UNIVERSITÄT
BERN

MASTER THESIS

Awarding the academic title of
Master of Science in Pharmacy (M Sc Pharm)

University of Bern

Machine Learning for the Prediction of Drug induced Liver Injury Patterns

Master Thesis submitted by

Arlene Jeanne Günter
Immatriculation Nr. 15-916-729

Handed in 10th of July 2021
For the degree of

Master of Science in Pharmacy (M Sc Pharm)

Supervisor: MD PhD Felix Hammann
Universitätsklinik für Allgemeine Innere Medizin, Klinische Pharmakologie & Toxikologie, Inselspital Bern

Co-supervisor: PhD Verena Schöning
Universitätsklinik für Allgemeine Innere Medizin, Klinische Pharmakologie & Toxikologie, Inselspital Bern

Table of Contents

1 Abstract

It is known that drugs can cause severe side effects, when taken, one of these is hepatotoxicity. *Drug induced liver injury* (DILI) is one of the most common reasons for the withdrawal of drugs from the market (32%) (1). Due to the uncommon nature of DILI (13.9 ± 2.4 per 100,000 patients) this severe side effect is often noticed only when the drug is already on the market (2). With the predictions from the machine learning (ML) models, the hepatotoxicity could partially be evaluated before the pre-clinical studies and therefore be used as additional information in drug discovery and development. Through this, possible damage to the liver of patients could be prevented or at the very least a better understanding as to what characteristic of a molecule is relevant for hepatotoxicity could be achieved. This way pharmaceutical companies can save resources, and clinical trials for possibly dangerous substances can be avoided.

Using *Quantitative structure-activity relationship* (QSAR) analysis ML methods can be trained to differentiate between hepatotoxic and non-hepatotoxic substances. Through this the ML method finds correlations between physio-chemical properties of the drug and its biological activity.

It is possible to make predictions on the hepatotoxicity of a drug. Of all the models that were developed the *Random Forest* (RF) models showed the best results for the hepatotoxic classification (balanced accuracy of 0.71 and 0.73). More detailed models which included the injury pattern of DILI performed the best with *k nearest neighbor* (kNN) or *support vector machine* (SVM) models. But their predicting power is weak with 0.32 balanced accuracy for the kNN model and 0.26 for the SVM model.

Analyzing the descriptors chosen by the models of kNN and *decision tree induction* (DTI) showed that the most conclusive physio-chemical properties are the ones that describe the surface of the drug. The four categories that were the most common were: *Atom type electro topological state, extended topochemical atom, ring count* and *charged partial surface area*.

The prediction if a substance will cause DILI or not is possible. With further addition of data, the models can improve their predictability power. Until then the investigation of drugs using QSAR and ML provides helpful information on the important properties involved in hepatotoxicity.

2 Introduction

Drugs can affect every part of the body beneficially by curing an injury or an illness. However, all drugs can cause side effects, some of them so serious that they are the reason for the drugs being withdrawn from the market. Throughout history there have been a lot of documented cases. Benoxaprofen got approved in April 1982 and was withdrawn in August 1982 due to liver and kidney failure caused by the medication. Bromfenac sodium was approved in July 1997 and withdrawn in June 1998 due to causing severe hepatitis and leading to liver failure. Hepatotoxicity was also a reason for the removal of troglitazone in March 2000 and alatrofloxacin mesylate (3). Drug induced liver injury patterns (DILI) are rather rare, a French study reported an incidence of 13.9 ± 2.4 per 100,000 patients (2). Nonetheless it is important, since it is one of the main reasons for market withdrawal (32%) and termination of clinical trials (22%) (1). If a patient continues using drugs with these side effects it can have severe consequences for the liver and may end up being fatal (4). The DILI risk of a drug can lead to withdrawal from

the market as was the case for Alatrofloxacin, which was removed worldwide due to severe liver toxicity in 2006 (5,6).

To catch these possibly severe side effects before marketing the medication, clinical trials are conducted. Predominantly phase 1 clinical trial assesses side effects and overall safety and uses healthy people as test subjects. Phase 2 has its emphasis on effectiveness and short-term side effects on actual patients. The last phase before the drug is authorized is phase 3. Here, more information about effectiveness and safety in different populations and in combination with different drugs are obtained to find out how well the drug performs in daily usage. An adverse reaction that occurs at a low rate (1 in 3,000 – 5,000) is unlikely to be detected in these phases, simply because the tested population is not large enough to find side effects with statistical safety (7). After the marketing authorization, phase 4 starts (post-marketing). Here the drug is monitored in a large and diverse public population. Hitherto unknown rare adverse events like hepatotoxicity are more likely to show up in this large population (7,8).

Withdrawing a drug *after* marketing authorization is not just expensive for the marketing authorization holder, but also harmful to patients. To protect the health of patients and to prevent unnecessary loss of resources, it would therefore be preferable to find such adverse reactions or at least a prediction about the possible events before even the pre-clinical studies start.

Artificial intelligence and machine-learning (ML) can be useful to detect possible hepatotoxic properties of drug candidates early in the development. Quantitative structure-activity relationship (QSAR) can be implemented using ML models with the help of molecular descriptors to find correlations between properties and activity. QSAR addresses the correlation between physio-chemical properties and the biological activity of a substance (9). These properties can be characteristics like weight, number of aromatic rings, and electronegativity.

Finding patterns within data can be done by humans, but when no longer facing two to five different features but instead 300 or even 1000 features, the human mind reaches its limit. Computers can compare and process problems with a very large number of features. Decision tree induction (DTI), for instance, remains relevant in drug discovery to this day (10). Deep Learning as well as Big Data are used more and more in this field.

In this study, I compared different ways to predict the hepatotoxicity of molecules solely from structural features through various ML algorithms. The models were trained using information from the curated database *LiverTox* (4). Hepatotoxicity can be described at different levels of detail. The simplest differentiation would be for example whether a molecule is hepatotoxic or not. A deeper differentiation is the type of pattern the molecule causes. Ideally, we should be able to predict the pattern of the injury, not only if the drug is toxic or not. Therefore, two datasets were built: one to compare the predictability of just the toxicity and one to make predictions on the different injury patterns. These two sets were used to train different supervised ML algorithms and to compare the results to each other as well as to a random guess to see if certain algorithms were better suited than others to differentiate between the different outcomes.

Prior studies have used ML to make predictions about the hepatotoxicity of, for instance, pyrrolizidine alkaloids on the basis of their chemical structure (11). The same methodology could be used on medications and their corresponding DILI patterns to make predictions on the hepatotoxicity for similar structures.

As mentioned before, ML is growing in relevance (10) and hopefully a better risk-benefit assessment of a drug can be developed in the future through the help of the computing and predicting power of computers.

3 Question and Objectives

The objective of this thesis is to find out if it is possible to use ML with data based on the *LiverTox* database (4) to create models with sufficient predictive powers to establish whether or not a drug is hepatotoxic or not. Furthermore, compare the predictive power of different supervised ML models in their predictive power when it comes to clinically relevant hepatotoxicity and generate a ranking of different methods.

4 Definitions and Terms

ALT	Alanine transaminase
aP	Alkaline phosphatase
AST	Aspartate aminotransferase
AUC	Area under the curve
DILI	Drug induced liver injury
DTI	Decision tree induction
GGT	Gamma-glutamyl transferase
kNN	K nearest neighbor
logReg	Logistic regression
ML	Machine learning
OvR	One-vs-rest
RF	Random forest
RFE	Recursive feature elimination
RFECV	Recursive feature elimination cross-validated
SMILES	Simplified molecular-input line-entry system
SMOTE	Synthetic minority oversampling technique
SVM	Support vector machine

In this thesis, features and descriptors are used as synonyms for the values that are trained to predict a specific outcome.

5 Materials and Methods

Drug induced liver injury (DILI) is often a diagnosis of exclusion after other common causes of liver injuries and diseases are ruled out. Unlike for other specific liver injuries like hepatitis A, there are no specific analytical tests to find a causality between injury and drug. Due to the challenging nature of the diagnosis, attempts have been made to generate a standardized way

to assess the causality. Methods like the *Roussel Uclaf Causality Assessment Method (RUCAM)*, *Maria and Victorino (M&V)* and the *Naranjo Probability Scale* are commonly used. The gold standard, however, remains the expert opinion (4).

In most cases of liver injuries that are related to drugs, it can be difficult to single out a specific drug. It is difficult mainly because either multiple drugs were taken, other risk factors like excessive alcohol intake or a history of pre-existing liver injury are present. In practice it is often quite complex to attribute the cause of a liver injury to a single drug (4).

The injury pattern is one way of classifying drug induced liver diseases. They consist of hepatocellular, cholestatic and mixed hepatocellular-cholestatic injuries. These three refer to the histological features and are usually based on the serum enzyme elevations pattern of alanine transaminase (ALT), aspartate aminotransferase (AST), gamma-glutamyl transferase (GGT), and alkaline phosphatase (aP) as shown in Table 1 (4).

Hepatocellular injury clinically resembles an acute viral hepatitis. Cholestatic drug induced liver injury resembles bile duct obstruction. Mixed hepatocellular-cholestatic injury is, as the name suggests, a mixture of cholestatic and hepatocellular injury and is typical for many drugs. It occurs mostly in DILI and rarely in other forms of acute liver injury (4).

Pattern	**ALT/AST**	**GGT**	**aP**	**Example drug**
Hepatocellular	Highly elevated	Moderately increased	≥ 5	Isoniazid
Cholestatic	Moderately increased	Highly elevated	≤ 2	Ciprofloxacin
Mixed hepatocellular-cholestatic	Elevated	Elevated	≥ 2 and ≤ 5	Phenytoin

Table 1: The different injury patterns and the corresponding elevations of serum enzymes.

5.1 Origin of the Data

Two databases were used for this thesis: *LiverTox* and *DILI-RANK*. These databases differ in the source of the DILI evidence. *DILI-Rank* (5) focuses on FDA drug labelling and DILI causality evidence (12). The FDA labeling process consists of a data collection from preclinical toxicological data, clinical trials, pharmacovigilance reports and literature (13). *LiverTox* on the other hand focuses on Human DILI case reports (12).

The *LiverTox* database was used as foundation to construct the databases on which the ML models were built (4). A total of 1160 substances were assessed based on *LiverTox*. Additional substances from *DILI-Rank* (5) were used to enlarge the amount of data to train the models. The nontoxic substances from *DILI-Rank* were also used for the multiclass classification of the injury type of hepatotoxicity, but not the toxic substances since they do not have an injury pattern. This added 163 nontoxic and 169 toxic substances to the assorted data. Leading to a final number of 1492 substances.

5.2 Processing of the Data

Substances and clinical outcomes were compiled manually from the corresponding databases. As a first classification, it was assessed if a substance is hepatotoxic or not. As a second

classification, the injury pattern (hepatocellular, cholestatic or mixed) was used, resulting in the following classification structure:

- Hepatotoxic substances
    - Hepatocellular injury
    - Cholestatic injury
    - Mixed injury
    - Some combinations of the above
- Not hepatotoxic substances

LiverTox offers additional classification according to clinical phenotypes, which are even more detailed and contain classes like: acute hepatic necrosis, pure or bland cholestasis, sinusoidal obstruction syndrome and others. After assessing them we decided to exclude them from the models due to the relatively small number of samples for some combinations. The same process was used for the likelihood for DILI noted in *LiverTox*. The likelihood is marked by a letter ranging from A: 'well known to be hepatotoxic' to E: 'not believed or unlikely to be hepatotoxic'. In unclear cases where *LiverTox* described the hepatotoxicity of the substance with a likelihood of 'unknown' (indicated by X) or 'possible but not proven' (indicated by E*) the database of *DILI-Rank* (5) and the corresponding information for professionals were consulted for the final classification (4).

In addition to the possible outcomes, the name as well as the *Simplified Molecular Input Line Entry Specification (SMILES)* code of the substance was noted. The SMILES codes were taken from *PubChem* (14). A *SMILES* is a 1D representation of the 3D structure of a molecule. For example, the SMILES of Ethene (CH_2CH_2) is written as follows:

HC(H)=C(H)(H)

The general structure of the database is indicated in Table 2.

SMILES (Isomeric and canonical)	**Substance name**	**Hepatotoxic?** (Yes/No)	**Include for Toxicity models**	**Injury Pattern** (Separated into hepatocellular, cholestatic and mixed)	**Include in Pattern models**	**Clinical patterns** (Different subcategories)

Table 2: An exemplary depiction of the structure of the processed data from LiverTox and DILI-Rank. The injury patterns as well as the clinical patterns contain more than one column but are represented here by one.

Substances were further labelled as 'included' and 'not included' for both the hepatotoxicity and the patterns to denote substances that could not be included due to technical reasons. For instance, some drugs were proteins, for which no SMILES are available. Protein structures are very large compared to the other substances and not well characterizable by the physicochemical descriptor calculation software mentioned in 5.3. The same goes for elemental compounds like arsenic and zinc. The dataset was fixed on February 5, 2021, and no more substances were added after that date.

5.3 Calculating the Descriptors

A chemical descriptor is a number representing chemical information about a molecule. This can be as simple as the molecular weight, the number of aromatic rings or the number of hydrogen bonds. But it can also be complex such as dipole moments and polarity indices and

orbital electron densities (15). Combining all the different chemical descriptors allows for a quantitative structure activity relationship (QSAR) using ML models to predict the biological activities of a molecule.

Once the database was finalized, the chemical descriptors were calculated for the different chemical structures. By using/applying *PaDelPy* (16), *OpenBabel* (17) and *Chemical Development Kit* (18) a set of 2163 descriptors was generated, including 2D as well as 3D descriptors for each substance. 1875 of these descriptors were calculated by *PaDelPy* which uses the *Chemical Development Kit* as a basis and adds a few additional descriptors. The remaining 288 were calculated using *OpenBabel* to generate mainly the 3D descriptors, which turned out 0 when calculated by *PaDelPy*. Some of the 3D descriptors overlapped, but since the ones generated by *PaDelPy* were 0, they were later removed through the feature selection process (cf. 5.5.2).

5.4 Data Sets

The collected dataset was split into training- and test-data using a ratio of 80:20 with stratification by class to ensure each split has approximately the same ratio of outcomes. Most of the splits had *Synthetic Minority Oversampling Technique (SMOTE)* (19) applied to them (cf. 5.6). For the models that dealt with the different pattern, a label was introduced for each combination. In the end we had the following 6 pairs of training- and test-data. The dataset splits will be identified by the abbreviations shown in Table 3.

Content of the set	**Abbreviation**	**Number of substances before SMOTE**	**Number of substances after SMOTE**
Tox/nonTox (*SMOTE*)	TOX	1228	1431
Tox/nonTox only in the range of the patterns (*SMOTE*)	TOX2	940	960
Pattern with all labels separate (*SMOTE*)	PAT_S	940	3276
Pattern with all labels separate (*nonSMOTE*)	PAT_nS	940	940
Pattern with two labels merged (*SMOTE*)	PAT2_S	940	1890
Pattern with two labels merged (*nonSMOTE*)	PAT2_nS	940	940

Table 3: Overview of the different sets of training- and test-data generated from the database.

As in the first two datasets considered as binary classification problems, they differentiate between hepatotoxic and non-hepatotoxic. The remaining four datasets have more than these two classifications, making it a multilabel problem.

Multilabel problems are present when we assign more than one class to the data (20). In the case of DILI this means for example an injury can be hepatocellular as well as cholestatic, not just one or the other. We need the model to make a classification for every one of the different classes. Multilabel problems are akin to multiclass problems. Multiclass problems are closer to the binary classification, because the outcome can only be one of the classes (20).

The multilabel problem from the dataset was transformed into a multiclass problem, *SMOTE* only works on multiclass and not on multilabel. To achieve this transformation artificial labels were generated for the substances.

A label was created for each combination of labels by generating a so-called '*powerlabel*', like a fingerprint hash, using the following formula:

$$Powerlabel = 8 * x[Mixed] + 4 * x[Cholestatic] + 2 * x[Hepatocellular] + 1 * x[Hepatotoxic]$$

The labels within the Brackets can be 1 (TRUE) or 0 (FALSE), generating the following *powerlabels*: (the ones not listed simply had no instances)

Powerlabel	Meaning of the powerlabel
0	Not hepatotoxic
3	Hepatotoxic with hepatocellular injury
5	Hepatotoxic with cholestatic injury
7	Hepatotoxic with cholestatic and hepatocellular injury (not mixed)
9	Hepatotoxic with mixed injury
11	Hepatotoxic with mixed and hepatocellular injury
13	Hepatotoxic with mixed and cholestatic injury
15	Hepatotoxic with all injury patterns

A depiction on the frequency of the different *powerlabel* is seen in Figure 1. The label 'toxic with only mixed pattern' only had 11 instances. A mixed pattern means that it is hepatocellular as well as cholestatic, technically showing all patterns. It could therefore be debated if 9 and 15 belong together, since a mixed pattern means that hepatocellular as well as cholestatic are present and it is possible that either of those can occur alone. This is similar for all the patterns including the mixed injury label. We chose to combine 9 and 15 in one of the test configurations because 9 only had 11 instances in total for the 768 instances present. This was an attempt to balance the dataset a bit.

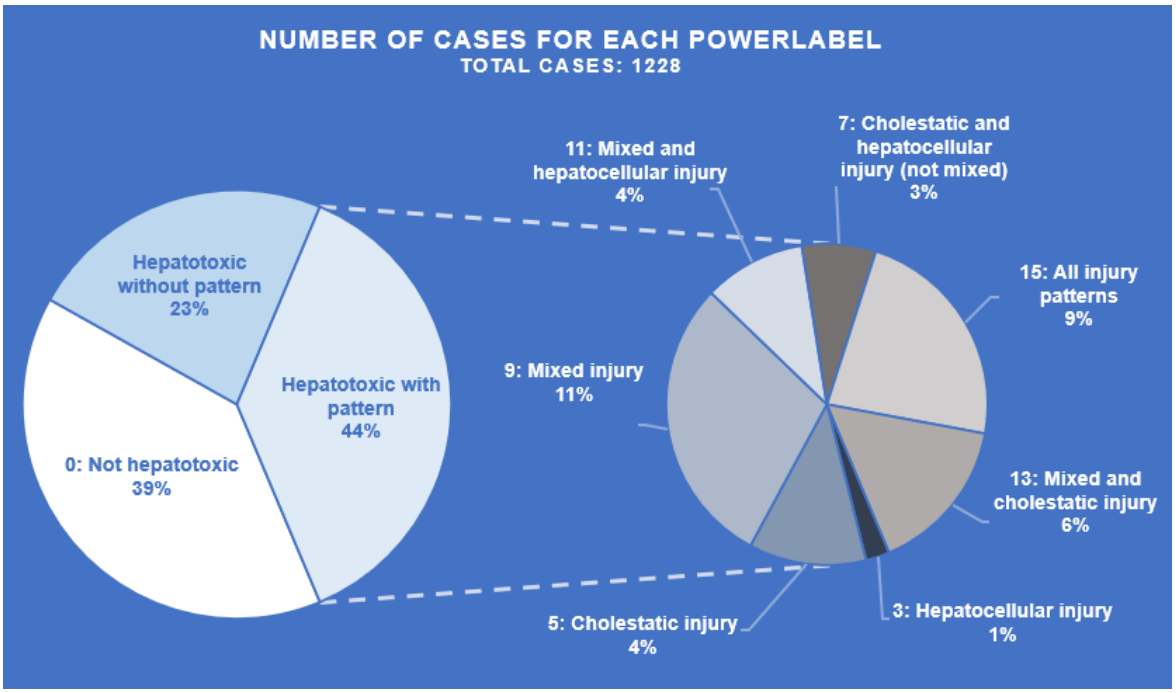


Figure 1: The distribution of the different cases within the binary and multiclass problem.

5.5 Pre-processing of the Data

After splitting up the datasets into training and test, pre-processing including feature selection was used to reduce the number of descriptors and to increase the precision of them. This was done after the split, to prevent information crossover. Pre-processing and feature selection in general is performed to improve the performance of the models by removing features with duplicated or unnecessary information and thus reducing interfering noise in the dataset (20,21).

In total five pre-processing steps were used to clean and reduce the descriptors:

- Normalization (min-max feature scaling)
- Low variance removal
- Intracorrelation removal
- Removal of columns with missing values
- Imputing of the still missing values, with the help of kNN

The feature reduction methods used for the other models do not work for kNN since it has no intrinsic way of weighing the different descriptors. Because of the importance of pre-processing, a loop was built to generate the ranking of the descriptors by hand and the best descriptors where then used to generate the kNN model (cf. 5.7.6).

5.5.1 Normalization

The first step was the normalization of the data to ensure that the range of the values for the descriptors is similar between [0, 1] using the following equation:

$$Normalized\ Value = \frac{original\ Value - minimal\ Value}{maximal\ Value - minimal\ Value}$$

This is also known as min-max scaling. Most ML algorithms, especially kNN, have problems when the values of the different features have different scales, since bigger values would have a bigger impact (21,22).

5.5.2 Low Variance removal

Low variance is about how many different feature values are within the data as well as how much they differ from one another. Since the data is normalized there is bound to be a value 1 and a value 0. A low variance would mean that most features are around the same value. A high variance on the other hand would mean that a lot of different values between 0 and 1 are taken. If there is next to no variance, the extreme case being only one value across the board, the variable holds no information. Low variance can hold information in combination with other features, if only concrete values can be taken by the variable, but this is rather unlikely. Using *VarianceThreshold* from *Scikit-Learn* features that are below a certain threshold of variance within themselves or in correlation with others (intercorrelation) can be removed (22). The formula used by *VarianceThreshold* is the following:

$$Var[x] = p(1-p)$$

The variance of the descriptor (x) is compared to the wanted threshold (p) and all descriptors that do not meet the criteria are removed. The threshold for low variance was set to reduce the number of descriptors to 900.

5.5.3 Intracorrelated Descriptors

Intracorrelation means that two features are not independent from each other, e.g., if x increases, then y increases/decreases in a parallel manner. The information of both descriptors is similar, therefore one of them can be dropped. One of the intracorrelated variables can therefore be dropped (21,22).

The threshold for intracorrelation was set to have only 400 descriptors remaining. Reducing dataset size boosts learning speed and helps to prevent overfitting.

5.5.4 Dealing with Missing Data

Missing data in the dataset are represented by NA. The NA stands for “not available”, i.e., values that are unknown. This can for example happen within the context of this thesis, if a structure is too complex to calculate a certain value of a descriptor or if the calculation timed out. As most ML models cannot handle NA values, these need to be removed by either deleting the whole descriptor or by imputing the value.

Through the removal of low variance descriptors, descriptors that only contain NA values have also been removed. In addition, descriptors with more than 25% NA values were removed, as the information gain is minimal.

For the remaining descriptors, the NA values imputed with kNN (cf. K). With this the columns with missing values[1] are filled in using the values of other substances. With the *KNNImputer* from *Scikit-Learn* the neighboring values are used to generate a likely value for each missing one. For each missing value the nearest neighbors were found and the mean value of them then used to create an artificial value (22).

[1] Please note that no more than 25% of the values are missing for one feature because of previous processing steps. If too much data is missing, the imputing leads to a crossover of values from other substances.

5.6 Balancing

A *balanced dataset* has about equal observations for each outcome. For a dataset with irregularly distributed observations, meaning one outcome is more common than the other, the term *unbalanced dataset* is used. In an *unbalanced dataset* the class with less observations is the *minority class*, whereas the more frequent one is called the *majority class* (23).

The dataset is unbalanced, as it contains more hepatotoxic substances than non-hepatotoxic ones. We balanced out the training dataset using *Synthetic Minority Oversampling Technique (SMOTE)* (19).

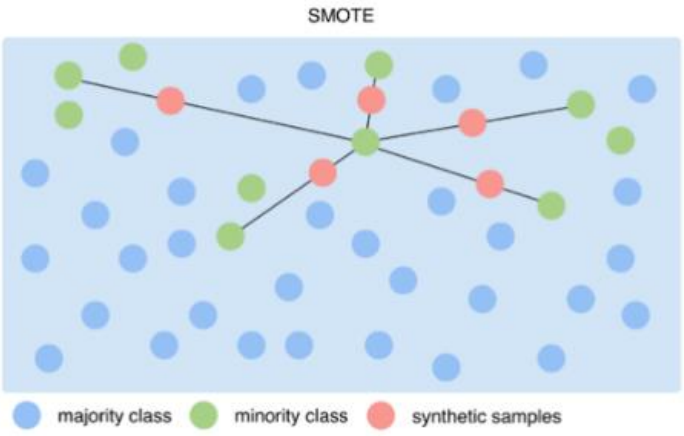


Figure 2: An example of how SMOTE generates new instances (23).

SMOTE is a way of resampling unbalanced datasets to achieve a more balanced one through synthetic oversampling. *Synthetic oversampling* is a term for generating synthetic observations or doubling observations of the minority class. *SMOTE* generates these new observations using a distance measure between the existing samples. The distance measure used is the nearest neighbor for one feature at a time and uses these values to fabricate a new instance, which is then multiplied with a random number between 1 and 0. Therefore *SMOTE* doesn't copy observations, it generates new synthetic ones from the data present (cf. Figure 2) (23).

After balancing with *SMOTE*, the different outcomes all have the same number of instances present in the training-data (Figure 3). To see the difference especially in the patterns we keep a version of the training data, that was not processed via *SMOTE*.

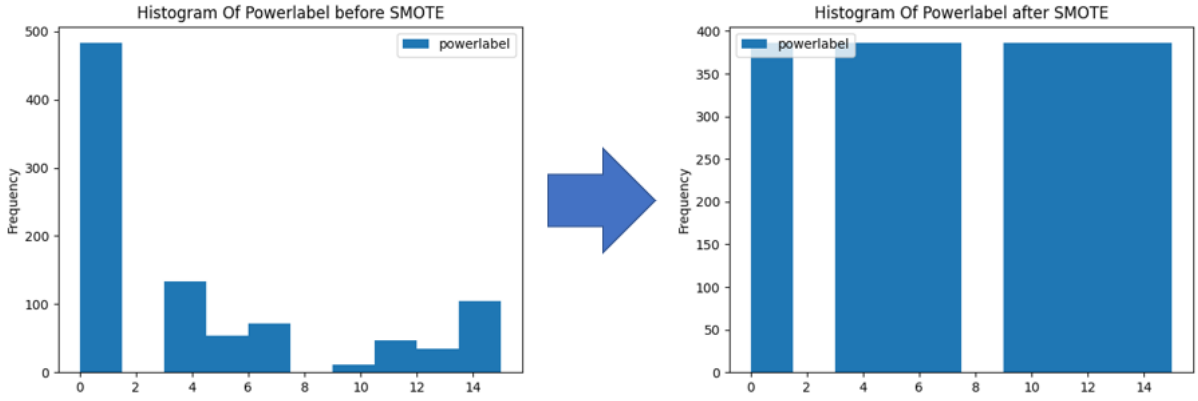


Figure 3: The influence of SMOTE on the frequency of the different powerlabels. After the use of SMOTE all labels are equally distributed.

5.7 Machine Learning Methods

Machine learning (ML) is an analytical technique to teach computers to learn from experience. The algorithms used in ML are not dependent on predetermined equations and "learn" directly from the data (24).

Each ML model possesses its own set of hyperparameters. *Hyperparameters* are parameters of the learning algorithm, which must be set before training and remain constant during training. They control different aspects of the model building process such as the number of iterations during fitting the model or the amount of regularization within the training. Proper tuning of the hyperparameters is an integral part of building ML systems. The different *hyperparameters* that were tuned for each model are explained in the appendix in 9.1.

5.7.1 Basic Concepts

5.7.1.1 Scoring Methods

Different scoring methods can be used to evaluate the predictive power of an ML model, e.g., '*accuracy'*, '*f1'*, '*precision'*, '*recall'* and '*roc_auc*'.

Accuracy is the number of correct predictions over the overall predictions made (21). *Precision* is a measurement of the positive prediction accuracy. It is the ratio of the correctly classified positive instances over the actual results. It is often used together with recall, also called sensitivity; it shows how many of the positive predicted values are indeed positive. Recall and precision are dependent on one another. Increasing recall reduces precision, and the other way around (20). The f1-score combines precision and recall into a single number. f1 is a *harmonic mean*, meaning that low values get more weight than they would if a *regular mean* is used. Therefore, uncommon values do not get neglected.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN} \qquad f1\ score = 2 * \frac{precision * recall}{precision + recall}$$

$$TP = true\ positive, FP = false\ positive, TN = true\ negative, FN = false\ negative$$

The *receiver operating characteristic* (*ROC*) curve shows the performance of a classifier with different discriminating thresholds by plotting the sensitivity vs 1-specificity. The *area under the curve* (*AUC*) is calculated for the graph generated by *ROC* and summarizes the curves information in one number (20,22).

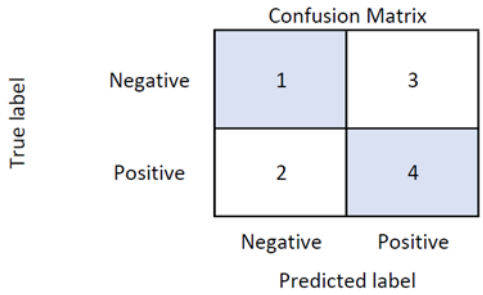
For a balanced accuracy (bal. ac.), also known as macro accuracy, the individual accuracy for the predictions of the different classes is calculated and given the same weight when adding them together. Through this the minority classes gain more weight and are not overlooked in favor of the majority classes in the model (22).

Macro values consider all classes to be of equal importance. If a model has trouble correctly classifying the minority class, this measurement will be impacted more than weighted values. Weighted values consider the frequency of the different classes and incorporate it in the calculation. This means that for an unbalanced dataset, the majority classes have more impact than the small classes (20,22).

$$weighted\ Score = Score_{class\ 1} * weight_1 + Score_{class\ 2} * weight_2 + \cdots + Score_{class\ n} * weight_n$$

$$macro\ Score = \frac{Score_{class\ 1} + Score_{class\ 2} + \cdots + Score_{class\ n}}{n}$$

A worked example of all the scores is shown in Figure 4.



$$accuracy = \frac{4+1}{4+1+3+2} = \frac{5}{10} = 0.5$$

$$precision = \frac{4}{4+1} = 0.8$$

$$recall = \frac{4}{4+3} = 0.57$$

$$f1\ score = 2 * \frac{0.8 * 0.57}{0.8 + 0.57} = 2 * \left(\frac{0.46}{1.37}\right) = 0.67$$

$$weighted\ precision = 0.8 * 0.6 + 0.33 * 0.4 = 0.48 + 0.132 = 0.61$$

$$macro\ precision = \frac{0.8 + 0.33}{2} = 0.57$$

Figure 4: Worked example of a confusion matrix along with a selection of commonly used performance measures.

We settled for *balanced accuracy* as the scoring method to optimize the models.

5.7.1.2 Cross-Validation

Cross-Validation is a statistical tool for calculating the generalization power of a model. It is done by repeatedly splitting the data to train the model at different points. *K-Fold Cross-Validation* is often used, K being the number of cross-validations that will be done. Each data point will be used as training as well as test data in different folds. Using stratification, folds are split according to the frequency of the outcome in the dataset, this prevents a minority class from being absent in a training or a test set. The *repeated cross-validation* repeats the entire process a defined number of times (cf. Figure 5) (25).

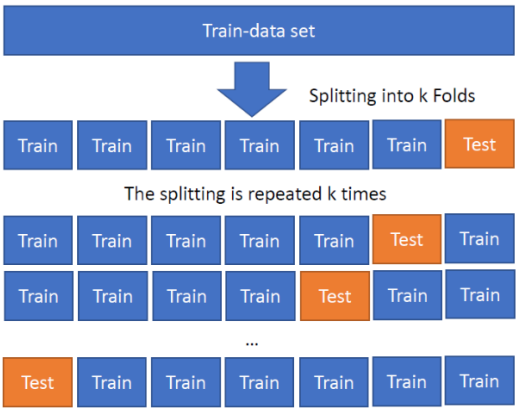


Figure 5: This process is called K-fold Cross-Validation, if it is repeated n times it becomes a repeated K-fold Cross-Validation (25).

The obtained results for the best hyperparameters within the scope of *GridsearchCV* were the hyperparameters used for the final models. The different models that were obtained were then compared to each other in respect to accuracy, f1-score, precision and recall using external validation through the test-set. Other than the comparison with each other, the models were evaluated on their expressiveness as well as the possibility of over- or underfitting of the data.

5.7.1.3 Over- and Underfitting

In *overfitting* the model overgeneralizes through the trainings data and memorizes them completely, meaning a model performing well on the training data but not on the test data or data outside the training set, since the generalization is not done well (20).

Underfitting is the opposite of overfitting. In this case the model is too simple to learn the structures that can be found in the model. The predictions are inaccurate on the training data as well as the test data, since reality is much more complex than the model (20).

5.7.1.4 GridsearchCV and RandomsearchCV

To tune the hyperparameters, *GridsearchCV* and *RandomsearchCV* were used (22,26).

GridsearchCV considers all parameter combinations given a group of possibilities needing precise values, whereas *RandomsearchCV* considers a given amount of combinations for a range of possibilities, which means it can be random numbers in a specified area (cf. Figure 6) (22).

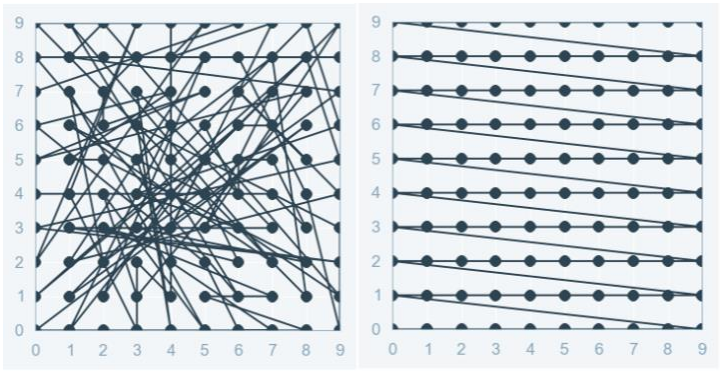


Figure 6: Visual representation of GridsearchCV (right) and RandomsearchCV (left) (22).

GridsearchCV was used after testing various scopes with *RandomsearchCV* to get a feel for what the best parameters are.

Here an example of the different scopes for the hyperparameter:

RandomsearchCV for SVM:
Gamma = [1000,500,100,50,10,5,1,0.05,0.01,0.005,0.001,0.0005,0.0001]
kernel = [’rbf’, ‘linear’, ‘poly’, ‘sigmoid’]
C = Range of 10^{-11} to 10^{5} with 10 multiplication steps, 16 values in total

GridsearchCV for SVM:
Gamma = [5,2,1,0.5,0.1,0.05,0.01,0.005,0.001]
kernel = [’rbf’]
C = Range of 10^{-3} to 10^{2} with 20 steps, therefore 20 values in total

Since for kernel only four different options are available, it was held constant for both searches, however, the parameters for gamma and C got a smaller scope around the optimal values obtained via *RandomsearchCV*.

5.7.2 Logistic Regression

Even though the model is named *logistic regression*, it is a linear algorithm for classification rather than a regression. It splits the descriptor space with a hyper-plane in two sub-spaces. On one side of the line, there is category 1 and on the other category 0 (cf. Figure 7). Therefore it is a binary classifier (20–22). This works well for binary classification problems such as the differentiation of toxic vs nontoxic substances. For multiclass problems, a *One-vs-Rest (OvR)* classifier is needed.

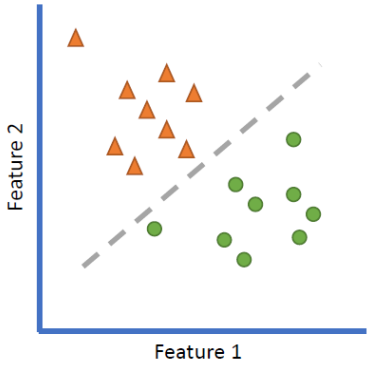


Figure 7: Logistic regression separates the data into sub-spaces.

OvR is a strategy which fits one outcome against all the other outcomes taken together, for example, nontoxic vs all toxic outcomes. This breaks the multiclass problem into multiple binary problems (22).

5.7.3 Decision Tree Induction

A decision tree is a hierarchical series of Yes/No decisions which leads to an outcome. For continuous features one must decide whether the feature is smaller or larger than a specific value (cf. Figure 8). For a prediction with a decision tree, all decisions from the root to the terminal node (the leaf) need to be made. If a tree is unconstrained, the fitting of the tree to the training data will be very tight and will most likely end in an overfitting. To avoid overfitting, the freedom of the tree needs to be restricted by regularizing the hyperparameters.

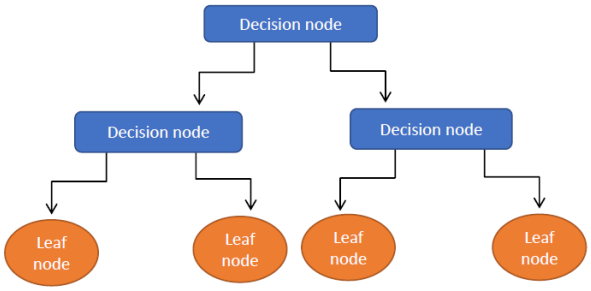


Figure 8: Starting with all the data, each decision node splits the data into smaller portions. In the leaf node the final classification is made.

One of the main issues concerning decision trees is the sensitivity to small variations in the training data, i.e., one variation could lead to an entirely different tree. *Scikit-Learn* uses a stochastic algorithm to train trees, which may lead to different results even for the same training data. This can be prevented by setting the random_state hyperparameter, it sets the random generator to a specific state to continuously return the same value, for each time it is used (20,21).

5.7.4 Random Forest

Random forests can be used to avoid the instability of single decision trees by calculating averages of all the predictions of the different trees (majority vote cf. Figure 9). It is an ensemble of decision trees built on different splits of the training set. Additional randomness is introduced due to growing the trees not for the best features sets, but on the best features of a random subset. Therefore there

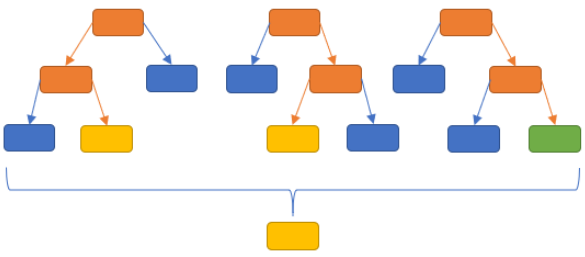


Figure 9: Each tree presents its own classification and with a majority vote the definitive classification.

are two different aspects of randomness: the choice of data points to build the tree and the choice of the tested features for each separation (20,21).

5.7.5 Support Vector Machine

Support Vector Machine (SVM) can be used to perform linear and non-linear classification as well as regression. The SVM classifier searches a decision plane that not only separates the classes but also is as far away as possible from the closest training data point (support vector). This type of classifier is well suited for small to medium size complex datasets (20).

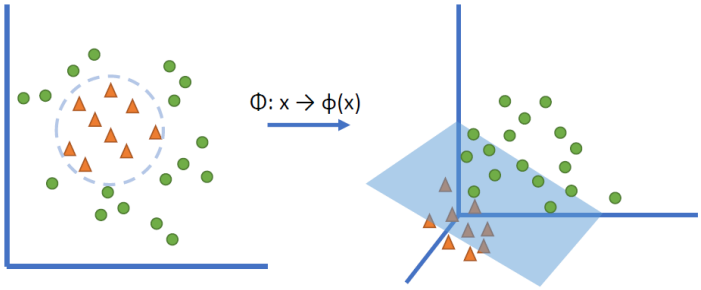


Figure 10: SVM makes separations through the generation of hyperplanes that can separate the data.

If a nonlinear separation is needed, the data is transformed to be in a higher dimensional space, with the goal that in this newly created space the data can be linearly separated (cf. Figure 10). The kernel function is a special kind of transformation that is useful in training SVM models. The so-called '*kernel trick*' reduces the computational cost drastically. The data is represented by the kernel method in pairwise similarity comparisons between two datapoints. By doing this the distance in the feature space can be calculated without explicitly calculating the extension (21).

5.7.6 K Nearest Neighbors

The training data is used to make predictions about new data points by searching the nearest points to it in the training set, the so-called *'nearest neighbors'* (20,21).

The pre-processing and feature reduction of the data is especially important for this classifier, since it often works poorly for large amounts of features and better with a small feature set (21).

The *'neighbor'* of a feature is the closest value in distance. The distance can be any measure, although commonly *Euclidian distance* is used. Often more than one *'neighbor'* is needed, taking the nearest values in distance to the feature in question until the necessary number of *'neighbors'* is reached. The parameter K defines the number of neighbors to be included in the analysis (cf. Figure 11). From the neighbors the most common class is chosen. To avoid ties for binary classification, the number of neighbors should be uneven (20,22).

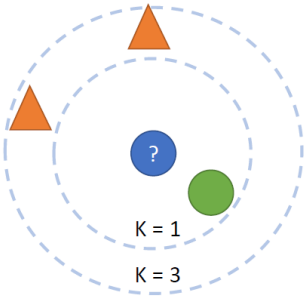


Figure 11: K is an important factor when running a kNN model. For K =1 the classification is a green circle, but for K = 3 it becomes an orange triangle.

The only hyperparameter tuned for kNN was the number of neighbors used.

The following is the pseudocode for the search of the best features for kNN as well as the ideal number:

- for the span of all features:
    - choose one after the other
    - calculate the score of the kNN model using one feature
    - take the best feature and save it
- for the next 30 iterations:
    - for all features which are not already chosen
        - add the feature to the previously chosen ones
        - calculate the score of the kNN model using one feature
        - take the best feature combination and save it
- use the number of selected features with the best score for the final model

5.8 Feature Reduction

Feature reduction was performed on all the models but kNN using recursive feature elimination (*RFECV*), cross validated from *Scikit-Learn* (27).

Recursive feature elimination (RFE) iteratively removes features starting with the full set. The importance of the features is gained by using the intrinsic method of a classifier. The least important features are dropped from the feature set. This is done repeatedly. *RFECV* is a form of recursive feature elimination (RFE*)* that has an integrated cross-validation loop to find the best features (22).

