied to data sets instead of choosing only one algorithm. The first choice in modelling fuel properties is to whatever to apply deep learning algorithms. Theoretically, neural networks can provide superior performance in QSAR prediction accuracy, because of their ability to learn new molecular representations, which could be informative and unknown to begin with [79]. On the other hand neural networks are complex in implementation and require many parameters to tune. In addition, it might take a lot of time to go through one parameter setup [123]. Lastly, deep learning algorithms require data sets with sophisticated sizes, i.e. at least in the order of thousands of training samples, which is not always present in fuels properties datasets.

Non-deep learning algorithms can include support vector machine (SVM), random forest (RF), gradient boosting machine (GBM) and K-nearest neighbours (KNN). SVMs perform well in cases where the number of variables is much larger than the number of samples, which is very applicable to most fuel properties data sets, where the number of descriptors is larger than the number of training samples. The SVM algorithm requires the input data

to be normalised, which can be skewed by outliers in data. In addition for the classification tasks, SVMs do not output probabilities and assign labels instead. Such a property will require algorithm calibration for classification tasks. The last disadvantage of SVMs is the computational expense. SVMs require the entire dataset to be stored in memory during training, making them less suitable for very large datasets where memory limitations may arise. Neural networks on the other hand update their parameters with batches of data, which eliminates memory limitations compared to SVMs.

Random forests are robust to over-fitting because they mitigate the risk of over-fitting by aggregating the predictions of multiple decision trees. Random forests can estimate a variable's importance, while neural networks and SVMs can not [141]. This information can be useful for variable selection and understanding the underlying data, for example which molecular descriptors influence the octane number the most. Random Forests can also handle missing data and outliers effectively by using techniques such as mean imputation or proximity-based methods [54]. Therefore random forests can be useful where some molecular descriptors values are absent. Such a property is missing in other algorithms discussed in this thesis, except for gradient-boosting machines. Another advantage of random forests is that they do not make strong assumptions about the distribution of the data [26], which in the case of fuel data sets is unknown. They can handle a wide range of data types and feature interactions. The last property is important since the interaction between descriptors used for fuel properties modelling and their interaction is not always clear. Random forests' disadvantage is the same as with SVMs, it requires the entire training dataset to be loaded into memory during the training phase. Another potential pitfall of random forests is a high bias towards variables with high cardinality, i.e. when the categorical variables range in the dataset is comparatively big compared to the dataset size [62]. Such an issue can easily happen when modelling fuel properties since the categorical descriptors range could be substantially high. These descriptors may have a higher chance of being selected in the tree splits, potentially impacting the importance estimation of other variables by random forest.

Gradient boosting machines share similar advantages and disadvantages with random forests since both are based on an ensemble of multiple decision trees [115]. The common advantages include estimation of variable importance, handling missing data, do not require normalization of the input data. The common disadvantages of random forests are poor performance with high cardinal variables and computational complexity. GBMs can model highly complex relationships by sequentially building decision trees [151]. However, this property can also lead to over-fitting and hence GBMs require more careful tuning of parameters compared to random forests. In addition, gradient-boosting machines are more computationally complex compared to random forests with the same number of trees, since

sequential tree building prohibits parallelization, because each tree is built one after another is complete.

The k-nearest neighbours (KNN) algorithm is a relatively simple and intuitive algorithm and requires only a few parameters to tune, which are the choice of similarity metric between data points and the number of neighbours. The unseen data-point predictions get predicted values as the average of the neighbors' values. The disadvantages of KNN include sensitivity of variables scaling and curse of dimensionality, which both influence the calculation of the distance between the data points and hence predictions. Variables with large scales can dominate the distance calculation and skew predictions. As the number of variables increases, the density of the data becomes sparse in the high-dimensional space, making it difficult for KNN to find meaningful neighbors and potentially degrading its performance.

In conclusion neural networks and gradient boosting machines have the highest ability in modelling complex relationships, but require careful parameter selection. Random forest algorithms are more robust to outliers, since they use average of many trees. Gradient boosting and random forests algorithms do not require variables normalisation, while the other algorithms can be influenced by the data normalisation process. KNN algorithm is the easiest to implement require the smallest parameters compared to algorithms to tune, however perform worst in high dimensional data. On the other hand support vector machines are much better in high dimensional data. Lastly neural networks can learn new molecular representations, while non-deep learning algorithms can not.

2.5 Variable selection

This part of the methodology discusses variable selection in QSAR modelling. When descriptor molecular representation is used it's unclear, which descriptors should be used as model inputs. Given n descriptors the total number of possible descriptors permutations is 2^n , so it becomes infeasible to try all combinations for large n . For example the number of descriptors which describe fuel properties calculated by [114] is between 1000 and 2000. However, variable selection is needed to avoid over-fitting risk in QSAR modelling [163]. Over-fitting is the problem when the QSAR model demonstrates poor external predictivity while performing much better on the validation part of the data-set. Contradictory [46] concluded that state of the art algorithms such as random forests, support vector machines do not gain much from the variable selection. However, there is no best or worst variable selection method. It can actually be considered as another parameter in the QSAR model optimization process.

There are 3 main types of variable selection techniques, including filter, wrapper and embedded methods [55]. Filter methods measure some dependency of descriptors on the target property value. Thereafter obtained measurement is used to decide whatever variable/descriptor contributes to the target property. An example of the filter method is a correlation-based or mutual information test. Filter methods include distance, information, dependency and consistency methods [56].

The wrapper method to variable selection uses a search algorithm to traverse the space of possible variables subsets and evaluates each subset by evaluating machine learning algorithm performance using the chosen subset. Sequential Forward Selection search algorithm [4] is an example of wrapper method, it reduces the number of descriptors sets for the QSAR model evaluation from the total 2^n to $n(n+1)/2$, where n is the number of variables.

The embedded variable selection method uses a learning model to perform training and variable selection simultaneously [63]. Regularized trees [41], lasso and ridge regressions are examples of embedded variable selections. The comparison by 5 criteria's between filter, wrapper and embedded variable selection methods is given in Table 2.3.

The first column in Table 2.3 checks whatever a variable selection method considers dependencies between variables. The second column checks if the variable selection method choice affects the learning model choice. Ideally, the QSAR model needs to select optimal variables and the learning parameters simultaneously, since the optimal model parameters for a given choice of variables might not be the best model. The third column looks at whether the variable selection method interacts with the training model. Fourth column checks if variable selection can lead to over-fitting. While the last column looks at the impact of variable selection on-time computer executes the method versus the number of features and data set size.

Some trade-offs become apparent in Table 2.3, like variable dependence versus over-fitting risk. The complex relationships between molecular descriptors and fuel properties require not ignoring the interdependence between descriptors. On the other hand some fuel properties data-sets might be too small to use wrapper methods for variable selection and to avoid over-fitting.

Alternative multioptimization method to QSAR variable selection was proposed by [118], where the number of selected variables and models accuracy are optimized simultaneously. In this method Pareto front of non-dominated solutions is created, where for each given number of variables the best set of variables is found, which gives the maximum model accuracy.

| Variable selection method | Variable dependence | Independent of the QSAR model | Interacts with QSAR model | Does not over-fit | Comp. complexity |
|---------------------------|---------------------|-------------------------------|---------------------------|-------------------|------------------|
| Filter univar | ✗ | ✓ | ✗ | ✓ | low |
| Filter multi-var | ✓ | ✓ | ✗ | ✓ | medium |
| Wrapper deterministic | ✓ | ✗ | ✓ | ✗ | high |
| Wrapper randomized | ✓ | ✗ | ✓ | ✗ | very high |
| Embedded | ✓ | ✓ | ✓ | ✗ | medium |

Table 2.3 Summary of variable selection methods

2.6 Validation strategies

Training machine learning models involves splitting data sets into train, validation and test sets. Train data-set is used to train machine learning models. Since training machine learning models involves finding appropriate model parameters, the validation part of the data-set is used to compare different model settings. In addition, the validation part of the data-set is used to compare different variable selection methods if descriptor molecular representation is used for modelling fuel properties. The test part of the data-set is not involved in any part of modelling and it's kept apart for an unbiased estimate of the model performance. In quantitative structure-activity modelling, the division of molecular data-sets into the train, validation and test sets can be categorized into the following three categories:

1. Random selection
2. Based on Y response - molecules targeted by the property of interest
3. Based on X response - molecules clustered by structural similarity or descriptors

The selection of one of these approaches remains an open question in QSAR modelling. Therefore best practices in pharmaceutical research were taken into consideration. [129] investigated the difference between published QSAR model results and the one remodelled with rational splitting technique and concluded that splitting techniques (2) and (3) lead to better external predictivity than random selection. In particular data sets with a small number of molecules showed the biggest improvement in external predictivity. In addition, [34] argued that for the best results, training set data should be well distributed over the full range

of endpoint values, i.e. the Y response. This is often not possible, for various reasons, but very poor distribution, such as two clusters of chemicals, or one or two chemicals far removed from the others, will exert strong model leverage and must be avoided. However, [106] found that models based on rational division (2) and (3) methods generate better statistical results for the test sets than models based on random division, but the predictive power of both types of models are comparable. Therefore all the three methods in data-set splitting were applied to molecules with a number of molecules less than 500.

On the other hand, [107] stated that if QSAR model is sensitive on the method of splitting and distributions on training and test sets this can lead to purposeful selection of train and test to guarantee a good statistical result. Consequently model accuracy based on the single split is insufficient to guarantee the true predictivity of the QSAR model. Therefore multiple splits into train, validation and test sets were used in this research for data-sets, which found to be sensitive on the method of splitting.

Lastly, the proportions of train, validation and test sets were varied. The impact of data-set size was investigated by [137] and the conclusion was drawn that the optimum size of the training set should be set based on particular data set and types of descriptors and statistical analysis is used.

2.7 QSAR modelling practices

2.8 Modeling fuel properties

This chapter reviews methods used to model fuel properties, mainly research octane numbers. Modelling fuel properties includes quantification of the fuel's chemical structure, variable selection, choice of learning algorithm and the model validation strategy, i.e. the same as in any QSAR modelling. Mainly the difference between published fuel properties models lies in the method of molecular description and consequent selection of these variables as model inputs. Occam's razor, or the principle of parsimony, states that all necessary information should be used for modeling and nothing more; thus, it is undesirable to include unnecessary variables in the model. In addition discrepancies in model validation strategies are present.

The first type of QSAR fuel models describe each fuel by functional groups [7],[90], [39], [94], where model inputs predetermined manually by the domain knowledge. Inorganic chemistry functional group represent a substituent in a molecule that causes the molecule's characteristic chemical reactions. Figure 2.19 shows functional groups as inputs to model to predict Cetane number (CN), RON and MON. Based on fact that the CN can be treated as reverse octane number, while RON and MON are closely related the same model inputs

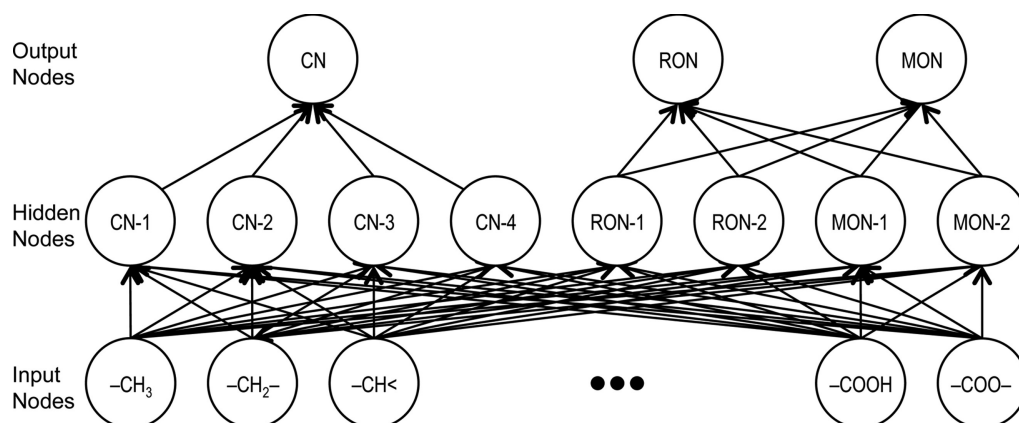


Fig. 2.19 Architecture of the ANN for predicting CN, RON, and MON from the group structure of a fuel molecule [90]

were used in [90]. In addition functional groups were used to model mixture of hydrocarbons octane number [8].

On the other hand physical properties can be used for modelling octane number [6],[160]. However the difference between these papers is that feature selection algorithm was used to determine most relevant physical properties (Table 2.5) in the first paper, while the second paper used domain knowledge to select them. Boruta algorithm and k-nn imputing were used to select physical properties in [160].

Alternately to functional groups and physical properties, molecular descriptors can be used to model fuel properties [98],[37],[80],[95]. The advantage of the descriptor-based approach is that molecules can be described in various levels of detail [61]. However, there is no domain knowledge compared to functional groups or physical properties, which can be used to select molecular descriptors to model octane numbers, so variable selection techniques must be applied.

Correlation-based approach was used to select relevant descriptors in [37] and [80]. Here the descriptors that correlate well with octane number and poorly correlate with other descriptors were kept. On the other hand to account for both the complexity and generalizability of models to resist overfitting, which commonly occurs when using a single-objective feature selection method, Multi-objective optimization with NSGA2 algorithm was used to select relevant descriptors in [98]. Table 2.4 shows the selected molecular descriptors. The optimization objectives included model accuracy and the number of selected descriptors. In this optimization Pareto front of non-dominant solutions was created, where an increase in one objective leads to a decrease in another one. Thereafter the number of relevant

| Symbol | Description | Descriptor type |
|-------------|--|-------------------------|
| ZM2V | Second zagreb index by valence vertex degrees | Topological indices |
| SPAN | Size of descriptors | Geometrical descriptors |
| nCp | Number of terminal primary C | Functional group counts |
| IC1 | Structural complexity per vertex | information indices |
| Mor03u | Signal 03/unweighted | 3D-MoRSE descriptors |
| Mor31m | Signal 31/weighted by mass | 3D-MoRSE descriptors |
| Mor10p | Signal 10/weighted by polarizability | 3D-MoRSE descriptors |
| Mor16m | Signal 16/weighted by mass | 3D-MoRSE descriptors |
| Mor29v | Signal 29/weighted by van der Waals volume | 3D-MoRSE descriptors |
| RDF020m | Radial distribution function -020/weighted by mass | RDF descriptors |
| ALOGP2 | Lipophilic contributions of atoms in the molecule | Molecular properties |
| ALOGPS logS | Solubility in water hydrophilic descriptors | Molecular properties |

Table 2.4 List of Molecular Descriptors Selected by Multi-objective Feature Selection

| features | weight | type |
|----------------|--------|------------|
| XLogP3 | 0.2689 | physical |
| autoignition | 0.1845 | physical |
| CCCCC | 0.1360 | structural |
| LogP | 0.1144 | physical |
| XLogP-AA | 0.0928 | physical |
| complexity | 0.0918 | physical |
| boiling point | 0.0488 | physical |
| vapor pressure | 0.0415 | physical |
| CCCCC | 0.0213 | structural |

Table 2.5 RON Predicting Features by Weight and Type

descriptors was chosen as such which gives a minimum decrease in the model accuracy. This approach in selecting relevant descriptors in QSAR modelling resembles published earlier [147] in pharmaceutical journal. In addition to octane number, bio-fuels viscosity [20] and auto-ignition temperature [142], cetane number[37] were modelled with descriptors as model inputs.

2.9 Knowledge gaps

This chapter summarises knowledge gaps in the application of machine learning techniques in fuels research. Fuel properties predictions modelling demonstrated that fuel research papers are based on advances made in QSAR modelling in pharmaceutical sciences. Therefore some of the main knowledge gaps in fuels research lie in advances made in pharmaceutical sciences. Three major knowledge gaps have been identified, which are the applicability domain, inverse QSAR modelling and the QSAR modes interpretations. The first knowledge gap is focused on the reliability of QSAR fuel models predictions. The gap demonstrates that prototype fuels can be suggested by generative models, which has not been investigated yet. However, the last knowledge gap lies in QSAR modelling techniques themselves, which is an interpretation of machine learning models, which are known for their black box predictions. The rest of this chapter will expand these knowledge gaps and provide references to literature to strengthen the evidence of these gaps.

2.9.1 Applicability domain

The domain of applicability (AD) is an important concept in quantitative structure activity relationships (QSAR) that allows one to estimate the uncertainty in the prediction of a particular molecule based on how similar it is to the compounds used to build the model [157]. This is particularly important in fuels properties modelling, so the QSAR models fuel properties predictions become reliable, especially with respect to prototype fuels.

However there is no single score function, which can define how a given molecule is similar to other set of molecules. The similarity function depends on how molecules been described by descriptors, graphs, e.t.c. For example if molecules are described by structural fragments they consist, then the the most common similarity is given by Tanimoto distance [15]. This score ranges from 1 if molecules are exact the same to zero if they consist of completely different fragments. There are various approaches for determining AD of QSAR models. The most commonly employed approaches for estimating the interpolation regions in a multivariate space include the followings [136]

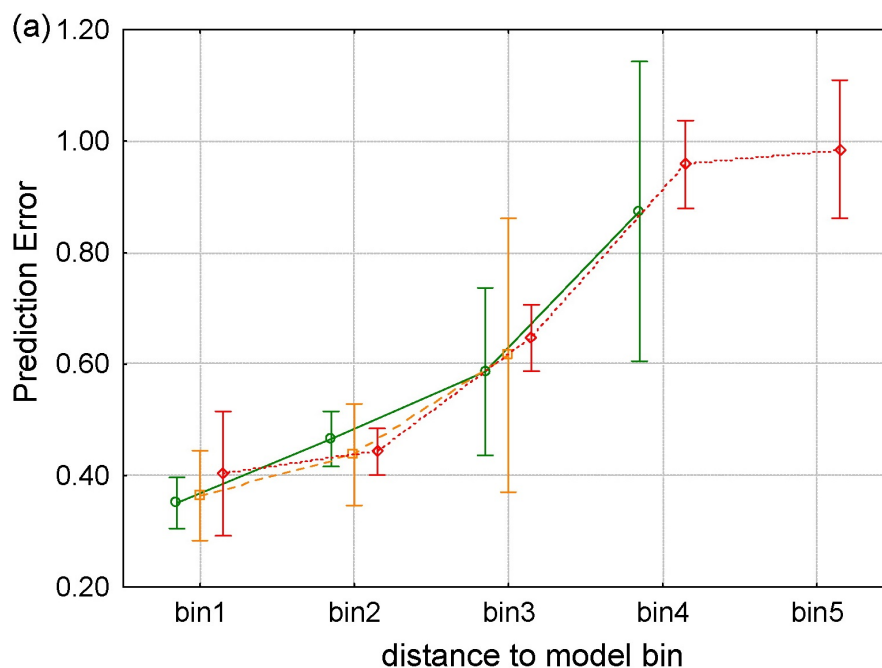


Fig. 2.20 Plot of the absolute error in prediction vs. the binned distance to model for the global plasma protein binding

1. Ranges in the descriptor space.
2. Geometrical methods.
3. Distance-based methods.
4. Probability density distribution
5. Range of the response variable

Here, the first four approaches are based on the methodology used for interpolation space characterization in the model descriptor space. On the contrary, the last one depends solely on response space of the training set molecules.

Various pharmaceutical QSAR studies investigated impact of applicability domain on models predictions. 30 QSAR models were developed using molecular fingerprints, two-dimensional descriptors, and five machine learning algorithms by [156] to predict *PPAR* Binding Affinity. In this study compounds were described by network similarity graphs, which then were used to compute compounds similarity score. Thereafter positive correlation between molecule predictions error and similarity scores were found. In another study [157] absolute QSAR model error predictions were clustered

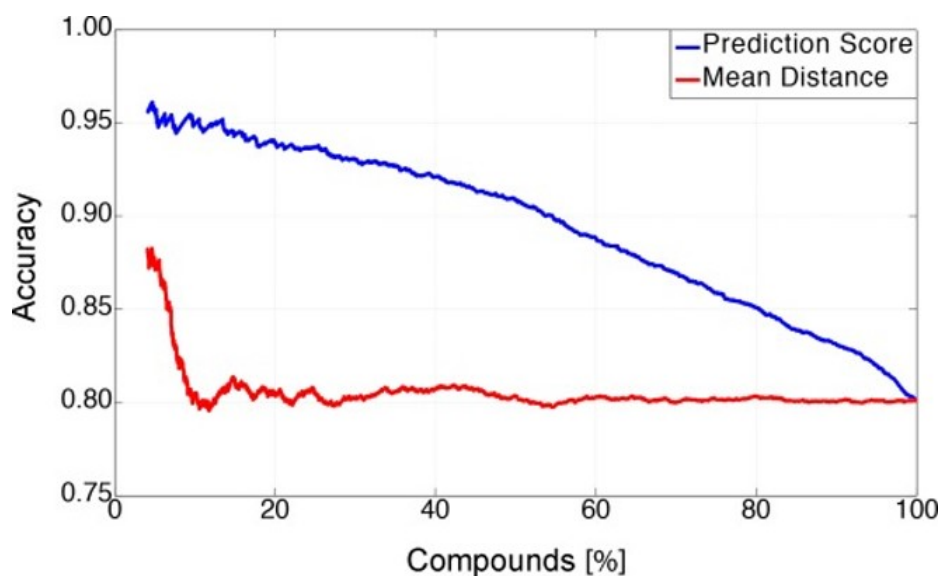


Fig. 2.21 Prediction accuracy vs novelty score [108]

To show how applicability domain impacts QSAR model predictions error in classifying bio-active compounds, model prediction error was plotted against compounds similarity distance score by [108]. The result is shown in Figure 2.21, where its apparent the larger proportion of compounds in test set, which compromise molecules with less similarity to the train set, the higher the prediction error.

2.9.2 Inverse QSAR

The literature review demonstrated the application of QSAR models to predict fuel properties. However, it is desirable to know what chemical structures can be used as new fuels. Such information can be applied to guide prototype fuels experiments. Theoretically compounds with desired properties can be found by inverse QSAR modelling [112]. Deep generative models have been shown powerful in generating novel molecules with desired chemical properties via their representations such as strings, trees or graphs [40]. Despite this, there remain relatively few examples of using generative models in real molecular discovery campaigns. This is because these campaigns are often characterized by a series of additional hurdles, making it difficult to directly deploy generative models for concrete applications [17]. Moreover application of inverse QSAR model to discover new fuels molecules is unexplored.

Recently inverse QSAR models demonstrated ability to make biologically active molecules [30]. However the inverse QSAR modelling application is not limited to drug design only. Polymer dielectrics are essential components in several applications such as electrical insula-

tion, capacitive energy storage, organic photovoltaics and flexible, stretchable and wearable electronic. New polymer dielectrics were discovered by inverse QSAR models by [103]. Organic light-emitting diodes (OLEDs) have great potential to be applied in the third-generation display devices. OLEDs have promising properties, including energy-saving and long-lasting working time [83], both of which outperform liquid crystal display (LCD) technology. New OLEDs were discovered by inverse QSAR models [53]. Moreover the experimentally determined external quantum efficiencies for these synthesized candidates were as large as 22%. Polymer solar cells admit numerous potential advantages including low energy payback time and scalable high-speed manufacturing, but the power conversion efficiency is currently lower than for their inorganic counterparts. Design of new polymeric materials by variational autoencoder model [88] was demonstrated by [82].

2.9.3 Model interpretations

QSAR models are widely applicable but can suffer from regards to the interpretation of the prediction [158]. That applies to fuel properties predictions, since these are a case of the QSAR modelling. Interpretation is the retrieval of useful knowledge from computational models. Models that do not allow for an interpretation of the cause behind the prediction can be underutilized, as the user cannot easily assess the prediction [126]. The problem with interpretation arises from the use of non-interpret-able variables used to describe chemical structures or non-interpret-able machine learning and statistical models in QSAR modelling. The scope of interpret-ability can be further divided into the local and global interpretation. In QSAR, global interpretation is information on structure-property relationships for a series of compounds, helping provide insight on mechanisms of action, activity and properties of compounds. Local interpret-ability gives information about how different parts of a single compound influence the molecular property and can be applied to identify structural motifs, which reduce or enhance the property in QSAR [48]. QSAR interpretation techniques can be grouped into the following categories.

1. Model – Descriptor – Structure Paradigm
 - (a) Machine learning model specific interpretation approaches
 - (b) Machine learning model agnostic interpretation approaches
2. Model – Structure - Paradigm

Machine learning agnostic interpretation in QSAR bio-fuels models was demonstrated by [161], by applying LIME algorithm [133] to developed bio-fuel octane number classification

model [160]. The bio-fuel octane classification model was altered not to include non-interpretable model inputs, despite some loss in accuracy, since LIME is a variable importance algorithm and requires inputs to be interpretable. In addition LIME algorithm provides only local interpretations.

Nevertheless, new approaches have been created to interpret machine-learning models, and therefore QSAR models, which include octane number models, for example, SHAP algorithm [101]. SHAP algorithm assigns variable importance as LIME. Alternatively to variable importance, rule extraction methods were created to interpret models build with support vector machines [16] and artificial neural networks [84]. These methods can provide global model interpretations, which [161] lack.

It is important for the QSAR model to be as accurate as possible to provide meaningful model interpretations. QSAR model interpretations by similarity maps [134], universal structural [127] and computationally matched do not require model inputs to be interpreted and therefore do not trade-off model accuracy with interpretations as in [160].