Since most models contain an intrinsic feature reduction, *RFECV* was no longer used in favor of computing time for the models.

5.9 Processing of the Multiclass Data

By using the *powerlabel* the data was in a form which the program could interpret with the same code and the same models as for the binary problem. In the same way as the binary datasets, the multiclass datasets were optimized using feature selection, *RandomsearchCV* and *GridsearchCV* using an internal repeated cross-validation.

In Figure 12 the complete process until the final models is depicted.

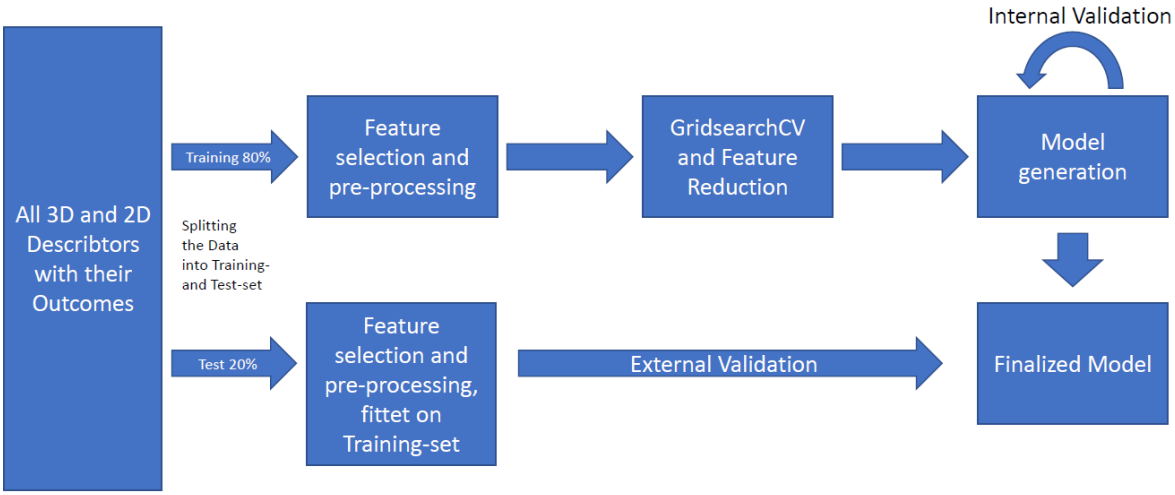


Figure 12: Overview of data processing. After the initial split, the feature-selection is made on the training-set and the test-set transformed accordingly. The models are built using GridsearchCV to find the

ideal hyperparameters and verified using cross-validation as an internal validation. The finalized model is validated using the test-set as external validation.

6 Results

6.1 Evaluation of the Descriptors Used in DTI and kNN

Using DTI and kNN we looked at the descriptors that the two models deemed important. SVM as well as logReg do not possess an intrinsic method to assign importance to the different descriptors. RF was not included in this comparison, because of the high computing time.

6.1.1 DTI

DTI allocates an importance to the features by itself, and the ranking can be called upon using *feature_importances_* a method from *scikit-learn* (cf. Figure 13).

Next to the feature importance, each tree was visualized using the *plot_tree* method as well as a text-based representation (cf. 9.4.2). The first and second splits of a decision tree are for the features with the highest differentiability. This is because of the restrictions put on the tree generation; the feature the model determines to have the highest importance will split earlier.

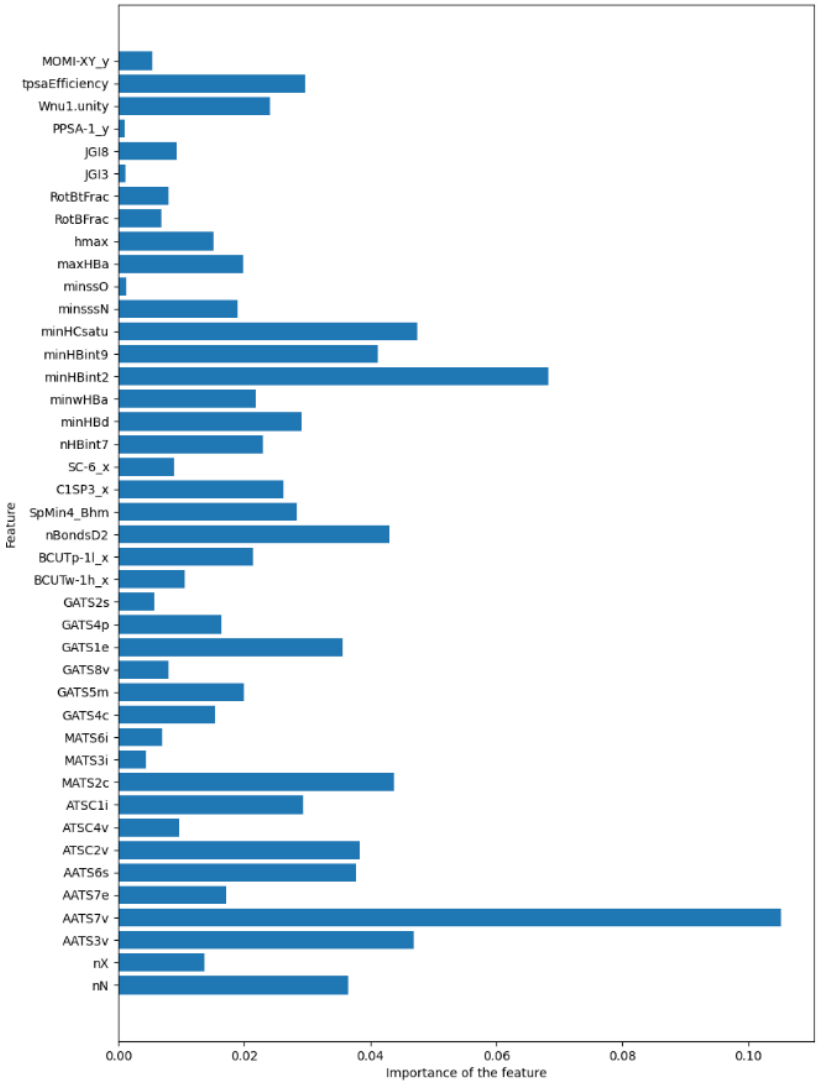


Figure 13: Feature importance from DTI model for TOX. The values for the importance of the features are rather low.

The second split contains two splits. They are on the same level and are therefore of the same importance. In the end we get the 3 most important features for each tree (Table 4). For a detailed explanation of a specific descriptor see the 'Handbook of Molecular Descriptors' (15), In the appendix (cf. 9.5) the abbreviations are attributed to the corresponding descriptor type.

Trainings set	First Split	Second Split (a)	Second Split (b)
TOX	AATS7v (0.51) *Autocorrelation*	minHBint2 (0.74) *Atom type electro topological state*	minHBd (0.49) *Atom type electro topological state*
TOX2	TDB1v (0.38) *3D autocorrelation*	nHBint2 (0.00) *Atom type electro topological state*	E1v (0.22) *WHIM*
PAT_S	ETA_Beta_ns_d (0.00) *Extended topochemical atom*	nBondsD2 (0.07) *Bond count*	Dv (0.43) *WHIM*
PAT_nS	AATS7v (0.51) *Autocorrelation*	minHdsCH (0.40) *Atom type electro topological state*	MAXDN (0.34) *Atom type electro topological state*
PAT2_S	C3SP3_x (0.00) *Carbon types*	nF10HeteroRing (0.36) *Ring count*	JGI6 (0.40) *Topological charge*
PAT2_nS	SHBint6 (0.36) *Atom type electro topological state*	CIC3 (0.51) *Information content*	JGI6 (0.23) *Topological charge*

Table 4: The different descriptors for each set used to separate the first three nodes of the decision tree. The values within the brackets are the thresholds for the splits. 0.51 means that the split is between the values above 0.51 and the ones below 0.51. Interesting are the values that are 0.00, meaning every value that is not 0 belongs to the same category. Below each value the type of the descriptor is noted.

6.1.2 kNN

Since kNN required a special method to achieve the feature reduction, the list that corresponded is sorted from most informative to least informative. The six sets deemed 144 features as important when given the task to select the best 30 out of 400. But only 31 occur in more than one model (Table 5). For details about the different descriptors that are not further explained, consult the 'Handbook of Molecular Descriptors' (15).

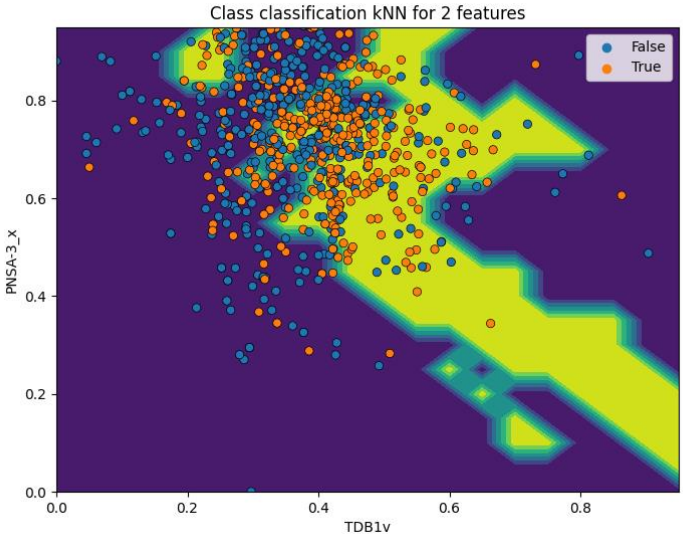


Figure 14: The two best features used in a kNN model using 11 neighbours for TOX2. The best features were determined using the feature_importances_ method from scikit learn.

The first 3 entries on each individual descriptor list, a list containing the 30 most informative descriptors, were used to generate a visualization of the kNN decision boundaries for the set they were representing (Figure 14).

TOX contains the lowest numbers of reoccurring descriptors with 6 out of 30, while PAT_nS contains the highest number with 14 out of 30.

Figure 14 shows that the data is quite convoluted and hard to separate. The decision boundaries set by the kNN model make sense and are like the boundaries a human would have set if given the same task. Adding additional dimensions makes it impossible to separate the data better than in this 2D representation.

Descriptor	Occurrences	Meaning
AATS8m	PAT_nS, PAT2_S	Autocorrelation, physico-chemical properties and molecular structure are transliterated in this topological descriptor.
BCUTc-1l_y	PAT_S, PAT2_nS	BCUT, calculated using the ‘Eigenvalues’ of a molecule
BCUTw-1h_x	PAT_nS, PAT2_S	BCUT, calculated using the ‘Eigenvalues’ of a molecule
C3SP3_y	PAT_nS, PAT2_nS	Carbon types
CrippenLogP	TOX, PAT_nS	Crippen log P
DPSA-1_x	TOX, PAT2_S	Charged partial surface area
E1m	PAT_nS, PAT2_S	WHIM
ETA_Beta	PAT_nS, PAT2_S	Extended topochemical atom
ETA_dBeta	PAT_nS, PAT2_S	Extended topochemical atom
ETA_Shape_Y	PAT_S, PAT_nS	Extended topochemical atom
FNSA-2_x	TOX, PAT_nS	Charged partial surface area
IC2	TOX2, PAT2_S	Information content
LipinskiFailures_y	PAT_nS, PAT2_S	Rule of five
MDEC-34_y	PAT_S, PAT2_nS	Molecular distance edge
mindsssP	PAT_nS, PAT2_nS	Atom type electro topological state
mintN	TOX, TOX2	Atom type electro topological state
n4HeteroRing	TOX2, PAT2_nS	Ring Count
nBase_x	TOX2, PAT2_nS	Basic group count
nF10HeteroRing	TOX2, PAT_nS	Ring Count
nF7HeteroRing	PAT_nS, PAT2_nS	Ring Count

Descriptor	Occurrences	Meaning
nFRing	PAT2_S, PAT2_nS	Ring Count
nHBint5	TOX, PAT_S, PAT2_nS	Atom type electro topological state
nHtCH	TOX2, PAT2_nS	Atom type electro topological state
nssO	TOX2, PAT2_nS	Atom type electro topological state
nT7Ring	TOX2, PAT2_nS	Ring Count
PNSA-3_x	TOX2, PAT2_S, PAT2_S	Charged partial surface area
PPSA-3_x	PAT_nS, PAT2_S	Charged partial surface area
RDF130u	TOX, PAT_S	RDF, radial distribution function
SIC0	TOX2, PAT_S	Information content
TDB1i	PAT_S, PAT2_S	3D autocorrelation
VC-5_y	TOX, PAT2_S	Chi cluster

Table 5: The descriptors used by the different kNN models that occurred in more than one model.

6.2 Evaluation of Hyperparameters from *GridsearchCV*

RF took by far the most computing time for one set of all the different models with *GridsearchCV*. It took 3 days for all 6 sets on an Intel® Core™ i7-3770 CPU with 16 GB RAM. Most other models took half a day or less for all 6 sets. A special case was kNN, since the 6 sets were calculated for different numbers of neighbors [1,3,5,7,9,11], needing 4 days to calculate all of them.

The results were obtained using the *classification_report* from *sklearn.metrics*, as well as the used hyperparameters from GridsearchCV. Here an example from logistic regression:

In Figure 15 the internal validation of the models achieved with the training date is marked by the best score (green). The best score depends on the scoring method used; we used the bal. ac. The accuracy (blue) refers to the external validation with the test set (it is a weighted value). The balance accuracy can be

TOX2 classification report for logReg

```
Fitting 50 folds for each of 39 candidates, totalling 1950 fits
Best Score: 0.6560705960705961
Best Hyperparameters: {'C': 0.1, 'max_iter': 5000,
                       'penalty': 'l2', 'solver': 'lbfgs'}
              precision    recall  f1-score   support

       False       0.67      0.62      0.65        97
        True       0.63      0.68      0.65        91

    accuracy                           0.65       188
   macro avg       0.65      0.65      0.65       188
weighted avg       0.65      0.65      0.65       188

balanced accuracy: 0.6499376911748046
```

PAT_S classification report for logReg

```
Fitting 50 folds for each of 39 candidates, totalling 1950 fits
Best Score: 0.9255261405261406
Best Hyperparameters: {'C': 100, 'max_iter': 5000,
                       'penalty': 'l2', 'solver': 'newton-cg'}
              precision    recall  f1-score   support

           0       0.64      0.58      0.61        97
           3       0.19      0.22      0.20        27
           5       0.12      0.09      0.11        11
           7       0.29      0.29      0.29        14
           9       0.00      0.00      0.00         2
          11       0.10      0.11      0.11         9
          13       0.12      0.14      0.13         7
          15       0.37      0.48      0.42        21

    accuracy                           0.42       188
   macro avg       0.23      0.24      0.23       188
weighted avg       0.44      0.42      0.43       188

balanced accuracy: 0.2382904895791494
```

Figure 15: An overview of the classification report as well as additional information about the best parameter and best score. In blue the accuracy score, in red the balanced accuracy, in green the internal validation score and in orange the macro value.

seen marked in red. The values for precision, f1 and recall in Table 6 are the same as the orange marked values in Figure 15. These are the macro average values, meaning an arithmetic mean of all the values from the different classes (28). Using these values, the models were evaluated and compared to each other as seen in Figure 15.

6.2.1 LogReg

The two hyperparameters tuned for the logistic regression were ‘solver’ and ‘C’. With two exceptions the ‘solver’ returned by *GridsearchCV* was ‘liblinear’. For TOX2 ‘lbfgs’ and for PAT_S ‘newton-cg’ were returned as the ‘solver’.
The hyperparameter ‘C’ showed more inconsistency. TOX, PAT_S and PAT2_S all returned a value of 100. This means that for these three the regularization was low compared to the other logReg models. TOX2 in comparison showed a high regularization with a value of 0.1, while the two sets without *SMOTE* both showed a value of 10.

6.2.2 DTI

GridsearchCV always returned the maximum of the ‘max_depth’ parameter available, which is 6. A *GridsearchCV* with a bigger depth was also conducted (cf. Figure 16), but the bal. ac. suffered due to overfitting of the tree.

The ‘max_features’ often returned the value of ‘sqrt’, meaning that for a split the number of features considered were equal to the square root of the number of descriptors, which would be 20 for the 400 features given.

The hyperparameter ‘min_samples_leaf’ mostly took on the value of 2,6 and 8. As for the ‘criterion’ hyperparameter, for the binary models ‘gini’ was preferred while for the multiclass models ‘entropy’ dominated.

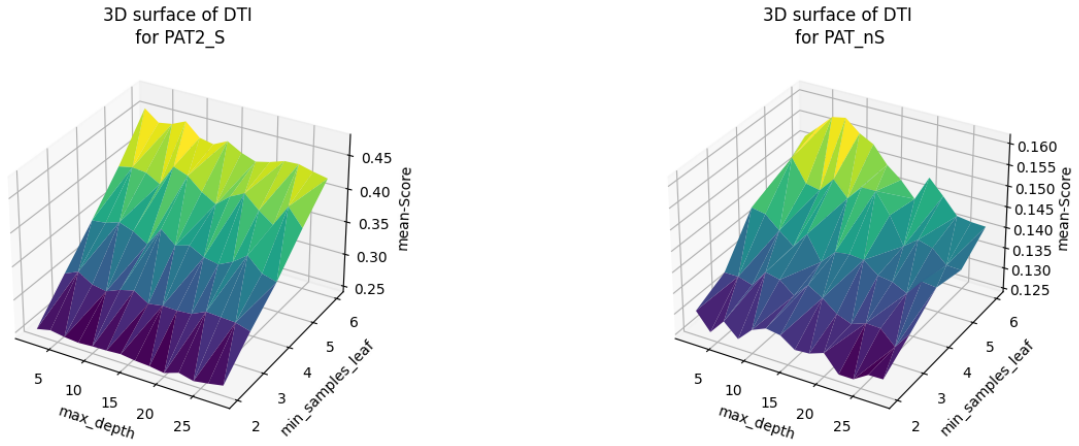


Figure 16: The influence of max_depth and min_samples_leaf on the overall balanced accuracy score for the different PAT models with the merged descriptors.

Figure 16 shows the interaction between the ‘max_depth’ and the ‘min_samples_leaf’ hyperparameter. The visualization illustrates that a high ‘min_samples_leaf’ together with a low ‘max_depth’ is preferable for the bal. ac. score. The ‘min_samples_leaf’ seems to have more impact on the accuracy than the ‘max_depth’.

6.2.3 RF

GridsearchCV showed that no restriction on the maximum of leaf nodes is preferred for the models. The ‘False’ for ‘bootstrap’ was taken by all the sets, meaning all the trees of each model were trained on the whole dataset. For the TOX and TOX2 ‘min_samples_leaf’ took the value of 4, while for the others it preferred the value 2. The hyperparameter ‘min_samples_split’ decided on the value 4 except for PAT_S and PAT2_nS where it was 2. The ‘max-features’ sway between ‘auto’ and ‘sqrt’ and no clear pattern of preference can be seen. A preference for the ‘criterion’ the Gini Impurity was observed.

6.2.4 SVM

results of hyperparameter tuning can be visualized as shown in Figure 16 and Figure 17 showing the influence of different hyperparameters on the bal. ac. of the model. Looking Figure 17 the value of ‘C’ seems to be best for the score, if it is low, while ‘gamma’ seems to be rather constant for each ‘C’.

The *GridsearchCV* leads to the default ‘kernel’ ‘rbf’. When looking at the other two hyperparameters ‘gamma’ and ‘C’ a pattern can be seen. TOX and TOX2 both show a ‘C’ of 1.64 and a ‘gamma’ of 0.5. This allows for a higher flexibility than the default value. The restriction value of ‘gamma’ is showing that the models are not highly restricted in the ‘rbf’ kernel.

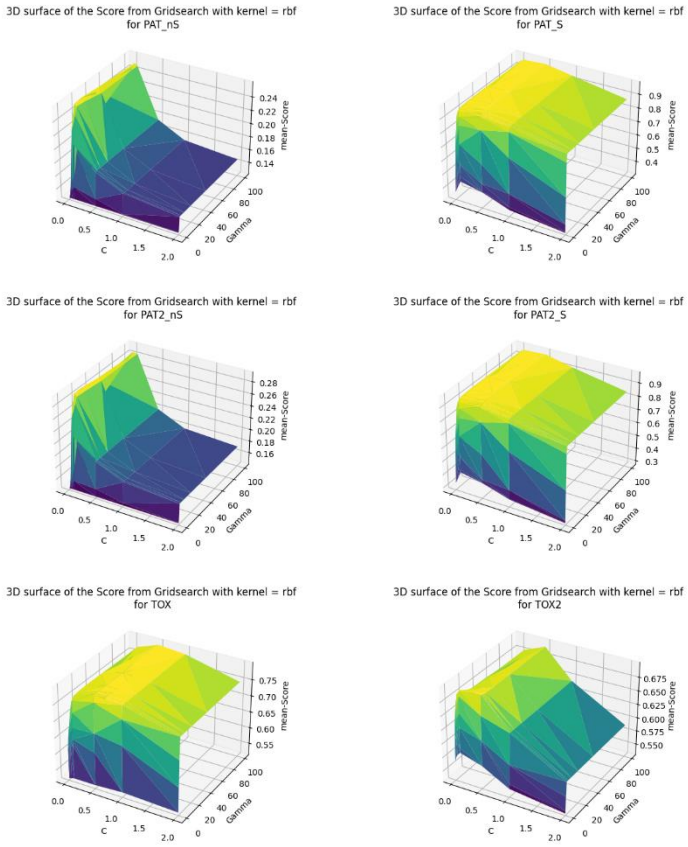


Figure 17: Influence of ‘gamma’ and ‘C’ on the balanced accuracy of the models.

The ‘gamma’ value remains the same for the *SMOTE* models, but the ‘C’ value changes to 8.48, allowing for an even higher curvature. The instances seem to be harder to separate und therefore need a higher flexibility for the separation.

The value of ‘C’ gets even higher for the models built from the sets without *SMOTE*, the value for both is 19.31. The value of 0.05 for ‘gamma’ means a lower restriction.

6.2.5 kNN

The only hyperparameter tuned for this model was the number of neighbors considered (‘k’) and the best models taken. TOX had an optimum at 5 neighbors, while TOX2 was at 11 neighbors. The two *SMOTE* sets performed better with a high number of neighbors (PAT_S = 9 and PAT2_S = 11), while the ones without *SMOTE* were at their best with 1 neighbor.

6.3 Performance

	balanced accuracy					macro precision					macro recall					macro f1-score				
Dataset	kNN	RF	DT	LogReg	SVM	kNN	RF	DT	LogReg	SVM	kNN	RF	DT	LogReg	SVM	kNN	RF	DT	LogReg	SVM
TOX	0.65	0.71	0.64	0.61	0.62	0.65	0.71	0.64	0.61	0.66	0.65	0.71	0.64	0.61	0.62	0.65	0.71	0.63	0.61	0.62
TOX2	0.71	0.73	0.58	0.65	0.63	0.71	0.73	0.58	0.65	0.65	0.71	0.73	0.57	0.65	0.63	0.71	0.73	0.55	0.65	0.63
PAT_S	0.24	0.25	0.23	0.24	0.18	0.22	0.30	0.19	0.23	0.34	0.23	0.25	0.23	0.24	0.18	0.19	0.26	0.17	0.23	0.19
PAT_nS	0.22 (0.2171)	0.22 (0.2240)	0.15	0.21	0.22 (0.2243)	0.26	0.40	0.13	0.23	0.27	0.22*	0.22*	0.15	0.21	0.22*	0.23	0.25	0.13	0.21	0.24
PAT2_S	0.32	0.25	0.22	0.24	0.17	0.28	0.31	0.20	0.23	0.27	0.32	0.25	0.22	0.24	0.17	0.27	0.26	0.18	0.23	0.16
PAT2_nS	0.23	0.22	0.17	0.24	0.26	0.27	0.41	0.20	0.26	0.28	0.23	0.22	0.17	0.24	0.26	0.24	0.23	0.16	0.24	0.27

*Table 6: Overview of the different models as well as their corresponding values for accuracy, precision, f1 and recall. Marked in green are the best values for the different models built from the corresponding dataset. The * next to the macro recall values means that this is the maximum of decimal places available through the classification report. The most important value to look at is the balanced accuracy.*

6.3.1 LogReg

The model for logistic regression is in the middle field of the models created looking at the bal. ac. (Table 6). The logReg models for the multiclass sets are all very close to each other when looking at the bal. ac. (~0.24).

The internal validation score for the models without *SMOTE* where 0.23 and 0.26, whereas with *SMOTE* they were 0.93 and 0.91. But looking at the actual bal. ac. in the external validation as well as the other scores the models do not differ significantly. The individual values for the different classes show that some classes present 0.00 across all values, meaning they are not being classified correctly at all. These are more common in the models without *SMOTE* than in the ones with *SMOTE*.

6.3.2 DTI

The DTI model was in the low range of the models when comparing the bal. ac. of the different models, especially for the multiclass classifications without *SMOTE*. It is the worst out of the 6 models for PAT_nS and PAT2_n.

6.3.3 RF

Table 6 shows that the RF models fare the best for the TOX and TOX2. The bal. ac. of 0.71 and 0.73 is the highest one present within all the different models. Looking at the individual recall values of TOX (toxic 0.76 and non-toxic 0.66) and TOX2 (toxic 0.76 and non-toxic 0.70), it becomes clear that the hepatotoxic substances are rightly classified more often.

For PAT_nS, the SVM and RF model have the same bal. ac. except for the fourth decimal point, which is neglectable. However, the higher macro values of precision, recall and f1-score suggest that the RF performs better.

6.3.4 SVM

The SVM models did not perform well except for the multiclass problems without *SMOTE*, where they performed better than the other models. This is the case of PAT_nS. There the bal. ac. of the models of SVM (0.22) and RF (0.22) are of the same quality.

6.3.5 kNN

The kNN model performs well when looking at the bal. ac. for nearly all the kNN models are the second best (except PAT2_nS and PAT2_S). PAT2_Sit even has the highest bal. ac. out of all the models with a value of 0.32.

7 Discussion

30 Models were successfully trained for hepatotoxic and DILI pattern detection. Assessments of their usability were made using the bal.ac. of each individual model. They were also compared to a random guess not only against each other. If a guess at random is made on which outcome belongs to which molecule, the following bal. ac. would be present:

0.50	For 2 different decisions (TOX, TOX2)
0.13	For 8 different decisions (PAT_S, PAT_nS)
0.14	For 7 different decisions (PAT2_S, PAT2_nS)

This differs in an unbalanced dataset where the safest guess would be using the majority class. The models did not do this since the scoring method of bal. ac. does not prefer them.

RF was shown to be superior for the hepatotoxicity classification problem over the other models. The bal. ac. of 0.71 for TOX and 0.73 for TOX2 are higher than a pure chance probability of 0.5. This means the models are about 22% more likely to predict the correct value than a guess.

The multiclass problem showed different priorities. SVM showed the best models for the two sets without *SMOTE* (bal.ac. PAT_nS = 0.22, PAT2_nS=0.26). For the sets with *SMOTE*, RF (0.22) and DTI (0.32) showed promising results for the classification.

7.1 Feature Importance

In Table 4 the first splits were listed with the corresponding decision boundary. Two descriptors show up twice (AATS7 and JGI6) indicating that these two show a higher importance for classification. For AATS7 the decision boundary remains the same. Comparing the feature importance of DTI in Figure 13 with the descriptors listed in Table 4, it can be seen that AATS7v, which is used in the first split, indeed has the highest feature importance (>0.1) for TOX. The descriptor minKBint2 shows the next highest values of feature importance and is also present in the next split. But minHBd shows a lower feature importance than other values, still it is contained within the next split. This could mean that the combination of the descriptor minHBd together with AATS7v and minKBint is better than the actual importance of minHBd.

Interestingly three descriptors show a decision value of 0.00 (nHBint2, ETA_Beta_ns_d and C3SP3_x), meaning that if this feature is present in any way, it has an influence on the classification. These three are different kinds of descriptors: *Atom type electro topological state, Extended topochemical atom* and *Carbon type* (which concerns sp^3 hybridization). Topochemical features as well as the electro topological state seem to play an important role on the hepatotoxicity of a molecule.

The importance of these specific types of descriptors is shown not only for DTI, but also when looking at the features from the kNN models. Four types of descriptors are more common than the others:

- Atom type electro topological state
- Extended topochemical atom
- Ring count
- Charged partial surface area

Especially '*Atom type electro topological state'* often occurs. As mentioned in 'Mechanisms of Hepatotoxicity', one of the main targets for hepatoxicity in drugs are the mitochondria (29). To influence the mitochondria of the liver cells a molecule would need to be able to enter through the cell membrane. Descriptors concerning the surface of the molecule as well as its ability to cross a cell membrane are exactly what we see predominantly selected.

The lack of reoccurrence of specific features suggests that the features within a descriptor class are all similar in importance and not that different from each other. This is the case for both DTI and kNN.

7.2 Hyperparameter Interpretation

7.2.1 LogReg

The penalty term 'C' is used to regulate against overfitting. The high values suggest that an overfitting is taking place. As for the 'solver' hyperparameter, the models all work with an 'l2' 'penalty', the only difference being that 'liblinear' uses an OvR approach, while 'lbfgs' and 'newton-cg' use a multinominal approach as well as an OvR one. Looking at the bal.ac. of the different solvers it does not seem to impact the model much, if at all.

7.2.2 DTI

The DTI model was in the low range of the models when looking at the bal.ac. especially for the multiclass classifications without *SMOTE*. This could be because of overfitting, as seen for the chosen hyperparameter. Further pruning (restriction of hyperparameters) of the tree could improve the model.

Interestingly a value of 8 is present for PAT_nS for 'min_sample_leaf' even though the smallest class has only 9 substances. The high number of 8 might lead to wrong classifications if for the smallest class 88% of all values need to be present. The chances of another class being present within the leaf node after the split is very high. That this model does not work well can be confirmed when looking at the bal. ac. which is 0.15. The 'min_samples_leaf' value of 2 is much more reasonable for the sets without *SMOTE*, since the sample size can be very small. For the *SMOTE* sets higher numbers would be expected, since they have equal amounts of instances, which are more than 100 each. A leaf node only needing 2 instances to justify a split could lead to strange splits that are too specific and end in overfitting.

The 'max_depth' parameter is maxed at the value of 6. This was chosen due to the model tending to pick the highest option, leading to overfitting.

Overfitting is one of the main problems we faced when making a DTI model. An idea as to how this could be solved would be to choose percentages for the values for 'min_samples_leaf'. This would lead to the hyperparameter being scaled with the size of the dataset used, thus

making the values usable for *SMOTE* and not *SMOTE* sets alike. However, this would not solve the problem with the minority classes and the tiny splits needed for them.

7.2.3 RF

GridsearchCV showed that no restriction on the maximum of leaf nodes allowed is preferred. This could lead to the suggestion of overfitting taking place. The low values for 'min_samples_leaf' and 'min_samples_split' suggest the same problem, namely that the generated trees tend to overfit towards the data for the sets with *SMOTE*. But RF models are very robust against overfitting, due to the sampling method used. If overfitting truly were a problem, the models would have performed like the DTI models.

7.2.4 SVM

The changes in the hyperparameters show an increase in flexibility in 3 steps. The least flexible are TOX and TOX2, PAT_S and PAT2_S are more flexible and PAT_nS and PAT2_nS are the most flexible when it comes to the kernel function. The higher the flexibility the more the function is bent and curved by the data. If the flexibility is too high, we end up with overfitting, if it is too low, an underfitting takes place.

Looking at the bal. ac. of the different SVM models, the most flexible models perform the best when compared with the other models. The need for a high flexibility leads to the conclusion that the data is quite convoluted and hard to separate with a straight line.

7.2.5 kNN

Considering the number of substances in each class, it is quite clear why the kNN models for the not oversampled models favor a low number of neighbors. For PAT_nS one minority class only contains 9 instances in the training set. If k = 11 is used, there are bound to be instances that are not the same class, simply because there are not 11 instances in the set. In general, the number of neighbors should not exceed the number of substances within the smallest class, or else the performance will be negatively influenced.

7.3 Performance in Comparison

The bal. ac. of TOX 0.71 and TOX2 0.73 is significantly better than a random guess. The model can predict whether a substance is toxic with an average bal. ac. of 72%. Looking at the individual recall values of TOX (toxic 0.76 and non-toxic 0.66) and TOX2 (toxic 0.76 and non-toxic 0.70), it becomes obvious that the toxic substances are correctly classified more often. This is favorable for drug discovery application since it would be more important to correctly classify a substance as toxic than non-toxic. The values are also higher for precision and f1-score when it comes to the toxic substances as opposed to the non-toxic ones.

When we compare our best model with the literature, we see that different models exist using the DILI-Rank database which reach an accuracy (not balanced) between 0.7 and 0.8. These models were made using a Bayesian machine learning model (30). TOX and TOX2 both reached an accuracy of 0.73 (not balanced) which places them within a similar range. In another publication a RF model was trained that reached an accuracy of 0.76 and was based on DILI-Rank. This model mainly differentiated between the substances with a high DILI concern and the ones without. Leaving out the substances with only a weak DILI concern (31), our

model differs from these examples mainly in terms of its basis, since our basis is LiverTox with the addition of DILI-Rank.

When looking into the detailed values for the kNN model for PAT2_S (Figure 18) we can see that the f1-score is overall quite evenly distributed except for the two smallest classes of 11 and 13. 0.39 for the recall of the negative hepatotoxicity means that in 39% of the classifications it is classified correctly. Over half the time the model would predict the wrong result. Even though this is better than predicting an outcome randomly, right now the kNN model for PAT2_S is not precise enough to be used as a proper classification.

```
maxvalue =  0.798218448218448
 max value position equal to number of features -1 =  29
              precision    recall  f1-score   support

           0       0.68      0.27      0.39        97
           3       0.28      0.41      0.33        27
           5       0.22      0.36      0.28        11
           7       0.29      0.57      0.38        14
          11       0.06      0.11      0.08         9
          13       0.06      0.29      0.10         7
          15       0.38      0.26      0.31        23

    accuracy                           0.31       188
   macro avg       0.28      0.32      0.27       188
weighted avg       0.48      0.31      0.34       188

balanced accuracy:  0.324029791661219
```

Figure 18: The classification report of the kNN model for PAT2_S.

7.4 Conclusion

It is possible to use the descriptors generated from the chemical structure to train ML models that can differentiate between the hepatotoxicity and the injury pattern. The RF model performed the best out of all the models tested for the binary classification with a bal.ac of 0.71 and 0.73. Whereas for the multiclass classification no clear preference for the RF model was visible. The kNN model performed by far the best out of all the multiclass models, but only for the PAT2_S set with a bal.ac of 0.32.

The models are limited because of the small amount of data available for the minority classes. With a bigger dataset as a basis, the predicting power of the model would be expected to improve significantly. Additionally, a more intense hyperparameter-tuning could increase the bal. ac. of the models slightly since some of the models showed signs of being overfitted.

In a further attempt all the low hepatotoxic substances could be excluded, and the model trained on only the highly and well-known toxic substances. After the training it would be interesting to see in which category the less hepatotoxic substances get categorized. This is similar to the approaches seen for the DILI-Rank based ML models in the literature mentioned in 7.3 (30,31).

It might also be interesting to test semi-supervised or unsupervised ML models with additional data. Unsupervised ML algorithms like Generative Adversarial Networks (GANs) could lead to intriguing models. Semi-supervised ML algorithms could be used to include unlabeled data, since from the original 1493 substances 269 could not be incorporated due to the lack of information about them. They were either categorized as unknown or not proven in *LiverTox.* These substances without a corresponding *DILI-Rank* and therefore without a proper label cannot be used in supervised learning.

Overall, the models show promising results in the prediction of DILI. With further improvements models like these could become an important asset for drug development regarding DILI in the future.

8 References

1. Watkins PB. Drug safety sciences and the bottleneck in drug development. Clin Pharmacol Ther. 2011;89(6):788–90.

2. Bell L, Chalasani N. Epidemiology of idiosyncratic drug-induced liver injury. Semin Liver Dis. 2009;29(4):337–47.

3. Qureshi ZP, Seoane-Vazquez E, Rodriguez-Monguio R, Stevenson KB, Szeinbach SL. Market withdrawal of new molecular entities approved in the United States from 1980 to 2009. Pharmacoepidemiol Drug Saf. 2011;20(7):772–7.

4. Bethesda (MD): National Institute of Diabetes and Digestive and Kidney Diseases. LiverTox: Clinical and Research Information on Drug-Induced Liver Injury [Internet]. [cited 2021 Feb 5]. Available from: *https://www.ncbi.nlm.nih.gov/books/NBK547852/*

5. FDA's National Center for Toxicological Research. Drug Induced Liver Injury Rank (DILIrank) Dataset [Internet]. [cited 2021 Mar 4]. Available from: *https://www.fda.gov/science-research/liver-toxicity-knowledge-base-ltkb/drug-induced-liver-injury-rank-dilirank-dataset*

6. Bedi O, Banerjee AB, Singh TG, Arora S, Gupta M. Ranitidine induced hepatotoxicity: a review. J Pharm Technol Res Manag. 2020;8(1):39–46.

7. Suvarna V. Phase IV of drug development. Perspect Clin Res. 2010;1(2):57–60.

8. NIH: National Institute on Aging. Clinical Trials: What Are Clinical Trials and Studies? [Internet]. [cited 2021 May 25]. Available from: *https://www.nia.nih.gov/health/what-are-clinical-trials-and-studies*

9. Benfenati E, Casalegno M, Cotterill J, Price N, Spreafico M, Toropov A. Characterization of chemical structures. In: Benfenati E, editor. Quantitative Structure-Activity Relationships (QSAR) for Pesticide Regulatory Purposes. Elsevier; 2007. p. 83–109.

10. Schöning V, Hammann F. How far have decision tree models come for data mining in drug discovery? Expert Opin Drug Discov. 2018;13(12):1067–9.

11. Schöning V, Hammann F, Peinl M, Drewe J. Identification of any structure-specific hepatotoxic potential of different pyrrolizidine alkaloids using random forests and artificial Neural Networks. Toxicol Sci. 2017;160(2):361–70.

12. Thakkar S, Chen M, Hong H, Liu Z, Fang H, Tong W. Drug-induced liver injury (DILI) classification and its application on human DILI risk prediction. In: Methods in Pharmacology and Toxicology. Springer; 2018. p. 45–59.

13. Murphy S, Roberts R. "Black box" 101: How the Food and Drug Administration evaluates, communicates, and manages drug benefit/risk. J Allergy Clin Immunol. 2006 Jan 1;117(1):34–9.

14. National Library of Medicine. PubChem [Internet]. [cited 2021 Jan 7]. Available from: *https://pubchem.ncbi.nlm.nih.gov/*

15. Todeschini R, Consonni V. Handbook of Molecular Descriptors. Mannhold R, Kubinyi H, Timmerman H, editors. Vol. 11, Methods and Principles in Medicinal Chemistry. New York: Wiley-VCH; 2000. p. 688

16. Yap CW. PaDEL-descriptor: An open source software to calculate molecular descriptors and fingerprints. J Comput Chem. 2011;32(7):1466–74.

17. O'Boyle NM, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR. Open Babel: An open chemical toolbox. J Cheminform. 2011;3(10).

18. Guha R. CDK Descriptor Calculator GUI (v 1.4.8) [Internet]. [cited 2021 May 22]. Available from: *http://www.rguha.net/code/java/cdkdesc.html*

19. Chawala N V., Bowyer KW, Hall LO, Kegelmeyer PW. SMOTE: Synthetic Minority Over-sampling Technique. J Artif Intell Res. 2002;(16):321–57.

20. Géron A. Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. O'Reilly. 2017.

21. Müller AC, Guido S. Einführung in Machine Learning mit Python: Praxiswissen Data Science. O'Reilly; 2017. p. 432

22. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. scikit-learn: Machine Learning in Python User Guide [Internet]. [cited 2021 Jun 14]. Available from: *https://scikit-learn.org/stable/user_guide.html*

23. Lahera G. Unbalanced Datasets & What To Do About Them [Internet]. 2019 [cited 2021 Jun 26]. Available from: *https://blog.strands.com/unbalanced-datasets*

24. Nguyen CN. Machine learning – kurz & gut : Eine Einführung mit Python, Pandas und Scikit-Learn. O'Reillys Taschenbibliothek; 2018.

25. Bobbitt Z. An Easy Guide to K-Fold Cross-Validation [Internet]. 2020 [cited 2021 May 25]. Available from: *https://www.statology.org/k-fold-cross-validation/*

26. Senapati D. Grid Search vs Random Search [Internet]. 2018. Available from: *https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318*

27. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res. 2011;12:2825–30.

28. Miyasato K. Classification Report: Precision, Recall, F1-Score, Accuracy [Internet]. 2020 [cited 2021 May 25]. Available from: *https://medium.com/@kennymiyasato/classification-report-precision-recall-f1-score-accuracy-16a245a437a5*

29. Jaeschke H, Gores GJ, Cederbaum AI, Hinson JA, Pessayre D, Lemasters JJ. Mechanisms of hepatotoxicity. Toxicol Sci [Internet]. 2002 Feb 1 [cited 2021 Jul 2];65(2):166–76. Available from: *https://academic.oup.com/toxsci/article/65/2/166/1634698*

30. Minerali E, Foil DH, Zorn KM, Lane TR, Ekins S. Comparing Machine Learning Algorithms for Predicting Drug-Induced Liver Injury (DILI). Mol Pharm. 2020;17(7):2628–37.

31. Liu A, Walter M, Wright P, Bartosik A, Dolciami D, Elbasir A, et al. Prediction and mechanistic analysis of drug-induced liver injury (DILI) based on chemical structure. Biol Direct. 2021;16(1).

32. Bishop CM. Pattern Recognition and Machine Learning. Reihe: Information Science and Statistics. Springer; 2006.

33. Kumar AM. C and Gamma in SVM [Internet]. 2018 [cited 2021 Jun 22]. Available from: *https://medium.com/@myselfaman12345/c-and-gamma-in-svm-e6cee48626be*

9 Appendix

All the Programming code can be found on the corresponding GitHub repository (*https://github.com/cptbern*) along with the excel-file containing the data used to generate the models.