Chapter 3

Methodology

3.1 Modelling overview

The topography of the thesis methodology is given in this chapter. The modeling consisted of two interdependent directions: from structure to property and the inverse property to structure, namely QSAR and molecular generative modeling. QSAR modeling was used to find relationships between molecular structures and fuel properties, such as research octane number, enthalpy of combustion, and other fuel physical properties by applying machine learning algorithms to molecular data sets with the corresponding fuel properties. Molecular generation on the other hand was used to suggest molecules with the desired properties, such as RON by applying machine learning generative models to large data sets consisting of small organic compounds, which were later assessed by QSAR physical properties models.

QSAR modeling includes data sets collection and processing, molecular representations, and machine learning algorithm training, which encompasses variable selection, algorithm parameter tuning, and model validation. Molecular representations are the process of describing molecules numerically. The variable selection step decides, which obtained molecular representations remain as the predictive model inputs. The algorithm parameter tuning step comprises optimizing the algorithm's parameters to improve its predictive capabilities, based on chosen metrics, such as R^2 in case of a regression task. Consequently validation is used to check the algorithm capabilities on unseen data.

Generative modeling consist of data-set collection, molecular representations, constrains setting, algorithm parameter tuning, and molecular generation followed by properties checking. Both constrains settings and property checking are QSAR models. Constraints settings guide algorithms to generate molecules with the specified constraints, such as increased research octane number or molecular weight in a specified range. The properties checking

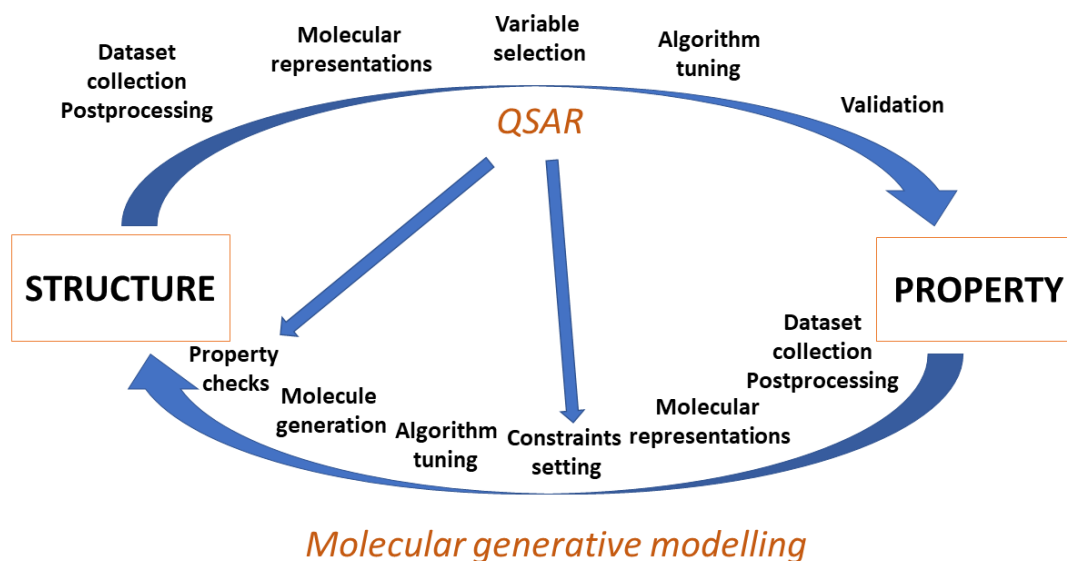


Fig. 3.1 Methodological workflow

involves examining generated molecules in terms of build QSAR fuel physical properties models .

Figure 3.1 summarises methodological workflow. As seen from the figure there are many common steps in QSAR and Molecular generative modelings such as data-set collection and processing, molecular representations, variable selection, and algorithm tuning. The rest of this chapter will provide a technical explanation behind these steps, excluding data-set collection and molecular generation, which are specific to each modeling task.

3.2 Molecular representations

Molecular representations used in fuel QSAR and generative modeling are introduced in this chapter. Molecular representations describe molecules numerically, so algorithms can be applied to molecular data sets. These representations can be grouped by learned/given, mathematical type such as vector or graph, descriptors theoretically or experimentally derived, and so on.

There is no specific rule, which tells what kind of molecular representation is the best. Therefore trial and error approach with respect to molecular representation in modeling fuel properties was present. However, a chosen representation must be compatible with an algorithm, which will be applied later.

| Compound | Smiles | RON |
|--------------------------|----------------|-------|
| Ethanol | CCO | 109.0 |
| 3-Methylpentane | CCC(C)CC | 74.5 |
| 2-Methyl-1,3-butadiene | CC(=C)C=C | 99.5 |
| 1-Octyne | CCCCCCC#C | 50.5 |
| 1-Phenyl-2-methylpropene | CC(=C)c1ccccc1 | 104.1 |
| 3-Heptene (cis) | CCCC=CCC | 90.2 |

Table 3.1 Example of smiles molecular representation in RON data

Smiles

Smiles are the most common molecular representation used in chemical data sets. This representation is in string format, which was derived from graph molecular structure representation [159]. The abbreviation of smiles stands for simplified molecular input line entry system. SMILES syntax consists rules for atoms, bonds, branching e.t.c. To make chemical rules even more strict canonization rules can be added to the SMILES syntax rules, the relating molecular representation is then named canonical smiles. Table 3.1 shows a section of research octane number database with molecules in smiles format.

The benefit of smiles representation is that each smiles string represents only one structure. With this property in mind, smiles molecular representation was used in the molecular generative modeling. In addition, all fuel QSAR modeling started with writing molecular data sets in smiles format.

Descriptors

Molecular descriptors are the most common numerical representation of molecules, which give a particular scalar value to them. For example, molecular weight or atom count are both molecular descriptors.

Molecular descriptors were used in modeling fuel properties, including the octane number. On the other hand, these were not used in molecular generative modeling, since the relationship between molecules and descriptors is one to many, and therefore obtaining a molecular structure from a set of descriptors is not feasible.

The molecular descriptors are divided into experimental and theoretical. Experimental descriptors are molecular properties obtained by specified experiments. Research octane number can be considered as experimental descriptor. Applying various algorithms to molecular graph representation derives theoretical descriptors. Two-dimensional circular connected fingerprint Rogers and Hahn [135] is an example of theoretically derived descriptor (Figure 3.2). Molecular fingerprints represent bit vectors or count vectors. These vectors are

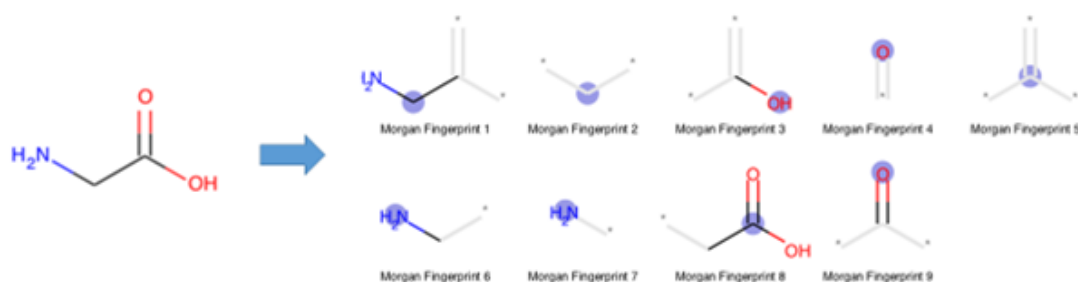


Fig. 3.2 Morgan Fingerprint generation

obtained by applying a kernel to a molecule. Typical kernels extract features from molecules, hash them, and use the hash to determine bits that should be set. Fingerprints encode presence or absence of a particular molecular fragment. This type of molecular representation was used to find similarity between generated compounds.

Theoretical descriptors can be grouped by dimensionality, which refers to the representation of molecules from which the descriptor value was derived.

- 0D-descriptors (i.e. constitutional descriptors, molecular formula)
- 1D-descriptors (i.e. list of structural fragments, fingerprints)
- 2D-descriptors (i.e. graph in-variants)
- 3D-descriptors (3D-MoRSE descriptors, WHIM, quantum-chemical descriptors)
- 4D-descriptors (such as those derived from GRID or CoMFA methods, Volsurf)

Besides, descriptors can be categorized by their nature, including constitutional, topological, geometrical, electronic and quantum chemical. Guha and Willighagen [61]. Constitutional descriptors characterize general properties of molecules and are fragment additive. These include molecular weight, ring count and functional groups. Topological descriptors

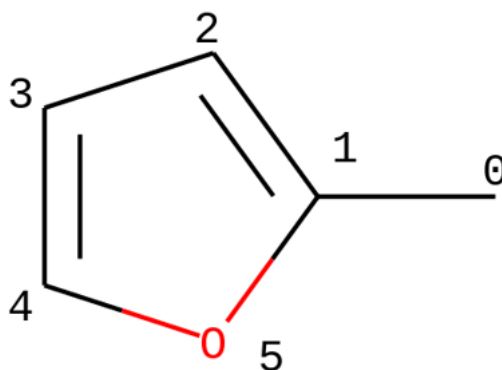


Fig. 3.3 Toluene

are concerned with molecular topology, such as the longest carbon chain or any other molecular structural attributes derived from graph molecular representation. Electronic and quantum chemical derived from physical simulations applied to molecular representation.

After all the benefit of descriptors is that a vast amount of information with respect to molecules can be obtained, which can be used to find relationships and correlations between structures and properties. On the other hand vast amount of information creates an ambiguity about which descriptors to keep as model inputs since the influence of descriptors on a target property can be interdependent.

Graphs

The most intuitive molecular representation is the molecular Graph, where atoms and bonds are graph nodes and edges respectively. Such molecular representation resembles the valence model. Mathematically molecular graph is an undirected graph described by adjacency, node, and edge feature matrices. Advantages of graph molecular representation include the fact that identical molecules represent identical graphs and vice versa. In addition mathematical theory behind the graphs is well understood. This molecular representation was used in both QSAR and molecular generative modelling.

For a molecule with N atoms and M bonds, the adjacency matrix is a binary matrix $A \in R^{N \times N}$ describing atoms connectivity, where A_{ij} is one if there is the bond between atoms i and j and zero if there is no bond. Important to note that the adjacency matrix for any molecular graph is symmetric, that is $A_{ij} = A_{ji}$ for any i, j . The node feature matrix is a matrix $X \in R^{N \times l}$, where each matrix row X_i is vector length l which describes atom i properties, including atom type, valence, e.t.c. The edge feature matrix $E \in R^{M \times g}$, where each row E_i vector size g describes i edge, including the edge type. Adjacency A , node feature X and edge feature E matrices are shown for toluene molecule (Figure 3.3).

$$A = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{vmatrix} \quad X = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \quad E = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix}$$

In this example node and edge feature matrices, rows are one-hot encoded vectors, which show type atoms and bonds respectively. Whereas columns in the edge feature matrix E are ordered by single, double, triple, and aromatic. Columns in the node feature matrix are ordered by carbon, oxygen, nitrogen, and silicon atoms.

Learned representations

Instead of defining the molecular representations before training like with previous molecular representations, these can be learned from data sets. From a machine learning perspective, it is a challenge to encode high-dimensional non-Euclidean molecular data into a feature vector. From wider perspective finding useful features from molecular graph is a case of representation learning from graphs. Two learning molecular representation algorithms developed by Duvenaud et al. [45] and *mol2vec* Jaeger et al. [76] were applied to fuels QSAR model development. The first algorithm shows that chemical structure representations can be found directly from molecular graphs during training procedure by substituting non-differential steps in circular fingerprint generation to differentiable to allow the error signal to back-propagate. The second algorithm uses the idea of natural language model *word2vec* applied to a large set of valid chemical compounds represented by smiles, with an idea that molecules are sentences, while smiles symbols are words. The second approach is also called transfer learning, since the representation was learned one data set and applied to another.

Both molecular representation approaches have advantage to skip variable selection step, since learning will allow to find optimal molecular representation. On the other the differentiable fingerprint Duvenaud et al. [45] is highly data dependant. That means if a dataset is not big or diverse enough, the algorithm might not learn effective molecular representation, which was the case with octane number dataset.

| Molecular representation | Python Library | Reference |
|--------------------------|-------------------|--|
| SMILES | RDKit | (author?) [Apache Software Foundation] |
| Descriptors | Mordred | Moriwaki et al. [114] |
| Graph | PyTorch Geometric | Fey and Lenssen [47] |
| Learned | Mol2vec, DeepChem | dee [1] |

Table 3.2 Python libraries for molecular representations

Python molecular APIs

As the molecular representations are diverse, the same applies to python open source libraries, which implement them. Table 3.2 lists python libraries used to make molecular representations

Molecular representations applied in this research

Each methodology chapter used their own molecular representations. Molecular descriptors were used in regression tasks, since the fuels data-sets sizes were not sufficiently large to learn new molecular representations. For the classification tasks mol2vec model [76] was used to find vector representation of molecules. Such representation was useful to train classification algorithms, since mol2vec model creates similar vectors for similar molecules. Compared to circular fingerprints, which are commonly used for QSAR modelling such representation overcomes bit collision problem [153]. To generate new fuel molecules graph and smiles representations were used. The advantage of graph representation was higher degree of variety of molecules to produce, while string based representation is easier to train with.

3.3 Interpretation

3.3.1 Introduction

The aim of this methodology chapter is to show that knowledge about fuels can be extracted from trained machine learning models, which are non-interpretable. The idea is to compare existing domain knowledge about fuel auto-ignition properties and interpretations behind classification models, which predict if a given molecule has a high octane rating. The data source and description chapter shows where the data to model octane classifications models was taken from. The modelling sub-chapter explains which algorithms were used to make the classification models and how they were optimized. The more detailed experiments set up is

written in the corresponding results chapter. Lastly, the method used to get interpretations behind trained classification models is explained with examples.

3.3.2 Data source and description

Two databases from the octane number modelling papers [98] and [160] were used in constructing the octane number prediction dataset. The first database consisted of 227 molecules, while the second 147. The first database represented fuel chemical structures by canonical smiles, while the second by PubChem ids. PubChem ids are unique compound identifiers in the PubChem database [18]. Pubchem ids were calculated for the first dataset and the two data-sets were merged according to PubChem ids.

Octane numbers were converted to binary values, where 0 represents an octane rating less than 94.4, with 1 representing octane numbers greater than this threshold, which represents a value typical of widely utilised fossil gasoline. In addition fuel chemical structures were standardized by ChemAxon® software, which included aromatization of cyclic bonds. For example the benzene rings in canonical smiles are written as a sequence of double-single carbon-carbon bonds, while in practice the ring includes aromatic bonds.

3.3.3 Modelling

Octane data-set was split by 20% test and 80% training and validation. Four classification models were developed by support vector machine (SVM), random forest (RF), gradient boosting machine (GBM) and k-nearest neighbours (KNN) algorithms. The hyper-parameters for each algorithm were found by Bayesian optimization, where the optimization target was 10-fold cross stratified validation on the train validation set and the validation metric utilised the ROC-AUC score. ROC-AUC is the area of True positive rate versus False negative rate under the different probabilistic thresholds, where an area closer to 1 indicates better classifier, and if the area is equal to 0.5, the classifier performance is equal to a random estimator. Figure 3.4 shows schematic representation of 10-fold stratified cross validation schematics. The parameters of each model, which gave the best score in the cross validation are then chosen. Thereafter models with these parameters are trained on the whole train-validation set.

The 10-fold cross-validation is necessitated by relatively small octane number data-set, since the sample variability between training and test part could be quite considerable. For example an algorithm could only be trained on paraffins and olefins but tested on cyclic compounds and oxygenates. Stratification was used to ensure similar distribution of high and

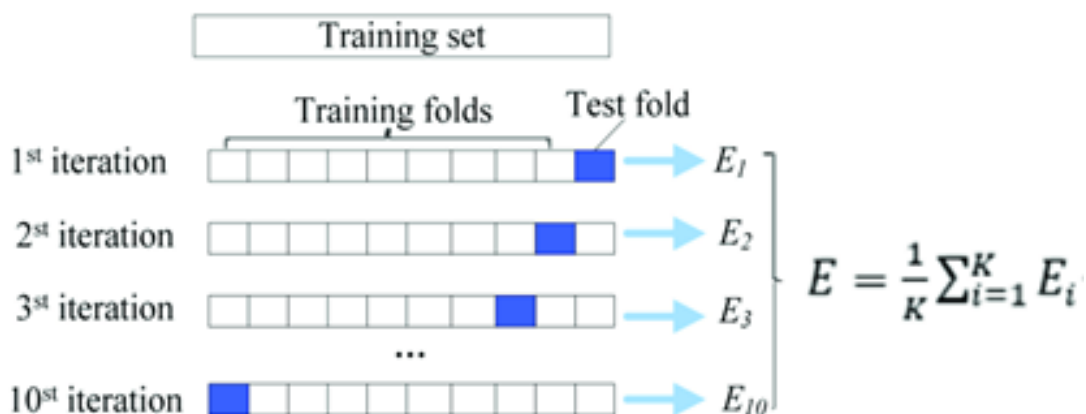


Fig. 3.4 10-fold cross validation

low octane fuels in each fold as in the whole data-set since the chosen ROC-AUC metric is sensitive towards the class imbalance.

Classification algorithms using SVM, RF, GBM and KNN are not probabilistic so their probability estimates need to be interpreted with caution. Calibration was therefore required to improve likelihood estimates of the models. Platt's scaling was used to calibrate the likelihood estimates and scaling is the application of logistic regression to output probabilities of classifiers and the true labels in the data-set. Calibration quality can be quantified by Brier's score and visualized by a reliability diagram, which plots the actual fraction of fuels having a high octane number versus probabilistic bins range.

For example, there were two groups of fuels, the first of which was predicted to have probability of having a high octane number in a range between 0 and 0.2. The second had a probability range between 0.2 and 0.4. However, the actual fraction of fuels having a high octane number was 0.25 and 0.4 for the first and the second group of fuels. Consequently, the reliability diagram will illustrate two-point (0.15,0.25) and (0.3,0.4). The closer these points to the diagonal in the reliability diagram the better are the likelihood estimates.

3.3.4 Interpretation method

This sub-chapter illustrates the interpretation method and results for the four octane number classification models. Fuels were described with mol2vec embedding, so the description of the fuel had neither physical nor explicit chemical meaning. Accordingly, interpretation by model structure paradigm must be chosen, since finding the variable importance or the rule extraction methods will not provide intuitive model interpretation. Secondly, different types of machine learning algorithms were chosen for the octane number classification model to reduce or find bias between different types of models. Therefore the interpretation

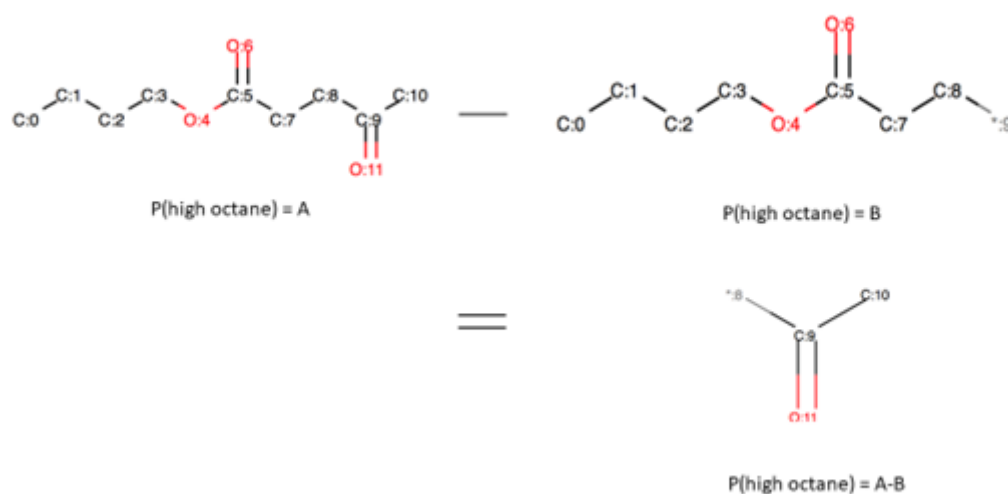


Fig. 3.5 Universal interpretation adapted from [127]

method must be model agnostic. Figure 3.5 illustrates the universal interpretation approach [127] applied in finding the ketone group contribution to the likelihood of Butyl Levulinate belonging to a high octane number range. Here two molecules compared, whose difference is only the presence of the ketone group. If ketone group presence increases the likelihood of fuel having a high octane number than it has positive contribution and vice versa. Such method has origin in chemical informatics drug discovery, known as Matched molecular pair analysis (MMPA) [42].

3.4 Applicability domain

3.4.1 Introduction

This methodology sub-chapter introduces a method to find fuel properties models' applicability domain. The aim of finding the applicability domain is to establish confidence in the model's predictions. New bio-fuel structures might be too dissimilar to molecules in fuel properties data sets, so there is a need to distinguish, which bio-fuel properties can be predicted or not. The applicability domain was established by finding the relationship between models predictions errors and the similarity of test compounds and compounds used to train fuel properties models. The goal of this study was to find such a similarity measure

that would correlate with model prediction errors and will work on multiple fuel properties data-sets.

The fuel properties data sets selected in this study included research octane number (RON), boiling point (BP), auto-ignition temperature (AIT), enthalpy of combustion (EC), density and dynamic viscosity. Four applicability domain techniques were applied to these fuel properties, including Tanimoto distance, Leverage distance, one class SVM and isolation forest.

The methodology is divided into fuel properties selection, data collection, QSAR modelling and applicability domain. The first sub-chapter explains the reason behind the selected fuel properties. Thereafter modelling behind fuel properties is shown. Algorithms parameter selection is shown in the corresponding results chapter. The applicability domain subchapter explains the chosen applicability domain methods.

It was found that anomaly detection algorithms OneClassSVM and IsolationForest could establish the applicability domain of fuel QSAR models compared to distance measures such as Leverage and Tanimoto by illustrating a constant decrease in the mean average error of the selected fuel molecules.

3.4.2 Fuel properties selection

There are 6 main characteristics of highly qualified automobile engine fuel. The first is research octane number, which is responsible for smooth combustion without engine knocking. The next one is the enthalpy of combustion which is directly related to the power output.

It is essential for spark ignition fuel to mix with air under engine operating conditions. Internal combustion fuels must have sufficient volatility to form a combustible mixture. The boiling point is a fuel property related to the mixing of fuels with air, which measures the temperature at which a substance evaporates at atmospheric pressure. Furthermore boiling point is related to other fuel physical properties such as vapour pressure, density, latent heat of vaporization, viscosity, and the fuel surface tension [167]. In general, lower boiling point fuels exhibited higher flame temperatures, less visible flame, and lower particulate emissions [145].

Similar to octane number auto-ignition temperature tells about fuel ignition. The auto-ignition temperature or kindling point of a substance is the lowest temperature at which it spontaneously ignites in a normal atmosphere without an external source of ignition, such as a flame or spark. This temperature is required to supply the activation energy needed for combustion.

Density and kinematic viscosity are two key properties of automotive fuels. Density is important because it relates volume directly with weight in the fuel tank and indirectly with the energy content. Viscosity influences the lubrication properties as well as the combustion properties of the fuel. Low viscosities lead to poor lubrication, which can cause excessive wear and leakage. Higher viscosities can cause an obstruction of the hoses or poor atomization of the fluid leading to poor combustion and an increase in exhaust gas emissions.

3.4.3 Data collection

This subchapter summarises and describes the sources of the selected fuel properties data. The octane number data-set came from 3 published papers [98], [160] and [154]. The first paper explored variable selection in predicting the octane number, while the second built classification model, which tells if a fuel has octane number higher or lower than 95. The last papers attempts to collect property database of fuel compounds with emphasis on spark-ignition engine applications. Boiling point data was taken from Handbook of thermodynamic and physical properties of chemical compounds [165], which collects physical, thermodynamic and transport properties for organic compounds covering C1 to C70 organics. This book was also used to get data for entropy. In addition entropy data was taken from paper on entropy predictions [9]. Reid vapor pressure P on the other hand was computed by Antoine (equation 3.1)

$$\log_{10}P = A - \frac{B}{T + C} \quad (3.1)$$

Where coefficients A, B and C for each compound were collected by web scrapping from NIST Chemistry WebBook [97], while temperature T was set to 38.7 degrees.

The autoignition temperature dataset was collected from 2 chemical engineering books [23] and [164], which collect data for organic compounds. The first book is a chemical engineering book for students. The second book contains almost 450,000 data records covering physical, thermophysical, thermodynamic, transport, safety, and environmental properties for over 5000 inorganic substances and over 35,000 organic substances. Lastly, data for autoignition temperature data was taken from paper [86], which predicts hydrocarbon auto-ignition temperature.

Last two properties density ρ and dynamic viscosity ν are temperature dependant. These properties can be computed by Rackett and GuzmanAndrade respectively, which take into account temperature.

$$\rho = \frac{1}{R} \frac{A}{\frac{B}{C} D^{1 + (1 - \frac{T}{B})^E}} \quad (3.2)$$

| Fuel property | Size |
|---------------------------|------|
| Octane number (RON) | 492 |
| Boiling point | 5549 |
| Auto ignition temperature | 944 |
| Enthalpy of combustion | 5172 |
| Reid vapour pressure | 403 |
| Density | 3930 |
| Dynamic viscosity | 6660 |

Table 3.3 Datasets sizes

$$v = A \exp\left(\frac{F}{T} + G\right) \quad (3.3)$$

Density data was taken from Physical Properties and Refractive Index table from [165]. The viscosity data was taken from published work on predicting viscosity for bio-fuels [142].