9.1 Hyperparameter

9.1.1 Hyperparameter logReg

‘Solver’ – it determines the algorithm used to find the optimum for the objective function. It can take the values of ‘newton-cg’, ‘lbfgs’, ‘liblinear’. These solvers try to find the ideal weights for the different parameters to minimize a cost function. ‘Liblinear’ uses the *OvR* approach for multiclass problems, whereas ‘lbfgs’ and ‘newton-cg’ use a generalization of the logistic regression. The multinominal approach should generate better calibrated probability estimates than the *OvR* one, since it can be thaught the true multinomial logistic regression model. In a multinominal approach all the outcomes are compared against each other and not one against the others as in *OvR* (22).

‘Penalty’ – it can take values of ‘l1’, ‘l2’, ‘elasticnet’, ‘none’. It specifies the norm used as penalization. Two of the used solvers (‘newton-cg’ and ‘lbfgs’) only support ‘l2’, which is also the default penalty. Therefore, we only used ‘l2’ (22).

‘C’ – a parameter for the inverse regularization strength. If ‘C’ is big, the regularization is low and vice versa (22).

9.1.2 Hyperparameter DTI

‘Max_features’ – defines the numbers of features to consider when looking for the best split.

‘Min_samples_leaf’ – defines the minimum number of samples required to be at a leaf node. Splits will only be considered if after the split this condition is met at all the ensuing nodes (22).

‘Criterion’ – defines the function that measures the quality of the split. It can take two values: ‘gini’ and ‘entropy’. ‘gini’ stands for the measurement for ‘Gini impurity’ and ‘entropy’ for a measurement for information gain (22).

‘Max_depth’ – defines the maximum depth of the tree and can only be an ‘int’ value (22).

9.1.3 Hyperparameter RF

‘Max_leaf_nodes’ – defines the maximum number of leaves per tree (22).

‘Min_samples_split’ – defines the minimum number of samples that need to be present in a node in order to be able to split it (22).

‘Bootstrap’ – can be ‘True’ or ‘False’. It activates or deactivates bootstrap samples for the building of trees. In case of ‘False’ each tree is built using the whole dataset (22).

‘Criterion’, ‘min_samples_leaf’ and ‘max_features’ – same as for decision tree.

9.1.4 Hyperparameter SVM

‘Kernel’ – specifies the type of kernel function used. The kernel function is a function that shows how similar two inputs are to each other. It can take values of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or callable. Pre-computed uses the kernel matrix from data matrices, it was not

used since no such data matrix was made. Only the values ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’ were taken into account (22,33).

‘Gamma’ – is a coefficient for the kernels ‘rbf’, ‘poly’ and ‘sigmoid’. It can either be ‘scale’, ‘auto’ or a ‘float’ value. This hyperparameter defines how much curvature is in the decision boundary. A high value means more curvature and vice versa (22,33).

‘C’ – is a regularization parameter; higher ‘C’ corresponds with a higher regularization (22,33).

9.2 Programming code

9.2.1 Generating the Descriptors

The 2D descriptors were generated with this code.

9.2.1.1 Describtor_with_Errorcatch.py

```
import pandas as pd
from padelpy import from_smiles

#csv data is read into the program
CSV_Data = pd.read_csv('Daten/27_5_2021/2021_6_9_Database_removeddou-
bles_cleaned.csv', delimiter=';')#reads in csv file
#print(CSV_Data)

#the SMILE and the Filter (if a substance is included) are generated
SMILES = CSV_Data['SMILE (isomeric)']
Filter = CSV_Data['Include? Hepatotoxic']
row = 0
leng = CSV_Data.shape[0]
labels = CSV_Data['Substance Name'] #the names are saved separately
print(leng)
report = open('report_descriptors.txt','w') c
#all the reports are to automatically write the terminal output
print(f"Column {row}")
report.write("This is the output from the descriptor generation \n")

#The descriptors for the first substance are made to set the size of the
#DataFrame
try:
    #generating descriptors with PaDelPy
    descriptors = from_smiles(SMILES[0], timeout=2000, removesalt=True)
    #removesalt was altered in the functions.py of PaDelPy as follows
    '''
    def from_smiles(smiles, output_csv: str = None, descriptors: bool =
        True,fingerprints: bool = False, timeout: int = 60, removesalt =
        True) -> OrderedDict:
                ...
        for attempt in range(3):
        try:
            padeldescriptor(
                mol_dir='{}.smi'.format(timestamp),
                d_file=output_csv,
                convert3d=True,
                retain3d=True,
                d_2d=descriptors,
                d_3d=descriptors,
                fingerprints=fingerprints,
                sp_timeout=timeout,
                retainorder=True
                removesalt = True
```

```
            )
    '''
except:
    #If an error occures
    print(f"Exception row {row}!")
    report.write(f"Exception row {row}! \n")
res = pd.DataFrame([descriptors])

#New descriptors are added to the DataFrame
for row in range(1, leng):
    print(f"Column {row}")
    #report.write(f"Column {row} \n")
    try:
        descriptors = from_smiles(SMILES[row], timeout=2000, re-
movesalt=True)
    except:
        print(f"Exception col {row}!")
        report.write(f"Exception row {row}! \n")
    res1 = pd.DataFrame([descriptors])
    frame = [res, res1]
    res = pd.concat(frame, axis=0)
#The descriptors are saved after the index is redone using the substance
#names.
report.close()
res.index = labels[0:leng]
res.to_csv('Daten/Describtor_22_2_2021.csv', index = True, header = True)
```

9.2.1.2 Babel_GUI_New_Describtor.py

Whereas the 3D descriptors were made by putting each sdf-file into *OpenBabel* by itself to generate the csv-file with the Descriptors. The sdf-files were made using the following program.

```
import pandas as pd
from openbabel import pybel
from babelwork import Molecule

CSV_Data = pd.read_csv('Daten\Stand_22_2_2021.csv', delimiter=';')
#reads in csv file
#print(CSV_Data)
'''
Remove the not not included substances and reset the index. The empty val-
ues are filled with 'FALSE'.
The new cleaned output is saved in an additional csv-file.
'''
CSV_Data.drop(CSV_Data[CSV_Data['Include? Hepatotoxic']==False].index,
                       inplace= True)
CSV_Data.reset_index(drop=True)
CSV_Data.fillna("FALSE")
CSV_Data.to_csv('Describtoren\Cleaned_Output_22_2_2021_m.csv', index=False,
                header=True)
print (CSV_Data)
print(CSV_Data['SMILE (isomeric)'])

#save the SMILES in a seperate variable
SMILES = CSV_Data['SMILE (isomeric)']
row = 0
leng = CSV_Data.shape[0]
print(leng)

report = open('report_sdf_generation_molecule.txt','w')
```

```
#all the reports are to automatically write the terminal output

#print(f"Column {row}")
report.write("This is the output from the describtor generation \n")

'''
For each SMILE an sdf-file is generated.
'''
for row in range(1, leng):
    print(f"Column {row}")
    #report.write(f"Column {row} \n")
    try:
        # read in .smi file into obabel, generate .sdf (subprocess.run)
        m = Molecule(smiles=SMILES.iloc[row], smilestype='iso')
        a = pybel.readstring('smi', SMILES.iloc[row])
        a.make3D() #makes the structure 3-dimensional
        a.removeh() #removes the H from the molecule
        #m.routine()
        ''' the sdf-file is saved seperatly for each substance'''
        a.write('sdf', f'Describtoren\sdf\File_for_CDK_m_{row}.sdf',
                overwrite=True)
        print(f"done {row}!")

    except:
        print(f"Exception col {row}!")
        report.write(f"Exception row {row}! \n")

report.close()
```

9.2.2 Combining Descriptors and Outcomes

9.2.2.1 Combination_Outcome_Desc.py

The 2D and 3D datasets are combined with the outcome.

```
import pandas as pd

#reads in the different datasets and filters them
Outcomes = pd.read_csv('Daten/27_5_2021/'
                       '2021_6_9_Database_removeddoubles_cleaned.csv',
                       delimiter=';')
Outcomes = Outcomes[Outcomes['Include?Hepatotoxic'] != False]
#filtering the data
Desc_3D = pd.read_csv('Daten/Describtor3D_CDK_without_Output.csv',
                       delimiter=',')
print(Desc_3D.shape) #controlling the shape
Desc_2D = pd.read_csv('Daten/Describtor_22_2_2021.csv', delimiter=',')
print(Desc_2D.shape) #controlling the shape

#one after the other the outcomes are merged with the 2D and the 3D de-
scriptors
Combination = Outcomes.merge(Desc_2D, on='Substance Name')
Final = Combination.merge(Desc_3D,on='Substance Name')

#the final combination is saved
Final.to_csv('Daten/2021_6_9_Merged_Describtor_and_Outcome_PatClean.csv',
                index=True)
print(Final.shape) #controlling the shape of the newly generated csv-file
```

9.2.3 Feature Selection

9.2.3.1 Multilabel_Preprocess.py

The data is split and processed with SMOTE if needed. Here the six sets are made from the original data.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import sys

printintofiles = open(r'Daten/27_5_2021/report_Separation2.txt', 'w')

#loading cleaned descriptors
#all the descriptors are here
data = pd.read_csv('Daten/'
                   '2021_6_9_Merged_Describtor_and_Outcome_PatClean.csv'
                   ,delimiter=',', low_memory=False) #dataset used
data = data.drop('Title',axis='columns')

#spliting test and training

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

'''
The X and y for the Tox/nonTox Models
'''
#Until the 36th entry are the outcomes, therefore the descriptors come
#afterwards.
Describtors = data.iloc[:,36:] #Origin X for the sets

#To see if it is Hepatotoxic or not
Outcome_Hep = data['Hepatotoxic?'] #Origin y for the sets
print(data['Hepatotoxic?'].value_counts())

#check if anything is empty
print(Describtors.isnull().sum())
print(Outcome_Hep.isnull().sum())

#vaiable to define for the split
ts_size = 0.2 #20% of data is test, the other train
randomstate = 42

#only tox and non tox (save this split!)
X_train1, X_test1, y_train1, y_test1 = train_test_split(Describtors,
                                       Outcome_Hep, test_size=ts_size,
                                       random_state=randomstate,
                                       stratify=Outcome_Hep)
#0.2 means 20% are test data and 80% are training


#saving the split with pickle
with open('pickels/tox_nontox/train1_pickle_externalval','wb') as f:
    pickle.dump([X_train1, y_train1], f)

with open('pickels/tox_nontox/test1_pickle_externalval', 'wb') as f:
    pickle.dump([X_test1, y_test1], f)
```

```
'''
The X and y for Pattern-Models
'''

#used for different categorization
data = data[data['Include?Pattern'] != False]
#new filter for the pattern models
print(data.shape)

#how many instances per label are present
print(data['Hepatotoxic?'].value_counts())
print(data['Hepatocellular?'].value_counts())
print(data['Cholestatic?'].value_counts())
print(data['Mixed?'].value_counts())
class_names = ['Hepatotoxic?','Hepatocellular?','Cholestatic?','Mixed?']

#check if DatetimeIndex
Describtors = data.iloc[:,36:]
#Outcomes are separated
Outcome_Pat = data[['Hepatocellular?','Cholestatic?','Mixed?',
                    'Hepatotoxic?']] #Pattern models
Outcome_Hep = data['Hepatotoxic?'] #TOX2

#check if anything is empty
print(Describtors.isnull().sum())
print(Outcome_Pat.isnull().sum())
print(Outcome_Hep.isnull().sum())

Test = Outcome_Pat.groupby(['Hepatotoxic?','Hepatocellular?',
                            'Cholestatic?','Mixed?']).size
print(Test)

Grouped = Outcome_Pat.groupby(['Hepatotoxic?','Hepatocellular?',
                            'Cholestatic?',
                            'Mixed?']).size().reset_index()\
                            .rename(columns={0:'count'})
print(Grouped)
print(Grouped, file=printintofiles)
Grouped.plot.bar()
plt.show()

#powerlabel and decoder for powerlabel
Powerlabel = pd.DataFrame()
Powerlabel['powerlabel'] = Outcome_Pat.apply(lambda x :
                                             8*x['Mixed?']+
                                             4*x['Cholestatic?']+
                                             2*x['Hepatocellular?']+
                                             1*x['Hepatotoxic?'],axis=1)
print(' formula for the powerlabels: '
       '8*x[Mixed?]+4*x[Cholestatic?]+2*x[Hepatocellular?]+'
       '1*x[Hepatotoxic?]'
      , file=printintofiles)
print('meaning: '
      '\n 0 = nontox'
      '\n 3 = Tox, hepatocellular pattern'
      '\n 5 = Tox, cholestatic pattern'
      '\n 7 = Tox, cholestatic and hepatocellular patterns (not mixed)'
      '\n 9 = Tox, mixed pattern (not singular hepatocellular'
               'or cholestatic)'
      '\n 11= Tox, mixed and hepatocellular (no singular cholestatic)'
      '\n 13= Tox, mixed and cholestatic (no singular hepatocellular)'
      '\n 15= Tox, all patterns exist'
```

```
      'only existing ones where explained.', file=printintofiles)

print('the new powerlabels: ', file=printintofiles)
print(Powerlabel, file=printintofiles)
Powerlabel.plot(kind='hist',title='Histogram Of Powerlabel before SMOTE',)
plt.xlabel = 'Powerlabel Nr.'
plt.ylabel = 'Occurences in the Dataset'
plt.show()


#0.2 means 20% are test data and 80% are training
#all labels nox combination, split
X_train_pl, X_test_pl, y_train_pl, y_test_pl = train_test_split(
                                                Describtors,Powerlabel,
                                                test_size=ts_size,
                                                random_state=randomstate,
                                                stratify=Powerlabel)

#saving the split with pickle
#smote pickles
with open('pickels/pattern_Non_fuse/train2_pickle_externalval_S_'
        'pattern','wb') as f:
    pickle.dump([X_train_pl, y_train_pl], f)
with open('pickels/pattern_Non_fuse/test2_pickle_externalval_S_pattern',
        'wb') as f:
    pickle.dump([X_test_pl, y_test_pl], f)

#powerlabel non smote pickles
with open('pickels/pattern_Non_fuse/train2_pickle_externalval'
        '_nS_pattern','wb') as f:
    pickle.dump([X_train_pl, y_train_pl], f)
with open('pickels/pattern_Non_fuse/test2_pickle_externalval'
        '_nS_pattern', 'wb') as f:
    pickle.dump([X_test_pl, y_test_pl], f)

'''
combination of mixed and all label
'''
Powerlabel['powerlabel'] = Powerlabel['powerlabel'].replace([9],15)
print('powerlabel 9 and 15 were merged.')
print('the new powerlabels after merging: ', file=printintofiles)
print(Powerlabel, file=printintofiles)
Powerlabel.plot(kind='hist',title='Histogram Of Powerlabel before SMOTE',)
plt.xlabel = 'Powerlabel Nr.'
plt.ylabel = 'Occurences in the Dataset'
plt.show()

X_train2_pl, X_test2_pl, y_train2_pl, y_test2_pl = train_test_split(
                                                Describtors,Powerlabel,
                                                test_size=ts_size,
                                                random_state=randomstate,
                                                stratify=Powerlabel)


#saving the split with pickle
#smote pickles
with open('pickels/pattern_Non_fuse/train2_pickle_externalval'
        '_S_merged_pattern','wb') as f:
    pickle.dump([X_train2_pl, y_train2_pl], f)
with open('pickels/pattern_Non_fuse/test2_pickle_externalval'
        '_S_merged_pattern', 'wb') as f:
    pickle.dump([X_test2_pl, y_test2_pl], f)
```

```
#powerlabel non smote pickles
with open('pickels/pattern_Non_fuse/train2_pickle_externalval_'
        'nS_merged_pattern','wb') as f:
    pickle.dump([X_train2_pl, y_train2_pl], f)
with open('pickels/pattern_Non_fuse/test2_pickle_externalval'
        '_nS_merged_pattern', 'wb') as f:
    pickle.dump([X_test2_pl, y_test2_pl], f)


#to use the pickle at a later stage:
# with open('train.pickle', 'rb') as f:
    #X_train, y_train = pickle.load(f)

'''
Tox that lies within the patterns
to see if there are difficult to categorize substances within the only
Tox/nonTox part.
'''

X_train2, X_test2, y_train2, y_test2 = train_test_split(Describtors,
                                       Outcome_Hep, test_size=ts_size,
                                       random_state=randomstate,
                                       stratify=Outcome_Hep)
#0.2 means 20% are test data and 80% are training


#saving the split with pickle
with open('pickels/tox_nontox/train2_pickle_externalval'
        '_tox_patfilter','wb') as f:
    pickle.dump([X_train2, y_train2], f)

with open('pickels/tox_nontox/test2_pickle_externalval'
        '_tox_patfilter', 'wb') as f:
    pickle.dump([X_test2, y_test2], f)
```

9.2.3.2 All_in_One.py

One program to call upon the feature selection methods from 9.2.3.3, 9.2.3.4 and 9.2.3.5.

```
#Merge the different Featureselections together into one executable python.
import pandas as pd

from normalization import normalize
from Intercorrelation_and_low_variance_binary_search import LV_and_Intra-
corr
from Nearest_Neighbour import kNN_Impute
import pickle
import matplotlib.pyplot as plt
'''
loading the compacted dataset
containing both the outcomes and the descriptors
'''

#loading of the pickles
#tox/nontox
with open('pickels/tox_nontox/train1_pickle_externalval', 'rb') as f:
    X_train_tox, y_train_tox = pickle.load(f)
with open('pickels/tox_nontox/test1_pickle_externalval', 'rb') as f:
    X_test_tox, y_test_tox = pickle.load(f)
```

```
#tox/nontox filter pattern
with open('pickels/tox_nontox/train2_pickle_externalval_tox_patfilter',
'rb') as f:
    X_train_tox2, y_train_tox2 = pickle.load(f)
with open('pickels/tox_nontox/test2_pickle_externalval_tox_patfilter',
'rb') as f:
    X_test_tox2, y_test_tox2 = pickle.load(f)

#Smote pattern
with open('pickels/pattern_Non_fuse/train2_pickle_externalval_S_pattern',
'rb') as f:
    X_train_patS, y_train_patS = pickle.load(f)
with open('pickels/pattern_Non_fuse/test2_pickle_externalval_S_pattern',
'rb') as f:
    X_test_patS, y_test_patS = pickle.load(f)

#nonSmote pattern
with open('pickels/pattern_Non_fuse/train2_pickle_externalval_nS_pattern',
'rb') as f:
    X_train_pat, y_train_pat = pickle.load(f)
with open('pickels/pattern_Non_fuse/test2_pickle_externalval_nS_pattern',
'rb') as f:
    X_test_pat, y_test_pat = pickle.load(f)

#Smote pattern merged
with open('pickels/pattern_Non_fuse/train2_pickle_externalval_S_'
        'merged_pattern', 'rb') as f:
    X_train2_patS, y_train2_patS = pickle.load(f)
with open('pickels/pattern_Non_fuse/test2_pickle_externalval_S_'
        'merged_pattern', 'rb') as f:
    X_test2_patS, y_test2_patS = pickle.load(f)

#nonSmote pattern merged
with open('pickels/pattern_Non_fuse/train2_pickle_externalval_'
        'nS_merged_pattern', 'rb') as f:
    X_train2_pat, y_train2_pat = pickle.load(f)
with open('pickels/pattern_Non_fuse/test2_pickle_externalval_nS_merged'
        '_pattern', 'rb') as f:
    X_test2_pat, y_test2_pat = pickle.load(f)


print('Starting with normalization of given Data...')
#Norm = normalize(data,25)
X_train_tox,X_test_tox = normalize(25,X_train_tox,X_test_tox,'TOX')
X_train_tox2,X_test_tox2 = normalize(25,X_train_tox2,X_test_tox2,'TOX2')
X_train_patS,X_test_patS = normalize(25,X_train_patS,
                                     X_test_patS,'PAT_SMOTE')
X_train_pat,X_test_pat = normalize(25,X_train_pat,
                                     X_test_pat,'PAT_nonSMOTE')
X_train2_patS,X_test2_patS = normalize(25,X_train2_patS,
                                     X_test2_patS,'PAT2_SMOTE')
X_train2_pat,X_test2_pat = normalize(25,X_train2_pat,
                                     X_test2_pat,'PAT2_nonSMOTE')
print('Finished with normalization of given Data.')

print('Starting with removal of Low Variance and Intracorrelation of '
        'given Data...')
#LV_Intra = LV_and_Intracorr(Norm)
X_train_tox,X_test_tox = LV_and_Intra-
corr(X_train_tox,X_test_tox,900,400,'TOX')
X_train_tox2,X_test_tox2 =
```

```
LV_and_Intracorr(X_train_tox2,X_test_tox2,900,400,'TOX2')
X_train_patS,X_test_patS  = LV_and_Intra-
corr(X_train_patS,X_test_patS,900,400,'PAT_SMOTE')
X_train_pat,X_test_pat = LV_and_Intra-
corr(X_train_pat,X_test_pat,900,400,'PAT_nonSMOTE')
X_train2_patS,X_test2_patS= LV_and_Intra-
corr(X_train2_patS,X_test2_patS,900,400,'PAT2_SMOTE')
X_train2_pat,X_test2_pat = LV_and_Intra-
corr(X_train2_pat,X_test2_pat,900,400,'PAT2_nonSMOTE')


print('Finished with removal of Low Variance and Intracorrelation of given
Data.')

#Data and amount of neighbors
print('Starting with imputing of missing Data in given Data...')
#Impute = kNN_Impute(LV_Intra,5)
X_train_tox,X_test_tox = kNN_Impute(5,X_train_tox,
       y_train_tox,X_test_tox,y_test_tox,
       'pickels/tox_nontox/train1_pickle_externalval',
       'pickels/tox_nontox/test1_pickle_externalval','TOX')
X_train_tox2,X_test_tox2 = kNN_Impute(5,X_train_tox2,
       y_train_tox2,X_test_tox2,y_test_tox2,
       'pickels/tox_nontox/train2_pickle_externalval_tox_patfilter',
       'pickels/tox_nontox/test2_pickle_externalval_tox_patfilter','TOX2')
X_train_patS,X_test_patS = kNN_Impute(5,X_train_patS,
       y_train_patS,X_test_patS,y_test_patS,
       'pickels/pattern_Non_fuse/train2_pickle_externalval_S_pattern',
       'pickels/pattern_Non_fuse/test2_pickle_externalval_S_pattern',
       'PATS1')
X_train_pat,X_test_pat = kNN_Impute(5,X_train_pat,
       y_train_pat,X_test_pat,y_test_pat,
       'pickels/pattern_Non_fuse/train2_pickle_externalval_nS_pattern',
       'pickels/pattern_Non_fuse/test2_pickle_externalval_nS_pattern',
       'PAT1')
X_train2_patS,X_test2_patS = kNN_Impute(5,X_train2_patS,
       y_train2_patS,X_test2_patS,y_test2_patS,
       'pickels/pattern_Non_fuse/
       train2_pickle_externalval_S_merged_pattern',
       'pickels/pattern_Non_fuse/
       test2_pickle_externalval_S_merged_pattern','PATS2')
X_train2_pat,X_test2_pat = kNN_Impute(5,X_train2_pat,
       y_train2_pat,X_test2_pat,y_test2_pat,
       'pickels/pattern_Non_fuse/
       train2_pickle_externalval_nS_merged_pattern',
       'pickels/pattern_Non_fuse/
       test2_pickle_externalval_nS_merged_pattern','PAT2')
print('Finished with imputing of missing Data in given Data.')

#Data, size sample for spilt and number of thresholds
print("Feature Selection done: Normalization, Low Variance, '
       ' Intracorrelation and Imputing.")

'''
SMOTE for the 3 SMOTE classes
'''
#SMOTE! AFTER the Splitting and feature selection!!
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state=42)
X_sm1, y_sm1 = sm.fit_resample(X_train_tox, y_train_tox)
```

```
#saving the split with pickle
with open('pickels/tox_nontox/train1_pickle_externalval','wb') as f:
    pickle.dump([X_sm1, y_sm1], f)


#after changing multilabel into multiclass, smote can be used


X_train1_sm,y_train1_sm = sm.fit_resample(X_train_patS,y_train_patS)

y_train1_sm.plot(kind='hist',title='Histogram Of Powerlabel after SMOTE',)
plt.xlabel = 'Powerlabel Nr.'
plt.ylabel = 'Occurences in the Dataset'
plt.show()

#saving the split with pickle
#smote pickles
with open('pickels/pattern_Non_fuse/train2_pickle_externalval_'
        'S_pattern','wb') as f:
    pickle.dump([X_train1_sm, y_train1_sm], f)


X_train2_sm,y_train2_sm = sm.fit_resample(X_train2_patS,y_train2_patS)

y_train2_sm.plot(kind='hist',title='Histogram Of Powerlabel after SMOTE',)
plt.xlabel = 'Powerlabel Nr.'
plt.ylabel = 'Occurences in the Dataset'
plt.show()

#saving the split with pickle
#smote pickles
with open('pickels/pattern_Non_fuse/train2_pickle_externalval'
        '_S_merged_pattern','wb') as f:
    pickle.dump([X_train2_sm, y_train2_sm], f)


X_sm2, y_sm2 = sm.fit_resample(X_train_tox2, y_train_tox2)

#saving the split with pickle
with open('pickels/tox_nontox/train2_pickle_externalval'
        '_tox_patfilter','wb') as f:
    pickle.dump([X_sm2, y_sm2], f)
```

9.2.3.3 normalization.py

Normalization of the data using a Min-Max-Scaler

```
import pandas as pd
import numpy as np
import sys
from sklearn.preprocessing import MinMaxScaler

def normalize (NAthres, X_train, X_test,SAVE):

    print('calculating normalized values...')

    pd.set_option('display.max_rows', None)
    pd.set_option('display.max_columns', None)
    pd.set_option('display.width', None)
    #print(values.dtypes)
```

```
    #Using a Min-Max Scaler the data is normalized
    trans = MinMaxScaler()
    #infinite values are changed to NA-values
    #since the sclaer can't handle them.
    X_train2 = X_train.mask(np.isinf(X_train))
    X_test2 = X_test.mask(np.isinf(X_test))

    X_train_Norm = pd.DataFrame(trans.fit_transform(X_train2),
                                columns=X_train2.columns)
    X_test_Norm = pd.DataFrame(trans.transform(X_test2),
                                columns=X_test2.columns)
    print('X_train_Norm:', X_train_Norm.head())
    print('X_test: ', X_test_Norm.head())


    print('creating NA report...')
    printintofiles = open(f'Daten/27_5_2021/'
                      f'report_NAs_27_5_2021_{SAVE}.txt', 'w')

    print('NAs in rows: ', X_train_Norm.isnull().sum(axis=0),
                file=printintofiles)  # column <NA>
    print('NAs in coloums: ', X_train_Norm.isnull().sum(axis=1),
                file=printintofiles)
    print('NAs in general: ', X_train_Norm.isna().sum().sum(),
                file=printintofiles)
    print('droping pure NA columns...')
    no_NA = X_train_Norm.dropna(axis=1, how='all')

    # Remove columns with more than in Nathres defined NA-values
    print('droping Data with more than', NAthres, '% NA...')
    X_train_NA_Norm = no_NA.drop(
        no_NA.loc[:, list((100 * (no_NA.isnull().sum() / len(no_NA.index))
        >= NAthres))].columns, 1)
    X_test_NA_Norm = X_test_Norm[X_train_NA_Norm.columns]

    print(X_train_NA_Norm.head())
    print(X_test_NA_Norm.head())
    print('NAs in general: ', X_train_Norm.isna().sum().sum())

    print('shape of X_train after: ', X_train_NA_Norm.shape,
          'shape of X_train befor: ', X_train_NA_Norm.shape)
    print('shape of X_test after: ', X_test_NA_Norm.shape,
          'shape of X_test befor: ', X_test_NA_Norm.shape)

    return X_train_NA_Norm,X_test_NA_Norm
```

9.2.3.4 Intercorrelation_and_low_variance_binary_search.py

A binary search to find the thresholds needed to reduce the amount descriptors to the wanted number.

```
import pandas as pd
from sklearn.feature_selection import VarianceThreshold
import numpy as np
import sys
import pickle

def LV_and_Intracorr(X_train,X_test,dnum1,dnum2,SAVE):
    report = open(f'Daten/27_5_2021/'
```

```
            f'report_descriptors_threshholds_27_5_2021_{SAVE}.txt', 'w')
report.write("This are the used thresholdes: \n")

np.set_printoptions(threshold=sys.maxsize)

low = 0.007
high = 0.15
oldfnum = X_train.shape[1]
thisfnum = 0

# binary search for the optimal threshold to meet the requested
# Number of descriptors
while (dnum1 != thisfnum and oldfnum != thisfnum):
    variance_threshold = (low + high) / 2.0
    selection = VarianceThreshold(threshold=variance_threshold)
    new_values = selection.fit_transform(X_train)
    oldfnum = thisfnum
    thisfnum = new_values.shape[1]
    print(thisfnum, variance_threshold);
    if thisfnum < dnum1:
        high =variance_threshold
    else:
        low = variance_threshold


print(thisfnum)
used_threshold =variance_threshold
values_LV = X_train.loc[:,VarianceThreshold(threshold=
           used_threshold).fit(X_train).get_support()]
report.write('Used thresholde Low variance : ')
report.write(str(used_threshold))
report.write('\n new shape: ')
report.write(str(new_values.shape[1]))
print(new_values.shape[1])

#Intraclass correlation
def trimm_correlated(values_in, threshold):
    values_corr =values_in.corr(method='pearson', min_periods=1)
    values_not_correlated = ~(values_corr.mask
           (np.tril(np.ones([len(values_corr)]*2, dtype=bool))).abs()
           > threshold).any()
    un_corr_idx = values_not_correlated.loc
                  [values_not_correlated[values_not_correlated.index]
                  == True].index
    values_out = values_in[un_corr_idx]
    return values_out
def return_correlated(values_in, threshold):
    values_corr = values_in.corr(method='pearson', min_periods=1)
    values_not_correlated = ~(values_corr.mask
           (np.tril(np.ones([len(values_corr)] * 2,
           dtype=bool))).abs() > threshold).any()
    corr_idx = values_not_correlated.loc[values_not_correlated
           [values_not_correlated.index] == False].index
    values_out = values_in[corr_idx]
    return values_out


low = 0.9
high = 1.0
oldfnum = X_train.shape[1]
thisfnum = 0
```

```
    # binary search the same as bevor only for intracorrelation
    while (dnum2 != thisfnum and oldfnum != thisfnum):
        threshold = (low + high) / 2.0
        new_values = trimm_correlated(values_LV, threshold)
        correlated_factors = return_correlated(values_LV, threshold)
        oldfnum = thisfnum
        thisfnum = new_values.shape[1]
        print(thisfnum, threshold);
        if thisfnum > dnum2:
            high =threshold
        else:
            low = threshold
    report.write('Used thresholde Low variance : ')
    report.write(str(threshold))
    report.write('\n new shape: ')
    report.write(str(new_values.shape[1]))
    report.close()
    uncorrelated_factors = pd.DataFrame(new_values)
    print(new_values.shape[1])

    print ('The uncorrelated factors are saved... \n')
    print('The correlated Factors, that got removed are saved \n')

    #printing the removed factors
    correlated_factors.to_csv(f'Daten/27_5_2021/'
                      f'Descriptors_corr_norm_lv_9_4_2021_{SAVE}.csv',
                      index=True, header=True)
    #adding outcomes back
    test_values_uncorr = X_test[uncorrelated_factors.columns]
    print(uncorrelated_factors.head())
    print(test_values_uncorr.head())

    print('shape of X_train after: ',uncorrelated_factors.shape,
          'before: ', X_train.shape)
    print('shape of X_test after: ',test_values_uncorr.shape,
          'before: ', X_test.shape)

    return uncorrelated_factors,test_values_uncorr
```

9.2.3.5 Nearest_Neighbour.py

Used to impute the missing data values.

```
#Imputing the missing values through kNN (nearest Neighbour)
from sklearn.impute import KNNImputer
import pandas as pd
import numpy as np
import sys
import pickle


def kNN_Impute(Neighbours, X_train,y_train,X_test,y_test,
               SAVE_train,SAVE_test,Saveimg):

    print("printing column name where infinity is present")
    col_name = X_test.columns.to_series()[np.isinf(X_test).any()]
    print(col_name)

    #Imputing of the missing data
    imputer = KNNImputer(n_neighbors=Neighbours)
```

```
    X_train_imput = pd.DataFrame(imputer.fit_transform(X_train),
                                 columns=X_train.columns)
    X_test_imput = pd.DataFrame(imputer.transform(X_test),
                                columns=X_train.columns)

    #check if all the NA-values got imputed
    print("printing column name where infinity is present")
    col_name = X_test_imput.columns.to_series()[np.isinf
               (X_test_imput).any()]
    print(col_name)
    print("printing column name where Nan is present")
    col_name = X_test_imput.columns.to_series()[np.isnan
               (X_test_imput).any()]
    print(col_name)

    # saving the intracorrelation,lowvar removal for the set
    with open(f'{SAVE_train}', 'wb') as f:
        pickle.dump([X_train_imput, y_train], f)

    with open(f'{SAVE_test}', 'wb') as f:
        pickle.dump([X_test_imput, y_test], f)

    return X_train_imput,X_test_imput
```

9.2.4 GridsearchCV and RandomsearchCV

9.2.4.1 Models_MC_and_bin.py

All the models are called upon using this program. GridsearchCV as well as RandomsearchCV is incorporated.

```
'''
Here all the different models can be called up for the different x and y.
The models are methods in different python files
'''
import pandas as pd
import pickle
import sys
import numpy as np
from KNN_Base import kNN
from Gridsearch_SVM import RandSearch_SVM, GridSearch_SVM
from Gridsearch_logReg import RandSearch_logReg,GridSearch_Log
from Gridsearch_RF import Random_RF,GridSearch_RF
from Gridsearch_DT import RandSearch_DT,GridSearch_DT
from scipy.stats import randint

scoring = 'balanced_accuracy'
#accuracy as well as f1_macro didn't seem to work for it...

#loading of the pickles
#to use the pickle at a later stage:
#tox/nontox
with open('pickels/tox_nontox/train1_pickle_externalval', 'rb') as f:
    X_train_tox, y_train_tox = pickle.load(f)
with open('pickels/tox_nontox/test1_pickle_externalval', 'rb') as f:
    X_test_tox, y_test_tox = pickle.load(f)
#tox/nontox filter pattern
with open('pickels/tox_nontox/'
          'train2_pickle_externalval_tox_patfilter', 'rb') as f:
    X_train_tox2, y_train_tox2 = pickle.load(f)
```

```
with open('pickels/tox_nontox/'
          'test2_pickle_externalval_tox_patfilter', 'rb') as f:
    X_test_tox2, y_test_tox2 = pickle.load(f)

#Smote pattern
with open('pickels/pattern_Non_fuse/'
          'train2_pickle_externalval_S_pattern', 'rb') as f:
    X_train_patS, y_train_patS = pickle.load(f)
with open('pickels/pattern_Non_fuse/'
          'test2_pickle_externalval_S_pattern', 'rb') as f:
    X_test_patS, y_test_patS = pickle.load(f)

#nonSmote pattern
with open('pickels/pattern_Non_fuse/'
          'train2_pickle_externalval_nS_pattern', 'rb') as f:
    X_train_pat, y_train_pat = pickle.load(f)
with open('pickels/pattern_Non_fuse/'
          'test2_pickle_externalval_nS_pattern', 'rb') as f:
    X_test_pat, y_test_pat = pickle.load(f)

#Smote pattern merged
with open('pickels/pattern_Non_fuse/'
          'train2_pickle_externalval_S_merged_pattern', 'rb') as f:
    X_train2_patS, y_train2_patS = pickle.load(f)
with open('pickels/pattern_Non_fuse/'
          'test2_pickle_externalval_S_merged_pattern', 'rb') as f:
    X_test2_patS, y_test2_patS = pickle.load(f)

#nonSmote pattern merged
with open('pickels/pattern_Non_fuse/'
          'train2_pickle_externalval_nS_merged_pattern', 'rb') as f:
    X_train2_pat, y_train2_pat = pickle.load(f)
with open('pickels/pattern_Non_fuse/'
          'test2_pickle_externalval_nS_merged_pattern', 'rb') as f:
    X_test2_pat, y_test2_pat = pickle.load(f)

print('shapes after SMOTE: ',
      'Tox: ', X_train_tox.shape, X_test_tox.shape,'\n'
      'Tox2: ', X_train_tox2.shape,X_test_tox2.shape,'\n'
      'PatSMOTE: ', X_train_patS.shape,X_test_patS.shape,'\n'
      'PAT2Smote: ', X_train2_patS.shape,X_test2_patS.shape,'\n'
      'PatnSMOTE: ', X_train_pat.shape,X_test_pat.shape,'\n'
      'PAT2nSmote: ', X_train2_pat.shape,X_test2_pat.shape,'\n'
      )


#Randomsearch models
RandSearch_logReg(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
                  'Gridsearch/Random/Tox_Report_Randomsearch_logReg.txt')
RandSearch_logReg(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,
                  scoring,'Gridsearch/Random/'
                  'Tox_Report_Randomsearch_Patfilter_logReg.txt')
RandSearch_logReg(X_train_patS,y_train_patS,X_test_patS,y_test_patS,
                  scoring,'Gridsearch/Random/'
                  'SMOTE_Report_Randomsearch_logReg.txt')
RandSearch_logReg(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
                  'Gridsearch/Random/'
                  'NonSMOTE_Report_Randomsearch_logReg.txt')
RandSearch_logReg(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,
                  scoring,'Gridsearch/Random/'
                  'SMOTE_Report_Randomsearch_merged_logReg.txt')
RandSearch_logReg(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,
```

```
                    scoring,'Gridsearch/Random/'
                   'NonSMOTE_Report_Randomsearch_merged_logReg.txt')

RandSearch_DT(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
                'Gridsearch/Random/Tox_Report_Randomsearch_DT.txt')
RandSearch_DT(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,scoring,
                'Gridsearch/Random/Tox_Report_Randomsearch_Patfilter_DT.txt')
RandSearch_DT(X_train_patS,y_train_patS,X_test_patS,y_test_patS,scoring,
                'Gridsearch/Random/SMOTE_Report_Randomsearch_DT.txt')
RandSearch_DT(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
                'Gridsearch/Random/NonSMOTE_Report_Randomsearch_DT.txt')
RandSearch_DT(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,
                scoring,'Gridsearch/Random/'
                 'SMOTE_Report_Randomsearch_merged_DT.txt')
RandSearch_DT(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,scoring,
                'Gridsearch/Random/'
               'NonSMOTE_Report_Randomsearch_merged_DT.txt')

Random_RF(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
           'Gridsearch/Random/Tox_Report_Randomsearch_RF.txt')
Random_RF(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,scoring,
           'Gridsearch/Random/Tox_Report_Randomsearch_Patfilter_RF.txt')
Random_RF(X_train_patS,y_train_patS,X_test_patS,y_test_patS,scoring,
           'Gridsearch/Random/SMOTE_Report_Randomsearch_RF.txt')
Random_RF(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
           'Gridsearch/Random/NonSMOTE_Report_Randomsearch_RF.txt')
Random_RF(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,scoring,
           'Gridsearch/Random/SMOTE_Report_Randomsearch_merged_RF.txt')
Random_RF(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,scoring,
           'Gridsearch/Random/NonSMOTE_Report_Randomsearch_merged_RF.txt')

RandSearch_SVM(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
                'Gridsearch/Random/Tox_Report_Randomsearch_SVM.txt')
RandSearch_SVM(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,scoring,
                'Gridsearch/Random/'
                'Tox_Report_Randomsearch_Patfilter_SVM.txt')
RandSearch_SVM(X_train_patS,y_train_patS,X_test_patS,y_test_patS,scoring,
                'Gridsearch/Random/SMOTE_Report_Randomsearch_SVM.txt')
RandSearch_SVM(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
                'Gridsearch/Random/NonSMOTE_Report_Randomsearch_SVM.txt')
RandSearch_SVM(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,
                scoring,'Gridsearch/Random/'
                'SMOTE_Report_Randomsearch_merged_SVM.txt')
RandSearch_SVM(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,scoring,
                'Gridsearch/Random/'
                'NonSMOTE_Report_Randomsearch_merged_SVM.txt')


#LogReg models with GridsearchCV
params0 = {"solver": ['newton-cg','lbfgs','liblinear'],
           "penalty": ['l2'],
           "C": [100, 50,20,10,5,2, 1.0,0.5,0.2, 0.1,0.05,0.02, 0.01],
           "max_iter": [5000]}
GridSearch_Log(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
                'Gridsearch/Grid_avac/'
                'NEW2_Tox_Report_Gridsearch_logReg_param0_repcv_balac.txt'
                ,params0)
GridSearch_Log(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,scoring,
                'Gridsearch/Grid_avac/'
                'NEW2_Tox_Report_Gridsearch_'
                'Patfilter_logReg_param0_repcv_balac.txt'
                ,params0)
```

```
GridSearch_Log(X_train_patS,y_train_patS,X_test_patS,y_test_patS,scoring,
               'Gridsearch/Grid_avac/'
               'NEW2_SMOTE_Report_Gridsearch_logReg_param0_repcv_balac.txt'
               ,params0)
GridSearch_Log(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
               'Gridsearch/Grid_avac/'
               'NEW2_NonSMOTE_Report_'
               'Gridsearch_logReg_param0_repcv_balac.txt'
               ,params0)
GridSearch_Log(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,
               scoring,'Gridsearch/Grid_avac/NEW2_SMOTE'
               '_Report_Gridsearch_merged_logReg_param0_repcv_balac.txt'
               ,params0)
GridSearch_Log(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,scoring,
               'Gridsearch/Grid_avac/'
               'NEW2_NonSMOTE_Report_'
               'Gridsearch_merged_logReg_param0_repcv_balac.txt'
               ,params0)