Table 3.3 shows the resulting data-sets sizes after combining data from different sources and removing duplicates

3.4.4 QSAR modelling

Descriptor-calculation software Mordred [114] was used to compute descriptors for each molecule in fuel properties datasets (Table 3.3). The number of descriptors computed for each compound was equal to 894. Problems can arise when the number of variables is greater than the number of samples; therefore, it is necessary to select a subset of descriptors which contribute to fuel property of interest. Firstly descriptors with normalised variance less than 0.01 were removed, because these descriptors were almost constant. Thereafter highly correlated descriptors were removed with the coefficient of determination larger than 95%.

Fuel QSAR models were built by a full factorial experiment involving 4 machine learning algorithms and 4 variable selection algorithms. Machine learning algorithms included multi-layer perceptron (MLP), support vector machine (SVM), gradient boosting regressor (GBM) and random forest (RF).

Variable selection methods included 2 wrapper methods "Wrapper + linear model" and "Wrapper + learner", elastic net and no variable selection. "Wrapper + linear model" and "Wrapper + learner" mean that the linear model was used to evaluate the selected descriptors, while in the second case the same algorithm was used in the wrapper method to select descriptors and the algorithm was used to model fuel properties. These two option choice was justified because wrapper methods are good at finding the set of descriptors for a particular model, i.e the second case. However wrapper methods are susceptible to over-

fitting, so wrapper with a simple model such as linear was used. The elastic net model is an embedded variable selection algorithm commonly used in QSAR modelling. [46]

3.4.5 Applicability domain

In this work, 4 applicability domain techniques were used. One-Class Support Vector Machine (denoted as 1-SVM), IsolationForest (denoted IF), Leverage and Tanimoto distances. This sub-chapter explains how these were used to establish the fuel properties applicability domain.

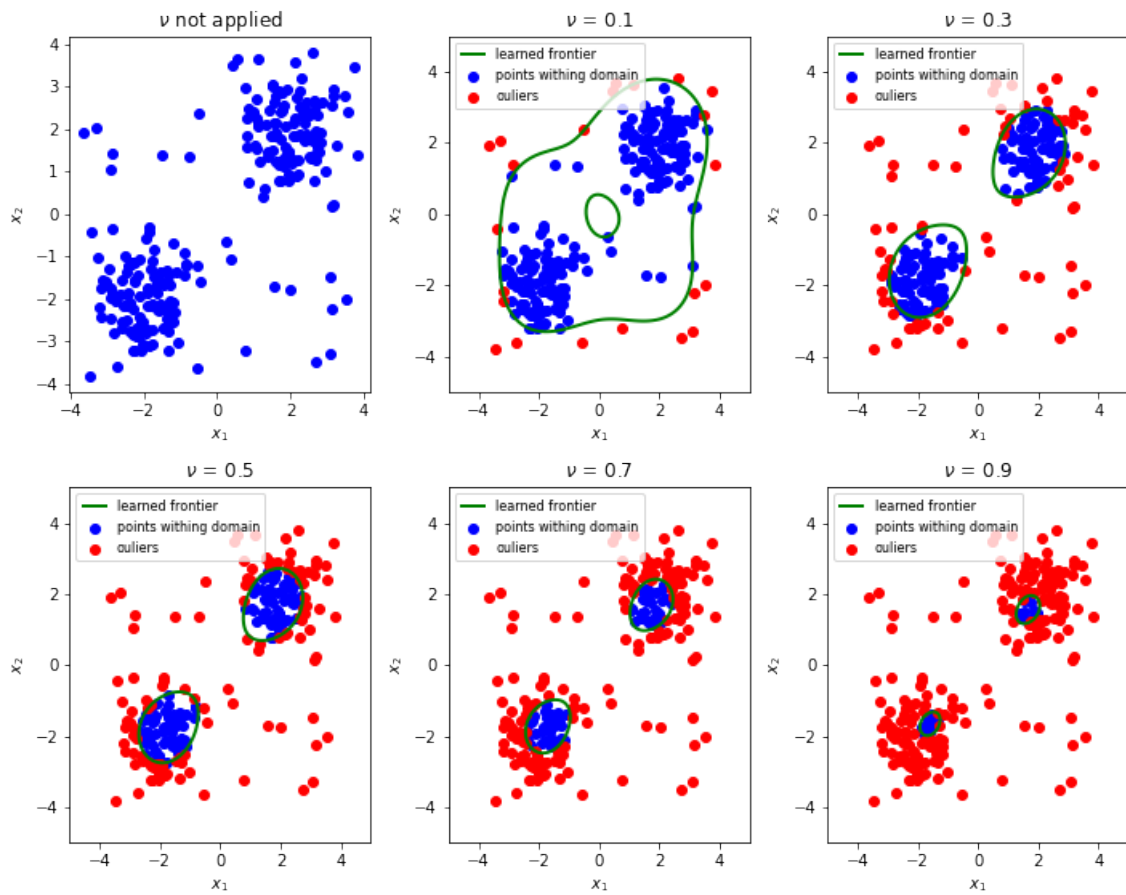
1-SVM and IF are algorithms used for outlier detection, i.e both find data points, which are considerably different to the rest of the data points. Both algorithms are classification models, which return two values, one for the data point being an outlier or not. These algorithms have a common parameter $\nu \in (0, 1)$, which controls what portion of data is considered to be outliers. Figure 3.6 illustrates how 1-SVM separates data points depending on ν applied to synthetic 2-dimensional data with two visibly clear clusters.

In 1-SVM and IF fuel properties applicability domain was found by the learned frontier, with data points within it being inside the applicability domain. The learned frontier is shown by green lines and points within the domain are coloured blue in Figure 3.6. The learned frontier was found by fitting 1-SVM and IF to train data used to model fuel properties. Thereafter test data points were classified by the learned frontier within and outside of the model's applicability domain. Consequently mean error of predictions for data points within the applicability domain was computed. Parameter ν was varied to change the learned frontier location and the respective mean of error within the domain was computed. Thereafter mean prediction errors within the domain were plotted against the ν value.

The other two methods Leverage and Tanimoto were distance based. Both methods compute a scalar value, which tells how a compound in the test set is different to training data. Tanimoto distance J is a common measure of similarity between compounds with values between 0 and 1, where 0 means no similarity and 1 is for identical compounds.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (3.4)$$

Where A and B are each respective molecule's fingerprints. The fraction in 3.4 means the ratio between the number of common fingerprints in two molecules divided by the number of unique fragments in both molecules combined. The similarity between a compound from the test set and train data was computed by finding the average for similarities between the compound from the test set and all the compounds in train data.

Fig. 3.6 Impact of contamination factor ν on data points classification

The leverage distance is based on the Mahalanobis distance to the centre of the training-set distribution. The leverage distance h between compounds in the data set is computed by

$$h = x^T (\mathbf{X}^T \mathbf{X})^{-1} x \quad (3.5)$$

Where x is a test compound descriptors value and \mathbf{X} is the training set descriptors values. Important to note is that equation 3.5 involves matrix inverse, so for some training set descriptors values \mathbf{X} the leverage distance can not be computed if $\mathbf{X}^T \mathbf{X}$ is not a full rank matrix.

Once these distances were computed, these were normalised by the maximum distance, so the compound with the largest distance had a value of 1. Consequently, mean prediction errors were computed for compounds with distance values less than $\alpha \in (0, 1)$. This was done in order to observe how the distance affects the test error predictions.

3.5 Generative modelling

3.5.1 Introduction

This methodology sub-chapter introduces 2 generative models used to generate fuel-like molecular structures. The purpose of this study is to investigate the generative model's ability to create new bio-fuel molecular structures. Bio-fuels can be potentially produced from a variety of sources and their molecular structure could be different to intuition. Similar to model interpretations domain knowledge about fuel structures will be used to make assessments of the generated fuel-like structures.

The role of generative models is to capture the underlying rules of a probability distribution. In this research, it was the distribution of molecular compounds of interest. Given a collection of training sample points $\{X_i\}$, a model is trained to match data distribution P_X , by means of a generative process P_Y , where Y P_Y resembles the real data X P_X . Compared to classification models, which find $P(Y|X)$ given data-set in form $\{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$, generative models find a joint distribution $P(X, Y)$.

There are four main machine-learning models, which can be applied to generate molecular structures. These include likelihood-based variation auto-encoders [88] and auto-regressive models [144], likelihood-free generative adversarial networks [58], and reinforcement approaches [173]. In this study, adversarial networks and auto-regressive models were used. The benefit of auto-regressive models is that they can be trained to modify a given molecule and are much easier to implement and train compared to reinforcement approaches. Adver-

serial networks don't assume any data distribution during training compared with variational auto-encoders and can come up with novel molecules. In addition study by [27] showed that auto-regressive models with LSTM networks outperformed other models in terms of generated molecular qualities.

The sub-chapter starts describing the first generative model used in this study, which was generative adversarial network. Description includes architecture choice for the generator and discriminator networks. Consequently the second generative model is introduced, which was auto-regressive model that replicates SMILES string generation by similar to words by language models. Thereafter metrics used to evaluate quality of generated molecular structures is given, followed by the models set up procedure.

3.5.2 Generative adversarial network

The second model that was trained to output valid organic molecules was a generative adversarial network. These networks do not explicitly define probability density functions as the above auto-regressive model. Instead, GANs take a game-theoretic approach by learning to generate from training distribution through a 2-player game. The game is defined by discriminative D and generative G networks with a mini-max training objective.

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (3.6)$$

Where p_{data} and p_z are unknown molecular distribution and normal distribution with mean zero and variance 1. The learning objective is to find the transformation function G , such that maps from the normal distribution p_z to molecular p_{data} . The discriminative network D learns to distinguish real molecules $X \sim p_{data}(x)$ and fake $G(z)$ where, $z \sim p_z$, while generator network G tries to fool the discriminator, so the discriminator will treat generated samples as real.

In this working generator, G consisted of three consecutive dense layers with a hidden number of nodes 128, 256, 512, and the output from the last layer is projected by 2 dense layers onto dimensions $N \times F$ and $N \times N \times T$. To convert the obtained continuous representations into discrete node feature matrix X and adjacency tensor A , the Gumbel-Soft Max activation function was used. Where N is the maximum number of atoms, excluding hydrogen. F is the number of atoms types, which was set to 3 and includes carbon (C), oxygen (O), and no atom (NA). A_{ij} describes the connection type between nodes i and j , which includes no connection. Generative model G always outputs symmetric A , such that $A_{ij} = A_{ji}$, i.e for any $i, j \leq N$, since molecules are undirected graphs.

The discriminator was a graph neural network with relational graph convolution layers described in the methodology section. It took the adjacency tensor A and node feature matrix X from both real molecules and generated and mapped them to a scalar between $(0, 1)$, which represented the probability if A and X represent a fake molecule.

3.5.3 Auto-regressive model

Given a sequence of words (w_1, \dots, w_n) language model can predict the distribution of the next $w_{(n+1)}$ word. For example, if a language model receives the ‘University College’ word London will have a higher probability than another word. Language models capture grammatical correctness and context. Similarly, language models can be applied to molecular generation, by modeling molecules as a sequence of smile symbols. Mathematically the model gives the likelihood of sequence of symbols $S = (s_1, \dots, s_n)$ by

$$P_{\theta}(s) = P_{\theta}(s_1) \prod_{i=2}^T P_{\theta}(s_i | s_{i-1}, \dots, s_1) \quad (3.7)$$

Where θ are learnable model parameters. Language models differ in terms of formulation of P_{θ} . In this work, recurrent neural networks were used to model P_{θ} . The model was trained by optimizing the log-likelihood of the next symbols by

$$\log P(S) = \sum_{i=1}^n \log P(s_i | s_1, s_2, \dots, s_{i-1}) \quad (3.8)$$

The input to the model is a sequence of symbols of molecular SMILES strings, which includes atoms and bond symbols. Each smile’s sequence was given a ‘start’ token at the beginning. This token was used to indicate the beginning of any sequence. If the length of a sequence was shorter by k symbols than the length of the longest one, additional ‘PAD’ tokens were appended to the sequence. The model ground truth labels consisted of the same sequence as the input but shifted with one step forward. Therefore the ‘start’ token was omitted, but the ‘end’ token was appended to indicate the end of the sequence. Figure 3.7 illustrates how the model processed the input ethanol molecule.

The trained auto-regressive model does not directly make sequences but rather computes conditional probabilities of the next symbols. Therefore sampling strategy is important to factor in the model inference. The most simple way to generate smile sequences from the trained model is to take the greedy approach. Here each sequence symbol is generated according to the maximum likelihood from the learned probability distribution P_{θ} . However, such a strategy will give only one molecule. To increase the diversity of generated molecules each

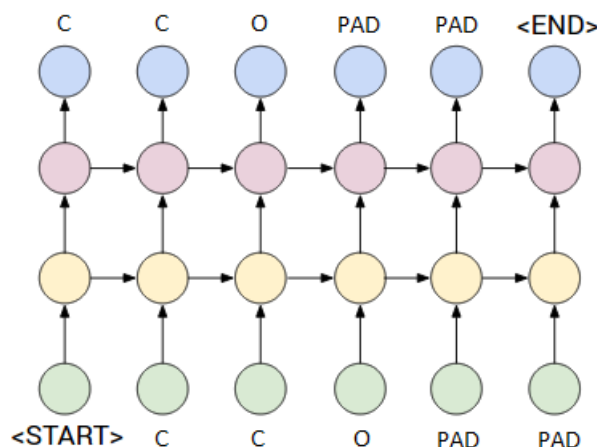


Fig. 3.7 Auto-regressive model set up

conditional token probability (Equation 3.7) was skewed with temperature hyperparameter T , which provides a trade-off between randomness and correctness in sampling.

$$P(x_{s+1} = k | s_1, \dots, s_n) = \frac{\exp\left(\frac{y_i^k}{T}\right)}{\sum_{i=1}^K \exp\left(\frac{y_i^k}{T}\right)} \quad (3.9)$$

Where T is a positive scalar when $T = 1$ the conditional distribution is equivalent to the learned one. On the other hand, $T \rightarrow 0$ will make the distribution equivalent to Dirac delta, while $T \rightarrow \infty$ will transform the distribution to uniform. The consequence of the first extreme is that the sampling will be equivalent to the greedy approach, but the second extreme will lead to completely random sampling. Here the model is given a ‘start’ token. Then the model gives the probability distribution of the second token by (Equation 3.9). The second token is sampled from the obtained distribution. Thereafter the model gives the third token distribution given the first two. This process repeats until the ‘end’ token is reached. It’s worth noting that the model input can be one or more tokens in length instead of ‘start’ only. This allows the model to generate molecules from a molecular fragment encoded as a smiles string.

Lastly, the trained model can be fine-tuned to output molecules with desired properties. In this work, a hill-climbing (Figure 3.8) algorithm was used to modify trained auto-regressive model weights.

3.5.4 Generated molecules assessment

To check generated molecule’s properties three criteria were used including validity, uniqueness, and novelty. The validity was measured as the ratio between the number of generated

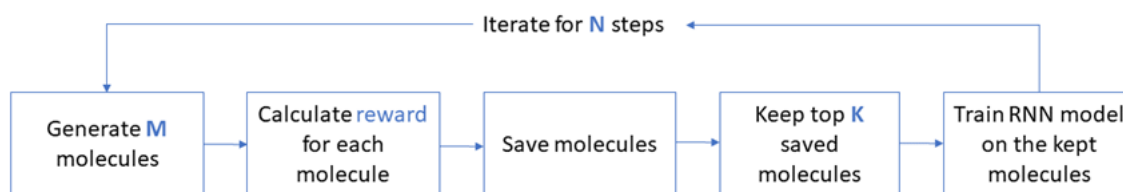


Fig. 3.8 Hill Climb algorithm

compounds and the ones which were actually valid. Valid, that is, whether they correspond to an (at least theoretically) realistic molecule. For example, molecules with an incorrect SMILES syntax, or with invalid valence, are penalized. RDKit python package was used to check the correctness of generated smiles string for the auto-regressive model and a written script was used to check the valence correctness for the molecular graph generated by the GAN model.

The uniqueness benchmark assesses whether models are able to generate unique molecules. It is desired for the model not to output repetitive samples. Therefore uniqueness was measured as a ratio of unique and valid molecules divided by the total number of generated compounds.

The optimal goal for the generative model is to be able to make molecules not seen in the training set used for generative models. Consequently, the novelty was defined as the ratio of molecules that are valid and unique and not present in the training set divided by the number of generated molecules.

3.5.5 Experiment strategy

To check the generative models' ability to make future bio-fuel structures three tests were given to generative models. The first test was a valid organic molecules test, which shows the model's ability to make valid organic molecular structures since bio-fuels are organic molecular structures. Both models were trained on QM9 data-set [130], which consists of organic molecular structures with several heavy atoms up to 9. Such size of heavy atoms, i.e. not including hydrogen is common in conventional hydrocarbon fuels used for spark ignition engines. Generated molecules were then checked in terms of validity in terms of chemical structures and later their uniqueness compared to qm9 data-set used for training.

Models which pass the first test are then checked for their ability to generate valid and unique organic molecules with high octane values. To accomplish that trained models on the qm9 data-set were fine-tuned by the developed octane regression model from the applicability domain chapter. Thereafter generated molecular structures will be checked again in terms of their uniqueness and validity. However in the second test uniqueness will be compared not only with QM9 data set but also the data-set used to train the octane regression model. Consequently generated unique molecules will be checked in terms of traits of high knock resistance molecular structures, such as presence of double bonds, branching and a short main carbon chain.

The last test will check models ability to generate unique molecular structures with high octane values and increased presence of the oxygen atoms, since bio-fuels contain oxygen atoms.

Chapter 4

Interpreting octane number predictions

4.1 Classification models validation

The quality of the developed models was evaluated by accuracy, precision, sensitivity, specificity and F1 score on the test set of molecules, which were not used in any part of the model development. Accuracy shows the overall model performance, while precision is the proportion of fuels which were predicted as having a high octane number actually had high octane number. Sensitivity is the proportion of fuels which had a high octane number and were predicted as such. Specificity is proportion of fuels which had low octane number were predicted as such. F1 score is a hybrid metric which takes into account precision and recall. Table 4.1 shows these metrics on the test set of octane database by the 4 developed models. Figure 4.1 shows the ROC-AUC curve for the developed classification models evaluated on the test set. The ROC-AUC curve shows the classification models false positive rate versus the true positive rate for the different classification thresholds. False positive rate is the ratio of fuels predicted to have a high octane number among fuels actually having an octane number in the low range. The true positive rate is the ratio between fuels predicted having a high octane number among fuels in the high octane range. The probability threshold converts probability of fuel belonging to high octane to binary range ,i.e a fuel is either in high or low octane number range.

The four developed octane number classification models (Table 4.1) displayed ROC scores 0.93,0.92,0.93 and 0.90, which is higher than 0.88 in [160]. Average accuracy was slightly less than 0.88 and was equal to 0.85,0.86,0.86 and 0.82. Sensitivity was almost identical, except for the KNN model with numbers being equal 0.91,0.89,0.89,0.77 compared to 0.88. Precision was also mostly identical with values equal to 0.90,0.89,0.9, and 0.80 compared to 0.88. This comparison indicates that the developed classification models have similar performance as in [160]. Reliability diagram is included in Figure 4.2

| Model | Accuracy | Precision | Sensitivity | Specificity | F1 score |
|-------|--------------|--------------|--------------|--------------|--------------|
| SVM | 0.849 | 0.909 | 0.914 | 0.789 | 0.845 |
| RF | 0.863 | 0.889 | 0.886 | 0.842 | 0.865 |
| GBM | 0.863 | 0.899 | 0.886 | 0.842 | 0.865 |
| KNN | 0.822 | 0.805 | 0.771 | 0.868 | 0.835 |

Table 4.1 Classification models test result

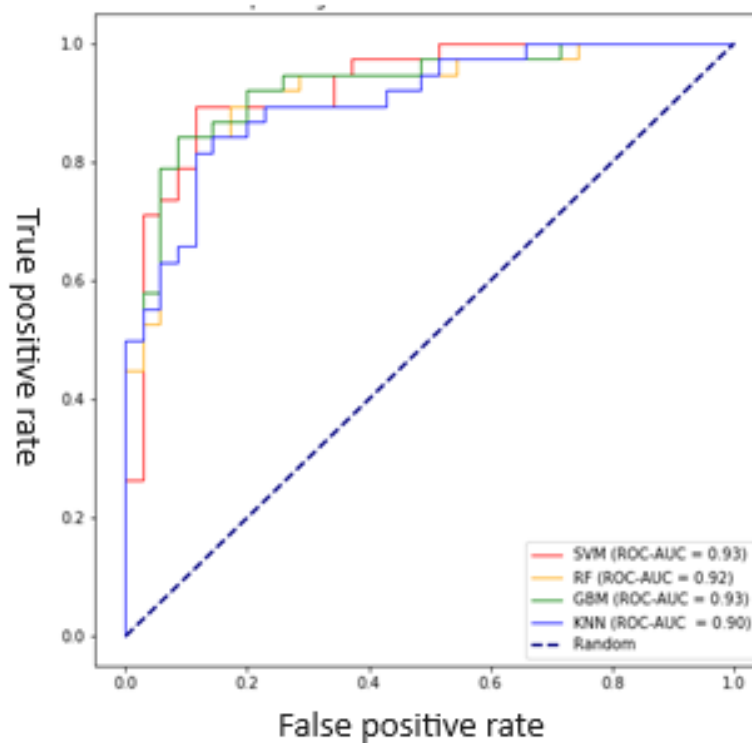


Fig. 4.1 ROC-AUC curve for the four classification models

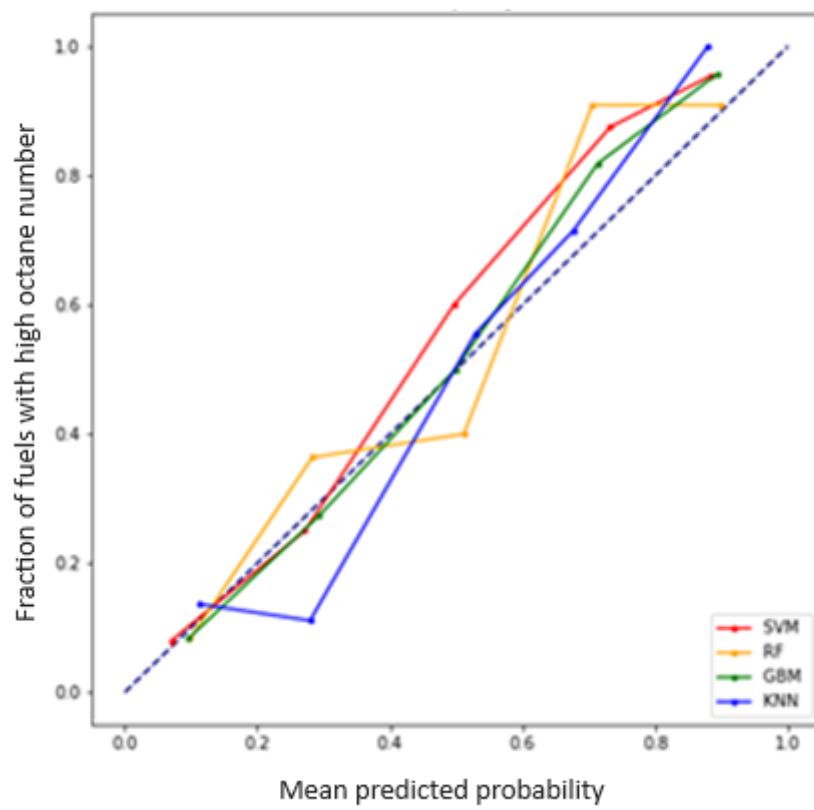


Fig. 4.2 Reliability diagram for the four classification models

4.2 Interpretation results

Six different interpretation cases were considered, which are summarised in Figure 4.3. In each case a compound with known octane number was modified in the same manner as in matched molecular pairs analysis [42]. Thereafter the developed octane number classification models reported the likelihood of the modified compounds having an octane number higher than 94.4. The rest of this sub-chapter discusses the observed changes in likelihoods and corresponding molecular structure modifications.

Case 1

The extent of branching was reduced in 2,2-dimethyl-3-ethylpentane. All four-classification models reported considerable drop in the likelihood of fuel belonging to high-octane class, while the original compound belongs to the high octane class with an octane number equal to 102.

Case 2

Dicyclopropylmethane has octane number equal to 96.0, and is classified as a high octane fuel. The cycloalkane structure was removed from Dicyclopropylmethane and four models reported almost equal probabilities of the modified compound to have either low or high octane number. The result can be interpreted as significant drop in ignition properties attributed to the removal of the cycloalkane. However, the original compound had an octane number close to 94.4, with the difference less than 2 octane points. Therefore equal probabilities of the modified compound can indicate that the modified molecule has an octane number close to the threshold, and that the removal of cycloalkane has no effect on ignition properties.

Case 3

Branching was reduced in T-butylbenzene in the same manner as with 2,2-dimethyl-3-ethylpentane. However, all four classification models reported an almost 100% likelihood of the modified compound to have high octane number. This result is completely opposite to that in Case 1. Therefore the influence of branching upon ignition properties might depend on the surrounding functional groups. In this case benzene ring was present, which is known to have high impact on ignition properties [168].

Case 4

The myrcene double bond was changed to a single bond, all four classification models reported likelihood of modified compound belonging to high octane twice lower than low octane range. The result can indicate that double bond removal did not significantly improve ignition properties of Myrcene, which was already in low octane range with octane number equal to 82.5. However the saturation from single to

Case 5

Branching was again investigated in the context of cyclic compounds. This time one carbon atom was removed from Sabinene. A significant drop of the likelihood of fuel belonging to the high-octane class was reported, which is opposite in the context to (Case 3) considering tert-Butylbenzene.

Case 6

Carbon was substituted to oxygen in Valero lactone. The tree based algorithms, RF and GBM reported significant drop in the likelihood. However, the Support vector machine and K-nearest neighbouring's algorithms did not report a significant decrease in the likelihood of the resultant molecule belonging to the high-octane range.

4.3 Conclusion

The interpretations of the classification models (Figure 4.3) were able to make a direct comparison of fuel structures with slightly different structural changes, as shown in case 6 when oxygen was substituted to carbon in Valero lactone. Furthermore, the interpretations obtained by this research method are able to illustrate the effect of small structural changes, such as carbon-carbon bond modification as in case 4. Higher granularity in developed octane number classification interpretation method compared to [161], was achieved by employing compact vectorised chemical information mol2vec [76] and machine learning model agnostic QSAR interpretation [127]. Case 1 demonstrated agreement with respect to branched alkanes known ignition properties [19]. By removing a carbon atom, one extra site for oxidation was available, and consequently the likelihood of the modified compound belonging to high octane class was low. Case 3 demonstrated agreement with [66], by showing structural dominance of benzoic bonds, where reduced branching did not show any significant influence on ignition properties, due to the nearby presence of the benzoic ring structure. Benzoic ring structures are known for high ignition resistance due to high stability of π -bond in the

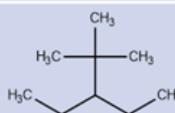
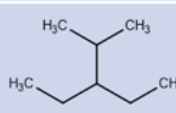
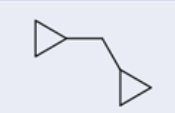

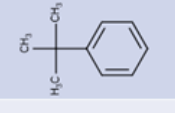
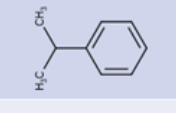
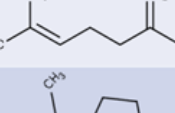
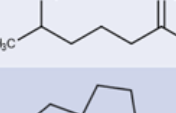
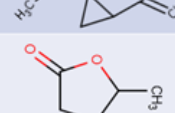
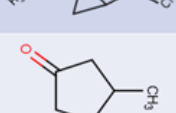

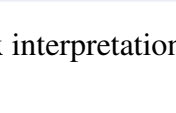
| Compound structure | Modified compound structure | Compound RON before modification | Investigated influence off | Predicted probability high octane class P(RON>94.4) by |
|--|---|----------------------------------|-------------------------------|--|
| 1)  |  | 102.0 Class 1 [RON>94.4] | Reduced branching | SVM: 0.26 RF: 0.14 GBM: 0.18 KNN: 0.00 |
| 2)  |  | 96.0 Class 1 [RON>94.4] | Cyclic group removal | SVM: 0.45 RF: 0.46 GBM: 0.51 KNN: 0.33 |
| 3)  |  | 108.0 Class 1 [RON>94.4] | Reduced branching | SVM: 0.99 RF: 0.99 GBM: 0.89 KNN: 1.00 |
| 4)  |  | 82.5 Class 0 [RON<94.4] | Double bond changed to single | SVM: 0.10 RF: 0.27 GBM: 0.30 KNN: 0.33 |
| 5)  |  | 107 Class 1 [RON>94.4] | Reduced branching | SVM: 0.54 RF: 0.30 GBM: 0.37 KNN: 0.33 |
| 6)  |  | 100 Class 1 [RON>94.4] | Oxygen substituted by carbon | SVM: 0.89 RF: 0.37 GBM: 0.41 KNN: 0.67 |

Fig. 4.3 Six interpretation cases by the four classification models

benzene ring. However, this was not the case, when branching was reduced in Case 5, where there was no presence of strong π bond. Case 2 was difficult to interpret since the structure before modification had octane number close to the 94.4 threshold and after modification, all four models reported almost equal likelihood of the fuel belonging to the high or low octane ranges. Investigating case 6 was quite ambiguous with the 4 models showing different results. This divergence could indicate that either the modified structure is too different from the rest of the octane number database or a modelling failure. In order to distinguish between these two sources of errors further work on models application domain needs to be done. However, the strategy of using different classification models and validating interpretations made such attention to detail as in case 6 possible. In addition to branching, ring functional group, carbon-carbon bonds and substitution of atoms on octane number was demonstrated in this report (Figure 4.3). This has not been achieved previously since the only octane QSAR model interpretation method [161] relies on much coarser fuels chemical structure representation, i.e molecular fingerprints. In addition, the obtained octane number classification models interpretations were mostly the same with respect to the different algorithms used. Therefore results obtained were not biased to a particular algorithm.

This chapter investigated interpretability behind the octane number quantitative structure activity models (QSAR), since quantitative structure activity models developed with the aid of machine learning algorithms provide high accuracy [6], [20] and [90], but lack interpretations behind them.

Chapter 5

Fuel properties models applicability domain

5.1 Fuel properties modelling result

This sub-chapter shows the result of regression models used to predict fuel properties. Regression modelling consisted in finding the best variables selection method in conjunction with the machine learning algorithm tuned. The variable selection methods included an embedded method of the elastic net, sequential feature selection method with 2 learning algorithms either linear model or support vector machine and no variable selection method. Each data set was split into the test set 20%, which was not used in any form of modelling. The rest of the data set was used for 5-fold cross-validation, which compared the choice of the variable selected with the machine learning algorithm trained. Negative mean squared error was used to measure the quality of cross-validation. Optuna python module [5] was used to find machine learning hyper-parameters such that maximise negative mean squared error on the cross validation. The parameter search grid is shown in Table 5.1

| Algorithm | Parameter | Parameter | Parameter |
|-----------|--------------------------|------------------------------|------------------------|
| SVM | $10^{-3} < C < 10^3$ | $10^{-3} < \gamma < 10^{-1}$ | $2 < d < 10$ |
| RF | $2 < maxdepth < 64$ | $4 < maxleafnodes < 20$ | $5 < maxfeatures < 30$ |
| MLP | $10^{-5} < lr < 10^{-2}$ | $10^{-4} < \alpha < 10^{-1}$ | $3 < bz < 10$ |
| GBM | $2 < maxdepth < 10$ | $5 * 10^{-2} < lr < 10^{-1}$ | $5 < maxfeatures < 30$ |
| GBM | $20 < ntrees < 300$ | $0.2 < subsample < 0.9$ | |

Table 5.1 Models parameters grid

| Property | Model | Model Parameters | Var selection |
|-----------------|-------|---|---------------|
| RON | SVM | $C = 993$ $d = 4$ $\gamma = 0.099$ | no var |
| AIT | GBM | $lr = 0.93$ $maxdepth = 6$ $ntrees = 189$ $subsample = 0.73$ $maxfeatures = 21$ | no var |
| Heat Combustion | GBM | $lr = 0.93$ $maxdepth = 5$ $ntrees = 157$ $subsample = 0.63$ $maxfeatures = 22$ | wrapper(lin) |
| Boiling Point | SVM | $C = 742$ $d = 5$ $\gamma = 0.087$ | wrapper(lin) |
| Density | GBM | $lr = 0.93$ $maxdepth = 8$ $ntrees = 132$ $subsample = 0.81$ $maxfeatures = 17$ | wrapper(lin) |
| Viscosity | SVM | $C = 876$ $d = 5$ $\gamma = 0.213$ | no var |

Table 5.2 Selected hyper-parameters in best performing models

Table 5.3 shows the result of QSAR fuel properties models on the test set, The table includes regression results in terms of coefficient of determination R^2 , mean average error (MAE) and mean average percentage error (MAPE) for the combination of different learners and variable selection methods. The best parameters found by the optuna package are shown in Table 5.3.

It's evident that heat combustion, boiling point and density were among the best performing models due to the dataset sizes. However, viscosity performed slightly worse than expected, which is due to the uneven distribution, which resembled exponential distribution in target values.

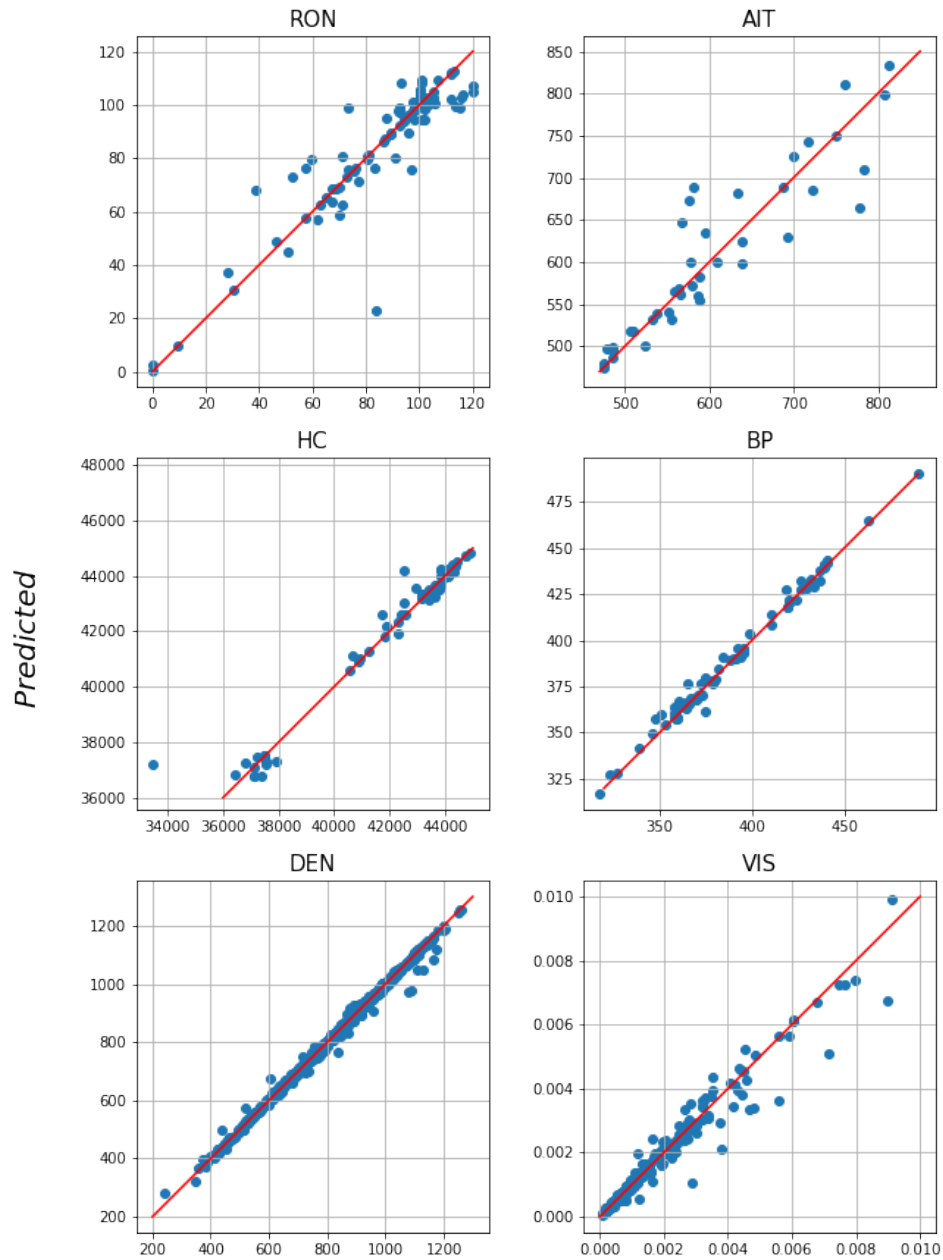
It is evident that the combination of variable selection and learners among different fuel properties was not the same for the best-performing models. However, SVM and GBM algorithms performed the best. This result can be partially explained by the way models were optimized. Models parameters were optimized with Bayesian optimization package optuna,

| Property | Dataset size | Model | Var selection | R^2 | MAE | MAPE |
|-----------------|--------------|-------|---------------|-------|------------------|------|
| RON | 492 | SVM | no var | 0.82 | 5.22 | 0.10 |
| | | GBM | elastic | 0.81 | 5.22 | 0.10 |
| | | MLP | wrapper(lin) | 0.76 | 7.26 | 0.15 |
| | | RF | wrapper(rf) | 0.69 | 7.62 | 0.12 |
| AIT | 492 | GBM | no var | 0.83 | 26.14 | 0.04 |
| | | SVM | wrapper(svm) | 0.79 | 26.5 | 0.04 |
| | | RF | wrapper(lin) | 0.80 | 26.82 | 0.04 |
| | | MLP | no var | 0.82 | 31.85 | 0.05 |
| Heat Combustion | 1000 | GBM | wrapper(lin) | 0.96 | 231.32 | 0.01 |
| | | RF | wrapper(RF) | 0.95 | 289.9 | 0.01 |
| | | SVM | wrapper(svm) | 0.95 | 308.71 | 0.01 |
| | | MLP | wrapper(lin) | 0.92 | 474.15 | 0.01 |
| Boiling point | 8 | SVM | wrapper(lin) | 0.99 | 2.64 | 0.01 |
| | | MLP | wrapper(lin) | 0.99 | 2.76 | 0.01 |
| | | GBM | wrapper(lin) | 0.98 | 3.24 | 0.01 |
| | | RF | wrapper(lin) | 0.93 | 5.98 | 0.02 |
| Density | 8 | GBM | wrapper(lin) | 0.99 | 4.68 | 0.01 |
| | | MLP | wrapper(lin) | 0.99 | 7.82 | 0.01 |
| | | SVM | no var | 0.99 | 10.37 | 0.01 |
| | | RF | wrapper(lin) | 0.9 | 31.85 | 0.04 |
| Viscosity | 8 | SVM | no var | 0.95 | $1.60 * 10^{-4}$ | 0.11 |
| | | MLP | wrapper(lin) | 0.95 | $1.80 * 10^{-4}$ | 0.13 |
| | | GBM | wrapper(lin) | 0.87 | $1.97 * 10^{-4}$ | 0.13 |

Table 5.3 QSAR fuel properties models performance

which builds a probabilistic model in models hyperparameter space and searches for the best hyperparameters. Since it takes longer for MLP and RF to perform one calculation compared to SVM and GBM, less steps were taken in the optimization process. The lower speed is attributed to the computational complexity of the selected machine learning algorithms.

Figure 5.1 illustrates expected values for fuel properties on the test set versus the predicted values for the best-performing models (Table 5.3) for each fuel property. The closer the points to the red line in Figure 5.1 better the predictions. The viscosity (VIS) plot in Figure 5.1 looks biased for larger viscosity values, this can be explained again by the much lower number of data available for larger viscosity values.



Expected

Fig. 5.1 Predicted and actual fuel properties, where RON(no units), AIT(K), HC(KJ/kg), BP(K), DEN(mg/cm³), VIS(mPa s)

5.2 Applicability domain results

The first approach to finding the influence of chemical structure on prediction error was measuring the mean Tanimoto distance between the tested compounds and all compounds on the train set used to build fuel property models.

Figure 5.3 illustrates the octane model prediction errors versus the mean Tanimoto distance. The four algorithms used to build the octane model illustrate similar patterns. However, the correlation between the errors and mean Tanimoto distance can not be observed. On the other hand, all four models show that the worst prediction in terms of absolute error was observed at the largest Tanimoto distance. In order to investigate the Tanimoto distance, the errors were grouped at a binned distance, which is illustrated in Figure 5.4. Again the pattern between the error distribution and bins was not observed. The conclusion was made not to proceed with Tanimoto distance to establish applicability domain for other fuel properties because in octane models the Tanimoto distance could not subgroup compounds in the test set in terms of prediction errors.

Thereafter tested compounds were grouped by outlier detection algorithms OneClassSVM and IsolationForest, by varying ν parameter, which influences which compounds are considered to be within the applicability domain or not. Figure 5.8 shows the mean absolute prediction error for compounds considered to be within the applicability domain according to the anomaly detection algorithms. The error values for $\nu = 0$ correspond to the error on the whole test set (Table 5.2).

All fuel properties and algorithms used for modelling illustrate a constant decrease in the prediction error for compounds considered to be inliers by anomaly detection algorithms with the ν parameter between 0 and 0.4. Therefore the anomaly detection algorithms could group compounds by structural information in such a way that correlates with models prediction errors. This can be explained by the fact that anomaly detection algorithms consider compounds within the applicability domain by their similarity in descriptor space to compounds used in the training set.

However, some differences in trends were evident in Figure 5.7, which can be attributed to the distribution of the train compounds in the descriptor space used for modelling different fuel properties. On the other hand, the trends in errors were similar for the models used to predict a given fuel property.

Since the octane number is a prime fuel property of interest error distribution for different ν values were plotted in Figure 5.5 and Figure 5.4. The difference between the two figures is that the first one includes the worst predictions, while the second does not. It can be observed that there is a constant decrease in error values variance as ν gets larger, with the largest

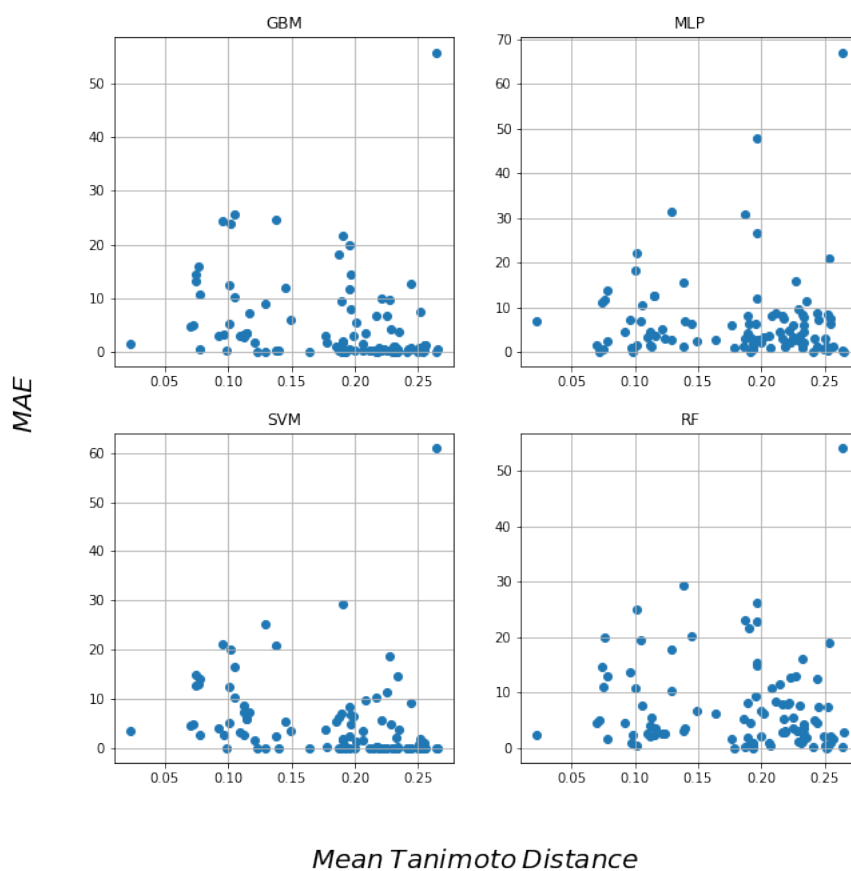


Fig. 5.2 Mean average absolute octane number prediction error vs mean Tanimoto distance

decrease observed for the SVM model, which was the most accurate for the octane number predictions (Table 5.2).

To investigate further if the worst predictions attributed to different compounds for the 4-built octane number model Table 5.8 was created. The table includes the worst 10 predictions for 4 octane number models built by different algorithms. It's evident that the same compounds showed the worst performance.

5.3 Conclusion

In this study 4 applicability domain techniques were investigated including 2 distance-based Tanimoto and Leverage, and 2 anomaly detection algorithms OneClassSVM and IsolationForest. These were applied to 6 fuel properties data-sets including research octane number, autoignition temperature, density, viscosity, boiling point and heat combustion.

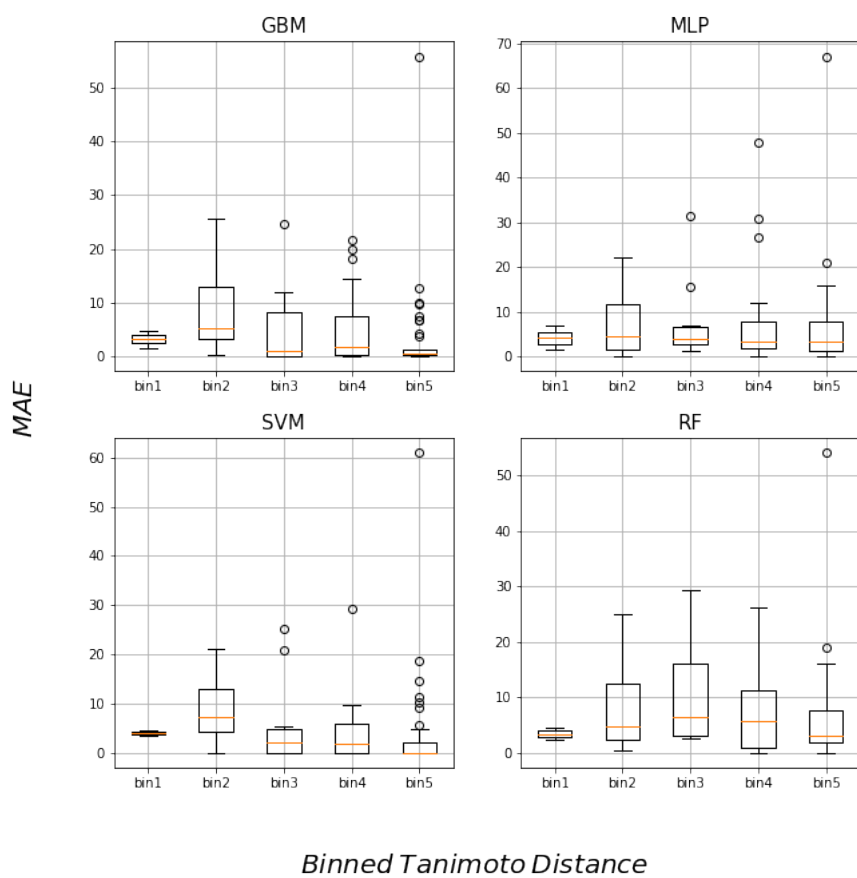


Fig. 5.3 Octane predictions errors distribution at different mean Tanimoto distance bins

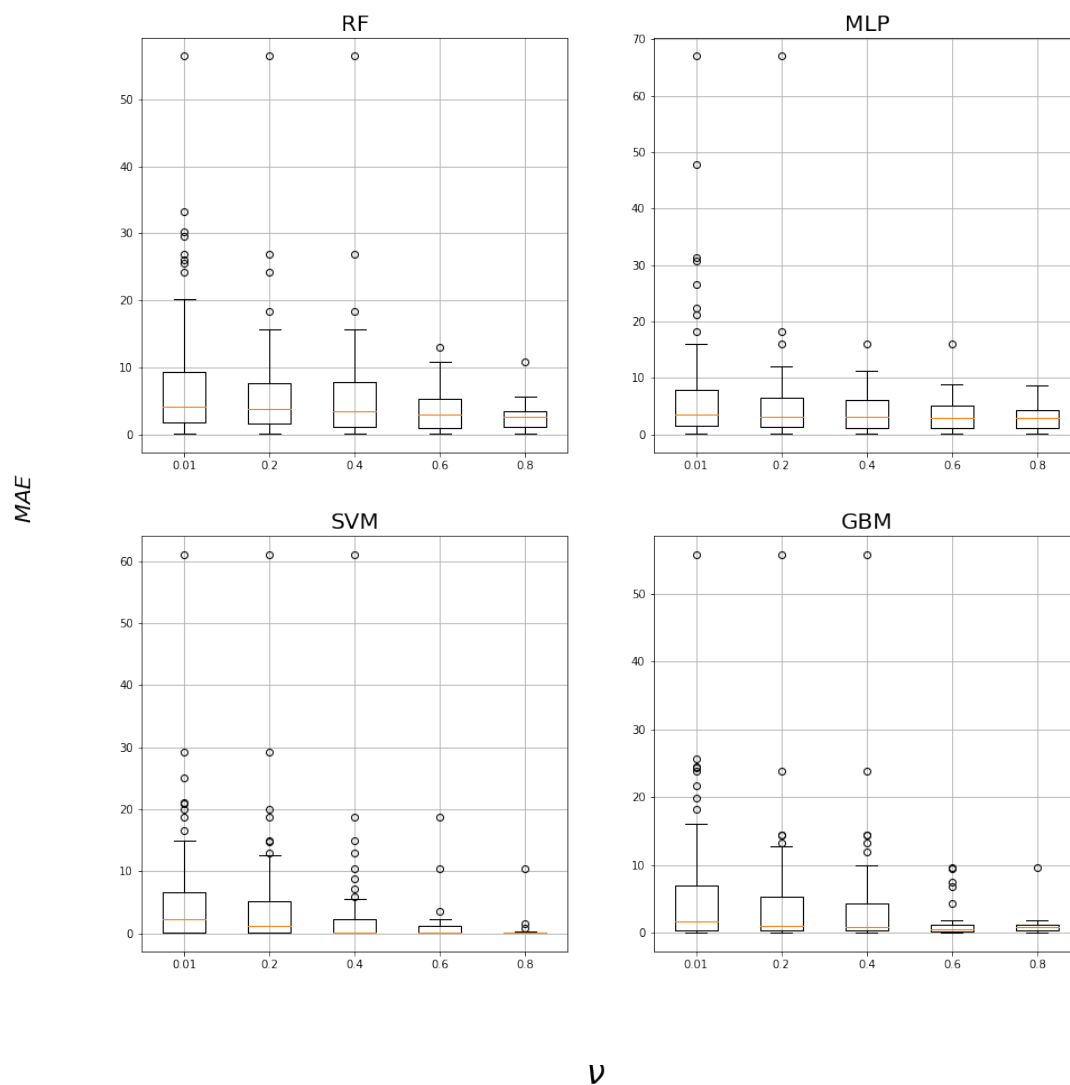


Fig. 5.4 Octane models mean average absolute error distribution vs contamination hyperparameter ν