#DTI models with GridsearchCV
params0 = {"max_features": ['auto', 'sqrt'],
              "min_samples_leaf": np.arange(2, 30,2),
              "criterion": ["gini", "entropy"],
            "max_depth": np.arange(2,7,1)}
#"max_leaf_nodes": np.arange(15,50,5)
GridSearch_DT(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_Tox_Report_Gridsearch'
              '_DT_param0_repcv_treeplot_balac.txt'
              ,params0,'TOX')
GridSearch_DT(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_Tox_Report_Gridsearch_Patfilter_'
               'DT_param0_repcv_treeplot_balac.txt'
              ,params0,'TOX2')
GridSearch_DT(X_train_patS,y_train_patS,X_test_patS,y_test_patS,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_SMOTE_Report_Gridsearch'
              '_DT_param0_repcv_treeplot_balac.txt'
              ,params0,'PAT_S')
GridSearch_DT(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_NonSMOTE_Report_Gridsearch'
              '_DT_param0_repcv_treeplot_balac.txt'
              ,params0,'PAT_nS')
GridSearch_DT(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,
             scoring,'Gridsearch/Grid_avac/'
              'NEW2_SMOTE_Report_Gridsearch'
              '_merged_DT_param0_repcv_treeplot_balac.txt'
              ,params0,'PAT2_S')
GridSearch_DT(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_NonSMOTE_Report_Gridsearch'
              '_merged_DT_param0_repcv_treeplot_balac.txt'
              ,params0,'PAT2_nS')

#RF models with Gridsearch
params0 = {"max_leaf_nodes": [35,40,45,None],
           "min_samples_split": np.arange(2, 6,2),
           "min_samples_leaf": np.arange(2, 6,2),
           "bootstrap": [True, False],
           "criterion": ["gini", "entropy"],
```

```
            "max_features": ['auto', 'sqrt']}
#"max_leaf_nodes": [35,40,45,None] maybe may depth if time ,
           #"max_depth": np.arange(2,8,1)
GridSearch_RF(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_Tox_Report_Gridsearch_'
              'RF_param0_repcv_balac.txt',params0)
GridSearch_RF(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_Tox_Report_Gridsearch_Patfilter_'
              'RF_param0_repcv_balac.txt',params0)
GridSearch_RF(X_train_patS,y_train_patS,X_test_patS,y_test_patS,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_SMOTE_Report_Gridsearch'
              '_RF_param0_repcv_balac.txt',params0)
GridSearch_RF(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_NonSMOTE_Report_Gridsearch'
              '_RF_param0_repcv_balac.txt',params0)
GridSearch_RF(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,
              scoring,'Gridsearch/Grid_avac/'
              'NEW2_SMOTE_Report_Gridsearch_'
              'merged_RF_param0_repcv_balac.txt',params0)
GridSearch_RF(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,scoring,
              'Gridsearch/Grid_avac/'
              'NEW2_NonSMOTE_Report_Gridsearch_'
              'merged_RF_param0_repcv_balac.txt',params0)

#SVM models with GridsearchCV
params0= {"gamma": [2,1,0.5,0.1,0.05,0.01,0.005,0.001],
            "kernel": ['rbf'],
            "C":  np.logspace(-3, 2, 15)}
#additional kernels: 'linear','poly','sigmoid'
#"C": [0.01,0.02,0.05,0.1,0.2,0.5,1,2,5,10,20,50,100,200,500,1000]
#"C":
[0.001,0.002,0.005,0.01,0.02,0.05,0.1,0.2,0.5,1,2,5,10,20,25,50,75,100]
#"gamma": [1,0.5,0.1,0.05,0.01,0.005,0.001]
#"C":  np.logspace(-9, 3, 13)

GridSearch_SVM(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
               'Gridsearch/Grid_avac/'
               'NEW2_Tox_Report_Gridsearch_SVM_param0__repcv_balac.txt'
               ,params0,'TOX')
GridSearch_SVM(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,scoring,
               'Gridsearch/Grid_avac/'
               'NEW2_Tox_Report_Gridsearch_Patfilter_'
               'SVM_param0__repcv_balac.txt'
               ,params0,'TOX2')
GridSearch_SVM(X_train_patS,y_train_patS,X_test_patS,y_test_patS,scoring,
               'Gridsearch/Grid_avac/'
               'NEW2_SMOTE_Report_Gridsearch_SVM_param0__repcv_balac.txt'
               ,params0,'PAT_S')
GridSearch_SVM(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
               'Gridsearch/Grid_avac/'
               'NEW2_NonSMOTE_Report_Gridsearch_'
               'SVM_param0__repcv_balac.txt'
               ,params0,'PAT_nS')
GridSearch_SVM(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,
               scoring,'Gridsearch/Grid_avac/'
               'NEW2_SMOTE_Report_Gridsearch'
               '_merged_SVM_param0_repcv_balac.txt'
               ,params0,'PAT2_S')
```

```
GridSearch_SVM(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,scoring,
               'Gridsearch/Grid_avac/'
               'NEW2_NonSMOTE_Report_Gridsearch_'
               'merged_SVM__repcv_balac.txt'
               ,params0,'PAT2_nS')

#kNN models with different k values
for i in (1,3,5,7,9,11):
    nei=i
    kNN(X_train_tox,y_train_tox,X_test_tox,y_test_tox,scoring,
        f'Gridsearch/Grid_avac/'
        f'NEW2_Tox_nonTox_Report_kNN{nei}.txt',
        f'Tox{nei}',nei)
    kNN(X_train_tox2,y_train_tox2,X_test_tox2,y_test_tox2,scoring,
        f'Gridsearch/Grid_avac/'
        f'NEW2_Tox_nonTox_Patt_Report_kNN{nei}.txt',
        f'Tox2{nei}',nei)
    kNN(X_train_patS,y_train_patS,X_test_patS,y_test_patS,scoring,
        f'Gridsearch/Grid_avac/'
        f'NEW2_Pattern_SMOTE_Report_kNN{nei}.txt',
        f'Pat_S{nei}',nei)
    kNN(X_train_pat,y_train_pat,X_test_pat,y_test_pat,scoring,
        f'Gridsearch/Grid_avac/'
        f'NEW2_Pattern_nonSMOTE_Report_kNN{nei}.txt',
        f'Pat_nS{nei}',nei)
    kNN(X_train2_patS,y_train2_patS,X_test2_patS,y_test2_patS,scoring,
        f'Gridsearch/Grid_avac/'
        f'NEW2_Pattern_SMOTE_merged_Report_kNN{nei}.txt',
        f'Pat_S_merged{nei}',nei)
    kNN(X_train2_pat,y_train2_pat,X_test2_pat,y_test2_pat,scoring,
        f'Gridsearch/Grid_avac/'
        f'NEW2_Pattern_NonSMOTE_merged_Report_kNN{nei}.txt',
        f'Pat_nS_merged{nei}',nei)
```

9.2.4.2 Gridsearch_logReg.py

The GridsearchCV and RandomsearchCV for the logistic Regression models.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV, RepeatedStratifiedKFold
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_validate
from scipy.stats import loguniform
from sklearn.metrics import classification_report,confusion_matrix, bal-
anced_accuracy_score


def RandSearch_logReg (X,y,X_test,y_test,SCORING,Savefile):
    printintofiles = open(Savefile, 'w')
    #neded to ravel y in order for it to work properly
    y = np.ravel(y)
    y_test = np.ravel(y_test)
    #define model:
    model = LogisticRegression()

    #repeated cv
    repeatcv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10)
```

```
    #define search space
    space = {"solver": ['newton-cg','lbfgs','liblinear'],
            "penalty": ['l2'],
            "C": [1000,500,200,100,50,20,10,5,2,1,0.5,0.2,0.1,0.05,0.02,
                  0.01,0.005,0.002,0.001,0.0005,0.0002,0.0001],
            "max_iter": [5000]}
    #the other penalties cause problems

    #define search
    search = RandomizedSearchCV(model,param_distributions=space,
                                scoring=SCORING, n_jobs=-1,
                                cv=repeatcv, random_state=1,
                                verbose=1, n_iter=100)

    #execute search
    result = search.fit(X,y)

    #summarize Results
    print('Best Score: %s' % result.best_score_)
    print('Best Hyperparameters: %s' % result.best_params_)

    print('Best Score: %s' % result.best_score_, file=printintofiles)
    print('Best Hyperparameters: %s' % result.best_params_,
          file=printintofiles)

    random_predict = result.predict(X_test)
    print(classification_report(y_test, random_predict))
    print(classification_report(y_test, random_predict),
          file=printintofiles)

    from sklearn.metrics import confusion_matrix
    cm1 = confusion_matrix(y_test, random_predict)
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict))
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict),
          file=printintofiles)
    print('confusion matrix: ', cm1, file=printintofiles)

def GridSearch_Log (X,y,X_test,y_test,SCORING,Savefile, paramgrid):
    printintofiles = open(Savefile, 'w')
    y = np.ravel(y)
    y_test = np.ravel(y_test)
    # define model:
    model = LogisticRegression()

    '''the search space is defined beforehand, this one is fitted around
    the parameters found for the randomsearch individually
    paramgrid is defined beforehand'''

    # repeated cv
    repeatcv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10)

    # define search
    search = GridSearchCV(model, param_grid=paramgrid,
                          scoring=SCORING, n_jobs=-1,
                          cv=repeatcv, verbose=1)

    # execute search
    result = search.fit(X, y)
```

```
    # summarize Results
    print('Best Score: %s' % result.best_score_)
    print('Best Hyperparameters: %s' % result.best_params_)

    print('Best Score: %s' % result.best_score_, file=printintofiles)
    print('Best Hyperparameters: %s' % result.best_params_,
          file=printintofiles)

    random_predict = result.predict(X_test)
    print(classification_report(y_test, random_predict))
    print(classification_report(y_test, random_predict),
          file=printintofiles)

    from sklearn.metrics import confusion_matrix
    cm1 = confusion_matrix(y_test, random_predict)
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict))
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict),
          file=printintofiles)
    print('confusion matrix: ', cm1, file=printintofiles)
```

9.2.4.3 *Gridsearch_DT.py*

The GridsearchCV and RandomsearchCV for the Decision tree induction models.

```
# Import necessary modules
from scipy.stats import randint
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV
import numpy as np
from sklearn.metrics import classification_report, balanced_accuracy_score
from sklearn.model_selection import GridSearchCV, RepeatedStratifiedKFold
import matplotlib.pyplot as plt

'''
Method to vizualize the different
Hyperparameter against each other
'''
def Plot3D(X,Y,Z,SAVEIMG):

   fig = plt.figure()
   plt.loglog(X, Y)
   ax = fig.gca(projection='3d')
   #lets see if the log here works as intended
   ax.plot_trisurf(Y, X, Z, cmap=plt.cm.viridis,
                   linewidth=0.2)
   ax.set_xlabel('max_depth')
   ax.set_ylabel('min_samples_leaf')
   ax.set_zlabel('mean-Score')

   ax.set_title('3D surface of DTI'
                f'\n for {SAVEIMG}')
   #plt.show()
   plt.savefig(f'Daten/Grafiken/Test/'
               f'3D_Plot_{SAVEIMG}_DT.png')

#Randomsearch definition
def RandSearch_DT(X, y,X_test,y_test, SCORING, Savefile):
```

```
    print('calculating randomsearch DT ...')
    printintofiles = open(Savefile, 'w')
    y = np.ravel(y)
    y_test = np.ravel(y_test)

    # repeated cv
    repeatcv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10)

    # Setup the parameters and distributions to sample from: param_dist
    param_dist ={"max_features": ['auto', 'sqrt'],
              "min_samples_leaf": np.arange(2, 55,2),
              "criterion": ["gini", "entropy"],
             "max_depth": np.arange(2, 12,1)}

    # Instantiate a Decision Tree classifier: tree
    tree = DecisionTreeClassifier()

    # Instantiate the RandomizedSearchCV object: tree_cv
    tree_cv = RandomizedSearchCV(tree, param_distributions=param_dist,
                                 cv=repeatcv,verbose=1,scoring=SCORING,
                                 n_jobs=-1,n_iter=100)

    # Fit it to the data
    tree_cv = tree_cv.fit(X,y)

    # Print the tuned parameters and score
    print('Best Score: %s' % tree_cv.best_score_)
    print('Best Hyperparameters: %s' % tree_cv.best_params_)
    print('Best Score: %s' % tree_cv.best_score_,
          file=printintofiles)
    print('Best Hyperparameters: %s' % tree_cv.best_params_,
          file=printintofiles)

    #evaluate the model
    random_predict = tree_cv.predict(X_test)
    print(classification_report(y_test, random_predict))
    print(classification_report(y_test, random_predict),
          file=printintofiles)

    from sklearn.metrics import confusion_matrix
    cm1 = confusion_matrix(y_test, random_predict)
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict))
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict),
          file=printintofiles)
    print('confusion matrix: ', cm1, file=printintofiles)

def GridSearch_DT(X, y,X_test,y_test, SCORING, Savefile,
paramgrid,Saveimg):
    from sklearn.metrics import classification_report
    print('calculating randomsearch DT ...')
    printintofiles = open(Savefile, 'w')
    y = np.ravel(y)
    y_test = np.ravel(y_test)

    # repeated cv
    repeatcv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10)

    # Instantiate a Decision Tree classifier: tree
    tree = DecisionTreeClassifier()
```

```
# Instantiate the RandomizedSearchCV object: tree_cv
tree_cv = GridSearchCV(tree, param_grid=paramgrid,
                       cv=repeatcv,verbose=1,
                       scoring=SCORING,n_jobs=-1)

# Fit it to the data
tree_cv = tree_cv.fit(X,y)

# Print the tuned parameters and score
print('Best Score: %s' % tree_cv.best_score_)
print('Best Hyperparameters: %s' % tree_cv.best_params_)
print('Best Score: %s' % tree_cv.best_score_,
      file=printintofiles)
print('Best Hyperparameters: %s' % tree_cv.best_params_,
      file=printintofiles)

random_predict = tree_cv.predict(X_test)
print(classification_report(y_test, random_predict))
print(classification_report(y_test, random_predict),
      file=printintofiles)
print('balanced accuracy: ',
      balanced_accuracy_score(y_test, random_predict))
print('balanced accuracy: ',
      balanced_accuracy_score(y_test, random_predict),
      file=printintofiles)


#print the best tree
from sklearn.tree import export_graphviz
import graphviz
from sklearn import tree
import matplotlib.pyplot as plt

#tree with the estimators generated through GridsearchCV
model_Hep = tree_cv.best_estimator_
# n_estimator stands for the number of trees used
model_Hep.fit(X, y)


from sklearn import tree
import matplotlib.pyplot as plt

#save the different names needed for the trees
featnames = X.columns.values
featnames_list = featnames.tolist()
clanames = np.unique(y)
clanames_list = str(clanames.tolist())
#print(clanames_list)
#print(featnames_list)
print('now the tree models are calculated')

#text representation of the tree
text_representation = tree.export_text(model_Hep,
                                       feature_names=featnames_list)
print(text_representation, file=printintofiles)
print(text_representation)

#printing the trees
#The first few splits of the tree
fig = plt.figure(figsize=(20, 15), dpi=300)
_ = tree.plot_tree(model_Hep, feature_names=featnames_list,
                   class_names=clanames_list, filled=True,
```

```
                    max_depth=2)
fig.savefig(f"Daten/Grafiken/Test/decision_tree_{Saveimg}.png")

#The whole tree
fig = plt.figure(figsize=(70, 50), dpi=300)
_ = tree.plot_tree(model_Hep, feature_names=featnames_list,
                   class_names=clanames_list, filled=True,
                   fontsize=10)
fig.savefig(f"Daten/Grafiken/"
            f"Test/decision_tree_no_max_depth_{Saveimg}.png")

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

#confusion matrix generation
cm1 = confusion_matrix(y_test, random_predict)

print('confusion matrix: ', cm1, file=printintofiles)


# Visualize it as a heatmap
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 7))
sns.heatmap(cm1, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.savefig(f'Daten/Grafiken/confusion_matrix_DT_{Saveimg}.tif')

import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import sys

# 3Dplot of the Gridsearchresults
np.set_printoptions(threshold=sys.maxsize)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)

results_dataframe = pd.DataFrame.from_dict(tree_cv.cv_results_)
# print (results_dataframe)
X_axis = results_dataframe['param_max_depth']
# print(X_axis)
Y_axis = results_dataframe['param_min_samples_leaf']
# print(Y_axis)
Z_axis = results_dataframe['mean_test_score']
# print(Z_axis)
Plot3D(X_axis, Y_axis, Z_axis, Saveimg)

# feature importance for the model
import pandas as pd

#print all the features with the importance
plt.figure(figsize=(10, 35))
n_features = X.shape[1]
plt.barh(range(n_features), model_Hep.feature_importances_,
         align='center')
plt.yticks(np.arange(n_features), featnames_list)
plt.xlabel('Importance of the feature')
plt.ylabel('Feature')
plt.savefig(f'Daten/Grafiken/Test/'
            f'Featureimportance_with0_{Saveimg}_DT.png')
```

```
    #remove the features without importance
    plt.figure(figsize=(10, 15))
    featureImp = model_Hep.feature_importances_
    featureImp_clean = pd.DataFrame({'name': featnames,
                                     'importance': featureImp},
                                    columns=['name', 'importance'])
    featureImp_clean = \
        featureImp_clean.loc[(featureImp_clean['importance'] != 0)]
    n_features = featureImp_clean.shape[0]
    plt.barh(range(n_features), featureImp_clean['importance'],
             align='center')
    plt.yticks(np.arange(n_features), featureImp_clean['name'])
    plt.xlabel('Importance of the feature')
    plt.ylabel('Feature')
    plt.savefig(f'Daten/Grafiken/Test/'
                f'Featureimportance_without0_{Saveimg}_DT.png')
```

9.2.4.4 Gridsearch_RF.py

The GridsearchCV and RandomsearchCV for the Random Forest models.

```
from sklearn.datasets import make_classification
from sklearn.feature_selection import RFECV
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV,Re-
peatedStratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from scipy.stats import randint as sp_randint
from sklearn.metrics import classification_report,confusion_matrix, bal-
anced_accuracy_score
import numpy as np
from sklearn.metrics import f1_score

#RandomsearchCV for RandomForest
def Random_RF(X,y,X_test,y_test,SCORING,Savefile):
    print('calculating randomsearch RF ...')
    printintofiles = open(Savefile, 'w')
    y = np.ravel(y)
    y_test = np.ravel(y_test)

    #Parameters
    grid = {"max_leaf_nodes": [20,25,30,35,40,45,None],
            "min_samples_split": np.arange(2, 10,2),
            "min_samples_leaf": np.arange(2, 10,2),
            "bootstrap": [True, False],
            "criterion": ["gini", "entropy"],
            "max_features": ['auto', 'sqrt']}

    estimator = RandomForestClassifier(n_estimators=500)

    #here an additonal RFECV could be implemented
    #It drastically increases computing time

    clf = RandomizedSearchCV(estimator, param_distributions=grid,
                             cv=5, scoring=SCORING,n_jobs=-1,
                             n_iter=100, verbose=1)
    clf.fit(X, y)

    #look at the hyperparameters and their score
    print('Best Score: %s' % clf.best_score_)
```

```
    print('Best Hyperparameters: %s' % clf.best_params_)
    print('Best Score: %s' % clf.best_score_, file=printintofiles)
    print('Best Hyperparameters: %s' % clf.best_params_,
          file=printintofiles)

    random_predict = clf.predict(X_test)

    #classification report as well as other values
    print(classification_report(y_test, random_predict))
    print(classification_report(y_test, random_predict),
          file=printintofiles)

    from sklearn.metrics import confusion_matrix
    cm1 = confusion_matrix(y_test, random_predict)
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict))
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict),
          file=printintofiles)
    print('confusion matrix: ', cm1, file=printintofiles)

def GridSearch_RF(X,y,X_test,y_test,SCORING,Savefile,paramgrid):
    print('calculating gridsearch RF ...')
    printintofiles = open(Savefile, 'w')
    y = np.ravel(y)
    y_test = np.ravel(y_test)

    estimator = RandomForestClassifier(n_estimators=500)

    # repeated cv
    repeatcv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10)

    clf = GridSearchCV(estimator, param_grid=paramgrid,
                       cv=repeatcv, scoring=SCORING,
                       n_jobs=-1, verbose=1)
    clf.fit(X, y)
    print('Best Score: %s' % clf.best_score_)
    print('Best Hyperparameters: %s' % clf.best_params_)
    print('Best Score: %s' % clf.best_score_, file=printintofiles)
    print('Best Hyperparameters: %s' % clf.best_params_,
          file=printintofiles)

    random_predict = clf.predict(X_test)

    #Scores and values achived by the model
    print(classification_report(y_test, random_predict))
    print(classification_report(y_test, random_predict),
          file=printintofiles)

    from sklearn.metrics import confusion_matrix
    cm1 = confusion_matrix(y_test, random_predict)
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict))
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict),
          file=printintofiles)
    print('confusion matrix: ', cm1, file=printintofiles)
```

9.2.4.5 Gridsearch_SVM.py

The GridsearchCV and RandomsearchCV for the Support Vector Machine models.

```
from sklearn.svm import SVC
import pandas as pd
import numpy as np
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV,Repeat-
edStratifiedKFold
from sklearn.feature_selection import RFECV
from sklearn.metrics import classification_report,confusion_ma-
trix,f1_score, balanced_accuracy_score
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import sys
import chart_studio.plotly as py
import plotly.graph_objects as go
import matplotlib.ticker as mticker


def Plot3D(X,Y,Z,SAVEIMG):
   #making a 3D plot for the Hyperparameters
   fig = plt.figure()
   #plt.loglog(X, Y)
   ax = fig.gca(projection='3d')
   #lets see if the log here works as intended
   ax.plot_trisurf(Y, X, Z, cmap=plt.cm.viridis, linewidth=0.2)
   ax.set_xlabel('C')
   ax.set_ylabel('Gamma')
   ax.set_zlabel('mean-Score')
   ax.set(title="Parameter overview")

   ax.set_title('3D surface of the Score from '
                'Gridsearch with kernel = rbf'
                f'\n for {SAVEIMG}')

   plt.savefig(f'Daten/Grafiken/3D_Plot_{SAVEIMG}.png')


def RandSearch_SVM(X, y,X_test,y_test, SCORING, Savefile):
    print('calculating randomsearch SVM ...')
    printintofiles = open(Savefile, 'w')
    y = np.ravel(y)
    y_test = np.ravel(y_test)

    # define model:
    estimator = SVC()

    # define evaluation
    selector = RFECV(estimator, step=1, cv=4)

    # define search space
    params = {"gamma": [1000,500,100,50,10,5,1,0.05,0.01,0.005,
                        0.001,0.0005,0.0001],
            "kernel":['rbf','linear','poly','sigmoid'],
            "C": np.logspace(-9, 3, 13)}

    # define search
    search = RandomizedSearchCV(estimator, param_distributions=params,
                                cv=5, scoring=SCORING,
```

```
                                n_jobs=-1, n_iter=100,verbose=1)

    # execute search
    result = search.fit(X, y)

    # summarize Results
    print('Best Score: %s' % result.best_score_)
    print('Best Hyperparameters: %s' % result.best_params_)

    print('Best Score: %s' % result.best_score_,
          file=printintofiles)
    print('Best Hyperparameters: %s' % result.best_params_,
          file=printintofiles)

    random_predict = result.predict(X_test)
    print(classification_report(y_test,random_predict))
    print(classification_report(y_test, random_predict),
          file=printintofiles)


    from sklearn.metrics import confusion_matrix
    cm1 = confusion_matrix(y_test, random_predict)
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict))
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict),
          file=printintofiles)
    print('confusion matrix: ', cm1, file=printintofiles)

def GridSearch_SVM(X, y,X_test,y_test, SCORING, Save-
file,paramgrid,SAVEIMG):
    print('calculating randomsearch SVM ...')
    printintofiles = open(Savefile, 'w')
    y = np.ravel(y)
    y_test = np.ravel(y_test)

    # repeated cv
    repeatcv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10)

    # define model:
    estimator = SVC()

    # define search
    search = GridSearchCV(estimator, param_grid=paramgrid,
                          cv=repeatcv, scoring=SCORING,
                          n_jobs=-1,verbose=1)

    # execute search
    result = search.fit(X, y)

    # summarize Results
    print('Best Score: %s' % result.best_score_)
    print('Best Hyperparameters: %s' % result.best_params_)

    print('Best Score: %s' % result.best_score_,
          file=printintofiles)
    print('Best Hyperparameters: %s' % result.best_params_,
          file=printintofiles)

    random_predict = result.predict(X_test)
    print(classification_report(y_test,random_predict))
    print(classification_report(y_test, random_predict),
```

```
          file=printintofiles)

    from sklearn.metrics import confusion_matrix
    cm1 = confusion_matrix(y_test, random_predict)
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict))
    print('balanced accuracy: ',
          balanced_accuracy_score(y_test, random_predict),
          file=printintofiles)
    print('confusion matrix: ',cm1, file=printintofiles)

    #3Dplot
    np.set_printoptions(threshold=sys.maxsize)
    pd.set_option('display.max_rows', None)
    pd.set_option('display.max_columns', None)
    pd.set_option('display.width', None)

    results_dataframe = pd.DataFrame.from_dict(result.cv_results_)
    #print (results_dataframe)
    X_axis = results_dataframe['param_C']
    #print(X_axis)
    Y_axis = results_dataframe['param_gamma']
    #print(Y_axis)
    Z_axis = results_dataframe['mean_test_score']
    #print(Z_axis)
    Plot3D(X_axis,Y_axis,Z_axis,SAVEIMG)
```

9.2.4.6 KNN_Base.py

This is the code used to find the best descriptors for the k nearest neighbor models.

```
#basic kNN to be called up from Models_MC_and_bin.py
import numpy as np
import pandas as pd
import sys
from sklearn.metrics import classification_report,confusion_matrix, bal-
anced_accuracy_score
from sklearn.model_selection import GridSearchCV, RepeatedStratifiedKFold

#variables: X,y, adittional info, name of the file for saving
def kNN (X_train, y_train, X_test, y_test, SCORING ,Saveplace,
Saveimg,neighbournr):
    print('calculating kNN model...')
    import matplotlib.pyplot as plt
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import cross_validate

    #set the display options to see all the data
    np.set_printoptions(threshold=sys.maxsize)
    pd.set_option('display.max_rows', None)
    pd.set_option('display.max_columns', None)
    pd.set_option('display.width', None)

    # repeated cv
    repeatcv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10)

    y_train = np.ravel(y_train)
    y_test = np.ravel(y_test)
```

```
leng = X_train.shape[1]
print(leng)
printintofiles = open(Saveplace, 'w')

#initiating different arrays
Score = 0
list_best_Desc = []
list_best_Desc_names = []
# search for the first best neighbour
for i in range(0, leng):
    KNN = KNeighborsClassifier(n_neighbors=neighbournr)
    #get the name of the i-th column
    colname = X_train.columns.values[i]
    #make a new X_train with only one variable
    dummy_X = X_train[[colname]]
    KNN.fit(dummy_X, y_train)
    cv = cross_validate(KNN, dummy_X, y_train,
                        scoring=SCORING, cv=repeatcv,n_jobs=-1)

    #save the score achived and compare it to the others
    zwischenspeicher = cv['test_score'].mean()
    if zwischenspeicher > Score:
        #only remember the highest value
        Score = cv['test_score'].mean()
        high = i
        # first one should always go through here
    else:
        continue
Scorelist = []
Scorelist.append(Score)
#List of names of the describtors
list_best_Desc_names.append(X_train.columns.values[high])
#the number of the describtor
list_best_Desc.append(high)
print(list_best_Desc)
print(list_best_Desc_names)
print(type(list_best_Desc_names))

namelist = list_best_Desc_names
print('first feature found, searching for additional features...')
# search for the next 30 neighbours
for j in range(1, 30):
    # reset of the variable score
    Score = 0

    print(namelist)
    print(type(namelist))
    print(f'searching for the {j} additional neighbour...')
    for i in range(0, leng):
        #see if i is not already used
        if i not in list_best_Desc:
            KNN = KNeighborsClassifier(n_neighbors=neighbournr,
                                       n_jobs=-1)
            colname = namelist
            #add the name to the selected ones
            colname.append( X_train.columns.values[i])
            #multiple columns are now present in dummy_X
            dummy_X = X_train[colname]
            KNN.fit(dummy_X, y_train)
            cv = cross_validate(KNN, dummy_X, y_train,
                                scoring=SCORING, cv=repeatcv,
                                n_jobs=-1)
```

```
                zwischenspeicher = cv['test_score'].mean()

                #check the score and compare
                if zwischenspeicher > Score:
                    Score = cv['test_score'].mean()
                    high = i

                    #remove the added name,
                    # otherwise it will stay
                    namelist.pop()
                else:
                    namelist.pop()
                    continue

            else:
                continue
        list_best_Desc_names.append(X_train.columns.values[high])
        list_best_Desc.append(high)
        print('used describtors: ', list_best_Desc)
        print('used describtors: ', list_best_Desc_names)
        Scorelist.append(Score)

        #save the best combination and continue
        namelist = list_best_Desc_names

    # list of the best describtors in order of value
    # the highest of all scores is taken
    # that may mean less than 30 descriptors
    maxvalue = max(Scorelist)
    print('maxvalue = ', maxvalue, file=printintofiles)
    print('used describtors: ', list_best_Desc,
          file=printintofiles)
    print('used describtornames: ', list_best_Desc_names,
          file=printintofiles)
    print('The different scores: ', Scorelist, file=printintofiles)
    print('number of neighbours: ', neighbournr)

    maxpos = Scorelist.index(maxvalue)
    print('maxvalue = ', maxvalue,
          '\n max value position equal to number of features -1 = ',
           maxpos)

    #make a model with the highest score
    Namelist = list_best_Desc_names[0:maxpos]
    Neig = KNeighborsClassifier(n_neighbors=neighbournr,
                                n_jobs=-1)
    X_train_DT = X_train[Namelist]
    X_test_DT = X_test[Namelist]
    Neig.fit(X_train_DT, y_train)

    from sklearn.metrics import confusion_matrix
    #test the model and save the values
    y_predicted1 = Neig.predict(X_test_DT)
    cm1 = confusion_matrix(y_test, y_predicted1)

    sensitivity = cm1[0, 0] / (cm1[0, 0] + cm1[0, 1])
    specificity = cm1[1, 1] / (cm1[1, 0] + cm1[1, 1])

    print(cm1, file=printintofiles)
    print('Sensitivity : ', sensitivity, file=printintofiles)
    print('Specificity : ', specificity, file=printintofiles)
    print(classification_report(y_test, y_predicted1))
```

```
print(classification_report(y_test, y_predicted1),
      file=printintofiles)
print('balanced accuracy: ',
      balanced_accuracy_score(y_test, y_predicted1))
print('balanced accuracy: ',
      balanced_accuracy_score(y_test, y_predicted1),
      file=printintofiles)
print('confusion matrix: ', cm1, file=printintofiles)

# Visualize it as a heatmap
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 7))
sns.heatmap(cm1, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.savefig(f'Daten/kNN/confusion_matrix_kNN_{Saveimg}.png')

#distribution for the first 2 features vizualized
from matplotlib.colors import ListedColormap

#for simplicity only the first three are used.
Namelist = list_best_Desc_names[0:2]
Namelist2 = list_best_Desc_names[1:3]
Neig = KNeighborsClassifier(n_neighbors=neighbournr,
                            n_jobs=-1)
X_train_DT = X_train[Namelist] #Dataframe
X_train2_DT = X_train[Namelist2]
#print(X_train_DT)
h = 0.05

Neig.fit(X_train_DT, y_train)
x_min = 0
x_max = 1
y_min = 0
y_max = 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
Z = Neig.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z)

# Plot also the training points
sns.scatterplot(x=X_train_DT.iloc[:, 0],
                y=X_train_DT.iloc[:, 1],
                hue=y_train, alpha=1.0,
                edgecolor="black")
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("Class classification kNN for 2 features")
plt.xlabel(list_best_Desc_names[0])
plt.ylabel(list_best_Desc_names[1])

plt.savefig(f'Daten/kNN/'
            f'2BestFeatures_kNN_{Saveimg}_'
            f'neigh{neighbournr}.png')

Neig.fit(X_train2_DT, y_train)
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
```

```
                        np.arange(y_min, y_max, h))
    Z = Neig.predict(np.c_[xx.ravel(), yy.ravel()])

    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.figure(figsize=(8, 6))
    plt.contourf(xx, yy, Z)

    # Plot also the training points
    sns.scatterplot(x=X_train_DT.iloc[:, 0],
                    y=X_train_DT.iloc[:, 1],
                    hue=y_train, alpha=1.0,
                    edgecolor="black")
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("Class classification kNN for 2 features")
    plt.xlabel(list_best_Desc_names[1])
    plt.ylabel(list_best_Desc_names[2])

    plt.savefig(f'Daten/kNN/2BestFeatures_kNN_{Saveimg}_'
                f'2_neigh{neighbournr}.png')
```

9.3 Classification Reports

shapes after SMOTE:

- Tox: (1186, 400) (245, 400)
- Tox2: (772, 400) (188, 400)
- PatSMOTE: (3088, 399) (188, 399)
- PAT2Smote: (2702, 400) (188, 400)
- PatnSMOTE: (752, 399) (188, 399)
- PAT2nSmote: (752, 400) (188, 400)

The thresholds used for the feature selection:

TOX:

This are the used thresholds:
Used threshold Low variance: 0.00863214111328125
New shape: 901
Used threshold Low variance: 0.91328125
New shape: 400

TOX2:

This are the used thresholds:
Used threshold Low variance: 0.0130921630859375
New shape: 900
Used threshold Low variance: 0.9281250000000001
New shape: 400

PAT_S:

This are the used thresholds:
Used threshold Low variance: 0.013185989379882815
New shape: 900
Used threshold Low variance: 0.9324218750000001
New shape: 399

PAT_nS:

This are the used thresholds:
Used threshold Low variance: 0.013185989379882815
New shape: 900
Used threshold Low variance: 0.9324218750000001
New shape: 399

PAT2_S:

This are the used thresholds:
Used threshold Low variance: 0.012991790771484376
New shape: 900
Used threshold Low variance: 0.9265625000000001
New shape: 400

PAT2_nS:

This are the used thresholds:
Used threshold Low variance: 0.012991790771484376
New shape: 900
Used threshold Low variance: 0.9265625000000001
New shape: 400

9.3.1 LogReg

9.3.1.1 TOX

Fitting 50 folds for each of 39 candidates, totaling 1950 fits

Best Score: 0.6775003560746333

Best Hyperparameters: {'C': 100, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'liblinear'}

	precision	recall	f1-score	support
False	0.53	0.53	0.53	97
True	0.69	0.70	0.69	148
accuracy			0.63	245
macro avg	0.61	0.61	0.61	245
weighted avg	0.63	0.63	0.63	245

balanced accuracy: 0.6108595709111173

9.3.1.2 TOX2

Fitting 50 folds for each of 39 candidates, totaling 1950 fits

Best Score: 0.6560705960705961

Best Hyperparameters: {'C': 0.1, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'lbfgs'}

	precision	recall	f1-score	support
False	0.67	0.62	0.65	97
True	0.63	0.68	0.65	91
accuracy			0.65	188
macro avg	0.65	0.65	0.65	188
weighted avg	0.65	0.65	0.65	188

balanced accuracy: 0.6499376911748046

9.3.1.3 PAT_S

Fitting 50 folds for each of 39 candidates, totaling 1950 fits

Best Score: 0.9255261405261406

Best Hyperparameters: {'C': 100, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'newton-cg'}

	precision	recall	f1-score	support
0	0.64	0.58	0.61	97
3	0.19	0.22	0.20	27
5	0.12	0.09	0.11	11
7	0.29	0.29	0.29	14
9	0.00	0.00	0.00	2
11	0.10	0.11	0.11	9
13	0.12	0.14	0.13	7
15	0.37	0.48	0.42	21
accuracy			0.42	188
macro avg	0.23	0.24	0.23	188
weighted avg	0.44	0.42	0.43	188

balanced accuracy: 0.2382904895791494

9.3.1.4 PAT_nS

Fitting 50 folds for each of 39 candidates, totaling 1950 fits

Best Score: 0.2249630279850868

Best Hyperparameters: {'C': 10, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'liblinear'}

	precision	recall	f1-score	support
0	0.61	0.72	0.66	97
3	0.20	0.19	0.19	27
5	0.17	0.09	0.12	11
7	0.25	0.14	0.18	14
9	0.00	0.00	0.00	2
11	0.17	0.11	0.13	9
13	0.00	0.00	0.00	7
15	0.43	0.43	0.43	21
accuracy			0.47	188
macro avg	0.23	0.21	0.21	188
weighted avg	0.43	0.47	0.44	188

balanced accuracy: 0.21003543039625516

9.3.1.5 PAT2_S

Fitting 50 folds for each of 39 candidates, totaling 1950 fits

Best Score: 0.9090005232862377

Best Hyperparameters: {'C': 100, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'liblinear'}

	precision	recall	f1-score	support
0	0.68	0.61	0.64	97
3	0.17	0.19	0.18	27
5	0.00	0.00	0.00	11
7	0.21	0.21	0.21	14
11	0.11	0.11	0.11	9
13	0.12	0.14	0.13	7
15	0.31	0.43	0.36	23
accuracy			0.42	188
macro avg	0.23	0.24	0.23	188
weighted avg	0.44	0.42	0.43	188

balanced accuracy: 0.24235274068788826

9.3.1.6 PAT2_nS

Fitting 50 folds for each of 39 candidates, totaling 1950 fits

Best Score: 0.2644320842065203

Best Hyperparameters: {'C': 10, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'liblinear'}

	precision	recall	f1-score	support
0	0.61	0.72	0.66	97
3	0.32	0.30	0.31	27
5	0.17	0.09	0.12	11
7	0.18	0.14	0.16	14
11	0.25	0.11	0.15	9
13	0.00	0.00	0.00	7
15	0.29	0.30	0.30	23
accuracy			0.47	188
macro avg	0.26	0.24	0.24	188
weighted avg	0.43	0.47	0.45	188

balanced accuracy: 0.23816727882809716

9.3.2 DTI

9.3.2.1 TOX

calculating randomsearch DT ...

Fitting 50 folds for each of 280 candidates, totaling 14000 fits

Best Score: 0.6329333428286569

Best Hyperparameters: {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 6}

	precision	recall	f1-score	support
False	0.53	0.68	0.60	97
True	0.74	0.61	0.67	148
accuracy			0.64	245
macro avg	0.64	0.64	0.63	245
weighted avg	0.66	0.64	0.64	245

balanced accuracy: 0.6442602396210644

9.3.2.2 TOX2

calculating randomsearch DT ...

Fitting 50 folds for each of 280 candidates, totaling 14000 fits

Best Score: 0.6025591075591076

Best Hyperparameters: {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2}

	precision	recall	f1-score	support
False	0.56	0.78	0.66	97
True	0.60	0.35	0.44	91
accuracy			0.57	188
macro avg	0.58	0.57	0.55	188
weighted avg	0.58	0.57	0.55	188

balanced accuracy: 0.5675767531437634

9.3.2.3 PAT_S

calculating randomsearch DT ...