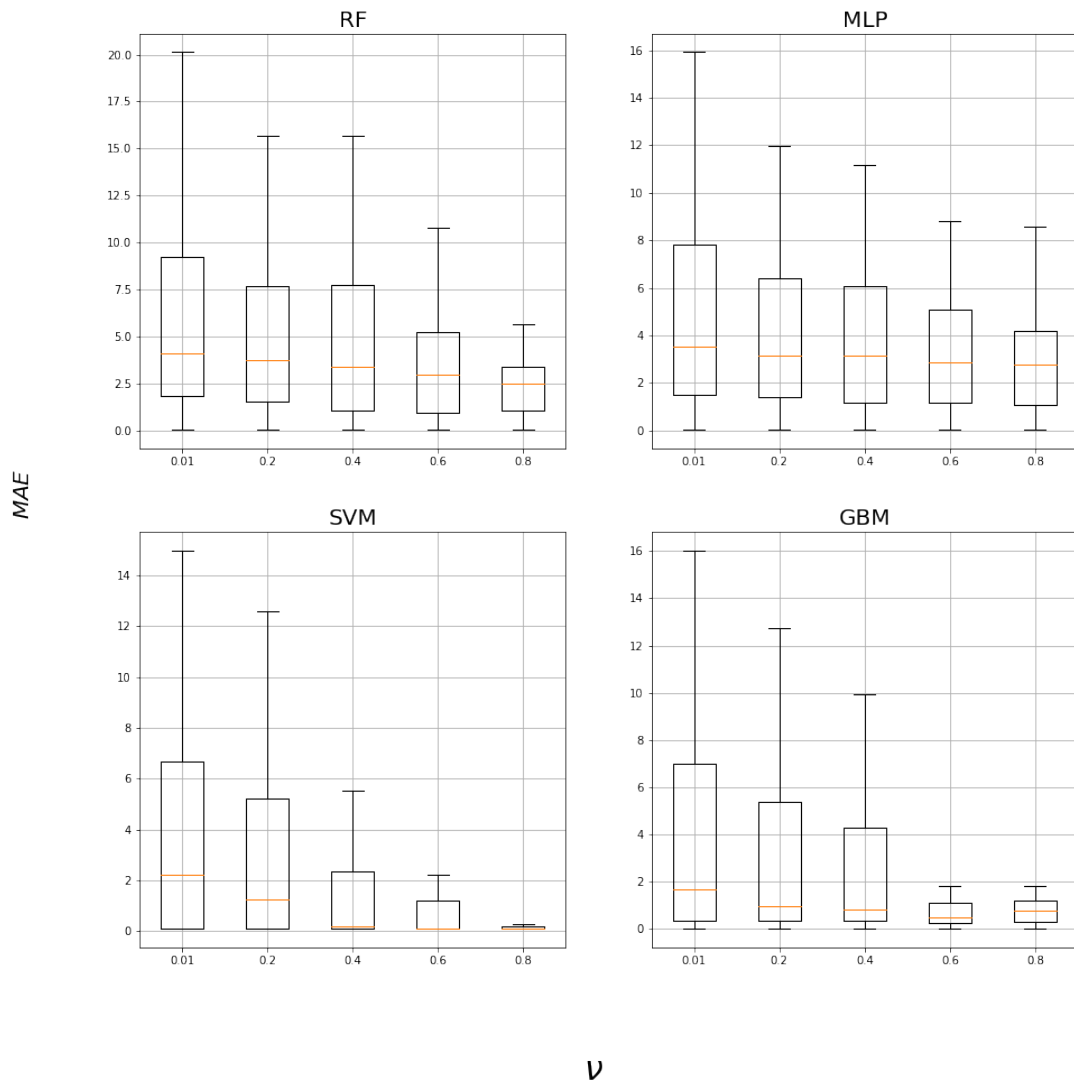


Fig. 5.5 Octane models mean average absolute error distribution vs contamination hyperparameter ν with outliers in previous figure excluded

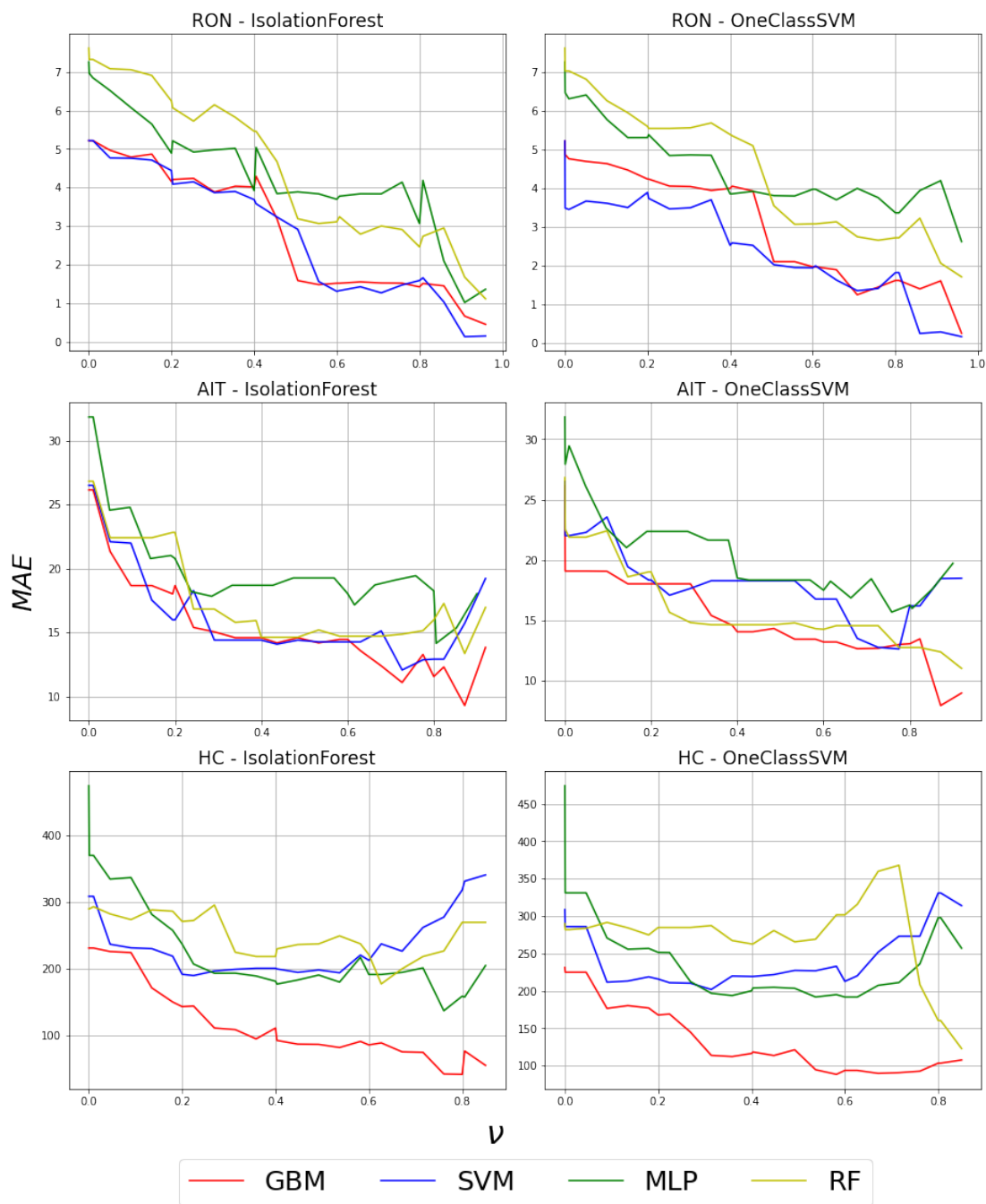


Fig. 5.6 Impact of contamination factor ν on predictions mean absolute error for different fuel properties

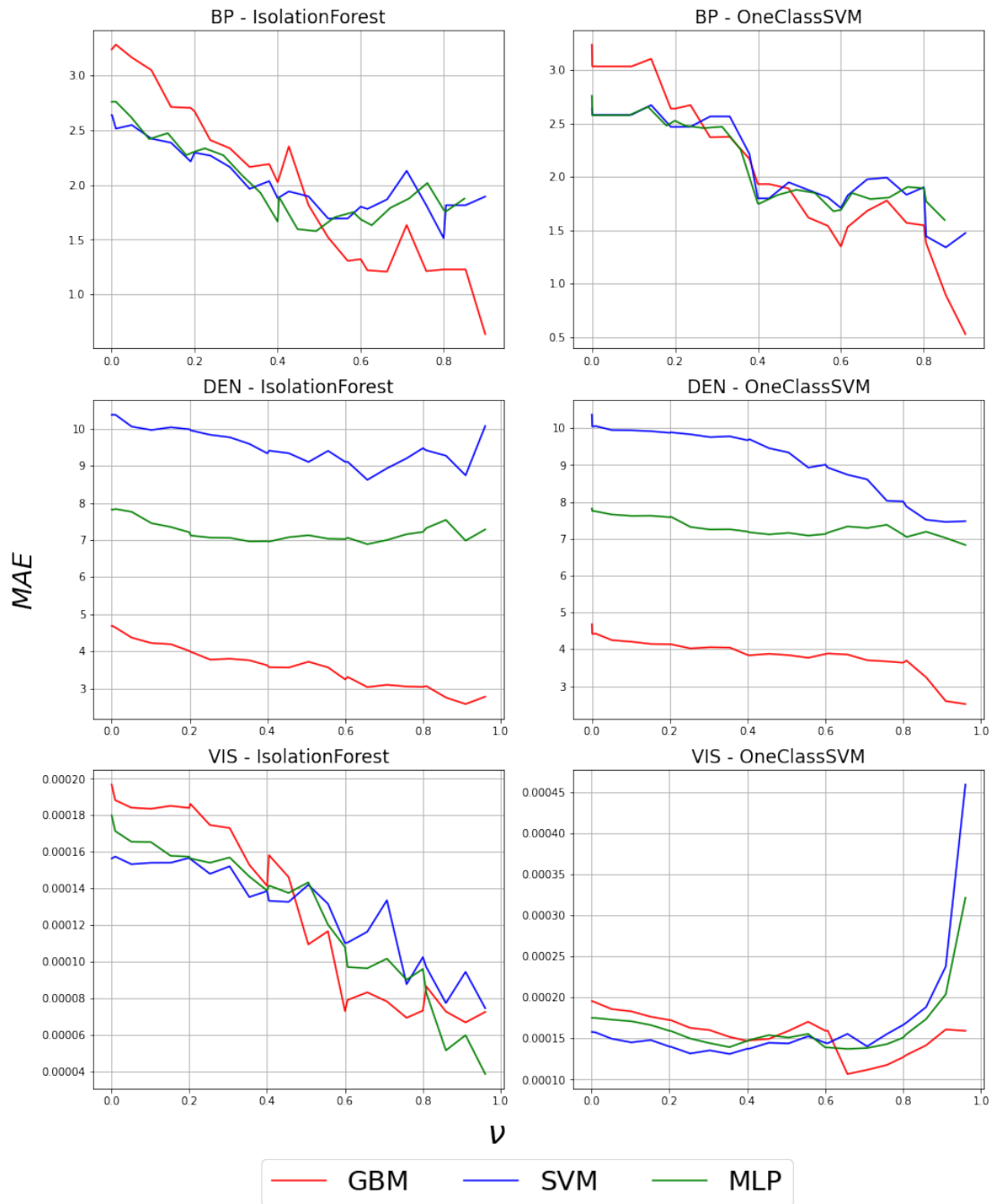


Fig. 5.7 Impact of contamination factor v on predictions mean absolute error for different fuel properties

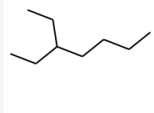
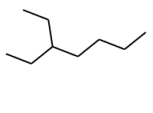
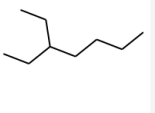
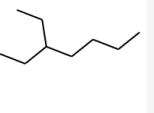
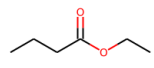
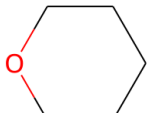
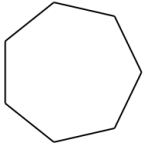
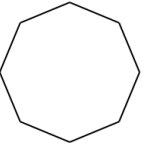
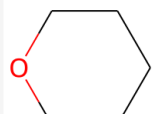
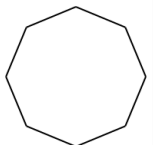
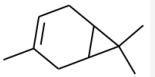
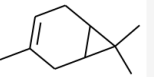
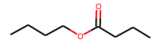
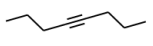
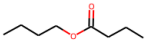
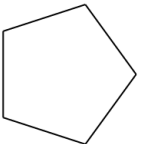
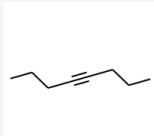
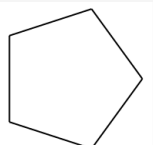
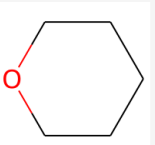
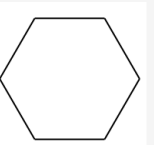
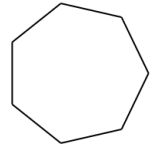
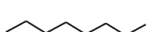


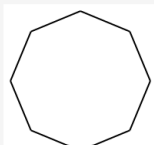
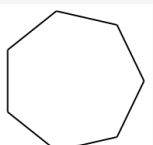
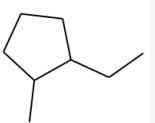
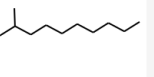
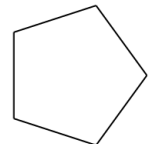
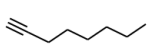
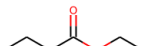
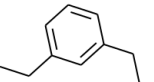
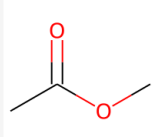
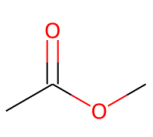
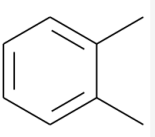
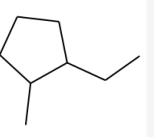
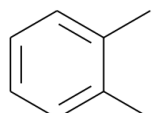
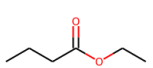
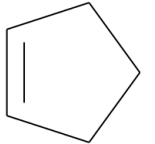
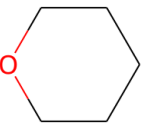
| GBM Molecule | [error] | SVM Molecule | [error] | MLP Molecule | [error] | RF Molecule | [error] |
|---|-----------|---|-----------|---|-----------|---|-----------|
|  | 55.667018 |  | 54.101226 |  | 60.958941 |  | 66.987094 |
|  | 25.691960 |  | 29.312501 |  | 29.189577 |  | 47.811559 |
|  | 24.529128 |  | 26.138059 |  | 25.121684 |  | 31.332078 |
|  | 24.359658 |  | 24.882834 |  | 21.036914 |  | 30.657856 |
|  | 23.812544 |  | 22.988832 |  | 20.954062 |  | 26.528723 |
|  | 21.653772 |  | 22.952966 |  | 19.965792 |  | 22.290318 |
|  | 19.920800 |  | 21.735122 |  | 18.795924 |  | 21.089858 |
|  | 18.267751 |  | 20.116500 |  | 16.563306 |  | 18.150872 |
|  | 16.005470 |  | 20.035250 |  | 14.953731 |  | 15.913208 |
|  | 14.365123 |  | 19.379861 |  | 14.732587 |  | 15.554403 |

Fig. 5.8 Octane models worst predictions

It was found that the anomaly detection algorithms could establish an applicability domain by sorting test compounds by their structural similarity to the training set. Thereafter sorting compounds in such a way showed a correlation to the test compound's prediction errors (Figure 5.6) and (Figure 5.7). For example the errors in admonition temperature decreased from 25 to 15 in Figure 5.6, given $v = 0.6$.

However, the distance metrics such as Tanimoto and Leverage could not sort compounds by their similarity to the train set in such a way that correlates with the prediction errors. Figure 5.2 shows poor correlation between mean Tanimoto distance and predictions errors.

The leverage distance, which measures the distance in descriptor space between the test compound and the mean of descriptors values for the training set was unstable. This was attributed to matrix inversion in (Equation 3.5). For the matrix inversion to be computed, the computed matrix must have full rank, i.e none of its columns can be represented as a linear combination of other columns. However since the descriptors can have such a relationship, the inversion was not possible for some descriptors. Therefore leverage distance was omitted since it can't be used to compare results in the applicability domain given a chosen descriptors definition.

Secondly, Tanimoto's average distance did not show a correlation between its values and octane QSAR model error (Fig 5.2). However, compared to Leverage distance, the Tanimoto measure was descriptors agnostic, since its calculation is based on finding the ratio between compounds' common number of fragments and the total for two compounds (equation 3.4) and therefore it could be computed no matter what descriptors were used for modelling fuel properties.

In addition compounds with the worst prediction tend to have the largest Tanimoto distance. The weak relationship between the Tanimoto distance and models predictions errors can be attributed to two factors. The first one is that the fuel properties were modelled with descriptors, not by the molecular fragments and so even if Tanimoto distance has chemical intuition in terms of similarity between compounds it can be misleading in terms of descriptors. Secondly, fuel properties datasets most likely had a clustering structure, with some compounds grouped together in descriptor space.

In conclusion anomaly detection algorithms could establish applicability domain for the chosen fuel properties. For example OneClassSVM algorithm in Figure 5.6 could consistently reduce model predictions errors up to $v = 0.5$. Research octane number predictions errors for support vector machine coloured in red reduced from 5 to 2, where $v = 0.5$.

Chapter 6

Generative modelling of high octane fuel molecules

6.1 Introduction

This chapter demonstrates results for generative modelling application in prototype fuel development. The goal of this chapter is to show how generative modelling can reveal fuel like molecular structures, which would not be found by intuition. The main findings include list of hydrocarbons and oxygenated compounds, which were generated to maximise the octane number. The structures found were in agreement with the domain knowledge of what knock resistant compounds might look like.

To make chapter results reproducible the experiment set up sub-chapter includes the generative models used architecture set up and hyper-parameter grid used to maximise the generative models performance. Valid organic molecules test demonstrates the generative models used ability to make valid organic chemical structures, which was the first models test, since bio-fuels come from organic sources. The test also included models ability to generate valid organic compounds with 2 properties, which can be found analytically. This was done in order to check the generative models ability to make property targeted molecular structure generation.

The last two subs-chapters show the final main results, which was the generation of fuel like molecules. Generating hydrocarbons with high octane sub-chapter shows molecules optimized for the octane number. While the last chapter uses LSTM models memory property and creates molecules starting from oxygen in order to increase oxygenated content in hydrocarbons structure generation to resemble bio fuels.

6.2 Experiment set up

Auto-regressive model

Auto-regressive model architecture included sequentially embedding layer, LSTM cell [70] and linear layer. The embedding layer weight size was equal to the input dimension times the embedding dimension, which was a variable parameter listed as *embed dim* in Table 6.1. The input dimension was equal to 27. The embedding layer was learning vector representation for each unique character used in the smiles string. LSTM cell had variable parameters the number of layers *n layers* and hidden dimension H_{hidden} (Table 6.1), these influenced the final hidden h_t and cell state vectors c_t shapes, which were equal to $(n\ layers, H_{hidden})$ and $(n\ layers, H_{hidden})$. Drop out was *drop out* (Table 6.1) applied to the embedding layer, LSTM cell and the linear layer, this parameter acted as regularization. The linear layer transformed the LSTM cell output dimension back to 27 size, since soft-max was used to compute probabilities of the next character.

Apart from architectural and regularisation parameters, learning rate *lr* (Table 6.1) controlled the optimization process of Adam optimizer [87]. Figure 6.1 shows the model architecture printed by Torchviz [31] python package. Table 6.1 includes column selected value, which shows parameters used to produce results in the following sub-chapters.

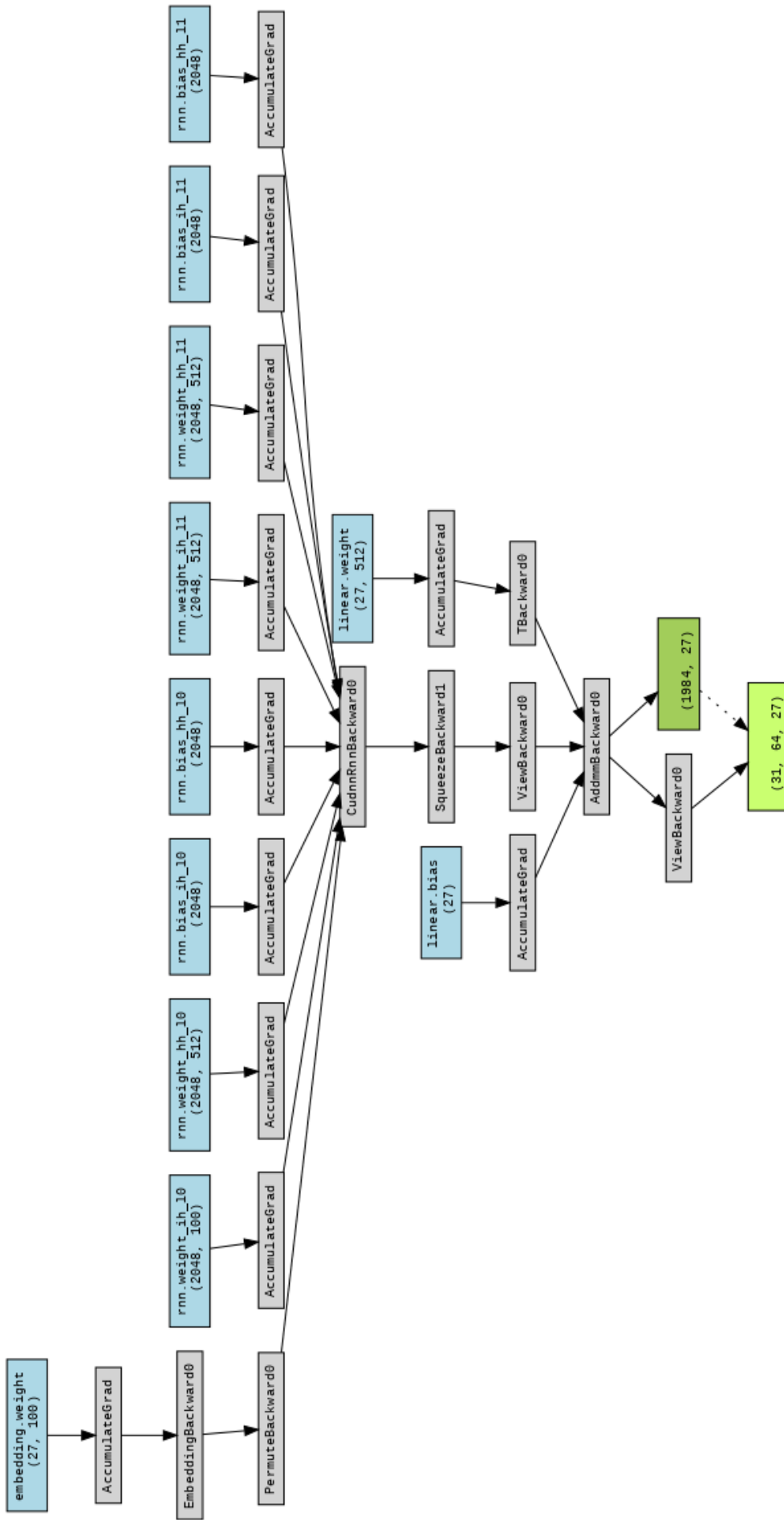


Fig. 6.1 Autoregressive model diagram

| Parameter | Search range | Selected value |
|-------------------|-----------------------|----------------|
| <i>embed dim</i> | 50 – 200 2 | 100 |
| <i>batch size</i> | 16 – 128 | 64 |
| <i>n layers</i> | 1 – 3 2 | 2 |
| <i>drop out</i> | 0.1 – 0.8 | 0.3 |
| <i>lr</i> | 10^{-5} – 10^{-1} | 10^{-3} |
| H_{hidden} | 32 – 512 | 512 |

Table 6.1 Auto-regressive model parameter search

GAN model

The GAN model included discriminator and generative networks. Discriminator receives graph as input and outputs scalar. Generator receives vector of variable size z and outputs adjacency tensor and node feature tensors with dimensions $(n_{moleclues}, 9, 9, 5)$ and $(n_{moleclues}, 9, 5)$ representing molecular graph. In adjacency tensor dimensions 9 and 5 represent the maximum number of atoms excluding hydrogen and the type of connection between them including, single, double, triple, aromatic bonds and no bond. The node feature tensor dimensions of 9 and 5 represent the number of atoms excluding hydrogen in one molecule and the atom types. These types included oxygen, carbon, nitrogen, fluorine, which were present in QM9 dataset and empty atom.

Discriminator consisted of two relational graph convolution layers [143], which propagated signal between graph nodes, followed by aggregation layer [96], which transformed graph input into vector form, and lastly series of dense layers. Figure 6.3 shows discriminator architecture used to make results for the rest of this chapter. Generator consisted of series of 3 consecutive linear layers, where the output of the third layer was projected two separate layers, which were edges and nodes in Figure 6.2. These were then reshaped into tensors from 405 to $(9, 9, 5)$ for edges and from 45 for node feature layer into $(9, 5)$. Later gamble-softmax activation [78] was applied along last dimensions in edge and node feature tensors in order to get discrete values for the type of bond and type of atoms respectively.

6.3 Valid organic molecules test

This sub-chapter shows results in model validation for GAN and auto-regressive models. Figure 6.4 shows the ratio of valid molecules made during each epoch step. The models were trained by stochastic gradient descent in mini-batches, consisting of 64 molecules at each step. Epoch denotes the number of batches when learning iterations cover the full training set. In this case, it was 120000 molecules from the QM9 dataset [130] small organic molecules.

1 G

```
Generator(  
  (lin1): Linear(in_features=32, out_features=128, bias=True)  
  (lin2): Linear(in_features=128, out_features=256, bias=True)  
  (lin3): Linear(in_features=256, out_features=512, bias=True)  
  (edges): Linear(in_features=512, out_features=405, bias=True)  
  (nodes): Linear(in_features=512, out_features=45, bias=True)  
  (drop_out): Dropout(p=0.1, inplace=False)  
  (act_): LeakyReLU(negative_slope=0.01)  
)
```

Fig. 6.2 Generator architecture

Figure 6.4 shows that the auto-regressive model learned to make valid molecules quickly, while the GAN model showed high oscillation in valid molecules produced and then diminished quality of molecules made. Since the auto-regressive model showed superior results it was chosen for the latter test to make molecules with target properties.

One of the major reasons why the GAN model did worse than the auto-regressive model is the difference in the optimization tasks for these models' equations 3.6 and 3.8. The solution to 3.6 is a saddle point, while the minimum is 3.8. Optimizers currently made for deep learning packages are fundamentally designed to find the minimum point, not the saddle. Figure 6.5 shows three cases of vector fields, the first one is with divergence and curls present, the second with no divergence, and the last without curl. Deep learning training can be represented by traversing in these fields. In the case of a single minimum, it's a traverse in a curl-free field, where a minimum can be reached with an appropriate step size. However in the case of GAN curl is present, which makes traverse with rotation present and it makes it hard for first-order optimizers to converge to the saddle point.

Figure 6.6 illustrates how sampling parameter T from equation 3.9 affects the number of novel molecules made. It was found that slight distortion of the original distribution from $T = 1$ to $T = 1.4$ made the maximum number of novel molecules. This sampling temperature will later be used for sampling in the hill climb algorithm (Figure 3.8) in order to modify molecules with the desired properties. In addition, as expected as the temperature parameter increased and the learned smiles symbols distribution was more distorted towards uniform distribution, the number of valid molecules diminished.

To test the ability of the generative model 2 tests were made including optimizing molecular structure to increase the octanol-water partition coefficient, known as $\log P$, and

1 D

```
R(  
  (drop_out): Dropout(p=0.1, inplace=False)  
  (conv1): Convolve(  
    (root): Linear(in_features=5, out_features=128, bias=True)  
    (single_): Linear(in_features=5, out_features=128, bias=False)  
    (double_): Linear(in_features=5, out_features=128, bias=False)  
    (triple_): Linear(in_features=5, out_features=128, bias=False)  
    (aromat_): Linear(in_features=5, out_features=128, bias=False)  
  )  
  (conv2): Convolve(  
    (root): Linear(in_features=133, out_features=64, bias=True)  
    (single_): Linear(in_features=133, out_features=64, bias=False)  
    (double_): Linear(in_features=133, out_features=64, bias=False)  
    (triple_): Linear(in_features=133, out_features=64, bias=False)  
    (aromat_): Linear(in_features=133, out_features=64, bias=False)  
  )  
  (agr): Aggregate(  
    (agg): GlobalAttention(gate_nn=gate_nn(  
      (lin1): Linear(in_features=192, out_features=1, bias=True)  
      (drop_out): Dropout(p=0.1, inplace=False)  
    ), nn=nn_  
      (lin2): Linear(in_features=192, out_features=128, bias=True)  
      (drop_out): Dropout(p=0.1, inplace=False)  
    ))  
  )  
  (linear): Sequential(  
    (0): Linear(in_features=128, out_features=128, bias=True)  
    (1): Dropout(p=0.1, inplace=False)  
    (2): LeakyReLU(negative_slope=0.01)  
    (3): Linear(in_features=128, out_features=64, bias=True)  
    (4): Dropout(p=0.1, inplace=False)  
    (5): LeakyReLU(negative_slope=0.01)  
    (6): Linear(in_features=64, out_features=1, bias=True)  
  )  
  (act_): LeakyReLU(negative_slope=0.01)  
)
```

Fig. 6.3 Discriminator architecture

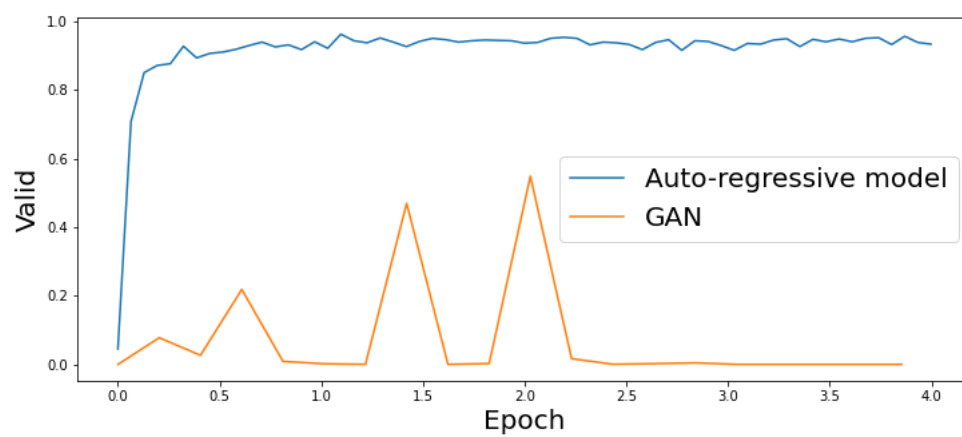


Fig. 6.4 Auto-regressive model vs GAN training

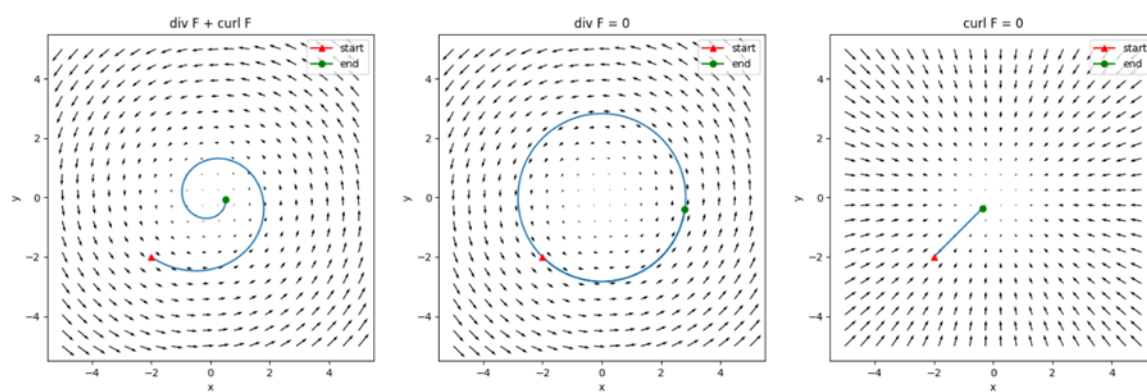


Fig. 6.5 Vector field comparison

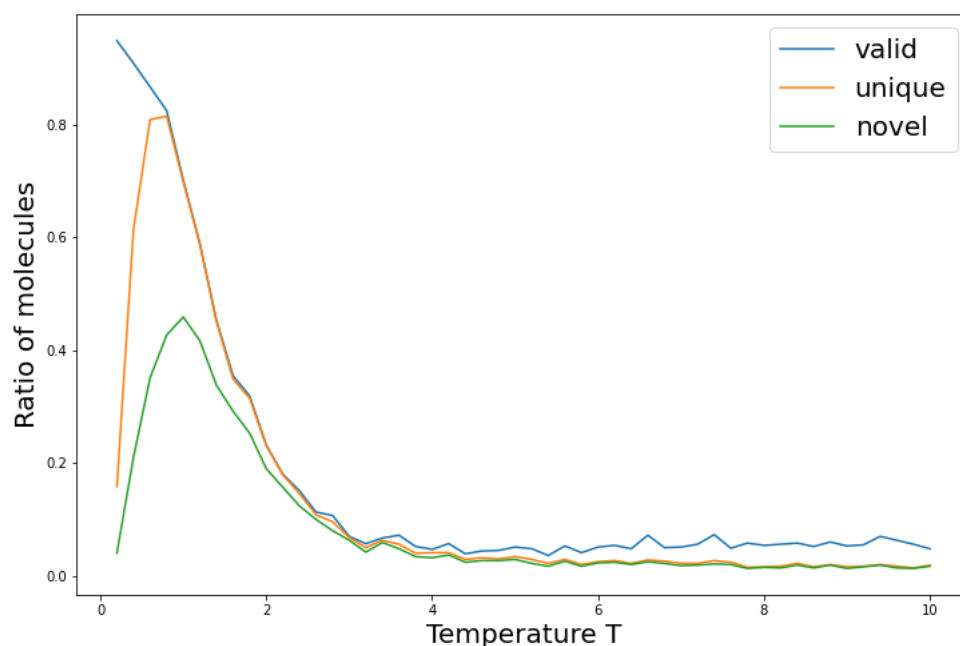


Fig. 6.6 Valid, Unique and Novel molecules vs sampling temperature T

altering the chemical structure to include the ketone group. Figure 6.6 shows the initial $\log P$ distribution for valid molecules produced and then for each iteration of the climb hill algorithm step N (Figure 6.7). It is evident that after 10 steps the median value for $\log P$ values of molecules made converges, which is due to the fact that similar molecules are made at each iterative step. To test the ability of the model to force chemical structure constraints on molecules produced, the hill climb algorithm was given a reward function that gives 1 if the ketone group is present and 0 if not. Figure 6.8 shows the initial ratio of molecules with the ketone group at steps 0 to step 10.

To conclude LSTM language model was the preferred choice, based on the proportion of valid compounds generated (Figure 6.4). The number of novel compounds reached 45% by getting the sampling temperature around 1.5 (Figure 6.6). Thereafter the generative model was fine-tuned for 2 tasks. One is to optimize the $\log P$ value and the proportion of compounds with the ketone group. These tasks were given to check the generative model capabilities since $\log P$ and the ketone group ratio can be validated analytically, unlike the octane number, which can only be determined by conducting an experiment. In conclusion, this verified technique of generating novel compounds with the property target set will be used to generate prototype fuel structures, which can not be found by intuition.

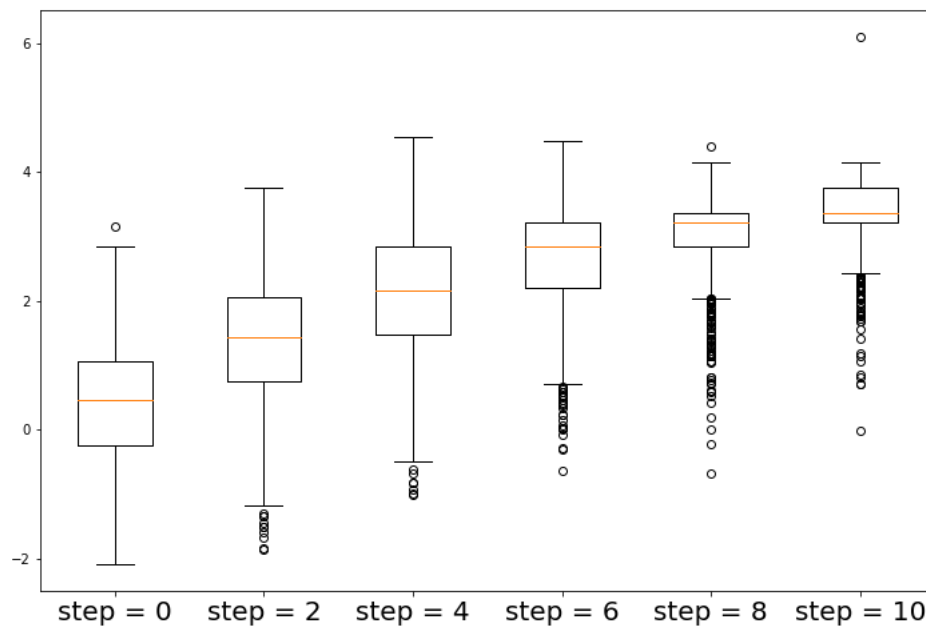


Fig. 6.7 Generated molecules $\log P$ values at each climb hill step

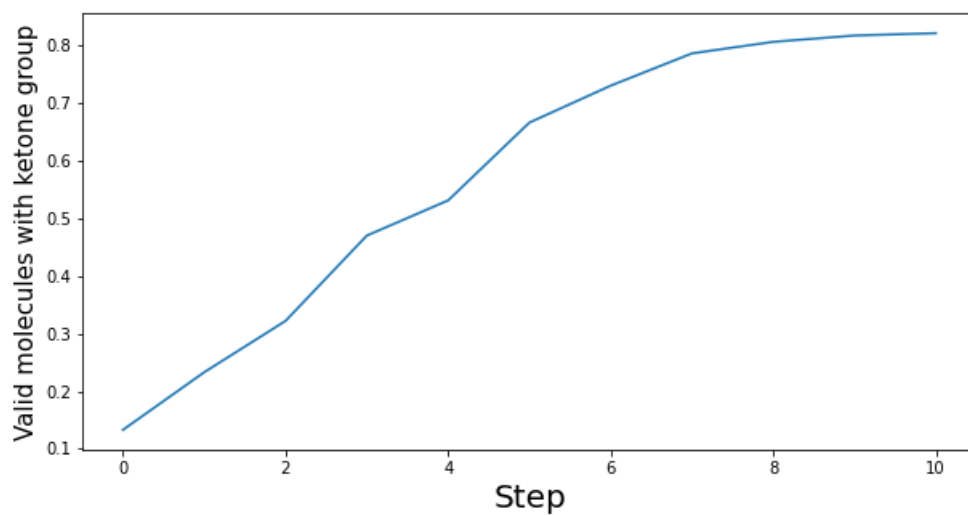


Fig. 6.8 Ratio of generated compounds with ketone group at each climb hill step

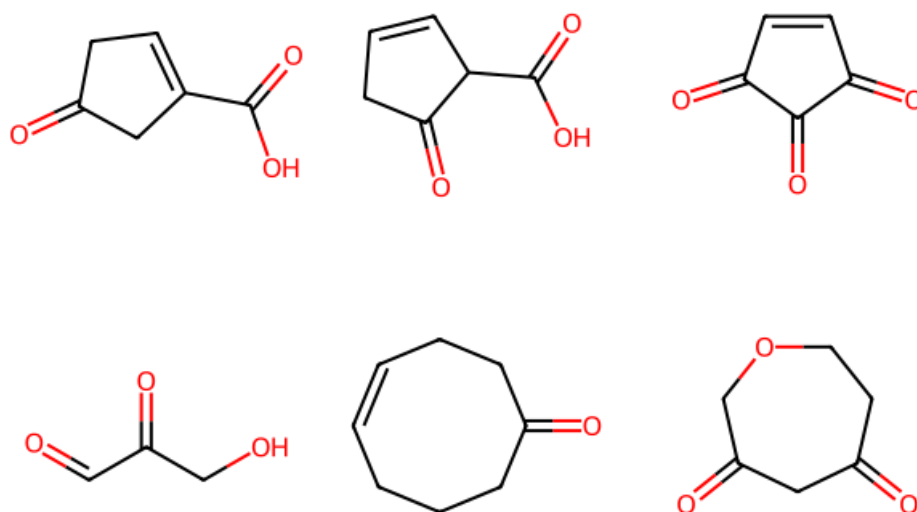


Fig. 6.9 Sample of generated molecules with ketone group

6.4 Generating hydrocarbons with high octane

In this chapter, molecules were optimized during generation for increasing the octane number. The same procedure of fine-tuning the LSTM model was taken with $\log P$ values and ketone group presence as in chapter 6. The only difference with chapter 6 molecular generation was the reward function, which returned the octane number predicted by the SVM model developed in chapter 5 (Table 5.3) multiplied by sign, which is only negative for compounds that contain nitrogen. This insured that molecules containing nitrogen will not be saved as top-performing and hence be used to fine-tune the model (Figure 6.10). Such a procedure introduces bias against generating molecular structures with nitrogen. Molecules with nitrogen were not considered as future fuels, since their combustion produces toxic nitrogen oxide emissions. Figure 6.10 shows the distribution of the reward function values of the generated molecules at each hill climb step. The hill climb algorithm was stopped when the median reward value converged to 100 at step 10 (Figure 6.10). Thereafter this model will be used to make compounds from scratch and start with the oxygen symbol. Important to note that the hill climb algorithm did not eliminate completely molecules with nitrogen present, which can be found as negative outliers in Figure 6.10

The model generated 500 compounds, which were later sorted by the octane number predicted by the same model used in the reward function. The best 30 molecules are shown in Figure 6.11. Table 6.2 lists 12 molecules out of these 30 which were not present in the QM9

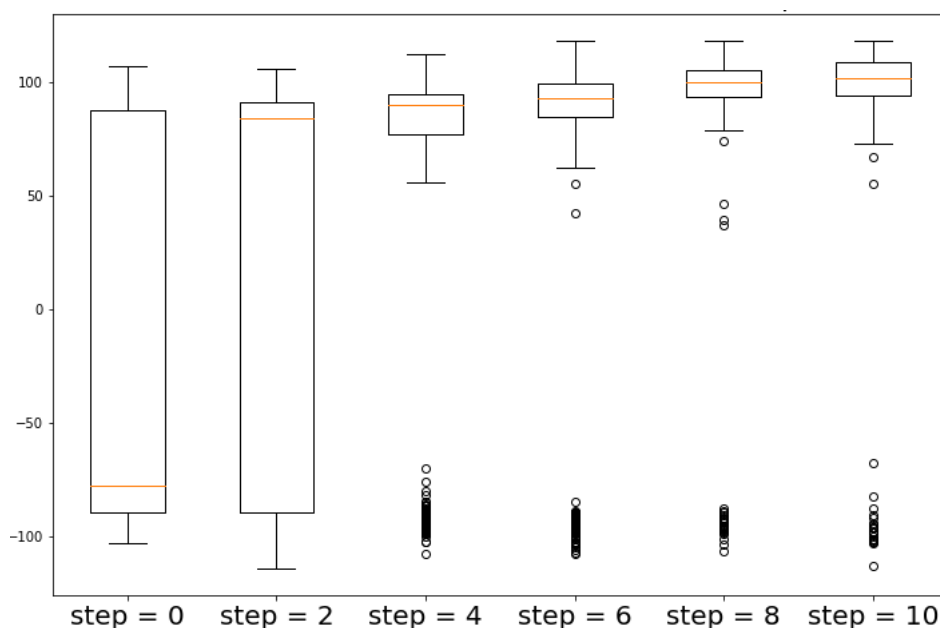


Fig. 6.10 Octane number distributions at different hill climb steps

dataset [130] of small organic compounds used to train the generative model. This selection was undertaken, so the new compounds can not be found by simply calculating the octane number for QM9 molecules. Pub-Chem identification number [18] was used to compare if molecules are identical since the same molecule can have different smile notations. These identification numbers were computed by open source package PubChemPy [Matt Swain].

Several molecules in Table 6.2 produced by the generative model and were already present in the octane database, such as 2,2,3-trimethylpentane. Mainly these duplicates were represented by alternative forms of smile's strings, for example an aromatic ring can written either c1ccccc1 as an aromatic ring or as an unsaturated ring C1==CC==C1. Subsequently, 5 molecules were left, which are not included in the QM9 and octane databases. The procedure undertaken used pubchem ids. Figure 6.12 shows the structure of these molecules left. Table 6.3 shows their predicted physical properties, made by models from chapter 5.

From Figure 6.12, it can be seen that in general two generated molecules are compact and incorporate branching. For example 2,2,3,4-tetramethyl, similar in structure to 2,2,3,3-tetramethyl pentane, which has a measured octane value of 114. It can also be seen that 40% of the molecules incorporate an aromatic ring. Both branching and aromatic rings are known to increase knock resistance [150], [85]. The smallest molecule 3-methylbutan-2-ol had the alcohol group. Finally, these 5 molecules had physical properties shown in table 6.3, which were in the adequate range for fuels.

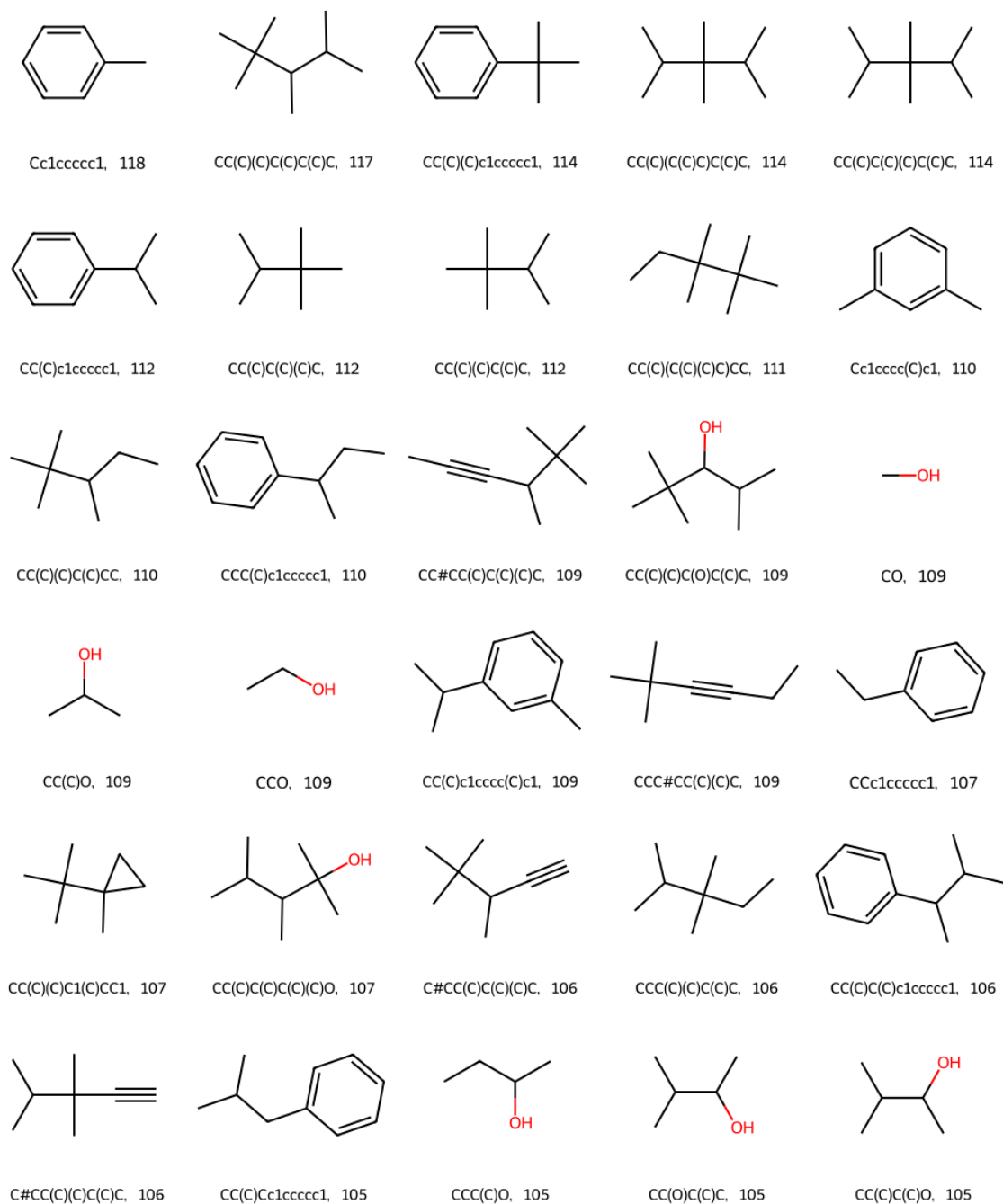


Fig. 6.11 Generated molecular structures with the highest octane values

| SMILES | Compound Name | Present_in_Octane_data |
|--------------------------------|---------------------------------|--|
| <chem>CC(O)C(C)C</chem> | 3-methylbutan-2-ol | False |
| <chem>CC(C)(C)C(C)CC</chem> | 2,2,3-trimethylpentane | <chem>CC(C)C(C)(C)C</chem> [112.1] |
| <chem>CC(C)Cc1ccccc1</chem> | (2-methylpropyl)benzene | <chem>CC(C)CC1=CC=CC=C1</chem> [101.6] |
| <chem>CC(C)(C)C(C)C(C)C</chem> | 2,2,3,4-tetramethylpentane | <chem>CC(C)C(C)C(C)(C)C</chem> [116.8] |
| <chem>CC(C)(C)c1ccccc1</chem> | tert-butylbenzene | <chem>CC(C)(C)C1=CC=CC=C1</chem> [103.0] |
| <chem>CCC(C)c1ccccc1</chem> | (butan-2-yl)benzene | <chem>CCC(C)C1=CC=CC=C1</chem> [100.7] |
| <chem>CC(C)c1ccc(C)c1</chem> | 1-methyl-3-(propan-2-yl)benzene | False |
| <chem>CC(C)(C)C(C)C</chem> | 2,2,3-trimethylbutane | <chem>CC(C)C(C)(C)C</chem> [112.1] |
| <chem>CC(C)(C)C(O)C(C)C</chem> | 2,2,4-trimethylpentan-3-ol | False |
| <chem>CC(C)C(C)c1ccccc1</chem> | (3-methylbutan-2-yl)benzene | False |
| <chem>CC(C)(C(C)C)C(C)C</chem> | 2,3,3,4-tetramethylpentane | False |
| <chem>CC(C)(C(C)(C)C)CC</chem> | 2,2,3,3-tetramethylpentane | <chem>CCC(C)(C)C(C)(C)C</chem> [99.1] |