Fitting 50 folds for each of 280 candidates, totaling 14000 fits

Best Score: 0.521055611055611

Best Hyperparameters: {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2}

	precision	recall	f1-score	support
0	0.58	0.33	0.42	97
3	0.18	0.22	0.20	27
5	0.11	0.18	0.13	11
7	0.14	0.29	0.19	14
9	0.09	0.50	0.15	2
11	0.00	0.00	0.00	9
13	0.04	0.14	0.06	7
15	0.38	0.14	0.21	21
accuracy			0.26	188
macro avg	0.19	0.23	0.17	188
weighted avg	0.39	0.26	0.29	188

balanced accuracy: 0.22567073533568377

9.3.2.4 PAT_nS

calculating randomsearch DT ...
Fitting 50 folds for each of 280 candidates, totaling 14000 fits

Best Score: 0.16135589084118493

Best Hyperparameters: {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 8}

	precision	recall	f1-score	support
0	0.54	0.88	0.67	97
3	0.20	0.07	0.11	27
5	0.00	0.00	0.00	11
7	0.00	0.00	0.00	14
9	0.00	0.00	0.00	2
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.29	0.24	0.26	21
accuracy			0.49	188
macro avg	0.13	0.15	0.13	188
weighted avg	0.34	0.49	0.39	188

balanced accuracy: 0.14855724649539082

9.3.2.5 PAT2_S

calculating randomsearch DT ...
Fitting 50 folds for each of 280 candidates, totaling 14000 fits

Best Score: 0.4760782075067789

Best Hyperparameters: {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 6}

	precision	recall	f1-score	support
0	0.62	0.24	0.34	97
3	0.14	0.15	0.14	27
5	0.25	0.27	0.26	11
7	0.08	0.29	0.12	14
11	0.10	0.11	0.11	9
13	0.13	0.43	0.20	7
15	0.07	0.09	0.08	23
accuracy			0.21	188
macro avg	0.20	0.22	0.18	188
weighted avg	0.38	0.21	0.24	188

balanced accuracy: 0.22433459572474748

9.3.2.6 PAT2_nS

calculating randomsearch DT ...
Fitting 50 folds for each of 280 candidates, totaling 14000 fits

Best Score: 0.1814718097274488

Best Hyperparameters: {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2}

	precision	recall	f1-score	support
0	0.55	0.90	0.68	97
3	0.17	0.04	0.06	27
5	0.12	0.09	0.11	11
7	0.25	0.07	0.11	14
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.29	0.09	0.13	23
accuracy			0.49	188
macro avg	0.20	0.17	0.16	188
weighted avg	0.37	0.49	0.39	188

balanced accuracy: 0.16903406251552502

9.3.3 RF

9.3.3.1 TOX

calculating randomsearch RF ...
Fitting 50 folds for each of 128 candidates, totaling 6400 fits

Best Score: 0.7794957983193278

Best Hyperparameters: {'bootstrap': False, 'criterion': 'entropy', 'max_features': 'sqrt', 'max_leaf_nodes': None, 'min_samples_leaf': 4, 'min_samples_split': 4}

	precision	recall	f1-score	support
False	0.66	0.63	0.65	97
True	0.76	0.79	0.78	148
accuracy			0.73	245
macro avg	0.71	0.71	0.71	245
weighted avg	0.72	0.73	0.73	245

balanced accuracy: 0.7097032599609919

9.3.3.2 TOX2

calculating randomsearch RF ...
Fitting 50 folds for each of 128 candidates, totaling 6400 fits

Best Score: 0.6976973026973028

Best Hyperparameters: {'bootstrap': False, 'criterion': 'gini', 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_leaf': 4, 'min_samples_split': 4}

	precision	recall	f1-score	support
False	0.76	0.70	0.73	97
True	0.70	0.76	0.73	91
accuracy			0.73	188
macro avg	0.73	0.73	0.73	188
weighted avg	0.73	0.73	0.73	188

balanced accuracy: 0.7296363430384049

9.3.3.3 PAT_S

calculating randomsearch RF ...
Fitting 50 folds for each of 128 candidates, totaling 6400 fits

Best Score: 0.9544671994671994

Best Hyperparameters: {'bootstrap': False, 'criterion': 'entropy', 'max_features': 'sqrt', 'max_leaf_nodes': None, 'min_samples_leaf': 2, 'min_samples_split': 2}

	precision	recall	f1-score	support
0	0.61	0.80	0.70	97
3	0.16	0.11	0.13	27
5	0.33	0.18	0.24	11
7	0.42	0.36	0.38	14
9	0.00	0.00	0.00	2
11	0.14	0.11	0.12	9
13	0.25	0.14	0.18	7
15	0.50	0.29	0.36	21
accuracy			0.51	188
macro avg	0.30	0.25	0.26	188
weighted avg	0.46	0.51	0.47	188

balanced accuracy: 0.249234800136862

9.3.3.4 PAT_nS

calculating randomsearch RF ...
Fitting 50 folds for each of 128 candidates, totaling 6400 fits

Best Score: 0.21828760251186718

Best Hyperparameters: {'bootstrap': False, 'criterion': 'gini', 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_leaf': 2, 'min_samples_split': 4}

	precision	recall	f1-score	support
0	0.56	0.91	0.70	97
3	0.23	0.11	0.15	27
5	0.50	0.09	0.15	11
7	0.67	0.29	0.40	14
9	0.00	0.00	0.00	2
11	0.33	0.11	0.17	9
13	0.33	0.14	0.20	7
15	0.60	0.14	0.23	21
accuracy			0.54	188
macro avg	0.40	0.22	0.25	188
weighted avg	0.50	0.54	0.46	188

balanced accuracy: 0.22397204742565566

9.3.3.5 PAT2_S

calculating randomsearch RF ...
Fitting 50 folds for each of 128 candidates, totaling 6400 fits

Best Score: 0.9425193853765282

Best Hyperparameters: {'bootstrap': False, 'criterion': 'gini', 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_leaf': 2, 'min_samples_split': 4}

	precision	recall	f1-score	support
0	0.63	0.85	0.72	97
3	0.25	0.15	0.19	27
5	0.20	0.09	0.13	11
7	0.43	0.21	0.29	14
11	0.10	0.11	0.11	9
13	0.17	0.14	0.15	7
15	0.38	0.22	0.28	23
accuracy			0.52	188
macro avg	0.31	0.25	0.26	188
weighted avg	0.46	0.52	0.47	188

balanced accuracy: 0.25286619091447166

9.3.3.6 PAT2_nS

calculating randomsearch RF ...
Fitting 50 folds for each of 128 candidates, totaling 6400 fits

Best Score: 0.24379127639278014

Best Hyperparameters: {'bootstrap': False, 'criterion': 'gini', 'max_features': 'sqrt', 'max_leaf_nodes': None, 'min_samples_leaf': 2, 'min_samples_split': 2}

	precision	recall	f1-score	support
0	0.57	0.92	0.70	97
3	0.20	0.07	0.11	27
5	0.00	0.00	0.00	11
7	1.00	0.07	0.13	14
11	0.33	0.11	0.17	9
13	0.20	0.14	0.17	7
15	0.56	0.22	0.31	23
accuracy			0.53	188
macro avg	0.41	0.22	0.23	188
weighted avg	0.49	0.53	0.44	188

balanced accuracy: 0.2191982824306574

9.3.4 SVM

9.3.4.1 TOX

calculating randomsearch SVM ...
Fitting 50 folds for each of 120 candidates, totaling 6000 fits

Best Score: 0.79686867967526

Best Hyperparameters: {'C': 1.6378937069540647, 'gamma': 0.5, 'kernel': 'rbf'}

	precision	recall	f1-score	support
False	0.63	0.40	0.49	97
True	0.68	0.84	0.76	148
accuracy			0.67	245
macro avg	0.66	0.62	0.62	245
weighted avg	0.66	0.67	0.65	245

balanced accuracy: 0.6233282251323489

9.3.4.2 TOX2

calculating randomsearch SVM ...
Fitting 50 folds for each of 120 candidates, totaling 6000 fits

Best Score: 0.6985581085581086

Best Hyperparameters: {'C': 1.6378937069540647, 'gamma': 0.5, 'kernel': 'rbf'}

	precision	recall	f1-score	support
False	0.62	0.79	0.69	97
True	0.68	0.47	0.56	91
accuracy			0.64	188
macro avg	0.65	0.63	0.63	188
weighted avg	0.65	0.64	0.63	188

balanced accuracy: 0.6331709527585816

9.3.4.3 PAT_S

calculating randomsearch SVM ...
Fitting 50 folds for each of 120 candidates, totaling 6000 fits

Best Score: 0.9772094572094571

Best Hyperparameters: {'C': 8.483428982440726, 'gamma': 0.5, 'kernel': 'rbf'}

	precision	recall	f1-score	support
0	0.53	0.92	0.67	97
3	0.20	0.04	0.06	27
5	0.33	0.09	0.14	11
7	0.40	0.14	0.21	14
9	0.00	0.00	0.00	2
11	0.00	0.00	0.00	9
13	0.50	0.14	0.22	7
15	0.75	0.14	0.24	21
accuracy			0.52	188
macro avg	0.34	0.18	0.19	188
weighted avg	0.46	0.52	0.42	188

balanced accuracy: 0.18425541621417912

9.3.4.4 PAT_nS

calculating randomsearch SVM ...
Fitting 50 folds for each of 120 candidates, totaling 6000 fits

Best Score: 0.259649624884919

Best Hyperparameters: {'C': 19.306977288832496, 'gamma': 0.05, 'kernel': 'rbf'}

	precision	recall	f1-score	support
0	0.60	0.80	0.69	97
3	0.23	0.19	0.20	27
5	0.20	0.09	0.13	11
7	0.36	0.29	0.32	14
9	0.00	0.00	0.00	2
11	0.00	0.00	0.00	9
13	0.20	0.14	0.17	7
15	0.55	0.29	0.37	21
accuracy			0.51	188
macro avg	0.27	0.22	0.24	188
weighted avg	0.45	0.51	0.46	188

balanced accuracy: 0.22431296271502457

9.3.4.5 PAT2_S

calculating randomsearch SVM ...
Fitting 50 folds for each of 120 candidates, totaling 6000 fits

Best Score: 0.9668674182959898

Best Hyperparameters: {'C': 8.483428982440726, 'gamma': 0.5, 'kernel': 'rbf'}

	precision	recall	f1-score	support
0	0.52	0.93	0.67	97
3	0.20	0.04	0.06	27
5	0.00	0.00	0.00	11
7	0.00	0.00	0.00	14
11	0.00	0.00	0.00	9
13	0.50	0.14	0.22	7
15	0.67	0.09	0.15	23
accuracy			0.50	188
macro avg	0.27	0.17	0.16	188
weighted avg	0.40	0.50	0.38	188

balanced accuracy: 0.170669393311386

9.3.4.6 PAT2_nS

calculating randomsearch SVM ...
Fitting 50 folds for each of 120 candidates, totaling 6000 fits

Best Score: 0.2929481892376629

Best Hyperparameters: {'C': 19.306977288832496, 'gamma': 0.05, 'kernel': 'rbf'}

	precision	recall	f1-score	support
0	0.65	0.79	0.72	97
3	0.28	0.26	0.27	27
5	0.00	0.00	0.00	11
7	0.44	0.29	0.35	14
11	0.00	0.00	0.00	9
13	0.17	0.14	0.15	7
15	0.40	0.35	0.37	23
accuracy			0.52	188
macro avg	0.28	0.26	0.27	188
weighted avg	0.47	0.52	0.49	188

balanced accuracy: 0.2613530296824143

9.3.5 kNN

balanced accuracy	neighbors					
Tox	1	3	5	7	9	11
TOX value intern	0.72	0.75	0.74	0.72	0.72	0.70
TOX value extern	0.63	0.58	0.65	0.61	0.61	0.59
Tox2	1	3	5	7	9	11
TOX2 value intern	0.72	0.68	0.72	0.74	0.72	0.72
TOX2 value extern	0.63	0.62	0.61	0.61	0.65	0.71
Pat_SMOTE	1	3	5	7	9	11
PatSMOTE value intern	0.93	0.90	0.88	0.85	0.83	0.82
PatSMOTE value extern	0.23	0.21	0.22	0.23	0.24	0.20
Pat_nSMOTE	1	3	5	7	9	11
Pat_nSMOTE value intern	0.32	0.29	0.25	0.27	0.27	0.26
Pat_nSMOTE value extern	0.22	0.19	0.17	0.17	0.17	0.14
Pat2_SMOTE	1	3	5	7	9	11
Pat2SMOTE value intern	0.92	0.89	0.86	0.84	0.81	0.80
Pat2SMOTE value extern	0.25	0.26	0.29	0.25	0.32 (0.318)	0.32 (0.324)
Pat2_nSMOTE	1	3	5	7	9	11
Pat2_nSMOTE value intern	0.35	0.31	0.29	0.30	0.27	0.28
Pat2_nSMOTE value extern	0.23	0.27	0.19	0.22	0.18	0.19

Table 7: Comparison of the different number of neighbors and the influence on the balanced accuracy score.

9.3.5.1 Best Models

9.3.5.1.1 TOX

used descriptors: [173, 237, 111, 201, 231, 313, 162, 185, 221, 159, 373, 183, 164, 308, 314, 319, 299, 320, 388, 309, 316, 169, 363, 167, 311, 399, 365, 273, 193, 353]

used descriptors: ['nssCH2', 'maxHBa', 'nBase_x', 'minHBint6', 'minsF', 'nF10Ring', 'nHBint5', 'naaO', 'minssssC', 'nwHBd', 'khs.sNH2', 'ndO', 'nHBint9', 'n7Ring', 'nF11Ring', 'n7HeteroRing', 'MDEO-22_x', 'nT7HeteroRing', 'MDEN-11_y', 'nG12Ring', 'nT7Ring', 'nHtCH', 'VC-5_y', 'nHss-NH', 'nF8Ring', 'XLogP_y', 'VP-7_y', 'IC2', 'SssS', 'nAcid_y']

number of neighbors: 5

maxvalue = 0.7434403931063952

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
False	0.56	0.64	0.60	97
True	0.74	0.67	0.70	148
accuracy			0.66	245
macro avg	0.65	0.65	0.65	245
weighted avg	0.67	0.66	0.66	245

balanced accuracy: 0.6540470883254388

9.3.5.1.2 TOX2

used descriptors: [286, 296, 375, 126, 261, 254, 251, 244, 382, 258, 167, 119, 336, 277, 138, 112, 47, 267, 117, 321, 17, 219, 51, 76, 217, 127, 137, 304, 50, 113]

used descriptors: ['TDB1v', 'PNSA-3_x', 'VPC-5_y', 'nsssCH', 'RotBFrac', 'n4HeteroRing', 'nF11Ring', 'n4Ring', 'khs.aaN', 'nF8HeteroRing', 'mintN', 'nHtCH', 'RDF80v', 'VR2_D', 'nssS', 'nHBd', 'GATS4e', 'JGI3', 'nHsNH2', 'RDF35m', 'ATSC0e', 'IC2', 'GATS5p', 'BCUTp-1l_x', 'IC0', 'ntsC', 'naaO', 'RPCG_x', 'GATS4p', 'nwHBd']

number of neighbors: 11

maxvalue = 0.7241741591741593

max value position equal to number of features -1 = 20

	precision	recall	f1-score	support
False	0.73	0.70	0.72	97
True	0.69	0.73	0.71	91
accuracy			0.71	188
macro avg	0.71	0.71	0.71	188
weighted avg	0.71	0.71	0.71	188

balanced accuracy: 0.7131528265548883

9.3.5.1.3 PAT_S

used descriptors: [73, 242, 310, 208, 45, 355, 322, 72, 356, 53, 217, 162, 378, 391, 36, 368, 215, 361, 319, 296, 297, 101, 22, 205, 230, 108, 198, 220, 398, 293]

used descriptors: ['BCUTw-1h_x', 'nRing', 'TPSA_x', 'ETA_EtaP', 'GATS2e', 'E1p', 'RDF35m', 'nBase_x', 'E2p', 'GATS4i', 'SIC0', 'minaasC', 'khs.aasC', 'MDEO-11_y', 'GATS4c', 'C3SP3_y', 'IC1', 'FNSA-3_y', 'RDF130u', 'PPSA-3_x', 'PNSA-1_x', 'VC-4_x', 'ATSC4i', 'ETA_dBeta', 'MDEC-13_x', 'VE1_Dt', 'ETA_dAlpha_B', 'SIC4', 'nRings7', 'TDB1i']

number of neighbors: 9

maxvalue = 0.8304836829836829

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
0	0.78	0.22	0.34	97
3	0.14	0.19	0.16	27
5	0.18	0.18	0.18	11
7	0.30	0.64	0.41	14
9	0.00	0.00	0.00	2
11	0.10	0.22	0.14	9
13	0.05	0.14	0.08	7
15	0.22	0.29	0.25	21
accuracy			0.24	188
macro avg	0.22	0.23	0.19	188
weighted avg	0.49	0.24	0.28	188

balanced accuracy: 0.23464362575187317

9.3.5.1.4 PAT_nS

used descriptors: [359, 250, 133, 248, 278, 174, 176, 177, 257, 110, 0, 142, 368, 147, 234, 6, 261, 251, 163, 134, 256, 116, 5, 337, 172, 114, 260, 390, 103, 127]

used descriptors: ['BCUTc-1l_y', 'nF10Ring', 'naaO', 'nF7Ring', 'VR2_D', 'mindsssP', 'minddssS', 'minsCl', 'nF7HeteroRing', 'nwHBd', 'nAcid_x', 'SssS', 'C3SP3_y', 'minHBint5', 'MDEC-24_x', 'nCl', 'nT7HeteroRing', 'nF11Ring', 'minssssC', 'nssS', 'n7HeteroRing', 'nHtCH', 'nF', 'RDF115s', 'minsOm', 'nHsNH2', 'nF10HeteroRing', 'MDEC-34_y', 'VC-6_x', 'nssssC']

number of neighbors: 1

maxvalue = 0.3173950924484013

max value position equal to number of features -1 = 26

	precision	recall	f1-score	support
0	0.63	0.71	0.67	97
3	0.25	0.19	0.21	27
5	0.22	0.18	0.20	11
7	0.20	0.21	0.21	14
9	0.00	0.00	0.00	2
11	0.07	0.11	0.09	9
13	0.50	0.14	0.22	7
15	0.24	0.19	0.21	21
accuracy			0.45	188
macro avg	0.26	0.22	0.23	188
weighted avg	0.44	0.45	0.44	188

balanced accuracy: 0.2171342164898866

9.3.5.1.5 PAT2_S

used descriptors: [77, 253, 219, 293, 166, 142, 294, 1, 349, 210, 302, 223, 9, 93, 363, 89, 204, 155, 354, 40, 352, 24, 212, 387, 304, 243, 92, 16, 299, 300]

used descriptors: ['BCUTw-1h_x', 'nFRing', 'nHBDon_Lipinski', 'TDB9m', 'minssCH2', 'SwHBa', 'TDB1v', 'ALogP_x', 'E1m', 'ETA_Beta', 'PPSA-3_x', 'IC2', 'AATS8m', 'C1SP2_x', 'FNSA-3_y', 'SpMin1_Bhe', 'MAXDN2', 'minHBint6', 'E3v', 'GATS7c', 'E1v', 'ATSC4i',

'ETA_dBeta', 'LipinskiFailures_y', 'PNSA-3_x', 'MDEO-11_x', 'SpMin1_Bhp', 'AATS8e', 'TDB1i', 'TDB10s']

number of neighbors: 11

maxvalue = 0.798218448218448

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
0	0.68	0.27	0.39	97
3	0.28	0.41	0.33	27
5	0.22	0.36	0.28	11
7	0.29	0.57	0.38	14
11	0.06	0.11	0.08	9
13	0.06	0.29	0.10	7
15	0.38	0.26	0.31	23
accuracy			0.31	188
macro avg	0.28	0.32	0.27	188
weighted avg	0.48	0.31	0.34	188

balanced accuracy: 0.324029791661219

9.3.5.1.6 PAT2_nS

used descriptors: [361, 256, 139, 98, 138, 119, 76, 257, 146, 115, 370, 260, 181, 121, 264, 254, 285, 140, 162, 165, 147, 129, 268, 391, 252, 144, 258, 253, 154, 389]

used descriptors: ['BCUTc-1l_y', 'nF10Ring', 'naaO', 'C3SP3_x', 'nssO', 'nHaaNH', 'nBase_x', 'nF11Ring', 'SHaaNH', 'nwHBd', 'C3SP3_y', 'n4HeteroRing', 'mindsssP', 'nHdCH2', 'nF7HeteroRing', 'nF7Ring', 'VR2_D', 'nssS', 'minHCsats', 'minsCH3', 'SHCsats', 'nsssCH', 'nT7HeteroRing', 'MDEC-34_y', 'n7Ring', 'SHBint6', 'nT7Ring', 'nFRing', 'minHBint5', 'MDEC-13_y']

number of neighbors: 1

maxvalue = 0.3487474868489906

max value position equal to number of features -1 = 25

	precision	recall	f1-score	support
0	0.60	0.63	0.61	97
3	0.22	0.26	0.24	27
5	0.33	0.09	0.14	11
7	0.21	0.21	0.21	14
11	0.20	0.11	0.14	9
13	0.17	0.14	0.15	7
15	0.15	0.17	0.16	23
accuracy			0.41	188
macro avg	0.27	0.23	0.24	188
weighted avg	0.41	0.41	0.41	188

balanced accuracy: 0.23160019161171755

9.3.5.2 Other Models

The order is always the following: TOX, TOX2, PAT_S, PAT_nS, PAT2_S, PAT2_nS
The best models are removed from this list, but it is noted where it was.

9.3.5.2.1 Neighbor 1

TOX

used descriptors: [359, 134, 118, 198, 191, 244, 184, 119, 167, 192, 27, 259, 251, 135, 127, 176, 254, 399, 141, 383, 310, 322, 60, 47, 76, 89, 277, 130, 252, 116]

used descriptors: ['BCUTc-1l_y', 'nsssN', 'nHaaNH', 'maxssssC', 'maxHBint10', 'n4Ring', 'maxHBint3', 'nHtCH', 'mintN', 'maxHsOH', 'MATS3c', 'nF10HeteroRing', 'nF11Ring', 'naasN', 'ntsC', 'mindsssP', 'n4HeteroRing', 'nRings7', 'SHsOH', 'khs.sssN', 'MOMI-Z_x', 'RDF45m', 'GATS1s', 'GATS4e', 'BCUTp-1l_x', 'SpMin1_Bhp', 'VR2_D', 'naaaC', 'nT7Ring', 'nHsOH']

number of neighbors: 1

maxvalue = 0.7194239094239094

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
False	0.63	0.66	0.65	97
True	0.62	0.59	0.61	91
accuracy			0.63	188
macro avg	0.63	0.63	0.63	188
weighted avg	0.63	0.63	0.63	188

balanced accuracy: 0.6266002039197915

TOX2

used descriptors: [359, 134, 118, 198, 191, 244, 184, 119, 167, 192, 27, 259, 251, 135, 127, 176, 254, 399, 141, 383, 310, 322, 60, 47, 76, 89, 277, 130, 252, 116]

used descriptors: ['BCUTc-1l_y', 'nsssN', 'nHaaNH', 'maxssssC', 'maxHBint10', 'n4Ring', 'maxHBint3', 'nHtCH', 'mintN', 'maxHsOH', 'MATS3c', 'nF10HeteroRing', 'nF11Ring', 'naasN', 'ntsC', 'mindsssP', 'n4HeteroRing', 'nRings7', 'SHsOH', 'khs.sssN', 'MOMI-Z_x', 'RDF45m', 'GATS1s', 'GATS4e', 'BCUTp-1l_x', 'SpMin1_Bhp', 'VR2_D', 'naaaC', 'nT7Ring', 'nHsOH']

number of neighbors: 1

maxvalue = 0.7194239094239094

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
False	0.63	0.66	0.65	97
True	0.62	0.59	0.61	91
accuracy			0.63	188
macro avg	0.63	0.63	0.63	188
weighted avg	0.63	0.63	0.63	188

balanced accuracy: 0.6266002039197915

PAT_S

used descriptors: [73, 397, 262, 333, 298, 204, 357, 79, 9, 31, 37, 72, 22, 368, 324, 231, 200, 126, 209, 356, 288, 246, 38, 214, 120, 42, 8, 117, 216, 39]

used descriptors: ['BCUTw-1h_x', 'nRings6', 'RotBFrac', 'RDF35v', 'PNSA-3_x', 'ETA_BetaP', 'E3p', 'SpMax7_Bhm', 'AATS5v', 'MATS2i', 'GATS5c', 'nBase_x', 'ATSC4i', 'C3SP3_y', 'RDF50m', 'MDEC-14_x', 'ETA_Shape_P', 'naaaC', 'ETA_EtaP_B', 'E2p', 'TDB1v', 'n7Ring', 'GATS7c', 'IC0', 'nHCsats', 'GATS6v', 'AATS8m', 'nHdCH2', 'IC2', 'GATS1m']

number of neighbors: 1

maxvalue = 0.9325437062937063

max value position equal to number of features -1 = 27

	precision	recall	f1-score	support
0	0.63	0.43	0.51	97
3	0.17	0.22	0.19	27
5	0.17	0.18	0.17	11
7	0.21	0.29	0.24	14
9	0.00	0.00	0.00	2
11	0.12	0.22	0.16	9
13	0.06	0.14	0.09	7
15	0.39	0.33	0.36	21
accuracy			0.34	188
macro avg	0.22	0.23	0.22	188
weighted avg	0.43	0.34	0.37	188

balanced accuracy: 0.2276446348611297

---- best model PAT_nS removed---

PAT2_S

used descriptors: [77, 248, 212, 308, 111, 302, 208, 125, 187, 18, 349, 387, 338, 87, 370, 118, 267, 9, 182, 4, 210, 181, 257, 301, 372, 19, 392, 141, 10, 390]

used descriptors: ['BCUTw-1h_x', 'nRing', 'ETA_dBeta', 'FNSA-2_x', 'CrippenLogP', 'PPSA-3_x', 'ETA_Shape_Y', 'nHCsatu', 'maxwHBa', 'ATSC1v', 'E1m', 'LipinskiFailures_y', 'RDF40v', 'SpMax1_Bhv', 'C3SP3_y', 'nHsNH2', 'nF10HeteroRing', 'AATS8m', 'minaaS', 'nN', 'ETA_Beta', 'mindsssP', 'nF11Ring', 'PPSA-1_x', 'SCH-5_y', 'ATSC4v', 'MDEO-11_y', 'SHBd', 'AATS5v', 'MDEC-14_y']

number of neighbors: 1

maxvalue = 0.921834831834832

max value position equal to number of features -1 = 27

	precision	recall	f1-score	support
0	0.61	0.53	0.57	97
3	0.20	0.22	0.21	27
5	0.00	0.00	0.00	11
7	0.16	0.21	0.18	14
11	0.18	0.22	0.20	9
13	0.17	0.29	0.21	7
15	0.28	0.30	0.29	23
accuracy			0.38	188
macro avg	0.23	0.25	0.24	188
weighted avg	0.41	0.38	0.39	188

balanced accuracy: 0.25350935234395566

---- best model PAT2_nS removed---

9.3.5.2.2 Neighbor 3

TOX

used descriptors: [341, 315, 385, 176, 305, 178, 117, 129, 185, 291, 381, 310, 232, 226, 103, 338, 357, 31, 278, 13, 366, 102, 40, 249, 294, 137, 186, 253, 353, 136]

used descriptors: ['BCUTc-1l_y', 'nF12Ring', 'MDEC-14_y', 'naaaC', 'n4Ring', 'ntN', 'nBondsT', 'C2SP1_x', 'naaO', 'MDEC-14_x', 'khs.aaO', 'nF7Ring', 'mindsssP', 'minssssNp', 'VR2_DzZ', 'SRW3', 'C1SP3_y', 'ATSC4e', 'SIC4', 'AATS4v', 'tpsaEfficiency', 'VE2_DzZ', 'MATS2c', 'maxssCH2', 'MDEC-24_x', 'SCH-4_x', 'nssS', 'meanI', 'nAcid_y', 'C4SP3_x']

number of neighbors: 3

maxvalue = 0.7516835208659736

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
False	0.51	0.49	0.50	97
True	0.67	0.68	0.68	148
accuracy			0.61	245
macro avg	0.59	0.59	0.59	245
weighted avg	0.61	0.61	0.61	245

balanced accuracy: 0.5886388966285874

TOX2

used descriptors: [359, 130, 399, 167, 254, 373, 198, 120, 247, 258, 237, 176, 206, 235, 175, 244, 197, 252, 119, 251, 127, 257, 283, 0, 165, 178, 13, 391, 390, 177]

used descriptors: ['BCUTc-1l_y', 'naaaC', 'nRings7', 'mintN', 'n4HeteroRing', 'VC-4_y', 'maxssssC', 'nHdCH2', 'n7Ring', 'nF8HeteroRing', 'MDEC-34_x', 'mindsssP', 'ETA_Shape_X', 'MDEC-24_x', 'minsF', 'n4Ring', 'maxaaaC', 'nT7Ring', 'nHtCH', 'nF11Ring', 'ntsC', 'n7HeteroRing', 'TDB9u', 'nAcid_x', 'minssssC', 'minddssS', 'AATS6e', 'MDEO-11_y', 'MDEC-34_y', 'minaaS']

number of neighbors: 3

maxvalue = 0.6845920745920746

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
False	0.62	0.69	0.65	97
True	0.62	0.55	0.58	91
accuracy			0.62	188
macro avg	0.62	0.62	0.62	188
weighted avg	0.62	0.62	0.62	188

balanced accuracy: 0.6200860994675428

PAT_S

used descriptors: [73, 397, 120, 305, 20, 124, 106, 395, 344, 335, 387, 386, 297, 306, 36, 355, 109, 337, 362, 22, 327, 101, 368, 58, 237, 373, 37, 41, 212, 278]

used descriptors: ['BCUTw-1h_x', 'nRings6', 'nHCsats', 'WNSA-3_x', 'ATSC1i', 'ndssC', 'CrippenLogP', 'nSmallRings', 'L3m', 'RDF45v', 'MDEC-12_y', 'LipinskiFailures_y', 'PNSA-1_x', 'RPCG_x', 'GATS4c', 'E1p', 'nHBd', 'RDF115s', 'RHSA_y', 'ATSC4i', 'RDF85m', 'VC-4_x', 'C3SP3_y', 'GATS1s', 'MDEO-11_x', 'VC-4_y', 'GATS5c', 'GATS1v', 'fragC_x', 'VR2_D']

number of neighbors: 3

maxvalue = 0.9012907925407926

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
0	0.75	0.39	0.51	97
3	0.13	0.19	0.15	27
5	0.18	0.18	0.18	11
7	0.19	0.36	0.25	14
9	0.00	0.00	0.00	2
11	0.06	0.11	0.07	9
13	0.07	0.14	0.09	7
15	0.30	0.29	0.29	21
accuracy			0.31	188
macro avg	0.21	0.21	0.19	188
weighted avg	0.47	0.31	0.36	188

balanced accuracy: 0.20694766764354394

PAT_nS

used descriptors: [359, 261, 259, 4, 369, 174, 246, 256, 347, 9, 231, 181, 388, 122, 83, 30, 107, 139, 127, 234, 374, 257, 138, 390, 299, 110, 297, 104, 230, 13]

used descriptors: ['BCUTc-1l_y', 'nT7HeteroRing', 'nF9HeteroRing', 'nS', 'SCH-5_y', 'mindsssP', 'n7Ring', 'n7HeteroRing', 'Dm', 'AATS5v', 'MDEC-14_x', 'maxHBint2', 'MDEC-13_y', 'nssCH2', 'SpMax1_Bhv', 'MATS1i', 'SpMAD_Dt', 'SHCsats', 'nssssC', 'MDEC-24_x', 'VC-5_y', 'nF7HeteroRing', 'SHaaNH', 'MDEC-34_y', 'DPSA-1_x', 'nwHBd',

'PNSA-1_x', 'SPC-4_x', 'MDEC-13_x', 'AATS6e']

number of neighbors: 3

maxvalue = 0.2909205541190836

max value position equal to number of features -1 = 27

	precision	recall	f1-score	support
0	0.58	0.82	0.68	97
3	0.12	0.07	0.09	27
5	0.33	0.18	0.24	11
7	0.25	0.21	0.23	14
9	0.00	0.00	0.00	2
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.40	0.19	0.26	21
accuracy			0.48	188
macro avg	0.21	0.19	0.19	188
weighted avg	0.40	0.48	0.42	188

balanced accuracy: 0.18567455358692472

PAT2_S

used descriptors: [77, 248, 212, 304, 29, 229, 80, 37, 150, 19, 387, 9, 58, 0, 377, 2, 255, 93, 346, 391, 333, 359, 312, 379, 129, 103, 218, 144, 237, 120]

used descriptors: ['BCUTw-1h_x', 'nRing', 'ETA_dBeta', 'PNSA-3_x', 'MATS3c', 'CIC2', 'nBondsD2', 'GATS1c', 'minHBd', 'ATSC4v', 'LipinskiFailures_y', 'AATS8m', 'GATS5i', 'nAcid_x', 'tpsaEfficiency', 'AMR_x', 'nF9Ring', 'C1SP2_x', 'Du', 'MDEC-34_y', 'RDF90m', 'E3p', 'RNCG_x', 'khs.aasC', 'nsssCH', 'VCH-6_x', 'fragC_x', 'SHBint6', 'MDEC-14_x', 'nHtCH']

number of neighbors: 3

maxvalue = 0.8868992911850053

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
0	0.68	0.40	0.51	97
3	0.12	0.15	0.13	27
5	0.08	0.09	0.09	11
7	0.29	0.50	0.37	14
11	0.05	0.11	0.07	9
13	0.11	0.29	0.16	7
15	0.29	0.30	0.30	23
accuracy			0.32	188
macro avg	0.23	0.26	0.23	188
weighted avg	0.44	0.32	0.36	188

balanced accuracy: 0.2631846168056708

PAT2_nS

used descriptors: [241, 129, 264, 396, 392, 0, 118, 137, 144, 399, 252, 258, 254, 285, 108, 263, 257, 115, 268, 139, 172, 179, 260, 376, 256, 50, 202, 14, 243, 116]

used descriptors: ['MDEC-33_x', 'nsssCH', 'nF7HeteroRing', 'nSmallRings', 'MDEO-11_y', 'nAcid_x', 'nHsNH2', 'naasN', 'SHBint6', 'nRings7', 'n7Ring', 'nT7Ring', 'nF7Ring', 'VR2_D', 'VC-6_x', 'n7HeteroRing', 'nF11Ring', 'nwHBd', 'nT7HeteroRing', 'naaO', 'mintN', 'minsOm', 'n4HeteroRing', 'VPC-5_y', 'nF10Ring', 'GATS5e', 'hmax', 'AATS6e', 'MDEO-11_x', 'nHBint4']

number of neighbors: 3

maxvalue = 0.3102625344580232

max value position equal to number of features -1 = 11

	precision	recall	f1-score	support
0	0.63	0.85	0.72	97
3	0.38	0.33	0.35	27
5	0.30	0.27	0.29	11
7	0.40	0.29	0.33	14
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.31	0.17	0.22	23
accuracy			0.54	188
macro avg	0.29	0.27	0.27	188
weighted avg	0.46	0.54	0.49	188

balanced accuracy: 0.2730069657136315

9.3.5.2.3 Neighbor 5

---- best model TOX removed---

TOX2

used descriptors: [287, 126, 142, 145, 112, 244, 181, 251, 254, 113, 399, 260, 277, 381, 172, 247, 324, 376, 223, 375, 322, 198, 137, 82, 390, 109, 176, 128, 23, 343]

used descriptors: ['TDB3v', 'nsssCH', 'SHCsats', 'minHBd', 'nHBd', 'n4Ring', 'maxHBa', 'nF11Ring', 'n4HeteroRing', 'nwHBd', 'nRings7', 'nT7HeteroRing', 'VR2_D', 'khs.dsN', 'mindO', 'n7Ring', 'RDF55m', 'tpsaEfficiency', 'SIC4', 'VPC-5_y', 'RDF45m', 'maxssssC', 'naaO', 'SpMin3_Bhm', 'MDEC-34_y', 'CrippenLogP', 'mindsssP', 'ndssC', 'AATSC0e', 'L3u']

number of neighbors: 5

maxvalue = 0.722947052947053

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
False	0.62	0.62	0.62	97
True	0.60	0.60	0.60	91
accuracy			0.61	188
macro avg	0.61	0.61	0.61	188
weighted avg	0.61	0.61	0.61	188

balanced accuracy: 0.6114761527132662

PAT_S

used descriptors: [73, 245, 120, 227, 298, 20, 357, 97, 356, 60, 124, 386, 230, 137, 348, 112, 378, 395, 334, 122, 114, 8, 295, 367, 260, 5, 48, 134, 329, 312]

used descriptors: ['BCUTw-1h_x', 'n6Ring', 'nHCsats', 'nAtomP_x', 'PNSA-3_x', 'ATSC1i', 'E3p', 'VCH-5_x', 'E2p', 'GATS4s', 'ndssC', 'LipinskiFailures_y', 'MDEC-13_x', 'SHsOH', 'E1v', 'nHBint4', 'khs.aasC', 'nSmallRings', 'RDF40v', 'nssCH2', 'nHsNH2', 'AATS8m', 'PPSA-1_x', 'C3SP2_y', 'nF10HeteroRing', 'nF', 'GATS1p', 'nssS', 'RDF95m', 'MOMI-Z_x']

number of neighbors: 5

maxvalue = 0.8770820845820846

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
0	0.76	0.32	0.45	97
3	0.13	0.22	0.16	27
5	0.15	0.18	0.17	11
7	0.23	0.36	0.28	14
9	0.00	0.00	0.00	2
11	0.11	0.22	0.14	9
13	0.06	0.14	0.09	7
15	0.30	0.29	0.29	21
accuracy			0.28	188
macro avg	0.22	0.22	0.20	188
weighted avg	0.48	0.28	0.33	188

balanced accuracy: 0.2164455676053614

PAT_nS

used descriptors: [359, 261, 256, 126, 176, 398, 110, 165, 278, 258, 246, 257, 243, 117, 249, 259, 219, 101, 147, 174, 385, 347, 14, 0, 93, 280, 80, 386, 268, 372]

used descriptors: ['BCUTc-1l_y', 'nT7HeteroRing', 'n7HeteroRing', 'naaaC', 'minddssS', 'nRings7', 'nwHBd', 'mintN', 'VR2_D', 'nF8HeteroRing', 'n7Ring', 'nF7HeteroRing', 'n4Ring', 'nHdCH2', 'nF9Ring', 'nF9HeteroRing', 'SIC2', 'VC-4_x', 'minHBint5', 'mindsssP', 'khs.aaS', 'Dm', 'AATS7e', 'nAcid_x', 'C3SP3_x', 'SRW5', 'SpMin2_Bhm', 'LipinskiFailures_y', 'JGI3', 'VCH-5_y']

number of neighbors: 5

maxvalue = 0.2506214639119051

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
0	0.55	0.86	0.67	97
3	0.17	0.07	0.10	27
5	0.25	0.09	0.13	11
7	0.22	0.14	0.17	14
9	0.00	0.00	0.00	2
11	0.50	0.11	0.18	9
13	0.00	0.00	0.00	7
15	0.25	0.10	0.14	21
accuracy			0.48	188
macro avg	0.24	0.17	0.17	188
weighted avg	0.39	0.48	0.40	188

balanced accuracy: 0.1712324521602872

PAT2_S

used descriptors: [77, 248, 212, 304, 65, 52, 357, 353, 231, 58, 31, 321, 148, 9, 141, 259, 155, 350, 233, 352, 364, 16, 17, 277, 117, 202, 370, 235, 128, 358]

used descriptors: ['BCUTw-1h_x', 'nRing', 'ETA_dBeta', 'PNSA-3_x', 'GATS5s', 'GATS4p', 'E1p', 'E2v', 'CIC4', 'GATS5i', 'MATS2e', 'RDF20u', 'SaaS', 'AATS8m', 'SHBd', 'nHeteroRing', 'minHBint6', 'E3m', 'nAtomP_x', 'E1v', 'RHSA_y', 'AATS8e', 'AATS6i', 'JGI5', 'nHsOH', 'hmax', 'C3SP3_y', 'MDEC-12_x', 'nssCH2', 'E2p']

number of neighbors: 5

maxvalue = 0.8601546072974646

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
0	0.71	0.31	0.43	97
3	0.19	0.26	0.22	27
5	0.18	0.27	0.21	11
7	0.29	0.50	0.37	14
11	0.08	0.22	0.12	9
13	0.05	0.14	0.08	7
15	0.32	0.35	0.33	23
accuracy			0.31	188
macro avg	0.26	0.29	0.25	188
weighted avg	0.47	0.31	0.34	188

balanced accuracy: 0.29345290493398324

PAT2_nS

used descriptors: [361, 264, 268, 140, 383, 0, 277, 139, 263, 115, 181, 387, 237, 14, 232, 15, 256, 149, 376, 246, 252, 112, 340, 104, 144, 240, 375, 138, 390, 118]

used descriptors: ['BCUTc-1l_y', 'nF7HeteroRing', 'nT7HeteroRing', 'nssS', 'khs.aaN', 'nAcid_x', 'JGI5', 'naaO', 'n7HeteroRing', 'nwHBd', 'mindsssP', 'LipinskiFailures_y', 'MDEC-14_x', 'AATS6e', 'ZMIC1', 'AATS7e', 'nF10Ring', 'SssS', 'VPC-5_y', 'piPC3', 'n7Ring',

'SpMax_Dt', 'RDF115s', 'SC-4_x', 'SHBint6', 'MDEC-24_x', 'VC-5_y', 'nssO', 'MDEC-14_y', 'nHsNH2']

number of neighbors: 5

maxvalue = 0.2944177868914711

max value position equal to number of features -1 = 23

	precision	recall	f1-score	support
0	0.58	0.88	0.70	97
3	0.26	0.19	0.22	27
5	0.25	0.09	0.13	11
7	0.50	0.14	0.22	14
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.00	0.00	0.00	23
accuracy			0.49	188
macro avg	0.23	0.19	0.18	188
weighted avg	0.39	0.49	0.42	188

balanced accuracy: 0.18503429696360474

9.3.5.2.4 Neighbor 7

TOX

used descriptors: [368, 111, 312, 136, 176, 308, 226, 381, 398, 230, 129, 297, 164, 309, 159, 168, 316, 310, 305, 374, 314, 372, 320, 388, 137, 103, 178, 204, 246, 337]

used descriptors: ['GRAV-5_y', 'nBase_x', 'nF9Ring', 'C4SP3_x', 'naaaC', 'n7Ring', 'minssss-Np', 'khs.aaO', 'nRings7', 'minsOm', 'C2SP1_x', 'MDEC-44_x', 'nHBint9', 'nG12Ring', 'nwHBd', 'nHaaNH', 'nT7Ring', 'nF7Ring', 'n4Ring', 'khs.aaNH', 'nF11Ring', 'khs.aaaC', 'nT7HeteroRing', 'MDEN-11_y', 'SCH-4_x', 'VR2_DzZ', 'ntN', 'minHBint9', 'maxHBint9', 'VR2_D']

number of neighbors: 7

maxvalue = 0.7160874519299244

max value position equal to number of features -1 = 24

	precision	recall	f1-score	support
False	0.51	0.59	0.55	97
True	0.70	0.64	0.67	148
accuracy			0.62	245
macro avg	0.61	0.61	0.61	245
weighted avg	0.63	0.62	0.62	245

balanced accuracy: 0.6113820005572583

TOX2

used descriptors: [287, 126, 385, 338, 0, 188, 190, 206, 9, 386, 382, 17, 137, 1, 309, 282, 310, 100, 42, 18, 117, 307, 254, 391, 144, 113, 115, 318, 339, 302]

used descriptors: ['TDB3v', 'nsssCH', 'LipinskiFailures_y', 'RDF90v', 'nAcid_x', 'maxHBint7', 'maxHBint9', 'ETA_Shape_X', 'AATS5v', 'nAtomLAC_y', 'khs.aaN', 'ATSC0e', 'naaO', 'ALogP_x', 'RHSA_x', 'TDB6u', 'MOMI-Z_x', 'SC-4_x', 'GATS1v', 'ATSC1e', 'nHsNH2', 'THSA_x', 'n4HeteroRing', 'MDEO-11_y', 'SssS', 'nwHBd', 'nHBint4', 'RDF130u', 'RDF105v', 'WNSA-1_x']

number of neighbors: 7

maxvalue = 0.7465068265068266

max value position equal to number of features -1 = 24

	precision	recall	f1-score	support
False	0.62	0.62	0.62	97
True	0.60	0.60	0.60	91
accuracy			0.61	188
macro avg	0.61	0.61	0.61	188
weighted avg	0.61	0.61	0.61	188

balanced accuracy: 0.6114761527132662

PAT_S

used descriptors: [73, 245, 320, 29, 17, 11, 351, 215, 229, 350, 289, 0, 324, 21, 341, 62, 219, 23, 80, 72, 386, 31, 372, 337, 395, 147, 202, 391, 86, 217]

used descriptors: ['BCUTw-1h_x', 'n6Ring', 'RDF15m', 'MATS2e', 'ATSC4v', 'AATS7v', 'Dv', 'IC1', 'MDEC-12_x', 'E3v', 'TDB3v', 'nAcid_x', 'RDF50m', 'ATSC3i', 'E2u', 'VE1_DzZ', 'SIC2', 'AATSC0e', 'SpMin2_Bhm', 'nBase_x', 'LipinskiFailures_y', 'MATS2i', 'VCH-5_y', 'RDF115s', 'nSmallRings', 'minHBint5', 'ETA_Shape_X',