Table 6.2 Generated compounds not present in QM9 data-set

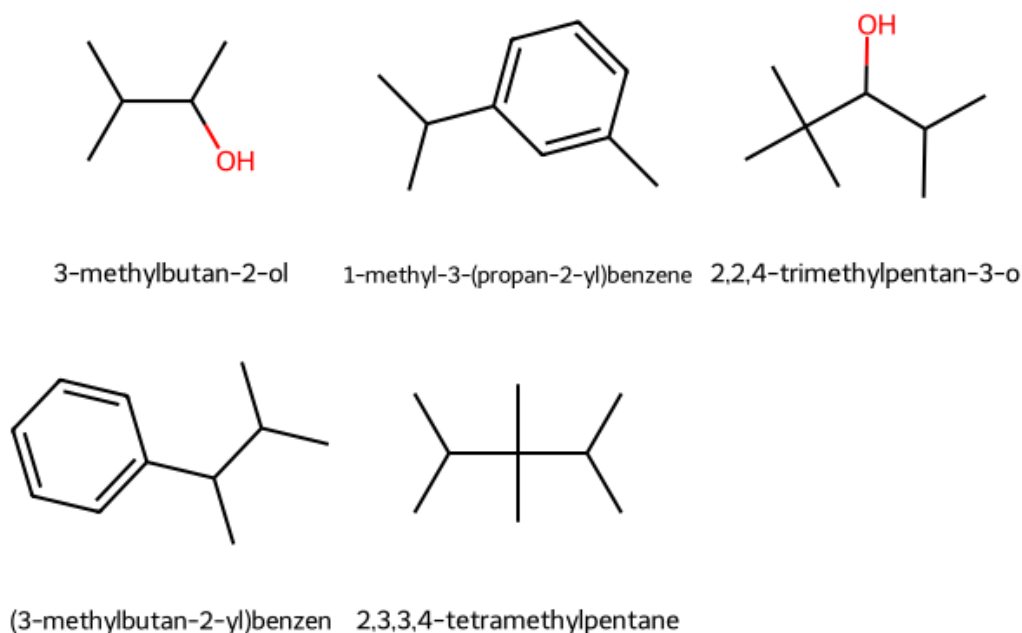


Fig. 6.12 Molecules not present in either QM9 and octane data-sets

| Compound name | RON | Density(mg cm ³) | Boiling point(K) | Viscosity(m Pa s) |
|---------------------------------|-----|------------------------------|------------------|-------------------|
| 3-methylbutan-2-ol | 105 | 806 | 369 | 0.86 |
| 1-methyl-3-(propan-2-yl)benzene | 108 | 846 | 452 | 0.68 |
| 2,2,4-trimethylpentan-3-o | 109 | 813 | 445 | 0.74 |
| (3-methylbutan-2-yl)benzen | 105 | 847 | 464 | 0.80 |
| 2,3,3,4-tetramethylpentane | 114 | 740 | 401 | 0.57 |
| gasoline | 95 | 743 | 368 | 0.70 |

Table 6.3 Physical properties of generated molecules not present in QM9 and octane datasets

6.5 Generating oxygenated molecules with high octane

The trained model was subsequently used to generate 3000 compounds where an oxygen atom was included in the starting token, thereby ensuring that the final structure also incorporates oxygen. The number of molecules generated increased from previous 500 in order to increase the variety of compounds since the additional constraint of oxygen was implemented. Figure 6.13 shows the generated molecules incorporating oxygen that were not present in either QM9 dataset or the database used to train the reward function. These generated RON numbers and physical properties including density and viscosity at 40 degrees and the boiling point at standard atmospheric pressure are shown in Table 6.4.

The first observation is that fewer aromatic species were generated (Figure 6.13) compared to molecules without a starting oxygen symbol (Figure 6.11). Started with oxygen, the majority of compounds contain more than one oxygen. It can also be seen that there the oxygenated compounds generated also featured a greater abundance of branching relative to the molecules generated without the oxygen starting token (Figure 6.11). It is also interesting to note that these molecules showed a greater abundance and variety of oxygen-bearing functional groups. For example ketones and ethers (Figure 6.13). The presence of the latter is interesting given that ether linkages are known to decrease octane number. *[findreference]* These molecules were hydroxymethyl acetate, 3-(hydroxymethyl)-4-methylpentan-2-one and 3-methylbutan-2-yloxymethanol. Some compounds like 4-methylhex-2-yn-1-ol and 4,5-dimethylhex-2-yn-1-o had triple bonds. These are known to increase the RON number, but also increase soot formation *[need ref here]*. One feature of the generated compounds 1-butan-2-ylcyclopropan-1-ol included triangular structure, which is not common in practical fuels.

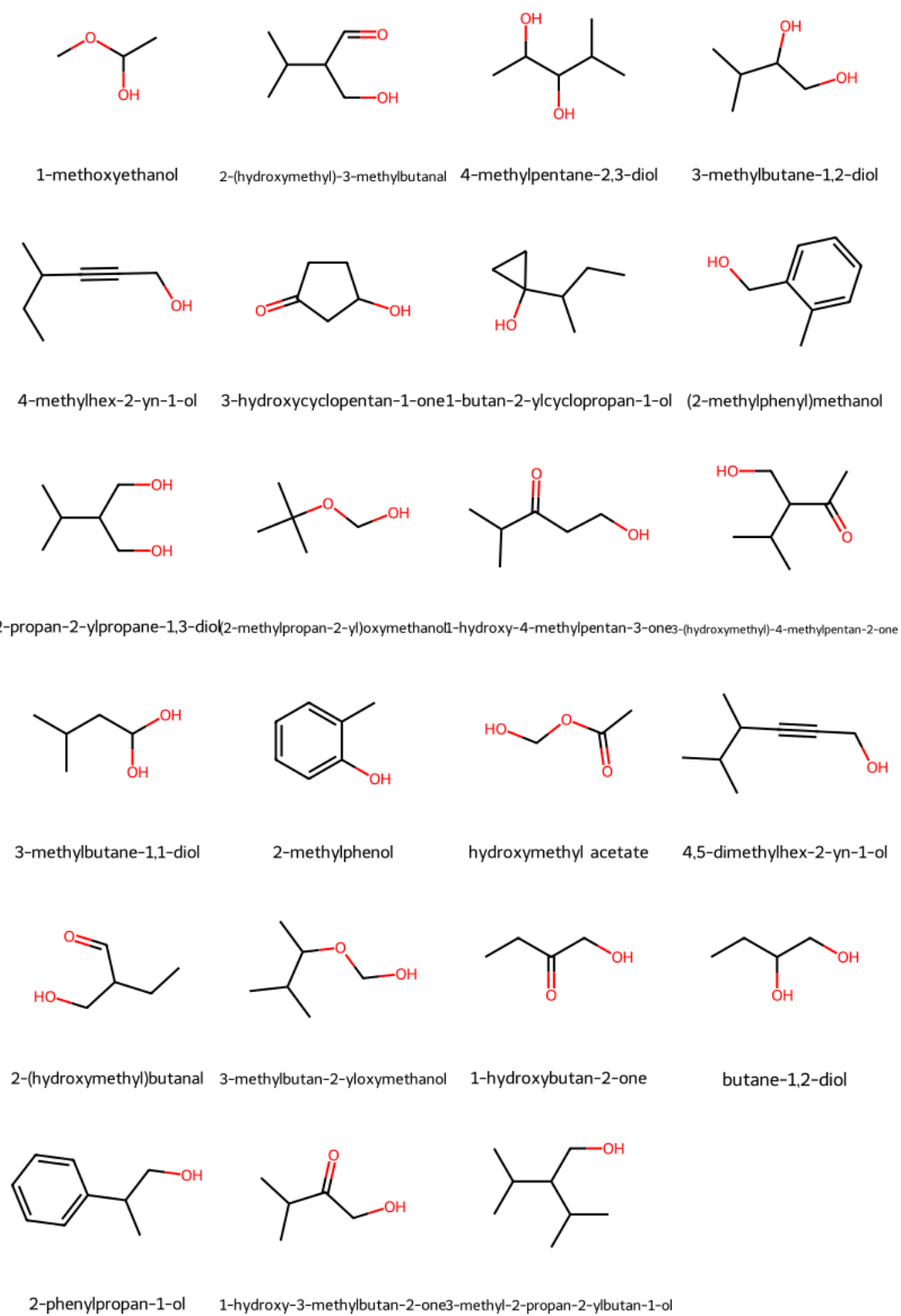


Fig. 6.13 Generated structures starting with oxygen symbol, which were not present in QM9 and octane datasets

| Compound name | SMILES | RON | DEN | BP | VIS |
|--|-------------------------------|-----|------|-----|------|
| 1-methoxyethanol | <chem>O(C)C(C)O</chem> | 102 | 952 | 358 | 0.99 |
| 2-(hydroxymethyl)-3-methylbutanal | <chem>OCC(C=O)C(C)C</chem> | 100 | 940 | 447 | 1.18 |
| 4-methylpentane-2,3-diol | <chem>OC(C(C)O)C(C)C</chem> | 100 | 950 | 455 | 0.98 |
| 3-methylbutane-1,2-diol | <chem>OC(CO)C(C)C</chem> | 99 | 962 | 432 | 0.97 |
| 4-methylhex-2-yn-1-ol | <chem>OCC#CC(C)CC</chem> | 98 | 829 | 450 | 0.68 |
| 3-hydroxycyclopentan-1-one | <chem>OC1CCC(=O)C1</chem> | 97 | 1085 | 416 | 0.96 |
| 1-butan-2-ylcyclopropan-1-ol | <chem>OC1(C(C)CC)CC1</chem> | 98 | 925 | 425 | 1.03 |
| (2-methylphenyl)methanol | <chem>OCc1ccccc1C</chem> | 97 | 1020 | 492 | 1.04 |
| 2-propan-2-ylpropane-1,3-diol | <chem>OCC(CO)C(C)C</chem> | 97 | 958 | 481 | 1.06 |
| (2-methylpropan-2-yl)oxymethanol | <chem>OCOC(C)(C)C</chem> | 99 | 901 | 411 | 0.90 |
| 1-hydroxy-4-methylpentan-3-one | <chem>OCCC(=O)C(C)C</chem> | 101 | 924 | 448 | 0.83 |
| 3-(hydroxymethyl)-4-methylpentan-2-one | <chem>OCC(C(=O)C)C(C)C</chem> | 97 | 914 | 466 | 0.87 |
| 3-methylbutane-1,1-diol | <chem>OC(O)CC(C)C</chem> | 100 | 948 | 451 | 1.15 |
| 2-methylphenol | <chem>Oc1ccccc1C</chem> | 103 | 1031 | 460 | 1.07 |
| hydroxymethyl acetate | <chem>OCOC(=O)C</chem> | 104 | 1125 | 464 | 1.47 |
| 4,5-dimethylhex-2-yn-1-ol | <chem>OCC#CC(C)C(C)C</chem> | 98 | 813 | 465 | 0.66 |
| 2-(hydroxymethyl)butanal | <chem>OCC(C=O)CC</chem> | 100 | 1025 | 426 | 1.28 |
| 3-methylbutan-2-ylloxymethanol | <chem>OCOC(C)C(C)C</chem> | 98 | 889 | 435 | 0.86 |
| 1-hydroxybutan-2-one | <chem>OCC(=O)CC</chem> | 105 | 1042 | 380 | 0.80 |
| butane-1,2-diol | <chem>OCC(O)CC</chem> | 97 | 991 | 404 | 1.04 |
| 2-phenylpropan-1-ol | <chem>OCC(C)c1ccccc1</chem> | 97 | 989 | 505 | 1.17 |
| 1-hydroxy-3-methylbutan-2-one | <chem>OCC(=O)C(C)C</chem> | 104 | 960 | 404 | 0.72 |
| 3-methyl-2-propan-2-ylbutan-1-ol | <chem>OCC(C(C)C)C(C)C</chem> | 101 | 811 | 455 | 0.84 |

Table 6.4 Generated compounds not present in QM9 and octane datasets

6.6 Conclusion

In attempt to generate molecules 2 methods were tried including GAN model and auto-regressive model, and the auto-regressive model performed all the tasks, in which were valid organic compounds test and generation of fuel like molecules.

Figure 6.4 showed that auto-regressive model performed better in valid organic molecules test, it reached 95% in the proportion of generated valid compounds during training, while the GAN model reached maximum of 50%. Consequently GAN model was not later used to create molecules with the desired properties.

Thereafter hill climb algorithm (Figure 3.9) was applied to tune the trained auto-regressive model to bias molecular generation towards chosen theoretical targets such as an increase in *logp* values and the ratio of compounds with ketone group. These tests could show models' ability to bias molecular generation and check it analytically, unlike the octane number, which can only be determined by experiment.

Median generated molecules *logp* values before applying the hill climb algorithm were around 0.5 at step 0 in Figure 6.7 increasing to 3.5 at step 10 in Figure 6.7. The ratio of molecules with the ketone group before and after the hill climb algorithm application increased from 0.15 to 0.8, step 0 and step 10 in Figure 6.8.

After testing the generative model's ability to bias molecular generation on theoretical targets, the model was fine tuned to increase the octane number. Figure 6.10 showed the octane number values of the generated compounds before fine tuning and after. Median octane number of generated compounds converged to 100 (Figure 6.10). In Figure 6.10 some compounds had negative octane rating, because predictions of the octane number for the compounds outside of THE applicability domain were assigned negative score in order to encourage model not to generate compounds outside the applicability domain of the regression octane number model.

Thereafter generated compounds with the high octane ratings were assessed qualitatively by their chemical structure properties. The model was able to reproduce already-known high-knock-resistant compounds (Table 6.2), which were not present in the QM9 data set used for the generative model training. Although such a result indicates that the model can produce knock-resistant compounds, it has limited practical use, as the model's goal is to propose novel high knock-resistant compounds. Therefore compounds in Figure 6.12, which were not used in any form of model development were further investigated. These compounds showed known chemical structure attributes of high knock resistance compounds, such as branching, cyclic rings, and compactness. Practical experiments and compound synthesis would be required to validate the octane rating of these molecules. Later the generative model

was forced to produce molecular structures starting with oxygen in smiles strings describing fuels chemical structure (Table 6.4). Again these compounds showed the traits of high knock resistant structures, such as compactness, presence of double bonds and branching.

Chapter 7

Conclusions and Future work

This chapter provides an overall summary of the thesis, by giving conclusion to the literature review, research results and future work. Literature review sub-chapter gives the main literature findings, which defined the main research directions. The results sub-chapter summarises the key findings from the modelling investigations undertaken. The future work provides research directions based on the obtained results and the limitations in methodology.

7.0.1 Literature review

Three major machine learning application gaps were found in the context of fuel research. These were interpretations of the models used to predict fuel properties, consideration of the applicability domain of the fuels properties models, and the application of inverse QSAR generative models.

The interpretation of predictions by a model enables further insights into the underlying relationship between the chemical structure and fuel properties. In addition model interpretations allow checks of the model predictions by comparing them with known chemical mechanisms.

In drug design, molecular property prediction models (QSAR) must have an applicability domain, which defines what compounds can be used as the model input. This stems from the fact that machine learning models can not confidently extrapolate, and so the data used to train a model defines what the model can predict or not. The cost of false prediction can be high in drug development. This approach was taken in the context of fuels in order to find boundaries of what can be predicted or not, based on the available fuel properties data.

Lastly advances in chemical informatics and machine learning demonstrated targeted molecular structure generation known as inverse QSAR or de-novo molecular design. Since combustion is a very complex process, future fuels structures can potentially be suggested

by inverse QSAR, i.e helping to overcome limitations in current combustion knowledge and intuition.

7.0.2 Results

This final chapter provides a summary of the thesis results and their significance. These include results from chapters 4,5 and 6 which were interpretations of the octane number models predictions, applicability domain in fuels properties regression models and generation of fuel like chemical structures.

To demonstrate interpretations behind the fuel property predictions model, 5 classification models were built, which predicted if a given compound had a research octane number above or below 95. This value was chosen as a reference for standard gasoline value. The models showed an average accuracy of 85%, which is comparable to the developed octane number classification model by [160]. Thereafter 6 model interpretations were compared to known relationships of fuel structural changes to the research octane number. Five out of six cases demonstrated agreement by four models where the structure changes lead to an increase or decrease of the octane number values. Since 4 out of 5 machine learning algorithms were fundamentally different, it demonstrated versatility in the chosen method behind octane number classification model interpretations.

The fuel properties applicability domain was investigated by conducting a cross-product experiment between 4 different machine learning algorithms, 6 fuel properties and 4 methods to compute the similarity between each compound in the test set and all other compounds in the train set. The goal was to find which similarity measure would correlate with the regression model's prediction errors. Methods included 2 distance-based and movement of classification boundaries by a chosen parameter.

The trained regression models showed satisfactory level of accuracy between 1% and 10% in mean average absolute error, for data-sets with more than 1000 samples and less. Distance based similarity measures did not correlate with the predictions errors and hence could not establish the fuels applicability domain. Such a result can be explained by clustered nature of fuel data-sets. Intuitively clustering can be result of the different types of chemicals, for example straight chain alkanes and compounds with aromatic ring.

The outlier detection algorithms such as OneClassSVM and IsolationForest established the relationship between predictions errors and compounds location inside and outside the classification boundary. All trained machine learning algorithm on all data-sets showed almost a 2 fold decrease in predictions errors, by choosing compound with the moving classification boundary to $\nu = 0.6$. However further increase in the contamination factor did not show predictions error reduction. This can be attributed to the fact that the classification

boundary at such a point can ignore another cluster of train data to which the tested compound is closer.

Two generative models were investigated to perform inverse QSAR for fuel like structure generation. The GAN model produced 50% valid organic compounds, while LSTM produced 95%. The later was than chosen to be fine tuned by a hill climb algorithm with the octane number predictions as the reward function in order to get structures with high knock resistance. The first result demonstrated the generation of already known highly knock resistant compounds, which were not used as input to generative model training. The other compounds generated, which were not included in any part of modelling including training the reward octane function, showed traits of high knock resistance such as branching, cyclic structures, and the compact size of the molecules.

Overall these three major results expand current applications of machine learning and chemical informatics applications to fuel research beyond the current race in the accuracy of predictions, where these accuracy results are compared on different sets of molecules in the test set. The outcome of these results is to gain better knowledge of combustion by utilizing model interpretations. Applicability domain results show that the boundary of what needs to be tested in the experiment and what can be predicted based on the available data can be established. Lastly, the generative modeling approaches can help direct compounds to be tested in experiments since not all combinations of fuel structures can be tested due to high combinations of fuel potential structures and limited knowledge in analytical methods that can define if a given structure is suitable as a fuel candidate.

7.0.3 Future work

This sub-chapter describes what future research steps can be taken based on the results of this research. These include octane number modeling, interpretations of fuel quantitative structure activities models, and inverse quantitative structure modeling by application of the generative models.

Fuel properties modeling showed a decrease in the mean absolute errors of predictions with fuel properties data-set size increase. One of the possible ways to improve the octane number modeling is to create domain-specific variables for the fuel molecules used as input to the training models. Since the research octane number data set is relatively small, algorithms might not find interim variables, which can capture relationship between the variables and octane number. An example of such variables could be some specific combustion attributes of the fuel molecules. For example the extent of branching of alkyl chains, presence of alcohol functional groups and some other oxygenated groups could be added as model inputs.

Interpretations of the octane number predictions used only one approach which was model and variable agnostic. However other interpretations of QSAR models could be applied. These include algorithm and variable specific interpretations. The goal of applying other approaches is to increase confidence in interpretations, where the domain knowledge is not sufficient to check the sensibility of the results. The applicability domain results showed that there is a relationship between test compounds error and there similarity to the data used to train models.

In terms of applicability domain results, the outlier detection algorithms could consistently sort compounds and showed the decreased mean absolute error in predictions as the contamination factor was increased. However, it's unclear what is the best value for contamination factor to use when considering new fuels. In this research, the proposed method is to use the value where the rate of error decrease start to converge. However such approach has no formal definition. Therefore, a probabilistic approach in treating confidence of predictions could have better formal definition and be more intuitive.

The generative model used was a language model, which uses string smiles notation. The main drawback with such approach is that the developed model can only add symbols sequentially. The generative model use could have much greater impact by updating existing chemical structures. In this case the model can only add symbols to starting structure, which limits how the given molecule can be updated. For example the model can not delete atoms or change a bond type. Therefore for future work generative model which uses graph molecular representation can overcome such limitations. Models such as [111] can be applied for fuels prototyping in order to investigate decreasing saturation of fuel molecules, which were used in drug discovery research. Such modelling would help to increase the range of biomass which can potentially be used as fuel source.

This research built applicability of the octane number classification model, which agreed with known structural properties of fuels in 5 by 6 cases. The applicability domain of fuel properties was established by outlier detection algorithms, which could separate test compounds in such way the prediction errors reduce twice. Lastly the generative modelling showed promising fuel like chemical structures, with a known high octane number of approximately 100

References

- [1] (2016). Democratizing deep-learning for drug discovery, quantum chemistry, materials science and biology. <https://github.com/deepchem/deepchem>.
- [2] 37th Joint Meeting of the Chemicals Committee et al. (2019). Oecd principles for the validation, for regulatory purposes, of (quantitative) structure–activity relationship models.
- [3] Adeniji, S. E., Uba, S., Uzairu, A., and Arthur, D. E. (2019). A derived qsar model for predicting some compounds as potent antagonist against mycobacterium tuberculosis: a theoretical approach. *Advances in preventive medicine*, 2019.
- [4] Aha, D. W. and Bankert, R. L. (1996). A comparative evaluation of sequential feature selection algorithms. In *Learning from data*, pages 199–206. Springer.
- [5] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.
- [6] Al-Fahemi, J. H., Albis, N. A., and Gad, E. A. (2014). Qspr models for octane number prediction. *Journal of Theoretical Chemistry*, 2014.
- [7] Albahri, T. A. (2003). Structural group contribution method for predicting the octane number of pure hydrocarbon liquids. *Industrial & engineering chemistry research*, 42(3):657–662.
- [8] Alboqami, F., van Oudenhoven, V. C., Ahmed, U., Zahid, U., Emwas, A.-H., Sarathy, S. M., and Abdul Jameel, A. G. (2022). A methodology for designing octane number of fuels using genetic algorithms and artificial neural networks. *Energy & Fuels*.
- [9] Aldosari, M. N., Yalamanchi, K. K., Gao, X., and Sarathy, S. M. (2021). Predicting entropy and heat capacity of hydrocarbons using machine learning. *Energy and AI*, 4:100054.
- [10] Anas, M., Paling, R., Nailon, W., and Cumming, D. (2004). Real-time combustion knock processing using a single instruction multiple data automotive powerpc system-on-a-chip. In *2004 5th Asian Control Conference (IEEE Cat. No. 04EX904)*, volume 1, pages 296–303. IEEE.
- [11] Andrae, J. (2011). A kinetic modeling study of self-ignition of low alkylbenzenes at engine-relevant conditions. *Fuel processing technology*, 92(10):2030–2040.