'MDEO-11_y', 'SpMin1_Bhe', 'SIC0']

number of neighbours: 7

maxvalue = 0.8531023143523144

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
0	0.77	0.28	0.41	97
3	0.24	0.30	0.27	27
5	0.12	0.18	0.14	11
7	0.29	0.57	0.38	14
9	0.00	0.00	0.00	2
11	0.05	0.11	0.07	9
13	0.05	0.14	0.07	7
15	0.38	0.29	0.32	21
accuracy			0.28	188
macro avg	0.24	0.23	0.21	188
weighted avg	0.51	0.28	0.33	188

balanced accuracy: 0.23344701308618832

PAT_nS

used descriptors: [235, 123, 257, 242, 250, 395, 112, 115, 319, 212, 375, 331, 138, 258, 165, 117, 110, 47, 237, 243, 44, 336, 42, 136, 279, 248, 240, 13, 14, 104]

used descriptors: ['MDEC-33_x', 'nsssCH', 'nF7HeteroRing', 'nRing', 'nF10Ring', 'nSmallRings', 'nHBint4', 'nHaaNH', 'RDF130u', 'fragC_x', 'VPC-5_y', 'RDF130m', 'SHaaNH', 'nF8HeteroRing', 'mintN', 'nHdCH2', 'nwHBd', 'GATS5e', 'MDEO-11_x', 'n4Ring', 'GATS1e', 'RDF110s', 'GATS6v', 'SHBint4', 'SRW3', 'nF7Ring', 'piPC3', 'AATS6e', 'AATS7e', 'SPC-4_x']

number of neighbours: 7

maxvalue = 0.270557610281875

max value position equal to number of features -1 = 24

	precision	recall	f1-score	support
0	0.54	0.89	0.67	97
3	0.25	0.15	0.19	27
5	0.20	0.09	0.13	11
7	0.60	0.21	0.32	14
9	0.00	0.00	0.00	2
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.00	0.00	0.00	21
accuracy			0.50	188
macro avg	0.20	0.17	0.16	188
weighted avg	0.37	0.50	0.41	188

balanced accuracy: 0.1674926114359104

PAT2_S

used descriptors: [77, 248, 212, 304, 241, 80, 320, 52, 53, 222, 347, 236, 14, 91, 25, 111, 356, 0, 67, 93, 372, 140, 205, 128, 328, 298, 364, 144, 294, 97]

used descriptors: ['BCUTw-1h_x', 'nRing', 'ETA_dBeta', 'PNSA-3_x', 'MDEC-33_x', 'nBondsD2', 'RDF10u', 'GATS4p', 'GATS5p', 'IC1', 'L3m', 'MDEC-13_x', 'AATS6e', 'SpMin7_Bhe', 'AATSC0e', 'CrippenLogP', 'E2e', 'nAcid_x', 'VE2_DzZ', 'C1SP2_x', 'SCH-5_y', 'nssS', 'ETA_dAlpha_B', 'nssCH2', 'RDF45m', 'TDB1e', 'RHSA_y', 'SHBint6', 'TDB1v', 'C2SP3_x']

number of neighbours: 7

maxvalue = 0.8365515436944009

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
0	0.76	0.26	0.38	97
3	0.24	0.41	0.31	27
5	0.07	0.09	0.08	11
7	0.27	0.43	0.33	14
11	0.05	0.11	0.07	9
13	0.05	0.14	0.07	7
15	0.21	0.30	0.25	23
accuracy			0.28	188
macro avg	0.24	0.25	0.21	188
weighted avg	0.48	0.28	0.31	188

balanced accuracy: 0.2489908522437177

PAT2_nS

used descriptors: [241, 129, 372, 152, 191, 135, 101, 261, 99, 386, 383, 126, 44, 243, 5, 307, 116, 375, 15, 14, 264, 117, 392, 276, 143, 253, 303, 254, 210, 268]

used descriptors: ['MDEC-33_x', 'nsssCH', 'SCH-5_y', 'minHBint2', 'maxHBint5', 'naaN', 'SCH-7_x', 'n5HeteroRing', 'SCH-5_x', 'khs.aaS', 'khs.aaN', 'nHother', 'GATS1v', 'MDEO-11_x', 'nS', 'FPSA-3_x', 'nHBint4', 'VC-5_y', 'AATS7e', 'AATS6e', 'nF7HeteroRing', 'nHsOH', 'MDEO-11_y', 'JGI4', 'SHBint4', 'nFRing', 'PNSA-1_x', 'nF7Ring', 'ETA_Beta', 'nT7HeteroRing']

number of neighbours: 7

maxvalue = 0.3035186371690131

max value position equal to number of features -1 = 27

	precision	recall	f1-score	support
0	0.59	0.84	0.69	97
3	0.20	0.15	0.17	27
5	0.60	0.27	0.37	11
7	0.30	0.21	0.25	14
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.11	0.04	0.06	23
accuracy			0.49	188
macro avg	0.26	0.22	0.22	188
weighted avg	0.40	0.49	0.43	188

balanced accuracy: 0.21624156320320756

9.3.5.2.5 Neighbor 9

TOX

used descriptors: [173, 156, 226, 305, 381, 311, 137, 388, 389, 157, 209, 182, 310, 313, 178, 315, 300, 309, 338, 297, 117, 291, 146, 237, 366, 1, 340, 32, 316, 103]

used descriptors: ['nssCH2', 'VR1_Dt', 'minssssNp', 'n4Ring', 'khs.aaO', 'nF8Ring', 'SCH-4_x', 'MDEN-11_y', 'MDEN-12_y', 'VR2_Dt', 'minHdCH2', 'naasN', 'nF7Ring', 'nF10Ring', 'ntN', 'nF12Ring', 'MDEN-12_x', 'nG12Ring', 'SRW3', 'MDEC-44_x', 'nBondsT', 'MDEC-14_x', 'VC-4_x', 'maxHBa', 'tpsaEfficiency', 'ALogP_x', 'XLogP_x', 'ATSC5p', 'nT7Ring', 'VR2_DzZ']

number of neighbours: 9

maxvalue = 0.7220502777382138

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
False	0.53	0.54	0.53	97
True	0.69	0.68	0.69	148
accuracy			0.62	245
macro avg	0.61	0.61	0.61	245
weighted avg	0.63	0.62	0.63	245

balanced accuracy: 0.6092574533296182

TOX2

used descriptors: [286, 296, 374, 265, 251, 18, 113, 244, 277, 167, 144, 298, 127, 258, 119, 254, 198, 391, 214, 303, 291, 138, 382, 137, 223, 322, 202, 33, 248, 115]

used descriptors: ['TDB1v', 'PNSA-3_x', 'VC-5_y', 'JGI1', 'nF11Ring', 'ATSC1e', 'nwHBd', 'n4Ring', 'VR2_D', 'mintN', 'SssS', 'FPSA-1_x', 'ntsC', 'nF8HeteroRing', 'nHtCH', 'n4HeteroRing', 'maxssssC', 'MDEO-11_y', 'ETA_EtaP_B_RC', 'WNSA-3_x', 'TDB1i', 'nssS', 'khs.aaN', 'naaO', 'SIC4', 'RDF45m', 'ETA_dAlpha_B', 'MATS2s', 'nFRing', 'nHBint4']

number of neighbours: 9

maxvalue = 0.7170579420579419

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
False	0.67	0.62	0.65	97
True	0.63	0.68	0.65	91
accuracy			0.65	188
macro avg	0.65	0.65	0.65	188
weighted avg	0.65	0.65	0.65	188

balanced accuracy: 0.6499376911748046

---- best model PAT_S removed---

PAT_nS

used descriptors: [98, 9, 382, 369, 134, 148, 248, 320, 136, 247, 237, 257, 92, 18, 289, 256, 298, 252, 112, 375, 390, 201, 58, 83, 270, 13, 246, 278, 202, 303]

used descriptors: ['VCH-6_x', 'AATS5v', 'khs.aaN', 'SCH-5_y', 'nssS', 'minHBint6', 'nF7Ring', 'RDF15m', 'SHBint4', 'nFRing', 'MDEO-11_x', 'nF7HeteroRing', 'C2SP3_x', 'ATSC0e', 'TDB3v', 'n7HeteroRing', 'PNSA-3_x', 'nT7Ring', 'nHBint4', 'VPC-5_y', 'MDEC-34_y', 'ETA_Shape_Y', 'GATS1s', 'SpMax1_Bhv', 'JGI5', 'AATS6e', 'n7Ring', 'VR2_D', 'ETA_Shape_X', 'FNSA-3_x']

number of neighbours: 9

maxvalue = 0.266908638134741

max value position equal to number of features -1 = 27

	precision	recall	1-score	support
0	0.60	0.88	0.71	97
3	0.31	0.19	0.23	27
5	0.29	0.18	0.22	11
7	0.17	0.07	0.10	14
9	0.00	0.00	0.00	2
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.14	0.05	0.07	21
accuracy			0.50	188
macro avg	0.19	0.17	0.17	188
weighted avg	0.40	0.50	0.43	188

balanced accuracy: 0.17029245573060006

PAT2_S

used descriptors: [77, 98, 398, 308, 214, 352, 379, 274, 369, 364, 62, 58, 226, 295, 210, 311, 5, 111, 386, 9, 141, 358, 155, 93, 230, 395, 144, 140, 243, 89]

used descriptors: ['BCUTw-1h_x', 'C3SP3_x', 'nRings6', 'FNSA-2_x', 'ETA_EtaP', 'E1v', 'khs.aasC', 'JGI2', 'C3SP2_y', 'RHSA_y', 'GATS1s', 'GATS5i', 'SIC2', 'TDB3v', 'ETA_Beta', 'RPCG_x', 'nS', 'CrippenLogP', 'khs.aaS', 'AATS8m', 'SHBd', 'E2p', 'minHBint6', 'C1SP2_x', 'CIC3', 'geomShape_y', 'SHBint6', 'nssS', 'MDEO-11_x', 'SpMin1_Bhe']

number of neighbours: 9

maxvalue = 0.8145050187907329

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
0	0.72	0.22	0.33	97
3	0.17	0.22	0.19	27
5	0.21	0.36	0.27	11
7	0.27	0.50	0.35	14
11	0.11	0.33	0.16	9
13	0.07	0.29	0.11	7
15	0.30	0.30	0.30	23
accuracy			0.27	188
macro avg	0.27	0.32	0.25	188
weighted avg	0.48	0.27	0.29	188

balanced accuracy: 0.3179641251934266

PAT2_nS

used descriptors: [151, 87, 183, 119, 257, 16, 115, 134, 5, 382, 285, 173, 252, 143, 132, 240, 392, 146, 399, 116, 144, 140, 396, 148, 46, 227, 47, 308, 376, 243]

used descriptors: ['minHBa', 'SpMax1_Bhv', 'minddssS', 'nHaaNH', 'nF11Ring', 'AATS8e', 'nwHBd', 'ndsN', 'nS', 'khs.dsN', 'VR2_D', 'mindsN', 'n7Ring', 'SHBint4', 'naaaC', 'MDEC-24_x',

'MDEO-11_y', 'SHaaNH', 'nRings7', 'nHBint4', 'SHBint6', 'nssS', 'nSmallRings', 'SaasC', 'GATS8v', 'SIC4', 'GATS1e', 'FNSA-2_x', 'VPC-5_y', 'MDEO-11_x']

number of neighbours: 9

maxvalue = 0.2686547182975754

max value position equal to number of features -1 = 28

	precision	recall	f1-score	support
0	0.56	0.75	0.64	97
3	0.12	0.11	0.12	27
5	0.00	0.00	0.00	11
7	0.25	0.14	0.18	14
11	0.25	0.11	0.15	9
13	0.00	0.00	0.00	7
15	0.19	0.13	0.15	23
accuracy			0.44	188
macro avg	0.20	0.18	0.18	188
weighted avg	0.36	0.44	0.39	188

balanced accuracy: 0.1782987810393842

9.3.5.2.6 Neighbor 11

TOX

used descriptors: [132, 368, 311, 337, 226, 305, 185, 314, 383, 374, 247, 159, 168, 232, 398, 356, 299, 381, 315, 316, 170, 372, 103, 246, 157, 308, 310, 137, 205, 164]

used descriptors: ['C3SP2_x', 'GRAV-5_y', 'nF8Ring', 'VR2_D', 'minssssNp', 'n4Ring', 'naaO', 'nF11Ring', 'LipinskiFailures_y', 'khs.aaNH', 'maxHBint10', 'nwHBd', 'nHaaNH', 'mindsssP', 'nRings7', 'C3SP2_y', 'MDEO-22_x', 'khs.aaO', 'nF12Ring', 'nT7Ring', 'nHdsCH', 'khs.aaaC', 'VR2_DzZ', 'maxHBint9', 'VR2_Dt', 'n7Ring', 'nF7Ring', 'SCH-4_x', 'minHBint10', 'nHBint9']

number of neighbours: 11

maxvalue = 0.6986597350804729

max value position equal to number of features -1 = 26

	precision	recall	f1-score	support
False	0.50	0.53	0.52	97
True	0.68	0.66	0.67	148
accuracy			0.61	245
macro avg	0.59	0.59	0.59	245
weighted avg	0.61	0.61	0.61	245

balanced accuracy: 0.5939676790192254

---- best model TOX2 removed---

PAT_S

used descriptors: [73, 395, 391, 204, 58, 269, 346, 293, 351, 311, 60, 372, 344, 356, 227, 54, 288, 219, 301, 257, 334, 18, 13, 258, 373, 122, 358, 287, 354, 200]

used descriptors: ['BCUTw-1h_x', 'nSmallRings', 'MDEO-11_y', 'ETA_BetaP', 'GATS1s', 'JGI4', 'E3m', 'TDB1i', 'Dv', 'RHSA_x', 'GATS4s', 'VCH-5_y', 'L3m', 'E2p', 'nAtomP_x', 'GATS5i', 'TDB1v', 'SIC2', 'FPSA-3_x', 'nF7HeteroRing', 'RDF40v', 'ATSC0e', 'AATS6e', 'nF8HeteroRing', 'VC-4_y', 'nssCH2', 'Dp', 'TDB9m', 'De', 'ETA_Shape_P']

number of neighbours: 11

maxvalue = 0.818028221778222

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
0	0.83	0.26	0.39	97
3	0.17	0.26	0.21	27
5	0.12	0.18	0.15	11
7	0.12	0.29	0.17	14
9	0.00	0.00	0.00	2
11	0.06	0.11	0.08	9
13	0.09	0.29	0.13	7
15	0.23	0.24	0.23	21
accuracy			0.24	188
macro avg	0.20	0.20	0.17	188
weighted avg	0.50	0.24	0.29	188

balanced accuracy: 0.20243054005940603

PAT_nS

used descriptors: [94, 98, 74, 129, 280, 237, 54, 136, 83, 257, 251, 252, 369, 391, 47, 61, 147, 118, 97, 40, 72, 117, 288, 110, 58, 0, 142, 20, 134, 372]

used descriptors: ['SCH-5_x', 'VCH-6_x', 'BCUTp-1l_x', 'naaN', 'SRW5', 'MDEO-11_x', 'GATS5i', 'SHBint4', 'SpMax1_Bhv', 'nF7HeteroRing', 'nF11Ring', 'nT7Ring', 'SCH-5_y', 'MDEO-11_y', 'GATS5e', 'GATS5s', 'minHBint5', 'nHdsCH', 'VCH-5_x', 'GATS8m', 'nBase_x', 'nHdCH2', 'TDB1v', 'nwHBd', 'GATS1s', 'nAcid_x', 'SssS', 'ATSC1i', 'nssS', 'VCH-5_y']

number of neighbours: 11

maxvalue = 0.25851173132423133

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
0	0.55	0.86	0.67	97
3	0.21	0.11	0.15	27
5	0.25	0.09	0.13	11
7	0.20	0.07	0.11	14
9	0.00	0.00	0.00	2
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.00	0.00	0.00	21
accuracy			0.47	188
macro avg	0.15	0.14	0.13	188
weighted avg	0.35	0.47	0.38	188

balanced accuracy: 0.1411398595676946

---- best model PAT2_S removed---

PAT2_nS

used descriptors: [108, 76, 249, 378, 253, 370, 181, 285, 172, 209, 252, 182, 391, 399, 139, 258, 390, 263, 265, 173, 349, 298, 9, 65, 257, 87, 161, 17, 222, 144]

used descriptors: ['VC-6_x', 'nBase_x', 'n4Ring', 'GRAV-5_y', 'nFRing', 'C3SP3_y', 'mindsssP', 'VR2_D', 'mintN', 'ETA_Shape_X', 'n7Ring', 'minaaS', 'MDEC-34_y', 'nRings7', 'naaO', 'nT7Ring', 'MDEC-14_y', 'n7HeteroRing', 'nF8HeteroRing', 'mindsN', 'E1m', 'TDB1e', 'AATS8m', 'GATS5s', 'nF11Ring', 'SpMax1_Bhv', 'minHaaCH', 'AATS6i', 'IC1', 'SHBint6']

number of neighbours: 11

maxvalue = 0.28024425115402557

max value position equal to number of features -1 = 29

	precision	recall	f1-score	support
0	0.54	0.82	0.65	97
3	0.15	0.07	0.10	27
5	0.29	0.18	0.22	11
7	0.25	0.14	0.18	14
11	0.00	0.00	0.00	9
13	0.00	0.00	0.00	7
15	0.38	0.13	0.19	23
accuracy			0.47	188
macro avg	0.23	0.19	0.19	188
weighted avg	0.38	0.47	0.40	188

balanced accuracy: 0.19341806419990445

9.4 Figures and Confusion Matrixes

9.4.1 LogReg

TOX : confusion matrix:
[[51 46]
 [45 103]]

TOX2 : confusion matrix:
[[60 37]
 [29 62]]

PAT_S : confusion matrix:
[[56 14 6 2 2 4 4 9]
 [13 6 1 3 0 2 1 1]
 [7 1 1 0 0 1 0 1]
 [1 5 0 4 0 1 0 3]
 [0 0 0 0 0 1 1 0]
 [3 2 0 1 0 1 0 2]
 [1 1 0 3 0 0 1 1]
 [6 3 0 1 0 0 1 10]]

PAT_nS : confusion matrix:
[[70 11 3 1 2 2 2 6]
 [14 5 1 3 0 2 1 1]
 [9 0 1 0 0 0 0 1]
 [4 5 1 2 0 0 1 1]
 [0 1 0 0 0 0 1 0]
 [5 1 0 0 0 1 0 2]
 [5 0 0 1 0 0 0 1]
 [8 2 0 1 0 1 0 9]]

PAT2_S : confusion matrix:
[[59 15 4 2 5 4 8]
 [11 5 1 4 3 1 2]
 [2 1 0 2 0 0 6]
 [4 4 2 3 0 0 1]
 [4 1 0 0 1 0 3]
 [2 1 0 1 0 1 2]
 [5 2 2 2 0 2 10]]

PAT2_nS : confusion matrix:
[[70 14 1 2 2 1 7]
 [12 8 0 4 1 1 1]
 [3 1 1 1 0 0 5]
 [8 1 2 2 0 0 1]
 [5 1 0 0 1 0 2]
 [5 0 0 1 0 0 1]
 [11 0 2 1 0 2 7]]

9.4.2 DTI

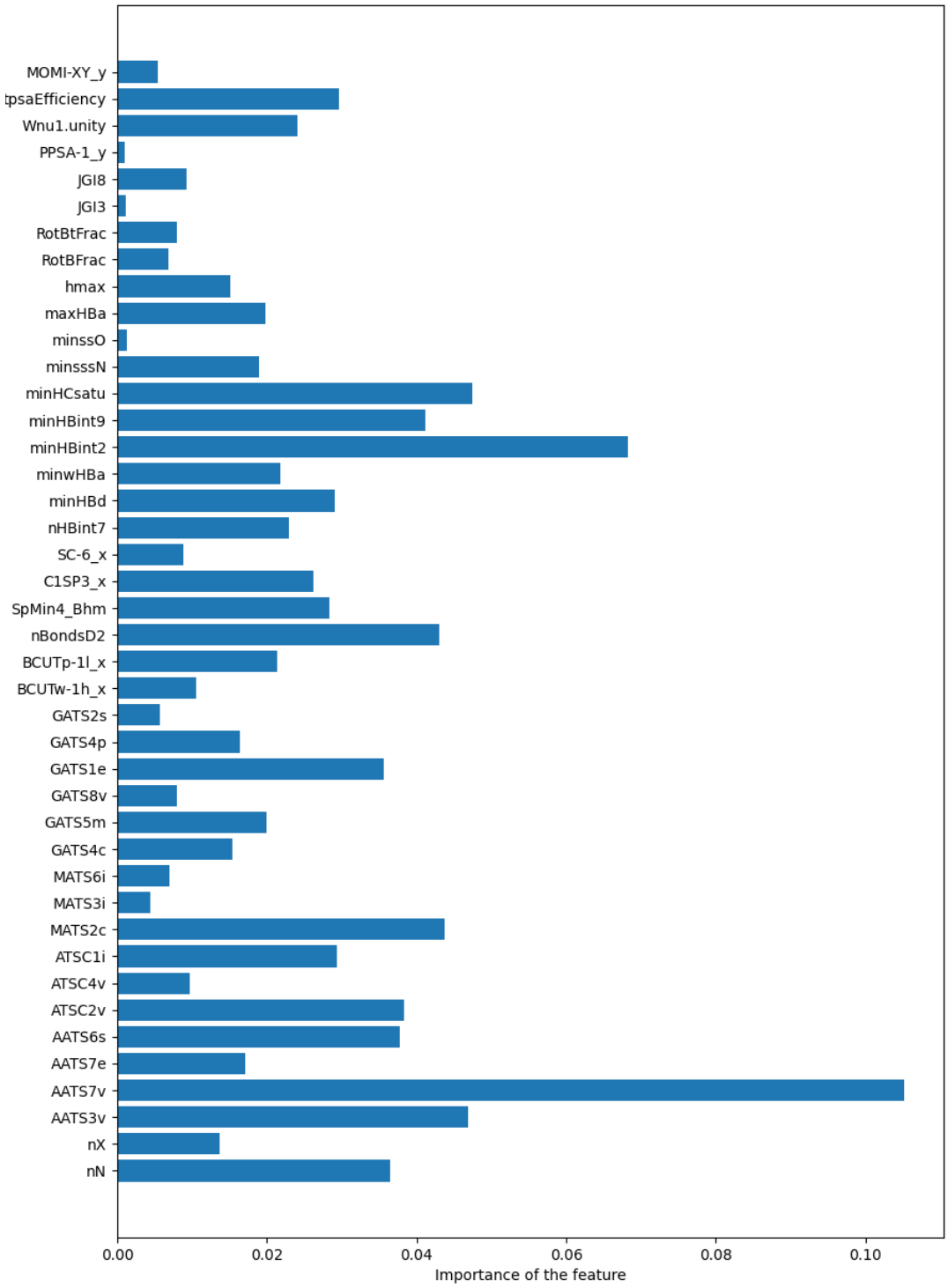


Figure 19:Feature Importance TOX, the y-axis only represents labels that are not 0 in importance.

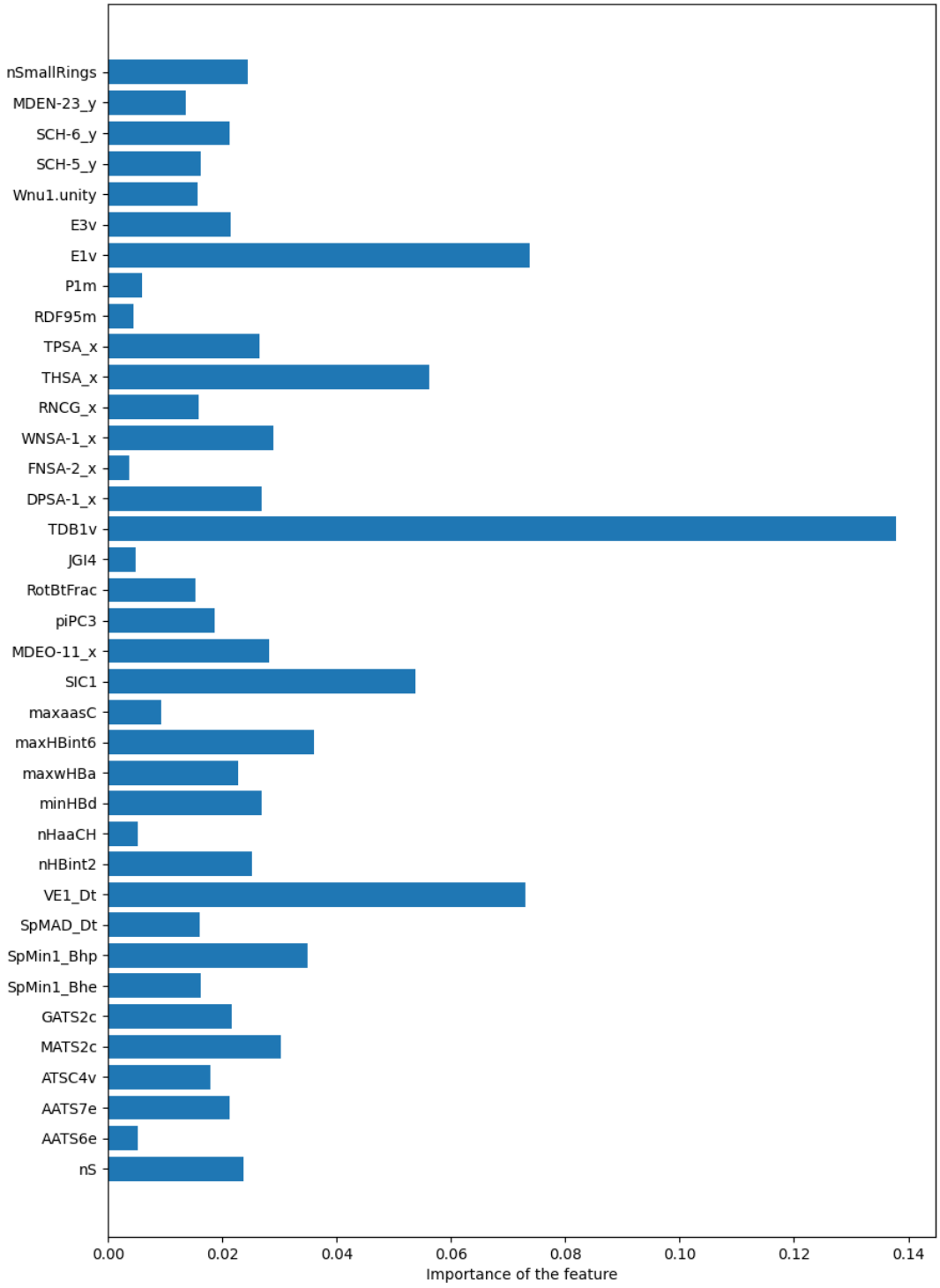


Figure 20: Feature Importance TOX2, the y-axis only represents labels that are not 0 in importance.

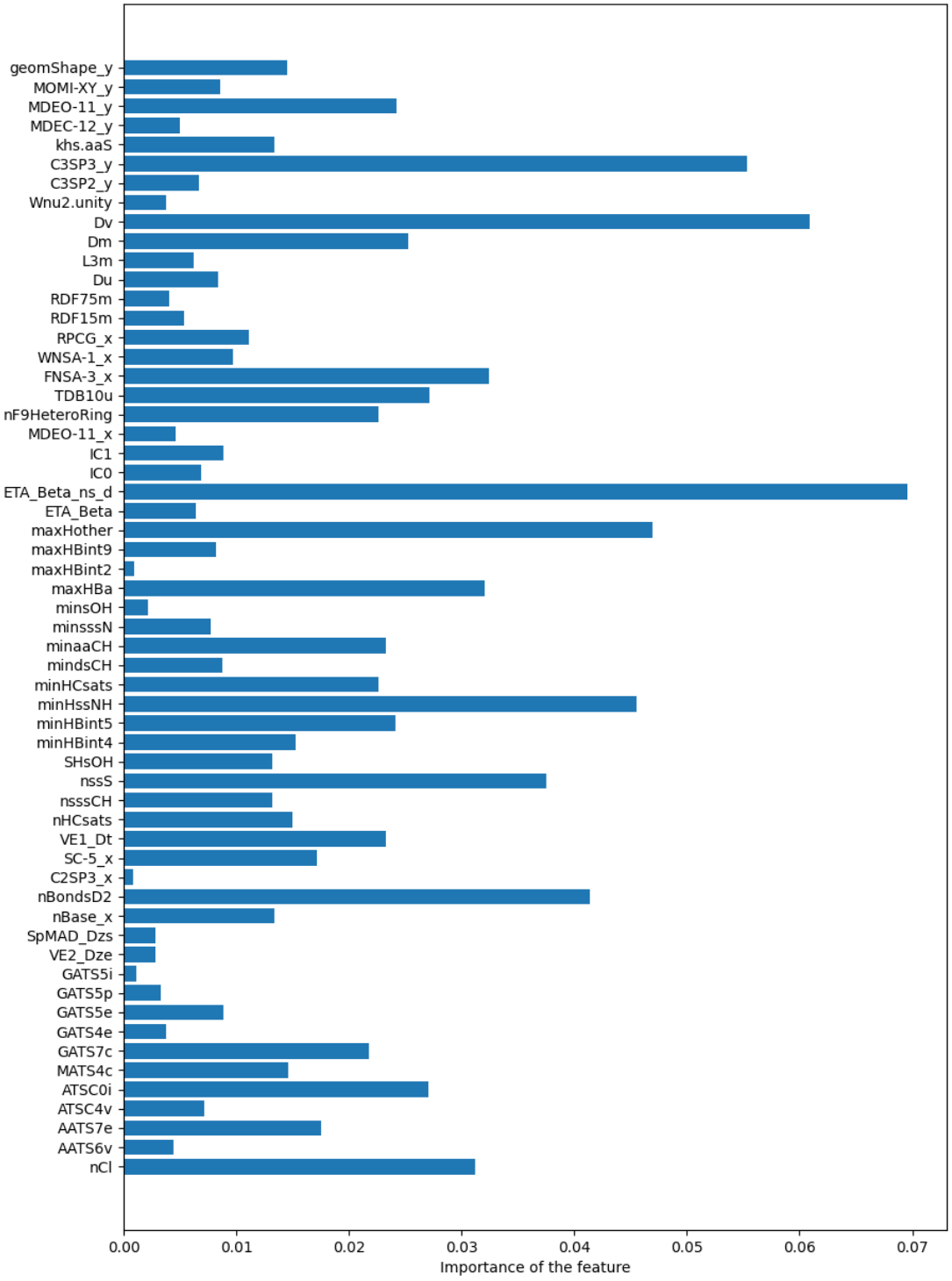


Figure 21: Feature Importance PAT_S, the y-axis only represents labels that are not 0 in importance.

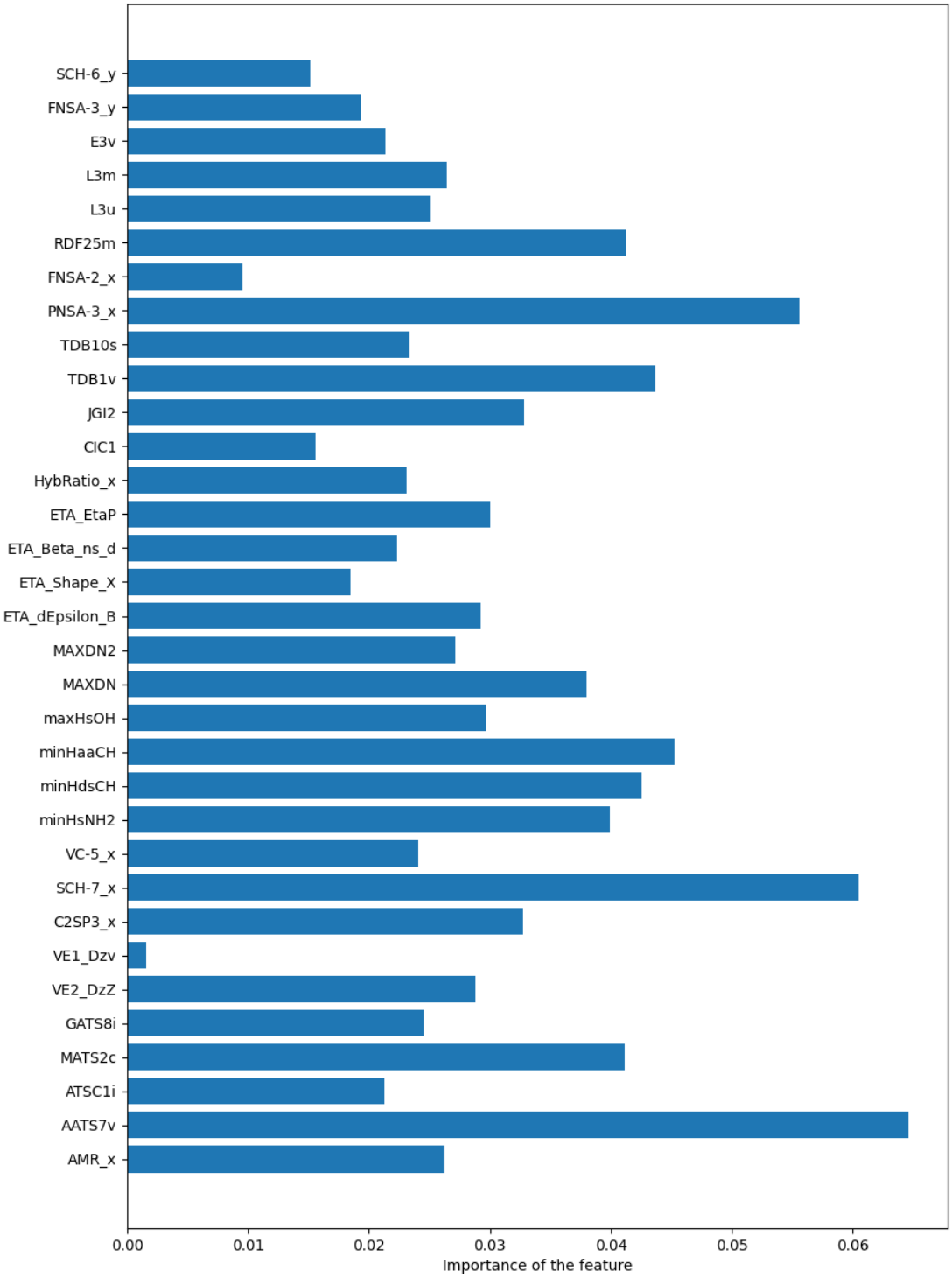


Figure 22: Feature Importance PAT_nS, the y-axis only represents labels that are not 0 in importance.

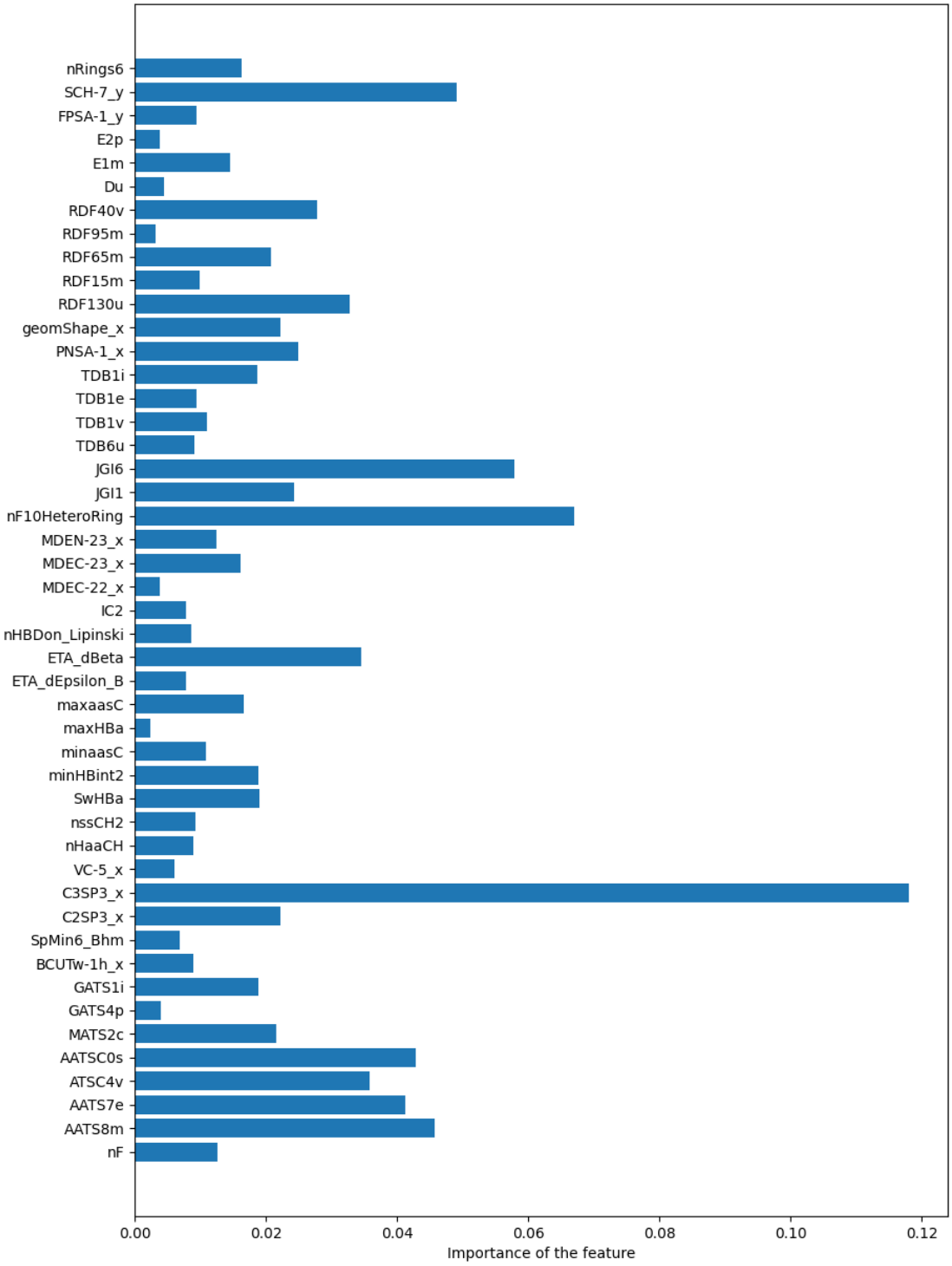


Figure 23: Feature Importance PAT2_S, the y-axis only represents labels that are not 0 in importance.

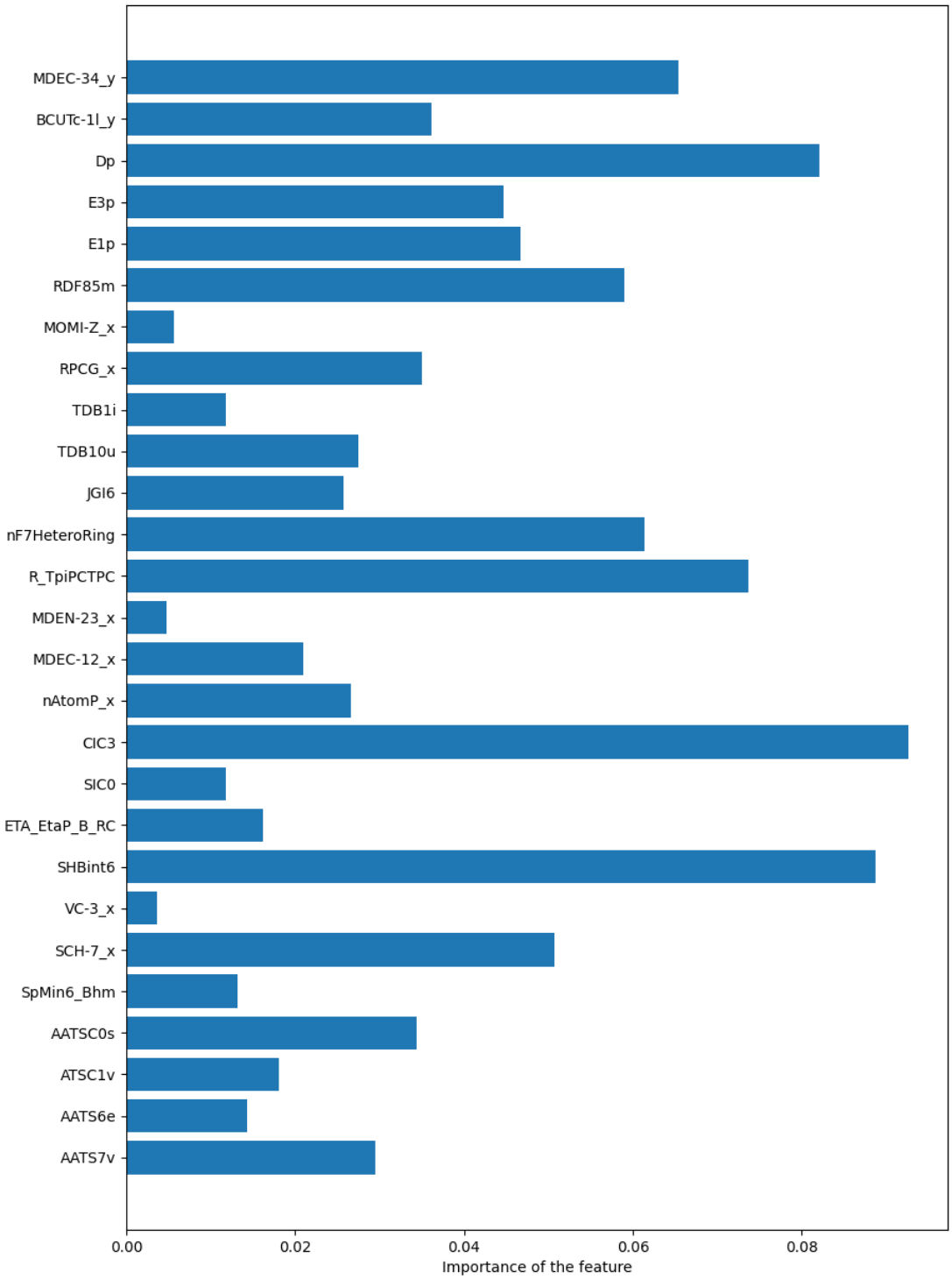


Figure 24: Feature Importance PAT2_nS, the y-axis only represents labels that are not 0 in importance.

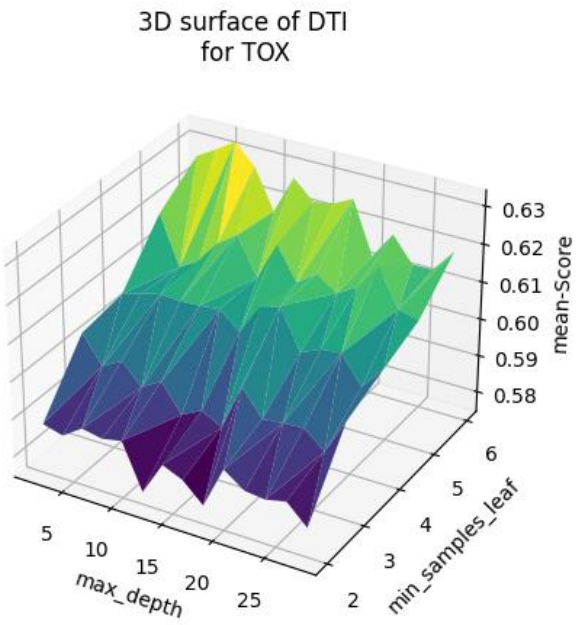


Figure 25: GridsearchCV 3D surface plot TOX

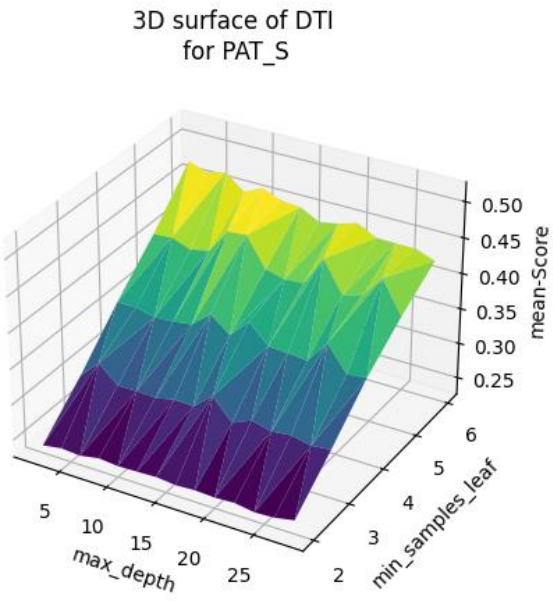


Figure 27: GridsearchCV 3D surface plot PAT_S

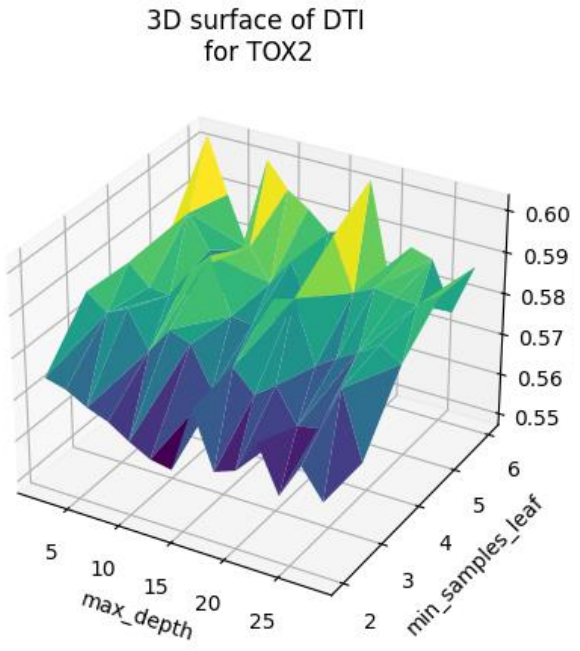


Figure 26: GridsearchCV 3D surface plot TOX2

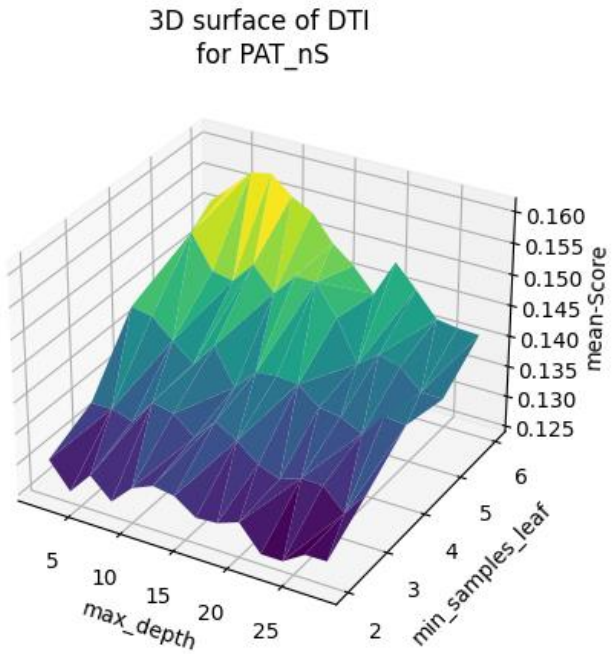


Figure 28: GridsearchCV 3D surface plot PAT_nS

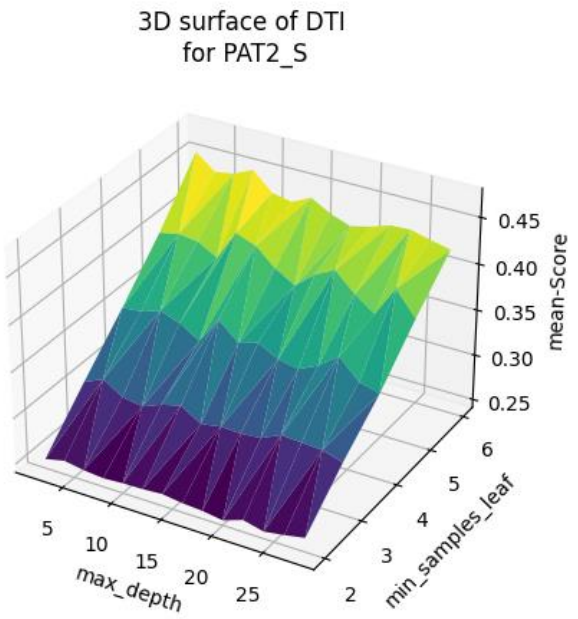


Figure 29: GridsearchCV 3D surface plot PAT2_S

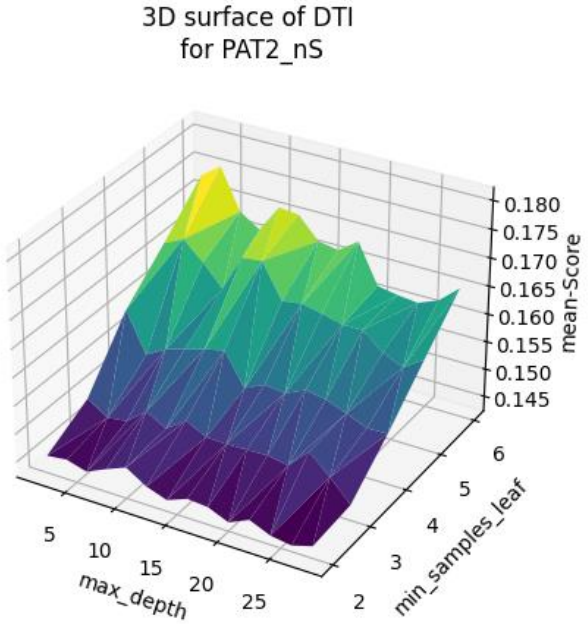


Figure 30: GridsearchCV 3D surface plot PAT2_nS

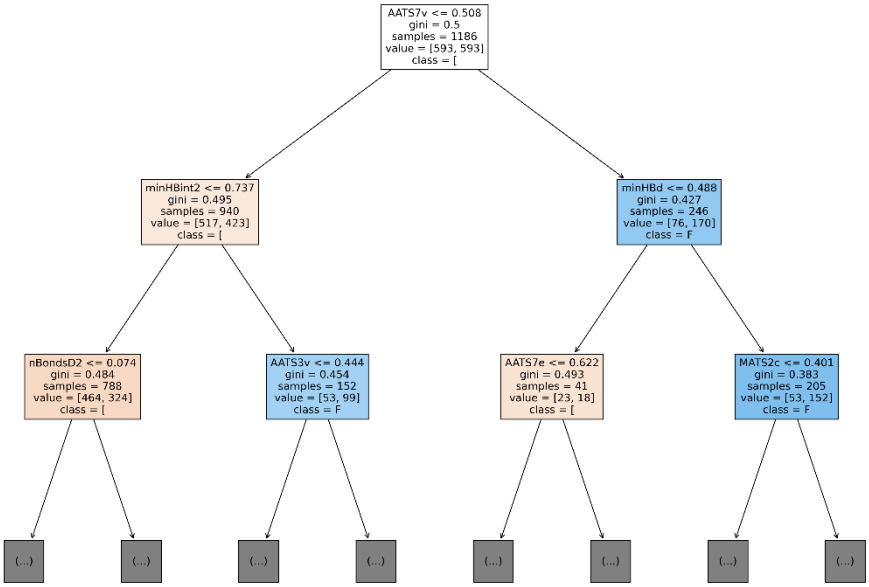


Figure 31: First few splits DTI of TOX

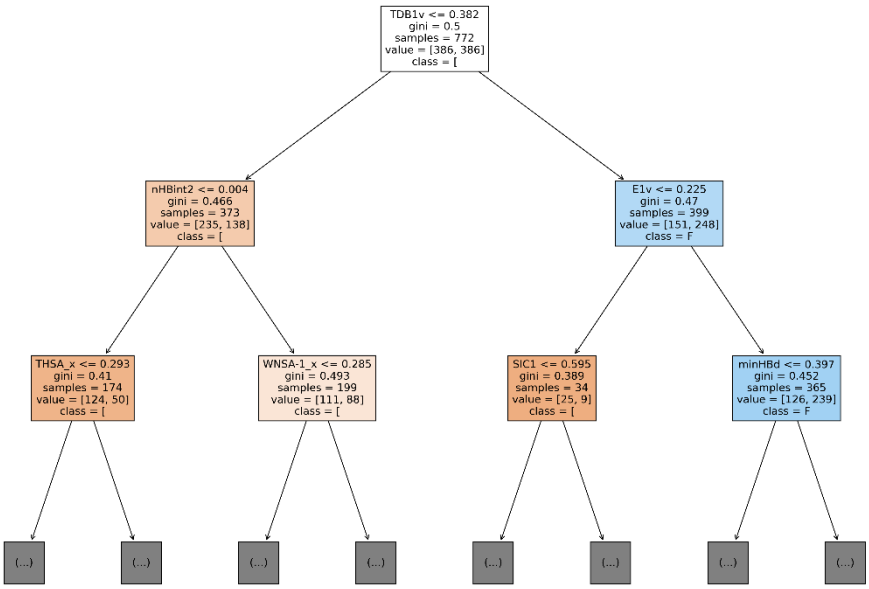


Figure 32: First few splits DTI of TOX2

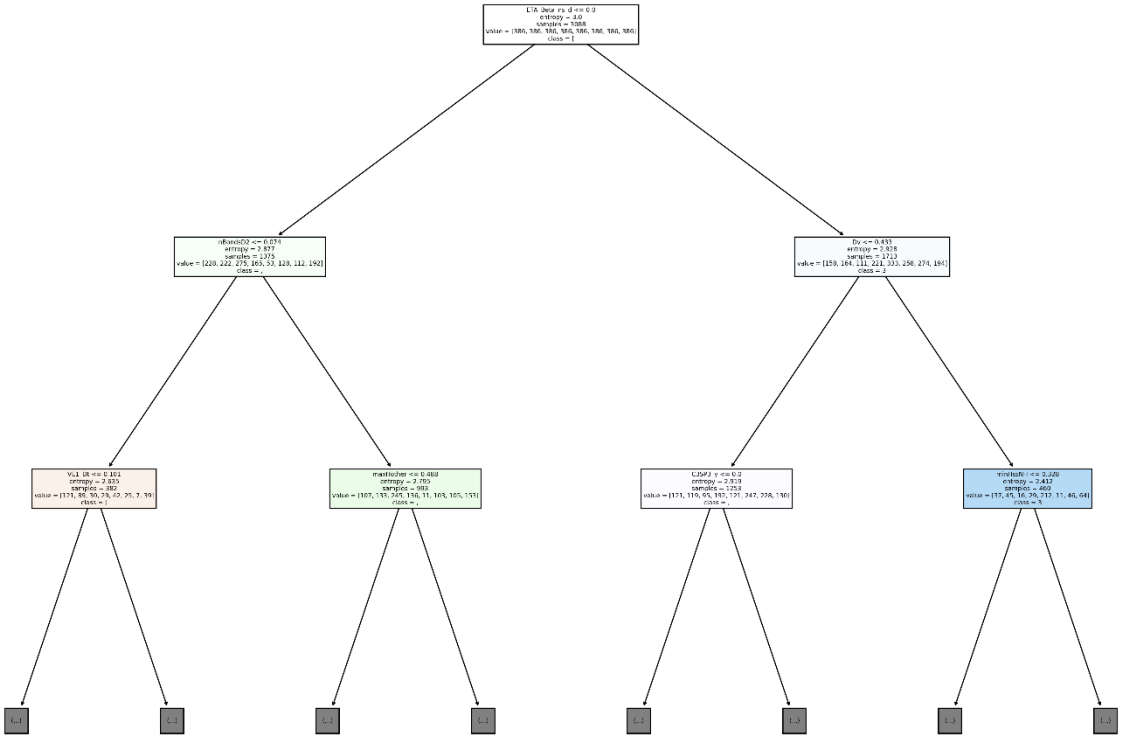


Figure 33: First few splits DTI of PAT_S

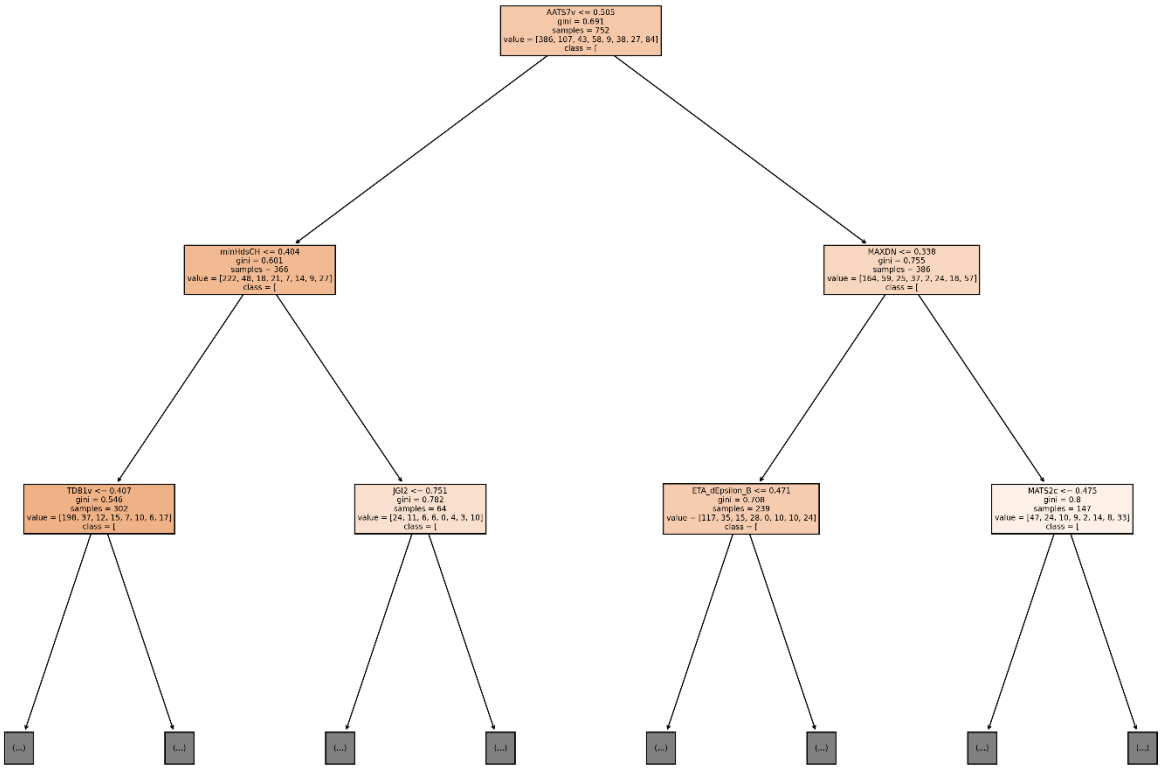


Figure 34: First few splits DTI of PAT_nS

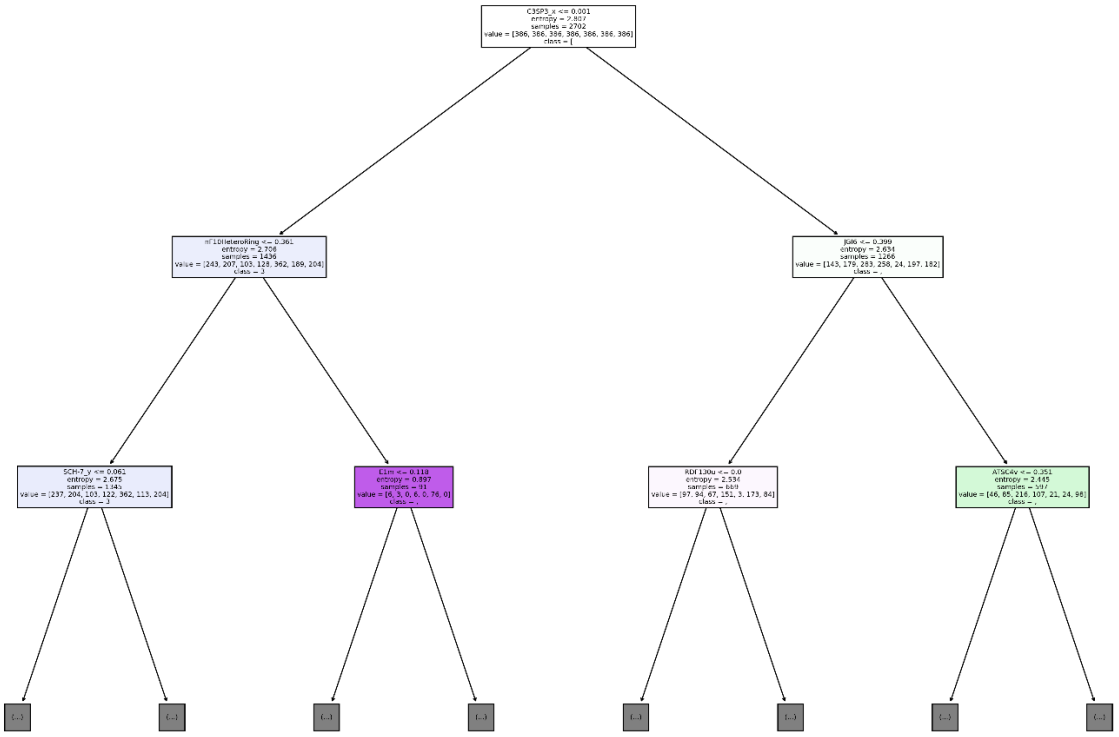


Figure 35: First few splits DTI of PAT2_S

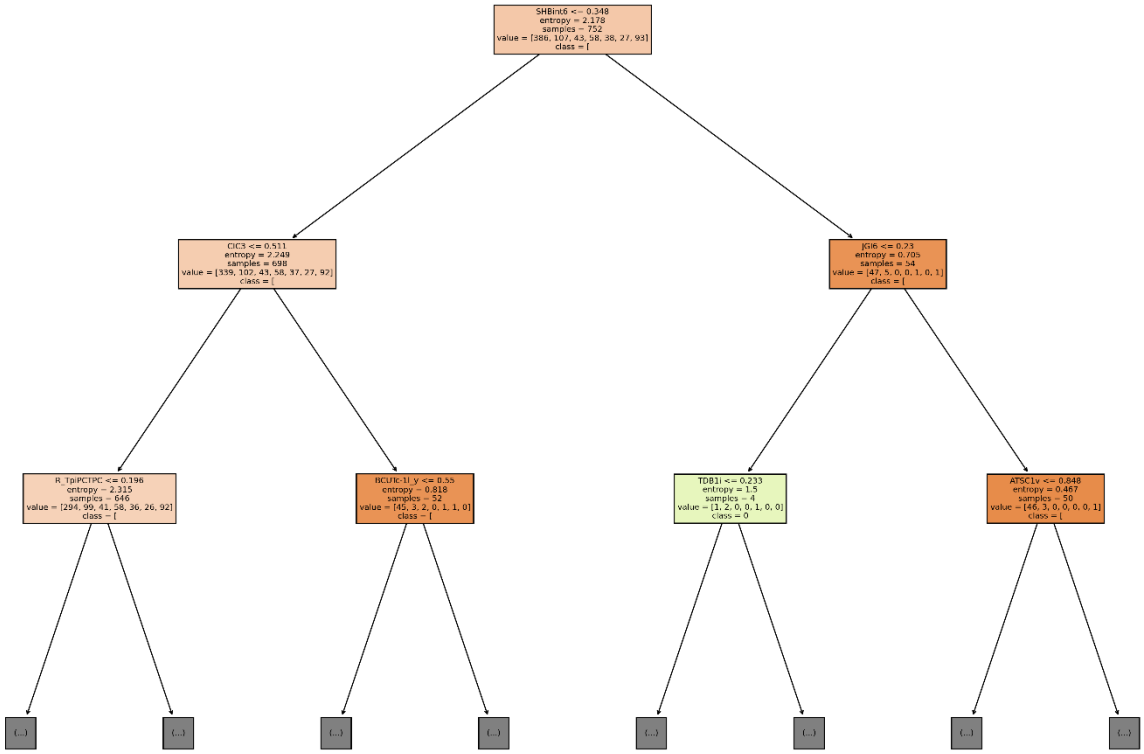


Figure 36: First few splits DTI of PAT2_nS

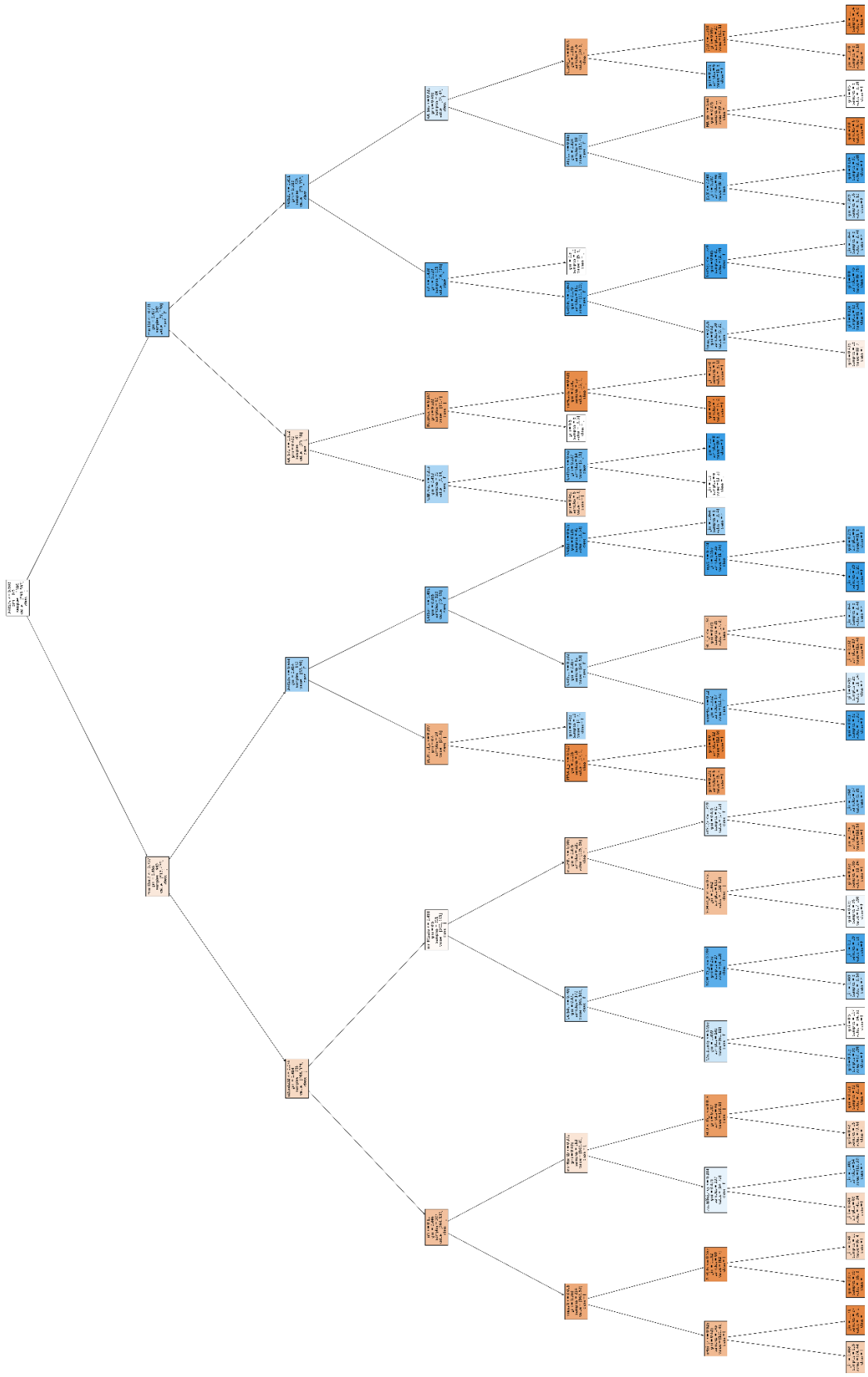


Figure 37: Decision tree of TOX, due to the size of the tree this only gives an idea of how it looks. For the values look at the text representation beneath.

```
|--- AATS7v <= 0.51
|   |--- minHBint2 <= 0.74
|   |   |--- nBondsD2 <= 0.07
|   |   |   |--- nN <= 0.11
|   |   |   |   |--- minsssN <= 0.02
|   |   |   |   |   |--- nHBint7 <= 0.03
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- nHBint7 >  0.03
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- minsssN >  0.02
|   |   |   |   |   |--- ATSC4v <= 0.56
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- ATSC4v >  0.56
|   |   |   |   |   |   |--- class: False
|   |   |   |--- nN >  0.11
|   |   |   |   |--- minHBint9 <= 0.39
|   |   |   |   |   |--- tpsaEfficiency <= 0.18
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- tpsaEfficiency >  0.18
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- minHBint9 >  0.39
|   |   |   |   |   |--- BCUTw-1h_x <= 0.02
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- BCUTw-1h_x >  0.02
|   |   |   |   |   |   |--- class: False
|   |   |--- nBondsD2 >  0.07
|   |   |   |--- minHCsatu <= 0.46
|   |   |   |   |--- GATS4p <= 0.70
|   |   |   |   |   |--- Wnu1.unity <= 0.06
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- Wnu1.unity >  0.06
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- GATS4p >  0.70
|   |   |   |   |   |--- MOMI-XY_y <= 0.09
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- MOMI-XY_y >  0.09
|   |   |   |   |   |   |--- class: True
|   |   |   |--- minHCsatu >  0.46
|   |   |   |   |--- maxHBa <= 0.81
|   |   |   |   |   |--- SpMin4_Bhm <= 0.45
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- SpMin4_Bhm >  0.45
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- maxHBa >  0.81
|   |   |   |   |   |--- ATSC2v <= 0.36
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- ATSC2v >  0.36
|   |   |   |   |   |   |--- class: True
|   |--- minHBint2 >  0.74
|   |   |--- AATS3v <= 0.44
|   |   |   |--- BCUTp-1l_x <= 0.40
|   |   |   |   |--- PPSA-1_y <= 0.03
|   |   |   |   |   |--- class: False
|   |   |   |   |--- PPSA-1_y >  0.03
|   |   |   |   |   |--- class: False
|   |   |   |--- BCUTp-1l_x >  0.40
|   |   |   |   |--- class: True
|   |   |--- AATS3v >  0.44
|   |   |   |--- GATS5m <= 0.49
|   |   |   |   |--- GATS1e <= 0.33
|   |   |   |   |   |--- minwHBa <= 0.22
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- minwHBa >  0.22
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- GATS1e >  0.33
|   |   |   |   |   |--- SC-6_x <= 0.08
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- SC-6_x >  0.08
|   |   |   |   |   |   |--- class: True
|   |   |   |--- GATS5m >  0.49
|   |   |   |   |--- MATS3i <= 0.67
|   |   |   |   |   |--- minssO <= 0.77
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- minssO >  0.77
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- MATS3i >  0.67
|   |   |   |   |   |--- class: True
|--- AATS7v >  0.51
|   |--- minHBd <= 0.49
|   |   |--- AATS7e <= 0.62
|   |   |   |--- RotBtFrac <= 0.20
|   |   |   |   |--- class: False
|   |   |   |--- RotBtFrac >  0.20
|   |   |   |   |--- GATS8v <= 0.45
|   |   |   |   |   |--- class: False
|   |   |   |   |--- GATS8v >  0.45
|   |   |   |   |   |--- class: True
|   |   |--- AATS7e >  0.62
|   |   |   |--- RotBFrac <= 0.15
|   |   |   |   |--- class: False
|   |   |   |--- RotBFrac >  0.15
|   |   |   |   |--- minHCsatu <= 0.44
|   |   |   |   |   |--- class: False
|   |   |   |   |--- minHCsatu >  0.44
|   |   |   |   |   |--- class: False
|   |--- minHBd >  0.49
|   |   |--- MATS2c <= 0.40
|   |   |   |--- nX <= 0.44
|   |   |   |   |--- GATS4c <= 0.44
|   |   |   |   |   |--- hmax <= 0.32
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- hmax >  0.32
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- GATS4c >  0.44
|   |   |   |   |   |--- GATS2s <= 0.53
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- GATS2s >  0.53
|   |   |   |   |   |   |--- class: True
|   |   |   |--- nX >  0.44
|   |   |   |   |--- class: False
|   |   |--- MATS2c >  0.40
|   |   |   |--- AATS6s <= 0.33
|   |   |   |   |--- ATSC1i <= 0.84
|   |   |   |   |   |--- JGI8 <= 0.35
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- JGI8 >  0.35
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- ATSC1i >  0.84
|   |   |   |   |   |--- MATS6i <= 0.44
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- MATS6i >  0.44
|   |   |   |   |   |   |--- class: False
|   |   |   |--- AATS6s >  0.33
|   |   |   |   |--- C1SP3_x <= 0.01
|   |   |   |   |   |--- class: True
|   |   |   |   |--- C1SP3_x >  0.01
|   |   |   |   |   |--- JGI3 <= 0.36
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- JGI3 >  0.36
|   |   |   |   |   |   |--- class: False
```

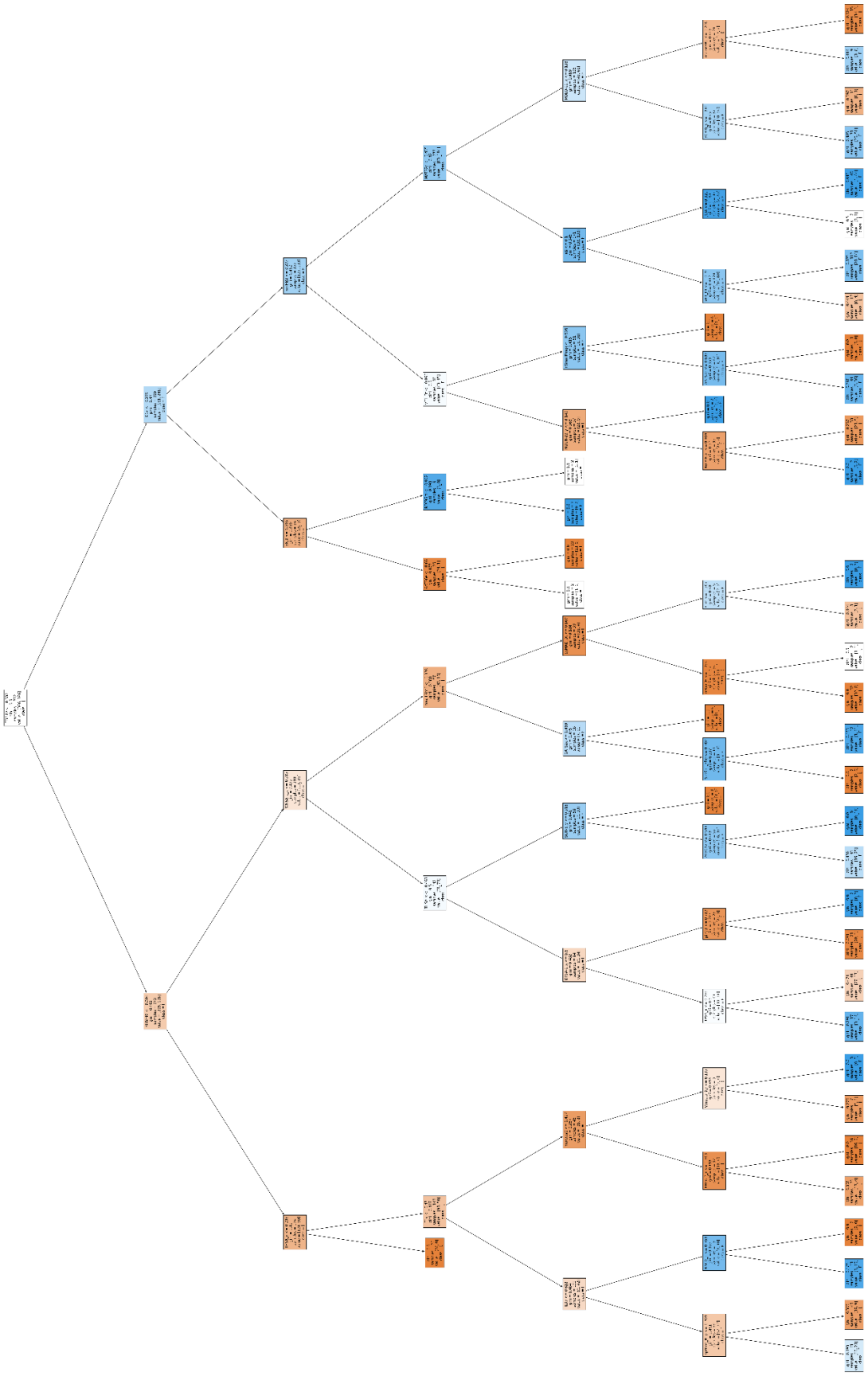


Figure 38: Decision tree of TOX2, due to the size of the tree this only gives an idea of how it looks. For the values look at the text representation beneath.

```
|--- TDB1v <= 0.38
|   |--- nHBint2 <= 0.00
|   |   |--- THSA_x <= 0.29
|   |   |   |--- class: False
|   |   |--- THSA_x >  0.29
|   |   |   |--- E1v <= 0.47
|   |   |   |   |--- E3v <= 0.62
|   |   |   |   |   |--- SpMin1_Bhp <= 0.77
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- SpMin1_Bhp >  0.77
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- E3v >  0.62
|   |   |   |   |   |--- RNCG_x <= 0.76
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- RNCG_x >  0.76
|   |   |   |   |   |   |--- class: False
|   |   |   |--- E1v >  0.47
|   |   |   |   |--- maxaasC <= 0.43
|   |   |   |   |   |--- FNSA-2_x <= 0.86
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- FNSA-2_x >  0.86
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- maxaasC >  0.43
|   |   |   |   |   |--- Wnu1.unity <= 0.14
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- Wnu1.unity >  0.14
|   |   |   |   |   |   |--- class: True
|   |--- nHBint2 >  0.00
|   |   |--- WNSA-1_x <= 0.28
|   |   |   |--- THSA_x <= 0.46
|   |   |   |   |--- DPSA-1_x <= 0.50
|   |   |   |   |   |--- TPSA_x <= 0.29
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- TPSA_x >  0.29
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- DPSA-1_x >  0.50
|   |   |   |   |   |--- piPC3 <= 0.79
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- piPC3 >  0.79
|   |   |   |   |   |   |--- class: True
|   |   |   |--- THSA_x >  0.46
|   |   |   |   |--- SCH-5_y <= 0.32
|   |   |   |   |   |--- ATSC4v <= 0.54
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- ATSC4v >  0.54
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- SCH-5_y >  0.32
|   |   |   |   |   |--- class: False
|   |   |--- WNSA-1_x >  0.28
|   |   |   |--- maxHBint6 <= 0.38
|   |   |   |   |--- GATS2c <= 0.50
|   |   |   |   |   |--- SpMin1_Bhe <= 0.71
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- SpMin1_Bhe >  0.71
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- GATS2c >  0.50
|   |   |   |   |   |--- class: False
|   |   |   |--- maxHBint6 >  0.38
|   |   |   |   |--- SpMAD_Dt <= 0.81
|   |   |   |   |   |--- nHaaCH <= 0.29
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- nHaaCH >  0.29
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- SpMAD_Dt >  0.81
|   |   |   |   |   |--- P1m <= 0.57
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- P1m >  0.57
|   |   |   |   |   |   |--- class: True
|--- TDB1v >  0.38
|   |--- E1v <= 0.22
|   |   |--- SIC1 <= 0.59
|   |   |   |--- AATS6e <= 0.65
|   |   |   |   |--- class: False
|   |   |   |--- AATS6e >  0.65
|   |   |   |   |--- class: False
|   |   |--- SIC1 >  0.59
|   |   |   |--- RDF95m <= 0.00
|   |   |   |   |--- class: True
|   |   |   |--- RDF95m >  0.00
|   |   |   |   |--- class: False
|   |--- E1v >  0.22
|   |   |--- minHBd <= 0.40
|   |   |   |--- VE1_Dt <= 0.07
|   |   |   |   |--- MDEN-23_y <= 0.15
|   |   |   |   |   |--- RotBtFrac <= 0.10
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- RotBtFrac >  0.10
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- MDEN-23_y >  0.15
|   |   |   |   |   |--- class: True
|   |   |   |--- VE1_Dt >  0.07
|   |   |   |   |--- nSmallRings <= 0.54
|   |   |   |   |   |--- AATS7e <= 0.63
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- AATS7e >  0.63
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- nSmallRings >  0.54
|   |   |   |   |   |--- class: False
|   |   |--- minHBd >  0.40
|   |   |   |--- MATS2c <= 0.45
|   |   |   |   |--- nS <= 0.10
|   |   |   |   |   |--- VE1_Dt <= 0.02
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- VE1_Dt >  0.02
|   |   |   |   |   |   |--- class: True
|   |   |   |   |--- nS >  0.10
|   |   |   |   |   |--- JGI4 <= 0.32
|   |   |   |   |   |   |--- class: False
|   |   |   |   |   |--- JGI4 >  0.32
|   |   |   |   |   |   |--- class: True
|   |   |   |--- MATS2c >  0.45
|   |   |   |   |--- MDEO-11_x <= 0.14
|   |   |   |   |   |--- SCH-6_y <= 0.34
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- SCH-6_y >  0.34
|   |   |   |   |   |   |--- class: False
|   |   |   |   |--- MDEO-11_x >  0.14
|   |   |   |   |   |--- maxwHBa <= 0.27
|   |   |   |   |   |   |--- class: True
|   |   |   |   |   |--- maxwHBa >  0.27
|   |   |   |   |   |   |--- class: False
```