- [Apache Software Foundation] Apache Software Foundation. Hadoop.
- [13] Arús-Pous, J., Johansson, S. V., Prykhodko, O., Bjerrum, E. J., Tyrchan, C., Reymond, J.-L., Chen, H., and Engkvist, O. (2019). Randomized smiles strings improve the quality of molecular generative models. *Journal of cheminformatics*, 11(1):1–13.
- [14] Bai, Y., Yang, E., Han, B., Yang, Y., Li, J., Mao, Y., Niu, G., and Liu, T. (2021). Understanding and improving early stopping for learning with noisy labels. *Advances in Neural Information Processing Systems*, 34:24392–24403.
- [15] Bajusz, D., Rácz, A., and Héberger, K. (2015). Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):1–13.
- [16] Barakat, N. and Bradley, A. P. (2010). Rule extraction from support vector machines: a review. *Neurocomputing*, 74(1-3):178–190.
- [17] Bilodeau, C., Jin, W., Jaakkola, T., Barzilay, R., and Jensen, K. F. (2022). Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, page e1608.
- [18] Bolton, E. E., Wang, Y., Thiessen, P. A., and Bryant, S. H. (2008). Pubchem: integrated platform of small molecules and biological activities. In *Annual reports in computational chemistry*, volume 4, pages 217–241. Elsevier.
- [19] Boot, M. D., Tian, M., Hensen, E. J., and Sarathy, S. M. (2017). Impact of fuel molecular structure on auto-ignition behavior—design rules for future high performance gasolines. *Progress in Energy and Combustion Science*, 60:1–25.
- [20] Borhani, T. N. G., Afzali, A., and Bagheri, M. (2016). Qspr estimation of the auto-ignition temperature for pure hydrocarbons. *Process Safety and Environmental Protection*, 103:115–125.
- [21] Bottou, L. et al. (1991). Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nimes*, 91(8):12.
- [22] Boulila, W., Driss, M., Alshantiti, E., Al-Sarem, M., Saeed, F., and Krichen, M. (2022). Weight initialization techniques for deep learning algorithms in remote sensing: Recent trends and future perspectives. *Advances on Smart and Soft Computing: Proceedings of ICACIn 2021*, pages 477–484.
- [23] Branan, C. (2002). Rules of thumb for chemical engineers: A manual of quick. *Accurate Solutions to Everyday Process Engineering Problems, 3rd ed.*; Gulf Professional Pub.: Amsterdam, The Netherlands.
- [24] Brecq, G., Bellettre, J., and Tazerout, M. (2003). A new indicator for knock detection in gas si engines. *International Journal of Thermal Sciences*, 42(5):523–532.
- [25] Brecq, G. and Le Corre, O. (2005). Modeling of in-cylinder pressure oscillations under knocking conditions: introduction to pressure envelope curve. *SAE technical paper*, (2005-01):1126.

- [26] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- [27] Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. (2019). Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108.
- [28] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and deep locally connected networks on graphs. arxiv. *arXiv preprint arXiv:1312.6203*.
- [29] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.
- [30] Button, A., Merk, D., Hiss, J. A., and Schneider, G. (2019). Automated de novo molecular design by hybrid machine intelligence and rule-driven chemical synthesis. *Nature machine intelligence*, 1(7):307–315.
- [31] Carilli, M. (2018). torchviz: A package for creating and rendering pytorch computation graphs. <https://github.com/szagoruyko/pytorchviz>.
- [32] Charu, C. A. (2018). *Neural networks and deep learning: a textbook*. Springer.
- [33] Chen, L., Li, T., Yin, T., and Zheng, B. (2014). A predictive model for knock onset in spark-ignition engines with cooled egr. *Energy Conversion and management*, 87:946–955.
- [34] Cherkasov, A., Muratov, E. N., Fourches, D., Varnek, A., Baskin, I. I., Cronin, M., Dearden, J., Gramatica, P., Martin, Y. C., Todeschini, R., et al. (2014). Qsar modeling: where have you been? where are you going to? *Journal of medicinal chemistry*, 57(12):4977–5010.
- [35] Correa, S. M. and Shyy, W. (1987). Computational models and methods for continuous gaseous turbulent combustion. *Progress in energy and combustion science*, 13(4):249–292.
- [36] Corti, E. and Moro, D. (2007). Knock indexes thresholds setting methodology. *SAE Transactions*, pages 1016–1024.
- [37] Creton, B., Dartiguelongue, C., de Bruin, T., and Toulhoat, H. (2010). Prediction of the cetane number of diesel compounds using the quantitative structure property relationship. *Energy & Fuels*, 24(10):5396–5403.
- [38] Curran, H. J. (2019). Developing detailed chemical kinetic mechanisms for fuel combustion. *Proceedings of the Combustion Institute*, 37(1):57–81.
- [39] Dahmen, M. and Marquardt, W. (2015). A novel group contribution method for the prediction of the derived cetane number of oxygenated hydrocarbons. *Energy & Fuels*, 29(9):5781–5801.
- [40] Dai Nguyen, H. and Tsuda, K. (2021). A generative model for molecule generation based on chemical reaction trees. *arXiv e-prints*, pages arXiv–2106.

- [41] Deng, H. and Runger, G. (2012). Feature selection via regularized trees. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.
- [42] Dossetter, A. G., Griffen, E. J., and Leach, A. G. (2013). Matched molecular pair analysis in drug discovery. *Drug Discovery Today*, 18(15-16):724–731.
- [43] Drew, K. L., Baiman, H., Khwaounjoo, P., Yu, B., and Reynisson, J. (2012). Size estimation of chemical space: how big is it? *Journal of Pharmacy and Pharmacology*, 64(4):490–495.
- [44] Duleep, K. G. (2017). The benefits of increasing fuel octane number on gasoline engine efficiency: A literature review.
- [45] Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*.
- [46] Eklund, M., Norinder, U., Boyer, S., and Carlsson, L. (2014). Choosing feature selection and learning algorithms in qsar. *Journal of Chemical Information and Modeling*, 54(3):837–843.
- [47] Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.
- [48] Fourches, D., Muratov, E., and Tropsha, A. (2010). Trust, but verify: on the importance of chemical structure curation in cheminformatics and qsar modeling research. *Journal of chemical information and modeling*, 50(7):1189.
- [49] Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.
- [50] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.
- [51] Galloni, E. (2012). Dynamic knock detection and quantification in a spark ignition engine by means of a pressure based method. *Energy conversion and management*, 64:256–262.
- [52] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feed-forward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.
- [53] Gómez-Bombarelli, R., Aguilera-Iparraguirre, J., Hirzel, T. D., Duvenaud, D., Maclaurin, D., Blood-Forsythe, M. A., Chae, H. S., Einzinger, M., Ha, D.-G., Wu, T., et al. (2016). Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach. *Nature materials*, 15(10):1120–1127.
- [54] Gómez-Méndez, I. and Joly, E. (2023). On the consistency of a random forest algorithm in the presence of missing entries. *Journal of Nonparametric Statistics*, pages 1–35.

- [55] Gonzalez, M. P., Teran, C., Saiz-Urra, L., and Teijeira, M. (2008). Variable selection methods in qsar: an overview. *Current topics in medicinal chemistry*, 8(18):1606–1627.
- [56] Goodarzi, M., Dejaegher, B., and Heyden, Y. V. (2012). Feature selection methods in qsar studies. *Journal of AOAC International*, 95(3):636–651.
- [57] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [58] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.
- [59] Grattarola, D., Zambon, D., Bianchi, F. M., and Alippi, C. (2022). Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.
- [60] Guha, R. (2013). On exploring structure–activity relationships. *In silico models for drug discovery*, pages 81–94.
- [61] Guha, R. and Willighagen, E. (2012). A survey of quantitative descriptions of molecular structure. *Current topics in medicinal chemistry*, 12(18):1946–1956.
- [62] Gupta, H. and Asha, V. (2020). Impact of encoding of high cardinality categorical data to solve prediction problems. *Journal of Computational and Theoretical Nanoscience*, 17(9-10):4197–4201.
- [63] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.
- [64] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.
- [65] Hellier, P., Ladommatos, N., Allan, R., and Rogerson, J. (2013). Combustion and emissions characteristics of toluene/n-heptane and 1-octene/n-octane binary mixtures in a direct injection compression ignition engine. *Combustion and flame*, 160(10):2141–2158.
- [66] Hellier, P., Talibi, M., Eveleigh, A., and Ladommatos, N. (2018). An overview of the effects of fuel molecular structure on the combustion and emissions characteristics of compression ignition engines. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 232(1):90–105.
- [67] Henderson, P. (2014). *Machine Learning an algorithmic perspective*. Chapman Hall/CRC, Cambridge.
- [68] Hinzman, L. D., Bettez, N. D., Bolton, W. R., Chapin, F. S., Dyurgerov, M. B., Fastie, C. L., Griffith, B., Hollister, R. D., Hope, A., Huntington, H. P., et al. (2005). Evidence and implications of recent climate change in northern alaska and other arctic regions. *Climatic change*, 72:251–298.

- [69] Hochreiter, S. and Schmidhuber, J. (1997a). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [70] Hochreiter, S. and Schmidhuber, J. (1997b). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- [71] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.
- [72] Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., and Shao, L. (2023). Normalization techniques in training dnns: Methodology, analysis and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [73] HURLEY, R. G. (2005). Journal of fuels and lubricants. *SAE transactions*, 114(4).
- [74] Ivanov, J., Polshakov, D., Kato-Weinstein, J., Zhou, Q., Li, Y., Granet, R., Garner, L., Deng, Y., Liu, C., Albaiu, D., et al. (2020). Quantitative structure–activity relationship machine learning models and their applications for identifying viral 3clpro-and rdrp-targeting compounds as potential therapeutics for covid-19 and related viral infections. *ACS omega*, 5(42):27344–27358.
- [75] Iyer, N., Thejas, V., Kwatra, N., Ramjee, R., and Sivathanu, M. (2021). Lrtuner: A learning rate tuner for deep neural networks. *arXiv preprint arXiv:2105.14526*.
- [76] Jaeger, S., Fulle, S., and Turk, S. (2018). Mol2vec: unsupervised machine learning approach with chemical intuition. *Journal of chemical information and modeling*, 58(1):27–35.
- [77] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2017). *An Introduction to Statistical Learning with Applications in R, Second Edition*. Springer, 2nd edition.
- [78] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.
- [79] Jiang, D., Wu, Z., Hsieh, C.-Y., Chen, G., Liao, B., Wang, Z., Shen, C., Cao, D., Wu, J., and Hou, T. (2021). Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of cheminformatics*, 13(1):1–23.
- [80] Jiao, L., Liu, H., Qu, L., Xue, Z., Wang, Y., Wang, Y., Lei, B., Zang, Y., Xu, R., Zhang, Z., et al. (2020). Qspr studies on the octane number of toluene primary reference fuel based on the electrotopological state index. *ACS omega*, 5(8):3878–3888.
- [81] Johnson, M. A. and Maggiora, G. M. (1990). *Concepts and applications of molecular similarity*. Wiley.
- [82] Jørgensen, P. B., Mesta, M., Shil, S., García Lastra, J. M., Jacobsen, K. W., Thygesen, K. S., and Schmidt, M. N. (2018). Machine learning-based screening of complex molecules for polymer solar cells. *The Journal of chemical physics*, 148(24):241735.
- [83] Jou, J.-H., Kumar, S., Agrawal, A., Li, T.-H., and Sahoo, S. (2015). Approaches for fabricating high efficiency organic light emitting diodes. *Journal of Materials Chemistry C*, 3(13):2974–3002.

- [84] Kamruzzaman, S. and Hasan, A. R. (2010). Rule extraction using artificial neural networks. *arXiv preprint arXiv:1009.4984*.
- [85] Karavalakis, G., Short, D., Vu, D., Russell, R., Hajbabaie, M., Asa-Awuku, A., and Durbin, T. D. (2015). Evaluating the effects of aromatics content in gasoline on gaseous and particulate matter emissions from si-pfi and sidi vehicles. *Environmental science & technology*, 49(11):7021–7031.
- [86] Keshavarz, M. H., Gharagheizi, F., and Ghanbarzadeh, M. (2013). A simple correlation for prediction of autoignition temperature of various classes of hydrocarbons. *Journal of the Iranian Chemical Society*, 10(3):545–557.
- [87] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [88] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- [89] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
- [90] Kubic Jr, W. L., Jenkins, R. W., Moore, C. M., Semelsberger, T. A., and Sutton, A. D. (2017). Artificial neural network based group contribution method for estimating cetane and octane numbers of hydrocarbons and oxygenated organic compounds. *Industrial & Engineering Chemistry Research*, 56(42):12236–12245.
- [91] Kwon, S., Bae, H., Jo, J., and Yoon, S. (2019). Comprehensive ensemble in qsar prediction for drug discovery. *BMC bioinformatics*, 20(1):1–12.
- [92] Lasocki, J. (2016). Engine knock detection and evaluation: a review. *Zeszyty Naukowe Instytutu Pojazdów*, 109(5):41–50.
- [93] LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.
- [94] Li, R., Herreros, J. M., Tsolakis, A., and Yang, W. (2020). Machine learning regression based group contribution method for cetane and octane numbers prediction of pure fuel compounds and mixtures. *Fuel*, 280:118589.
- [95] Li, R., Herreros, J. M., Tsolakis, A., and Yang, W. (2022). Integrated machine learning-quantitative structure property relationship (ml-qspr) and chemical kinetics for high throughput fuel screening toward internal combustion engine. *Fuel*, 307:121908.
- [96] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.
- [97] Linstrom, P. J. and Mallard, W. G. (2001). The nist chemistry webbook: A chemical data resource on the internet. *Journal of Chemical & Engineering Data*, 46(5):1059–1063.

- [98] Liu, Z., Zhang, L., Elkamel, A., Liang, D., Zhao, S., Xu, C., Ivanov, S. Y., and Ray, A. K. (2017). Multiobjective feature selection approach to quantitative structure property relationship models for predicting the octane number of compounds found in gasoline. *Energy & Fuels*, 31(6):5828–5839.
- [99] Lopes, M. E., Wu, S., and Lee, T. C. (2020). Measuring the algorithmic convergence of randomized ensembles: The regression setting. *SIAM Journal on Mathematics of Data Science*, 2(4):921–943.
- [100] Lovell, W. G. (1948). Knocking characteristics of hydrocarbons. *Industrial & engineering chemistry*, 40(12):2388–2438.
- [101] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777.
- [102] Maia, E. H. B., Assis, L. C., De Oliveira, T. A., Da Silva, A. M., and Taranto, A. G. (2020). Structure-based virtual screening: from classical to artificial intelligence. *Frontiers in chemistry*, 8:343.
- [103] Mannodi-Kanakithodi, A., Pilania, G., Huan, T. D., Lookman, T., and Ramprasad, R. (2016). Machine learning strategy for accelerated design of polymer dielectrics. *Scientific reports*, 6(1):1–10.
- [104] March, J. et al. (1968). *Advanced organic chemistry*.
- [105] Marsland, S. (2014). *Machine Learning: An Algorithmic Perspective, Second Edition*. Chapman & Hall/CRC, 2nd edition. Hardcover.
- [106] Martin, T. M., Harten, P., Young, D. M., Muratov, E. N., Golbraikh, A., Zhu, H., and Tropsha, A. (2012). Does rational selection of training and test sets improve the outcome of qsar modeling? *Journal of chemical information and modeling*, 52(10):2570–2578.
- [107] Masand, V. H., Mahajan, D. T., Nazeruddin, G. M., Hadda, T. B., Rastija, V., and Alfeefy, A. M. (2015). Effect of information leakage and method of splitting (rational and random) on external predictive ability and behavior of different statistical parameters of qsar model. *Medicinal Chemistry Research*, 24(3):1241–1264.
- [108] Mathea, M., Klingspohn, W., and Baumann, K. (2016). Chemoinformatic classification methods and their applicability domain. *Molecular Informatics*, 35(5):160–180.
- [Matt Swain] Matt Swain. pubchempy.
- [110] Mello, P., Wildner, F., de Andrade, G. S., Cataluna, R., and da Silva, R. (2014). Combustion time of the oxygenated and non-oxygenated fuels in an otto cycle engine. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 36(2):403–410.
- [111] Méndez-Lucio, O., Baillif, B., Clevert, D.-A., Rouquié, D., and Wichard, J. (2020). De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nature communications*, 11(1):10.

- [112] Miyao, T., Kaneko, H., and Funatsu, K. (2016). Inverse qspr/qsar analysis for chemical structure generation (from y to x). *Journal of chemical information and modeling*, 56(2):286–299.
- [113] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.
- [114] Moriwaki, H., Tian, Y.-S., Kawashita, N., and Takagi, T. (2018). Mordred: a molecular descriptor calculator. *Journal of cheminformatics*, 10(1):1–14.
- [115] Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21.
- [116] Neves, B. J., Braga, R. C., Melo-Filho, C. C., Moreira-Filho, J. T., Muratov, E. N., and Andrade, C. H. (2018). Qsar-based virtual screening: advances and applications in drug discovery. *Frontiers in pharmacology*, 9:1275.
- [117] Neves, B. J., Dantas, R. F., Senger, M. R., Melo-Filho, C. C., Valente, W. C., de Almeida, A. C., Rezende-Neto, J. M., Lima, E. F., Paveley, R., Furnham, N., et al. (2016). Discovery of new anti-schistosomal hits by integration of qsar-based virtual screening and high content screening. *Journal of medicinal chemistry*, 59(15):7075–7088.
- [118] Nicolaou, C. A. and Brown, N. (2013). Multi-objective optimization methods in drug design. *Drug Discovery Today: Technologies*, 10(3):e427–e435.
- [119] Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*.
- [120] Oshiro, T. M., Perez, P. S., and Baranauskas, J. A. (2012). How many trees in a random forest? In *Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings 8*, pages 154–168. Springer.
- [121] Panzani, G., Östman, F., and Onder, C. H. (2016). Engine knock margin estimation using in-cylinder pressure measurements. *IEEE/ASME transactions on Mechatronics*, 22(1):301–311.
- [122] Panzani, G., Pozzato, G., Savaresi, S. M., Rösgren, J., and Onder, C. H. (2019). Engine knock detection: an eigenpressure approach. *IFAC-PapersOnLine*, 52(5):267–272.
- [123] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.
- [124] Pauls, A. and Yoder, J. (2018). Determining optimum drop-out rate for neural networks. In *Midwest Instructional Computing Symposium (MICS)*.
- [125] Perdih, A. and Perdih, F. (2006). Chemical interpretation of octane number. *Acta chimica slovenica*, 53(3).

- [126] Polishchuk, P. (2017). Interpretation of quantitative structure–activity relationship models: past, present, and future. *Journal of chemical information and modeling*, 57(11):2618–2639.
- [127] Polishchuk, P. G., Kuz'min, V. E., Artemenko, A. G., and Muratov, E. N. (2013a). Universal approach for structural interpretation of qsar/qspr models. *Molecular informatics*, 32(9-10):843–853.
- [128] Polishchuk, P. G., Madzhidov, T. I., and Varnek, A. (2013b). Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679.
- [129] Puzyn, T., Mostrag-Szlichtyng, A., Gajewicz, A., Skrzyński, M., and Worth, A. P. (2011). Investigating the influence of data splitting on the predictive ability of qsar/qspr models. *Structural Chemistry*, 22(4):795–804.
- [130] Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1.
- [131] Reiser, P., Neubert, M., Eberhard, A., Torresi, L., Zhou, C., Shao, C., Metni, H., van Hoesel, C., Schopmans, H., Sommer, T., et al. (2022). Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93.
- [132] Reymond, J.-L. (2015). The chemical space project. *Accounts of Chemical Research*, 48(3):722–730.
- [133] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.
- [134] Riniker, S. and Landrum, G. A. (2013). Similarity maps—a visualization strategy for molecular fingerprints and machine-learning methods. *Journal of cheminformatics*, 5(1):1–7.
- [135] Rogers, D. and Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754.
- [136] Roy, K., Kar, S., and Ambure, P. (2015). On a simple approach for determining applicability domain of qsar models. *Chemometrics and Intelligent Laboratory Systems*, 145:22–29.
- [137] Roy, P. P., Leonard, J. T., and Roy, K. (2008). Exploring the impact of size of training sets for the development of predictive qsar models. *Chemometrics and Intelligent Laboratory Systems*, 90(1):31–42.
- [138] Roy, S. and Askari, O. (2020). A new detailed ethanol kinetic mechanism at engine-relevant conditions. *Energy & Fuels*, 34(3):3691–3708.
- [139] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

- [140] Ruiz, L., Gama, F., and Ribeiro, A. (2021). Graph neural networks: Architectures, stability, and transferability. *Proceedings of the IEEE*, 109(5):660–682.
- [141] Saarela, M. and Jauhiainen, S. (2021). Comparison of feature importance measures as explanations for classification models. *SN Applied Sciences*, 3:1–12.
- [142] Saldana, D. A., Starck, L., Mougin, P., Rousseau, B., Ferrando, N., and Creton, B. (2012). Prediction of density and viscosity of biofuel compounds using machine learning methods. *Energy & fuels*, 26(4):2416–2426.
- [143] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.
- [144] Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. (2018). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131.
- [145] Sholes, K. R., Odaka, M., Goto, Y., Ishii, H., and Suzuki, H. (2002). Study of the effect of boiling point on combustion and pm emissions in a compression ignition engine using two-component n-paraffin fuels. Technical report, SAE Technical Paper.
- [146] Skorski, M., Temperoni, A., and Theobald, M. (2021). Revisiting weight initialization of deep neural networks. In *Asian Conference on Machine Learning*, pages 1192–1207. PMLR.
- [147] Soto, A. J., Cecchini, R. L., Vazquez, G. E., and Ponzoni, I. (2009). Multi-objective feature selection in qsar using a machine learning approach. *QSAR & Combinatorial Science*, 28(11-12):1509–1523.
- [148] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.
- [149] Stanley, M., Bronskill, J. F., Maziarz, K., Misztela, H., Lanini, J., Segler, M., Schneider, N., and Brockschmidt, M. (2021). Fs-mol: A few-shot learning dataset of molecules. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- [150] Stauffer, E., Dolan, J., and Newman, R. (2008). Flammable and combustible liquids. *Fire debris analysis*, 2008:199–233.
- [151] Stawiski, M., Meier, P., Dornberger, R., and Hanne, T. (2022). Using the light gradient boosting machine for prediction in qsar models. In *International Joint Conference on Advances in Computational Intelligence*, pages 99–111. Springer.
- [152] Techspace USICT (Publication date not available). Normalization techniques in deep neural networks. Accessed on 2023-09-21.
- [153] Varnek, A. (2017). *Tutorials in chemoinformatics*. John Wiley & Sons.

- [154] vom Lehn, F., Cai, L., Tripathi, R., Broda, R., and Pitsch, H. (2021). A property database of fuel compounds with emphasis on spark-ignition engine applications. *Applications in Energy and Combustion Science*, 5:100018.
- [155] Wang, S. (2017). *Shock Tube/Laser Absorption Study of Aldehydes Kinetics*. PhD thesis, Stanford University.
- [156] Wang, Z., Chen, J., and Hong, H. (2021). Developing qsar models with defined applicability domains on pparγ binding affinity using large data sets and machine learning algorithms. *Environmental Science & Technology*, 55(10):6857–6866.
- [157] Weaver, S. and Gleeson, M. P. (2008). The importance of the domain of applicability in qsar modeling. *Journal of Molecular Graphics and Modelling*, 26(8):1315–1326.
- [158] Webb, S. J. (2015). *Interpretation and mining of statistical machine learning (Q) SAR models for toxicity prediction*. PhD thesis, University of Surrey.
- [159] Weininger, D. (1988). Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36.
- [160] Whitmore, L. S., Davis, R. W., McCormick, R. L., Gladden, J. M., Simmons, B. A., George, A., and Hudson, C. M. (2016a). Biocompoundml: a general biofuel property screening tool for biological molecules using random forest classifiers. *Energy & Fuels*, 30(10):8410–8418.
- [161] Whitmore, L. S., George, A., and Hudson, C. M. (2016b). Mapping chemical performance on molecular structures using locally interpretable explanations. *arXiv preprint arXiv:1611.07443*.
- [162] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.
- [163] Yasri, A. and Hartsough, D. (2001). Toward an optimal procedure for variable selection and qsar model building. *Journal of chemical information and computer sciences*, 41(5):1218–1227.
- [164] Yaws, C. L. (2012). *Yaws' critical property data for chemical engineers and chemists*. Knovel.
- [165] Yaws, C. L. and Gabbula, C. (2003). *Yaws' Handbook of thermodynamic and physical properties of chemical compounds*. Knovel.
- [166] Ying, R., You, J., Morris, C., Ren, X., Hamilton, W. L., and Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*.
- [167] Yuan, W., Hansen, A., and Zhang, Q. (2005). Vapor pressure and normal boiling point predictions for pure methyl esters and biodiesel fuels. *Fuel*, 84(7-8):943–950.

- [168] Zabryanskii, E. (1967). Knock resistance of aromatic hydrocarbons and catalytically cracked and reformed naphthas. *Chemistry and Technology of Fuels and Oils*, 3:355–358.
- [169] Zhang, L., Fourches, D., Sedykh, A., Zhu, H., Golbraikh, A., Ekins, S., Clark, J., Connelly, M. C., Sigal, M., Hodges, D., et al. (2013). Discovery of novel antimalarial compounds enabled by qsar-based virtual screening. *Journal of chemical information and modeling*, 53(2):475–492.
- [170] Zhang, S., Tong, H., Xu, J., and Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23.
- [171] Zhen, X., Wang, Y., Xu, S., Zhu, Y., Tao, C., Xu, T., and Song, M. (2012). The engine knock analysis—an overview. *Applied Energy*, 92:628–636.
- [172] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open*, 1:57–81.
- [173] Zhou, Z., Kearnes, S., Li, L., Zare, R. N., and Riley, P. (2019). Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):10752.
- [backend=biber, style=nature,]bibtex References/references