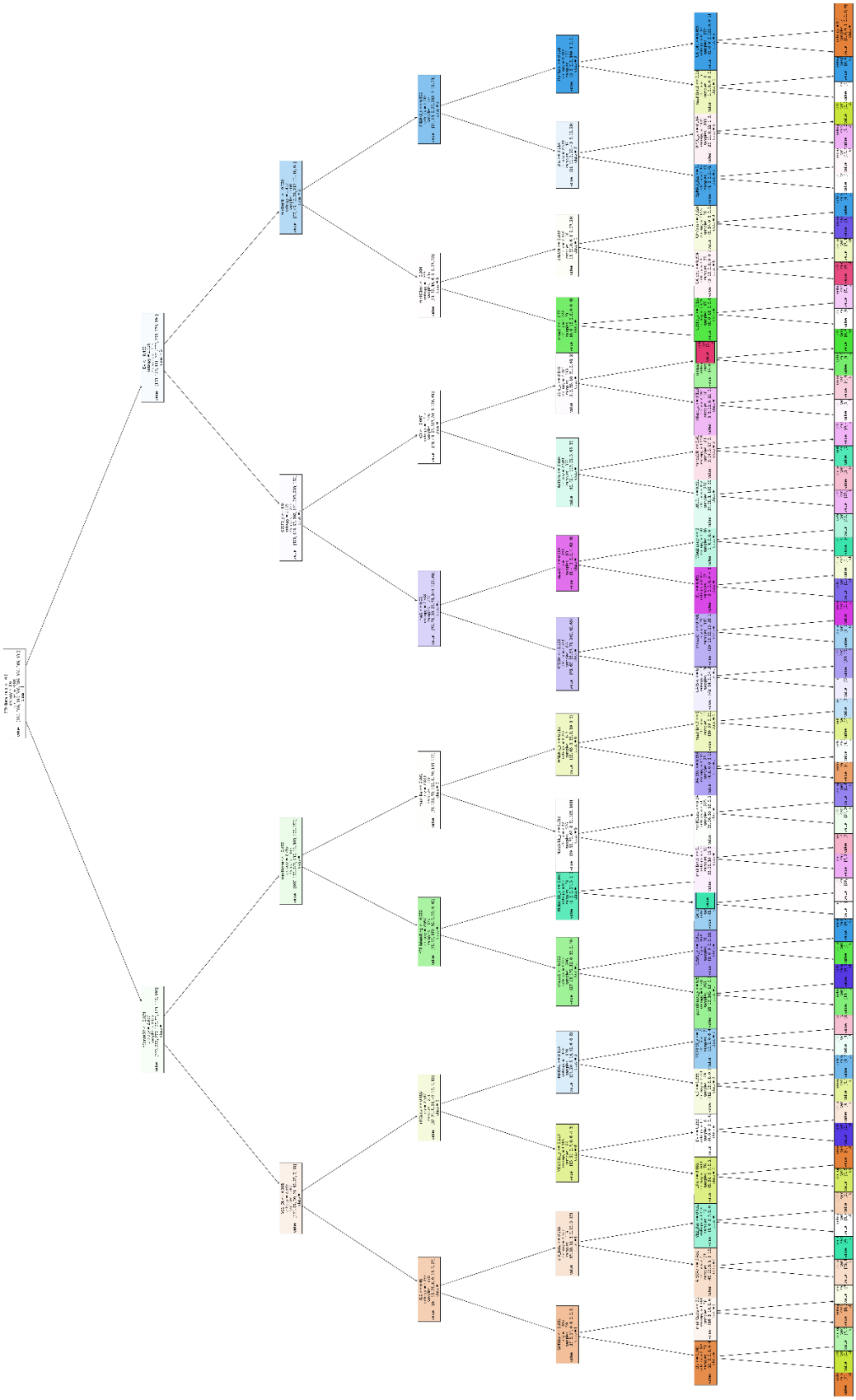


Figure 39: Decision tree of PAT_S, due to the size of the tree this only gives an idea of how it looks. For the values look at the text representation beneath.

```
|--- ETA_Beta_ns_d <= 0.00
|   |--- nBondsD2 <= 0.07
|   |   |--- VE1_Dt <= 0.10
|   |   |   |--- IC1 <= 0.48
|   |   |   |   |--- GATS5p <= 0.50
|   |   |   |   |   |--- Du <= 0.95
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Du >  0.95
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |--- GATS5p >  0.50
|   |   |   |   |   |--- minHCsats <= 0.35
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |--- minHCsats >  0.35
|   |   |   |   |   |   |--- class: 0
|   |   |   |--- IC1 >  0.48
|   |   |   |   |--- ETA_Beta <= 0.28
|   |   |   |   |   |--- ATSC4v <= 0.45
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |   |--- ATSC4v >  0.45
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- ETA_Beta >  0.28
|   |   |   |   |   |--- VE2_Dze <= 0.03
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- VE2_Dze >  0.03
|   |   |   |   |   |   |--- class: 5
|   |   |--- VE1_Dt >  0.10
|   |   |   |--- nHCsats <= 0.09
|   |   |   |   |--- MDEC-12_y <= 0.12
|   |   |   |   |   |--- L3m <= 0.09
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- L3m >  0.09
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |--- MDEC-12_y >  0.12
|   |   |   |   |   |--- Dv <= 0.28
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Dv >  0.28
|   |   |   |   |   |   |--- class: 11
|   |   |   |--- nHCsats >  0.09
|   |   |   |   |--- MATS4c <= 0.53
|   |   |   |   |   |--- IC0 <= 0.39
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- IC0 >  0.39
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |--- MATS4c >  0.53
|   |   |   |   |   |   |   |--- MOMI-XY_y <= 0.15
|   |   |   |   |   |   |   |   |--- class: 9
|   |   |   |   |   |   |   |--- MOMI-XY_y >  0.15
|   |   |   |   |   |   |   |   |--- class: 7
|   |--- nBondsD2 >  0.07
|   |   |--- maxHother <= 0.49
|   |   |   |--- nF9HeteroRing <= 0.00
|   |   |   |   |--- khs.aaS <= 0.01
|   |   |   |   |   |   |   |   |   |--- geomShape_y <=
0.67
|   |   |   |   |   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |   |   |   |   |--- geomShape_y >  0.67
|   |   |   |   |   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |   |   |   |--- khs.aaS >  0.01
|   |   |   |   |   |   |   |   |   |--- C3SP2_y <= 0.01
|   |   |   |   |   |   |   |   |   |   |--- class: 11
|   |   |   |   |   |   |   |   |   |--- C3SP2_y >  0.01
|   |   |   |   |   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |   |   |--- nF9HeteroRing >  0.00
|   |   |   |   |   |   |   |   |--- MDEO-11_x <= 0.05
|   |   |   |   |   |   |   |   |   |--- GATS5i <= 0.40
|   |   |   |   |   |   |   |   |   |   |--- class: 9
|   |   |   |   |   |   |   |   |   |--- GATS5i >  0.40
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- MDEO-11_x >  0.05
|   |   |   |   |   |   |   |   |   |--- class: 7
|   |   |   |--- maxHother >  0.49
|   |   |   |   |--- maxHBa <= 0.80
|   |   |   |   |   |--- MDEO-11_y <= 0.06
|   |   |   |   |   |   |--- minHBint5 <= 0.35
|   |   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |   |--- minHBint5 >  0.35
|   |   |   |   |   |   |   |--- class: 13
|   |   |   |   |   |--- MDEO-11_y >  0.06
|   |   |   |   |   |   |--- minHCsats <= 0.34
|   |   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |--- minHCsats >  0.34
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |--- maxHBa >  0.80
|   |   |   |   |   |--- WNSA-1_x <= 0.18
|   |   |   |   |   |   |--- AATS6v <= 0.56
|   |   |   |   |   |   |   |--- class: 11
|   |   |   |   |   |   |--- AATS6v >  0.56
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- WNSA-1_x >  0.18
|   |   |   |   |   |--- maxHBint9 <= 0.30
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- maxHBint9 >  0.30
|   |   |   |   |   |   |--- class: 3
|--- ETA_Beta_ns_d >  0.00
|   |--- Dv <= 0.43
|   |   |--- C3SP3_y <= 0.00
|   |   |   |--- nssS <= 0.01
|   |   |   |   |--- ATSC0i <= 0.19
|   |   |   |   |   |--- GATS7c <= 0.28
|   |   |   |   |   |   |--- class: 9
|   |   |   |   |   |--- GATS7c >  0.28
|   |   |   |   |   |   |--- class: 11
|   |   |   |   |--- ATSC0i >  0.19
|   |   |   |   |   |--- minaaCH <= 0.70
|   |   |   |   |   |   |--- class: 11
|   |   |   |   |   |--- minaaCH >  0.70
|   |   |   |   |   |   |--- class: 9
|   |   |   |--- nssS >  0.01
|   |   |   |   |--- nsssCH <= 0.00
|   |   |   |   |   |--- Du <= 0.47
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |   |--- Du >  0.47
|   |   |   |   |   |   |--- class: 11
|   |   |   |   |--- nsssCH >  0.00
|   |   |   |   |   |--- Wnu2.unity <= 0.64
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |   |--- Wnu2.unity >  0.64
|   |   |   |   |   |   |--- class: 7
|   |   |--- C3SP3_y >  0.00
|   |   |   |--- nCl <= 0.01
|   |   |   |   |--- AATS7e <= 0.60
|   |   |   |   |   |--- TDB10u <= 0.81
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- TDB10u >  0.81
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |--- AATS7e >  0.60
|   |   |   |   |   |--- mindsCH <= 0.42
|   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |--- mindsCH >  0.42
|   |   |   |   |   |   |--- class: 7
|   |   |   |--- nCl >  0.01
|   |   |   |   |--- SC-5_x <= 0.18
|   |   |   |   |   |--- nBase_x <= 0.12
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |   |--- nBase_x >  0.12
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |--- SC-5_x >  0.18
|   |   |   |   |   |--- minsssN <= 0.02
|   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |--- minsssN >  0.02
|   |   |   |   |   |   |--- class: 5
|--- Dv >  0.43
|   |--- minHssNH <= 0.33
|   |   |--- minHBint4 <= 0.30
|   |   |   |--- minsOH <= 0.77
|   |   |   |   |--- class: 15
|   |   |   |--- minsOH >  0.77
|   |   |   |   |--- C2SP3_x <= 0.15
|   |   |   |   |   |--- class: 5
|   |   |   |   |--- C2SP3_x >  0.15
|   |   |   |   |   |--- class: 5
|   |   |--- minHBint4 >  0.30
|   |   |   |--- SHsOH <= 0.10
|   |   |   |   |--- GATS5e <= 0.39
|   |   |   |   |   |--- class: 13
|   |   |   |   |--- GATS5e >  0.39
|   |   |   |   |   |--- class: 15
|   |   |   |--- SHsOH >  0.10
|   |   |   |   |--- RDF75m <= 0.33
|   |   |   |   |   |--- class: 3
|   |   |   |   |--- RDF75m >  0.33
|   |   |   |   |   |--- class: 11
|   |--- minHssNH >  0.33
|   |   |--- FNSA-3_x <= 0.80
|   |   |   |--- Dm <= 0.21
|   |   |   |   |   |--- SpMAD_Dzs <= 0.49
|   |   |   |   |   |   |--- class: 9
|   |   |   |   |   |--- SpMAD_Dzs >  0.49
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- Dm >  0.21
|   |   |   |   |   |--- RPCG_x <= 0.20
|   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |--- RPCG_x >  0.20
|   |   |   |   |   |   |--- class: 13
|   |   |--- FNSA-3_x >  0.80
|   |   |   |--- RDF15m <= 0.23
|   |   |   |   |--- maxHBint2 <= 0.16
|   |   |   |   |   |--- class: 3
|   |   |   |   |--- maxHBint2 >  0.16
|   |   |   |   |   |--- class: 0
|   |   |   |--- RDF15m >  0.23
|   |   |   |   |--- GATS4e <= 0.49
|   |   |   |   |   |--- class: 9
|   |   |   |   |--- GATS4e >  0.49
|   |   |   |   |   |--- class: 0
```

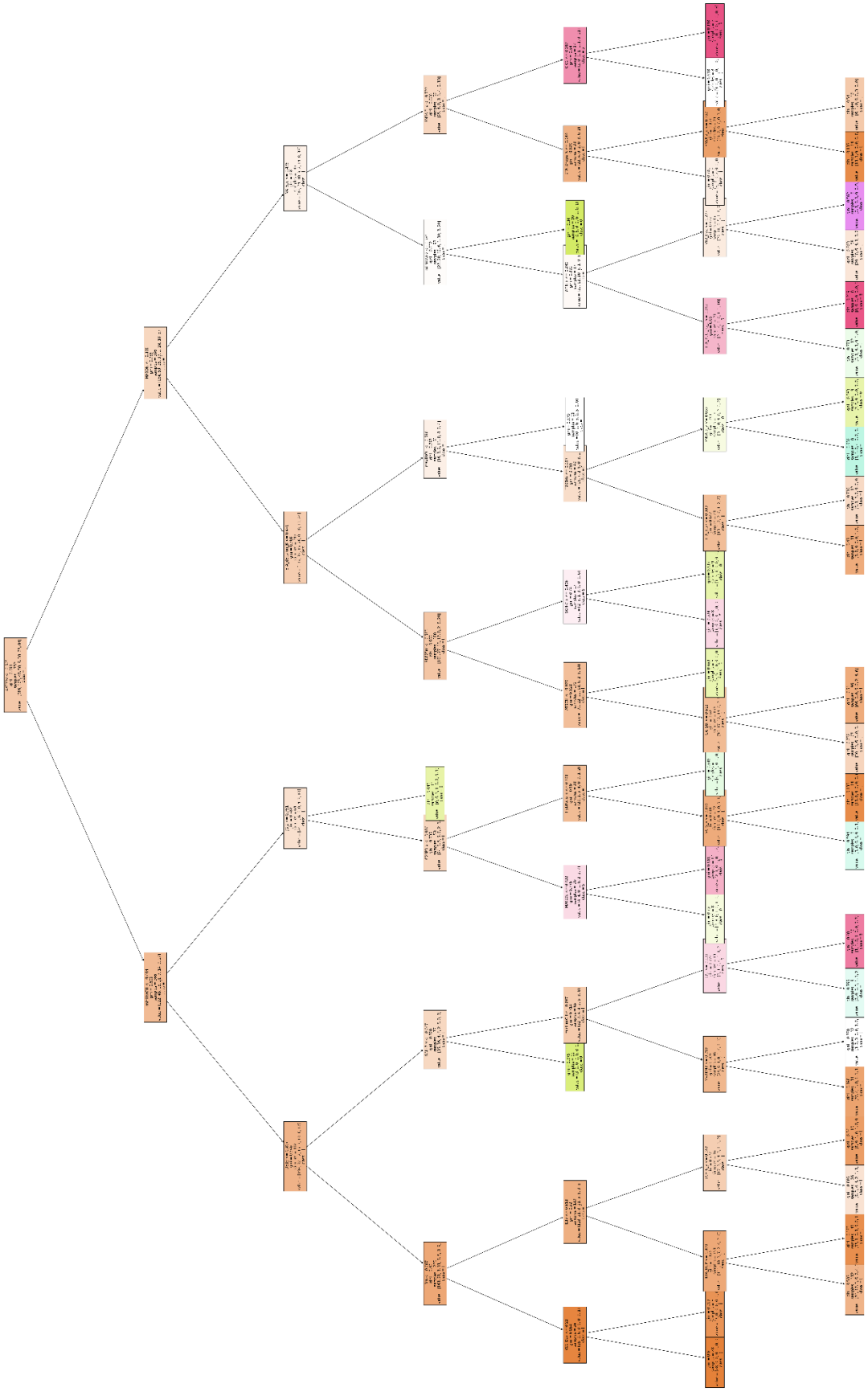


Figure 40: Decision tree of PAT_nS, due to the size of the tree this only gives an idea of how it looks. For the values look at the text representation beneath.

```
|--- AATS7v <= 0.51
|   |--- minHdsCH <= 0.40
|   |   |--- TDB1v <= 0.41
|   |   |   |--- L3m <= 0.07
|   |   |   |   |--- VE1_Dzv <= 0.22
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- VE1_Dzv >  0.22
|   |   |   |   |   |--- class: 0
|   |   |   |--- L3m >  0.07
|   |   |   |   |--- E3v <= 0.62
|   |   |   |   |   |--- ETA_EtaP <= 0.47
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- ETA_EtaP >  0.47
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- E3v >  0.62
|   |   |   |   |   |--- SCH-6_y <= 0.24
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- SCH-6_y >  0.24
|   |   |   |   |   |   |--- class: 0
|   |   |--- TDB1v >  0.41
|   |   |   |--- SCH-7_x <= 0.05
|   |   |   |   |--- class: 3
|   |   |   |--- SCH-7_x >  0.05
|   |   |   |   |--- minHaaCH <= 0.67
|   |   |   |   |   |--- MAXDN2 <= 0.30
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- MAXDN2 >  0.30
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- minHaaCH >  0.67
|   |   |   |   |   |--- L3u <= 0.19
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- L3u >  0.19
|   |   |   |   |   |   |--- class: 15
|   |--- minHdsCH >  0.40
|   |   |--- JGI2 <= 0.75
|   |   |   |--- C2SP3_x <= 0.09
|   |   |   |   |--- MATS2c <= 0.43
|   |   |   |   |   |--- class: 3
|   |   |   |   |--- MATS2c >  0.43
|   |   |   |   |   |--- class: 15
|   |   |   |--- C2SP3_x >  0.09
|   |   |   |   |--- HybRatio_x <= 0.78
|   |   |   |   |   |--- VC-5_x <= 0.07
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- VC-5_x >  0.07
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- HybRatio_x >  0.78
|   |   |   |   |   |--- class: 5
|   |   |--- JGI2 >  0.75
|   |   |   |--- class: 3
|--- AATS7v >  0.51
|   |--- MAXDN <= 0.34
|   |   |--- ETA_dEpsilon_B <= 0.47
|   |   |   |--- RDF25m <= 0.58
|   |   |   |   |--- ATSC1i <= 0.93
|   |   |   |   |   |--- GATS8i <= 0.62
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- GATS8i >  0.62
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- ATSC1i >  0.93
|   |   |   |   |   |--- class: 3
|   |   |   |--- RDF25m >  0.58
|   |   |   |   |--- SCH-7_x <= 0.41
|   |   |   |   |   |--- class: 15
|   |   |   |   |--- SCH-7_x >  0.41
|   |   |   |   |   |--- class: 3
|   |   |--- ETA_dEpsilon_B >  0.47
|   |   |   |--- maxHsOH <= 0.60
|   |   |   |   |--- TDB10s <= 0.19
|   |   |   |   |   |--- ETA_EtaP <= 0.31
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- ETA_EtaP >  0.31
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- TDB10s >  0.19
|   |   |   |   |   |--- FNSA-3_y <= 0.66
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- FNSA-3_y >  0.66
|   |   |   |   |   |   |--- class: 3
|   |   |   |--- maxHsOH >  0.60
|   |   |   |   |--- class: 7
|   |--- MAXDN >  0.34
|   |   |--- MATS2c <= 0.47
|   |   |   |--- minHsNH2 <= 0.58
|   |   |   |   |--- AMR_x <= 0.25
|   |   |   |   |   |--- ETA_Beta_ns_d <= 0.06
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |--- ETA_Beta_ns_d >  0.06
|   |   |   |   |   |   |--- class: 15
|   |   |   |   |--- AMR_x >  0.25
|   |   |   |   |   |--- VE2_DzZ <= 0.28
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- VE2_DzZ >  0.28
|   |   |   |   |   |   |--- class: 13
|   |   |   |--- minHsNH2 >  0.58
|   |   |   |   |--- class: 3
|   |   |--- MATS2c >  0.47
|   |   |   |--- PNSA-3_x <= 0.72
|   |   |   |   |--- ETA_Shape_X <= 0.14
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- ETA_Shape_X >  0.14
|   |   |   |   |   |--- FNSA-2_x <= 0.73
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- FNSA-2_x >  0.73
|   |   |   |   |   |   |--- class: 0
|   |   |   |--- PNSA-3_x >  0.72
|   |   |   |   |--- CIC1 <= 0.37
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- CIC1 >  0.37
|   |   |   |   |   |--- class: 15
```

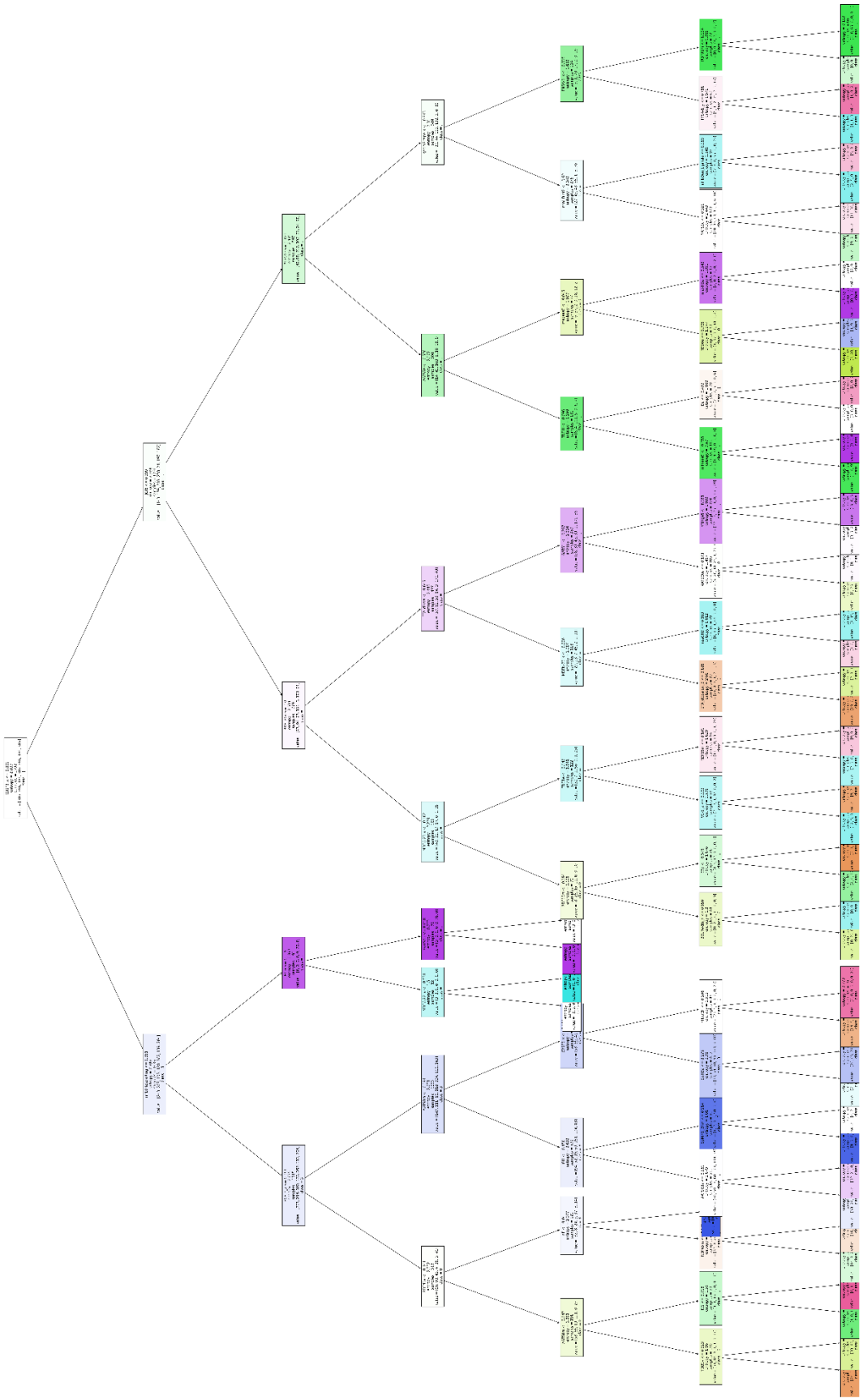


Figure 41: Decision tree of PAT2_S, due to the size of the tree this only gives an idea of how it looks. For the values look at the text representation beneath.

```
|--- C3SP3_x <= 0.00
|   |--- nF10HeteroRing <= 0.36
|   |   |--- SCH-7_y <= 0.06
|   |   |   |--- RDF40v <= 0.14
|   |   |   |   |--- AATS8m <= 0.15
|   |   |   |   |   |--- TDB1v <= 0.22
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- TDB1v >  0.22
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |--- AATS8m >  0.15
|   |   |   |   |   |--- IC2 <= 0.72
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |--- IC2 >  0.72
|   |   |   |   |   |   |--- class: 15
|   |   |   |--- RDF40v >  0.14
|   |   |   |   |--- nF <= 0.04
|   |   |   |   |   |--- RDF65m <= 0.14
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |--- RDF65m >  0.14
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- nF >  0.04
|   |   |   |   |   |--- class: 11
|   |   |--- SCH-7_y >  0.06
|   |   |   |--- AATS8m <= 0.19
|   |   |   |   |--- JGI1 <= 0.46
|   |   |   |   |   |--- AATSC0s <= 0.16
|   |   |   |   |   |   |--- class: 11
|   |   |   |   |   |--- AATSC0s >  0.16
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |--- JGI1 >  0.46
|   |   |   |   |   |--- SpMin6_Bhm <= 0.21
|   |   |   |   |   |   |--- class: 11
|   |   |   |   |   |--- SpMin6_Bhm >  0.21
|   |   |   |   |   |   |--- class: 0
|   |   |   |--- AATS8m >  0.19
|   |   |   |   |--- C2SP3_x <= 0.13
|   |   |   |   |   |--- SwHBa <= 0.27
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- SwHBa >  0.27
|   |   |   |   |   |   |--- class: 11
|   |   |   |   |--- C2SP3_x >  0.13
|   |   |   |   |   |--- nHaaCH <= 0.21
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- nHaaCH >  0.21
|   |   |   |   |   |   |--- class: 15
|   |--- nF10HeteroRing >  0.36
|   |   |--- E1m <= 0.12
|   |   |   |--- MDEC-22_x <= 0.53
|   |   |   |   |--- class: 0
|   |   |   |--- MDEC-22_x >  0.53
|   |   |   |   |--- class: 7
|   |   |--- E1m >  0.12
|   |   |   |--- GATS4p <= 0.66
|   |   |   |   |--- class: 13
|   |   |   |--- GATS4p >  0.66
|   |   |   |   |--- class: 0
|--- C3SP3_x >  0.00
|   |--- JGI6 <= 0.40
|   |   |--- RDF130u <= 0.00
|   |   |   |--- MDEC-23_x <= 0.18
|   |   |   |   |--- RDF15m <= 0.19
|   |   |   |   |   |--- BCUTw-1h_x <= 0.10
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |   |--- BCUTw-1h_x >  0.10
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |--- RDF15m >  0.19
|   |   |   |   |   |--- E2p <= 0.34
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |--- E2p >  0.34
|   |   |   |   |   |   |--- class: 0
|   |   |   |--- MDEC-23_x >  0.18
|   |   |   |   |--- TDB1e <= 0.15
|   |   |   |   |   |--- VC-5_x <= 0.10
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- VC-5_x >  0.10
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- TDB1e >  0.15
|   |   |   |   |   |--- RDF65m <= 0.10
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- RDF65m >  0.10
|   |   |   |   |   |   |--- class: 15
|   |   |--- RDF130u >  0.00
|   |   |   |--- ETA_dBeta <= 0.65
|   |   |   |   |--- MDEN-23_x <= 0.01
|   |   |   |   |   |--- ETA_dEpsilon_B <= 0.19
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- ETA_dEpsilon_B >  0.19
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |--- MDEN-23_x >  0.01
|   |   |   |   |   |--- nssCH2 <= 0.18
|   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |--- nssCH2 >  0.18
|   |   |   |   |   |   |--- class: 7
|   |   |   |--- ETA_dBeta >  0.65
|   |   |   |   |--- GATS1i <= 0.27
|   |   |   |   |   |--- AATSC0s <= 0.17
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |   |--- AATSC0s >  0.17
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |--- GATS1i >  0.27
|   |   |   |   |   |--- nRings6 <= 0.18
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |   |--- nRings6 >  0.18
|   |   |   |   |   |   |--- class: 13
|   |--- JGI6 >  0.40
|   |   |--- ATSC4v <= 0.35
|   |   |   |--- AATS7e <= 0.58
|   |   |   |   |--- TDB1i <= 0.10
|   |   |   |   |   |--- minaasC <= 0.46
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |--- minaasC >  0.46
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |--- TDB1i >  0.10
|   |   |   |   |   |--- Du <= 0.49
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Du >  0.49
|   |   |   |   |   |   |--- class: 15
|   |   |   |--- AATS7e >  0.58
|   |   |   |   |--- maxaasC <= 0.04
|   |   |   |   |   |--- TDB6u <= 0.80
|   |   |   |   |   |   |--- class: 3
|   |   |   |   |   |--- TDB6u >  0.80
|   |   |   |   |   |   |--- class: 11
|   |   |   |   |--- maxaasC >  0.04
|   |   |   |   |   |--- maxHBa <= 0.54
|   |   |   |   |   |   |--- class: 13
|   |   |   |   |   |--- maxHBa >  0.54
|   |   |   |   |   |   |--- class: 0
|   |   |--- ATSC4v >  0.35
|   |   |   |--- geomShape_x <= 0.84
|   |   |   |   |--- minHBint2 <= 0.62
|   |   |   |   |   |--- MATS2c <= 0.32
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |--- MATS2c >  0.32
|   |   |   |   |   |   |--- class: 15
|   |   |   |   |--- minHBint2 >  0.62
|   |   |   |   |   |--- nHBDon_Lipinski <= 0.16
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- nHBDon_Lipinski >  0.16
|   |   |   |   |   |   |--- class: 15
|   |   |   |--- geomShape_x >  0.84
|   |   |   |   |--- PNSA-1_x <= 0.32
|   |   |   |   |   |--- FPSA-1_y <= 0.71
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- FPSA-1_y >  0.71
|   |   |   |   |   |   |--- class: 15
|   |   |   |   |--- PNSA-1_x >  0.32
|   |   |   |   |   |--- RDF95m <= 0.12
|   |   |   |   |   |   |--- class: 5
|   |   |   |   |   |--- RDF95m >  0.12
|   |   |   |   |   |   |--- class
```

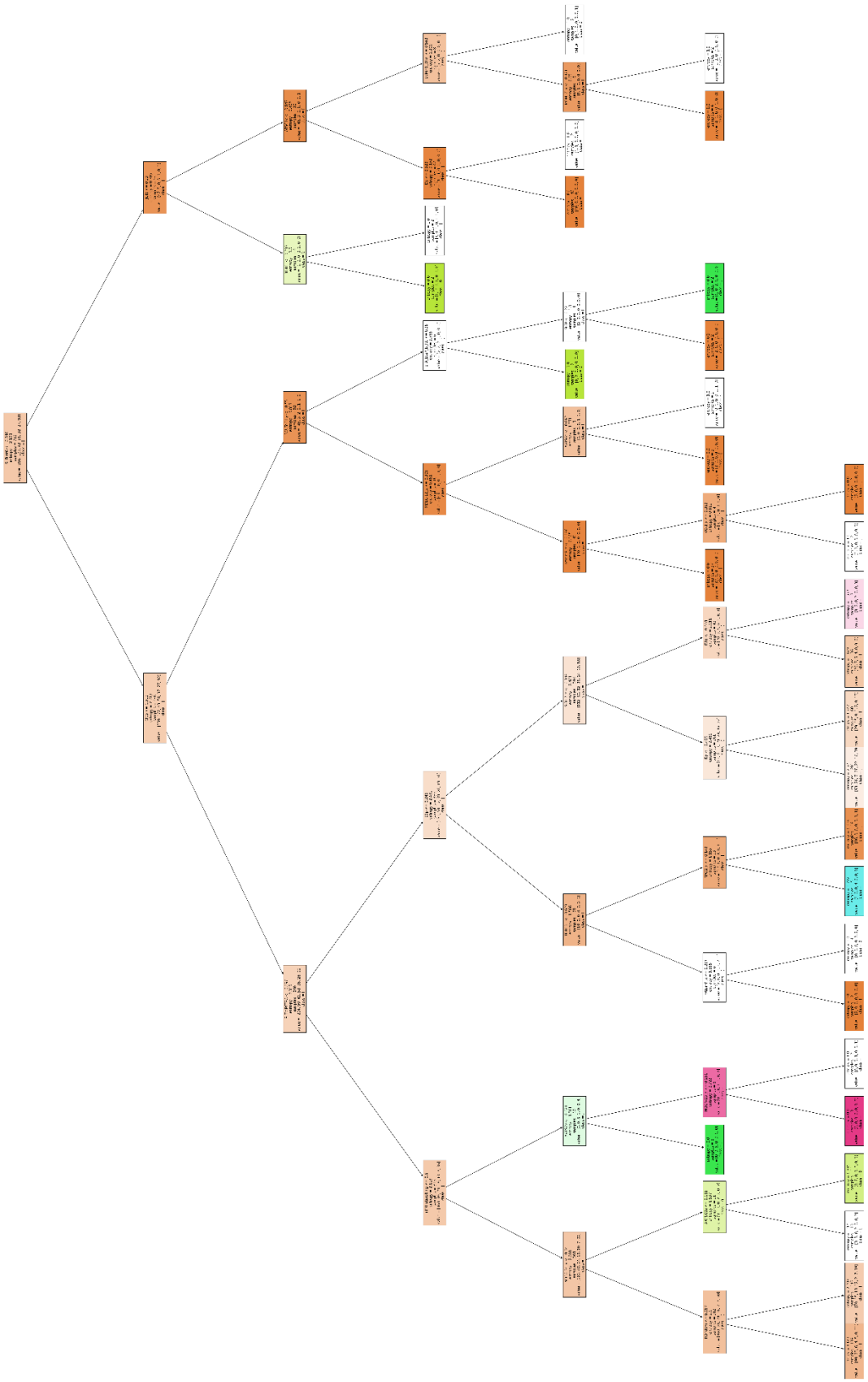


Figure 42: Decision tree of PAT2_nS, due to the size of the tree this only gives an idea of how it looks. For the values look at the text representation beneath.

```
|--- SHBint6 <= 0.35
|   |--- CIC3 <= 0.51
|   |   |--- R_TpiPCTPC <= 0.20
|   |   |   |--- nF7HeteroRing <= 0.50
|   |   |   |   |--- MDEC-34_y <= 0.34
|   |   |   |   |   |--- RDF85m <= 0.06
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- RDF85m >  0.06
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- MDEC-34_y >  0.34
|   |   |   |   |   |--- AATSC0s <= 0.17
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- AATSC0s >  0.17
|   |   |   |   |   |   |--- class: 3
|   |   |   |--- nF7HeteroRing >  0.50
|   |   |   |   |--- AATS7v <= 0.62
|   |   |   |   |   |--- class: 5
|   |   |   |   |--- AATS7v >  0.62
|   |   |   |   |   |--- MDEN-23_x <= 0.25
|   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |--- MDEN-23_x >  0.25
|   |   |   |   |   |   |--- class: 7
|   |   |--- R_TpiPCTPC >  0.20
|   |   |   |--- E1p <= 0.25
|   |   |   |   |--- TDB10u <= 0.63
|   |   |   |   |   |--- nAtomP_x <= 0.26
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- nAtomP_x >  0.26
|   |   |   |   |   |   |--- class: 11
|   |   |   |   |--- TDB10u >  0.63
|   |   |   |   |   |--- RPCG_x <= 0.15
|   |   |   |   |   |   |--- class: 7
|   |   |   |   |   |--- RPCG_x >  0.15
|   |   |   |   |   |   |--- class: 0
|   |   |   |--- E1p >  0.25
|   |   |   |   |--- SCH-7_x <= 0.35
|   |   |   |   |   |--- Dp <= 0.40
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Dp >  0.40
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- SCH-7_x >  0.35
|   |   |   |   |   |--- E3p <= 0.45
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- E3p >  0.45
|   |   |   |   |   |   |--- class: 15
|   |   |--- CIC3 >  0.51
|   |   |   |--- BCUTc-1l_y <= 0.55
|   |   |   |   |--- MDEC-12_x <= 0.30
|   |   |   |   |   |--- nAtomP_x <= 0.19
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- nAtomP_x >  0.19
|   |   |   |   |   |   |--- VC-3_x <= 0.35
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- VC-3_x >  0.35
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- MDEC-12_x >  0.30
|   |   |   |   |   |--- AATS6e <= 0.66
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- AATS6e >  0.66
|   |   |   |   |   |   |--- class: 11
|   |   |   |--- BCUTc-1l_y >  0.55
|   |   |   |   |--- ETA_EtaP_B_RC <= 0.28
|   |   |   |   |   |--- class: 3
|   |   |   |   |--- ETA_EtaP_B_RC >  0.28
|   |   |   |   |   |--- SIC0 <= 0.22
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- SIC0 >  0.22
|   |   |   |   |   |   |--- class: 5
|--- SHBint6 >  0.35
|   |--- JGI6 <= 0.23
|   |   |--- TDB1i <= 0.23
|   |   |   |--- class: 3
|   |   |--- TDB1i >  0.23
|   |   |   |--- class: 0
|   |--- JGI6 >  0.23
|   |   |--- ATSC1v <= 0.85
|   |   |   |--- Dp <= 0.52
|   |   |   |   |--- class: 0
|   |   |   |--- Dp >  0.52
|   |   |   |   |--- class: 0
|   |   |--- ATSC1v >  0.85
|   |   |   |--- SpMin6_Bhm <= 0.56
|   |   |   |   |--- MOMI-Z_x <= 0.42
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- MOMI-Z_x >  0.42
|   |   |   |   |   |--- class: 0
|   |   |   |--- SpMin6_Bhm >  0.56
|   |   |   |   |--- class: 3
```

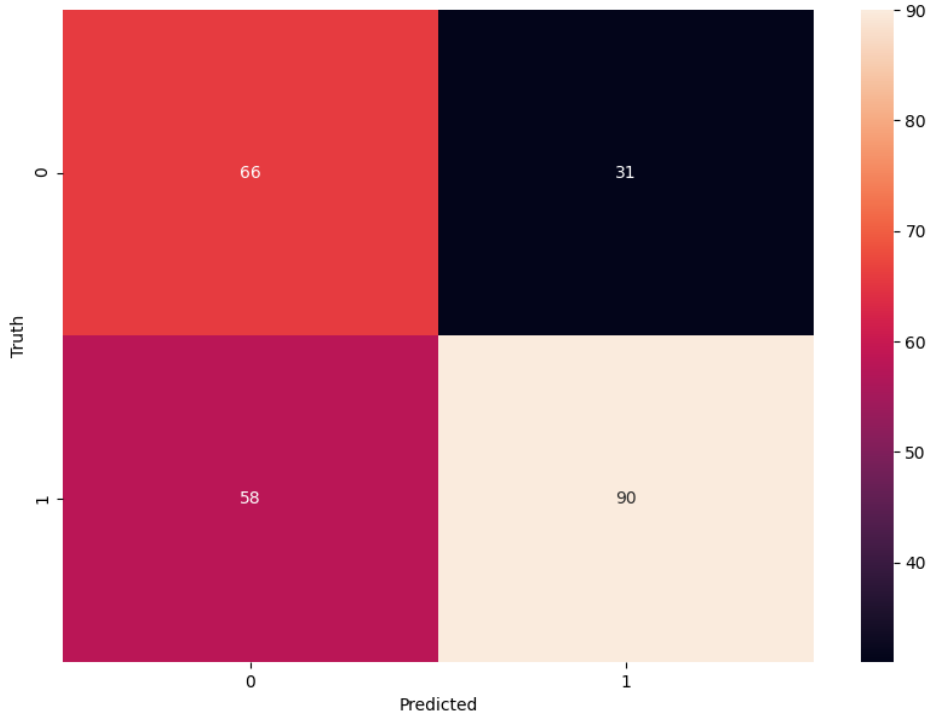


Figure 43: confusion matrix of TOX

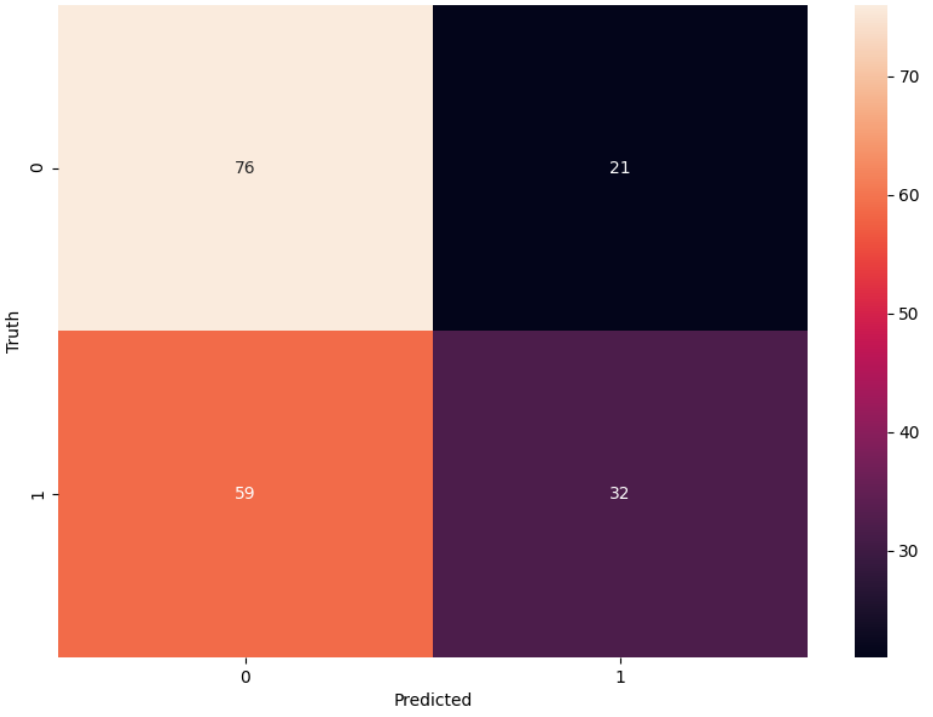


Figure 44: confusion matrix of TOX2

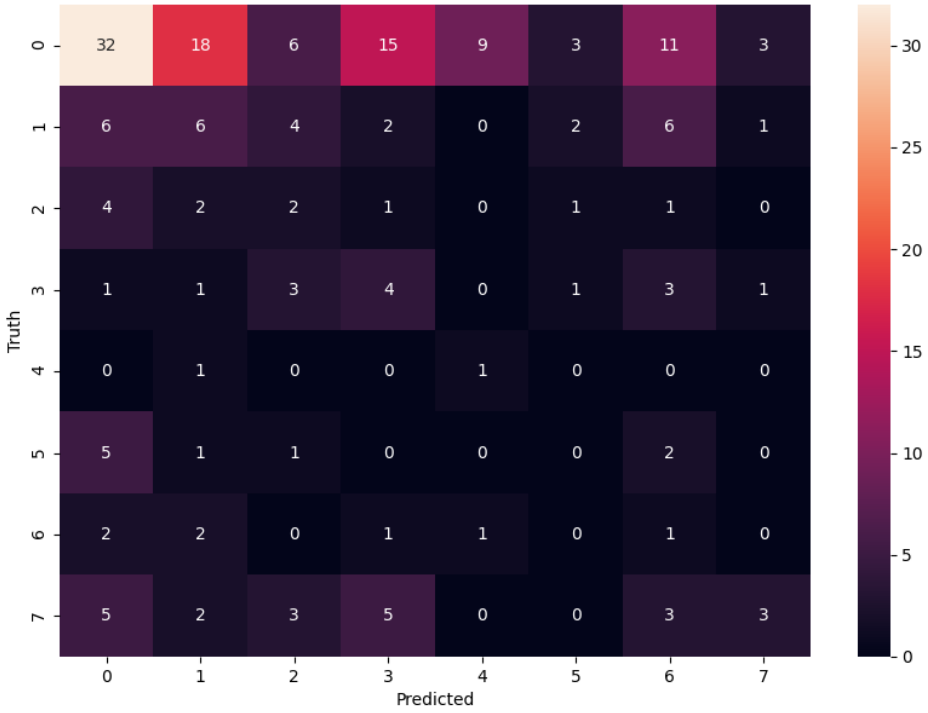


Figure 45: confusion matrix of PAT_S

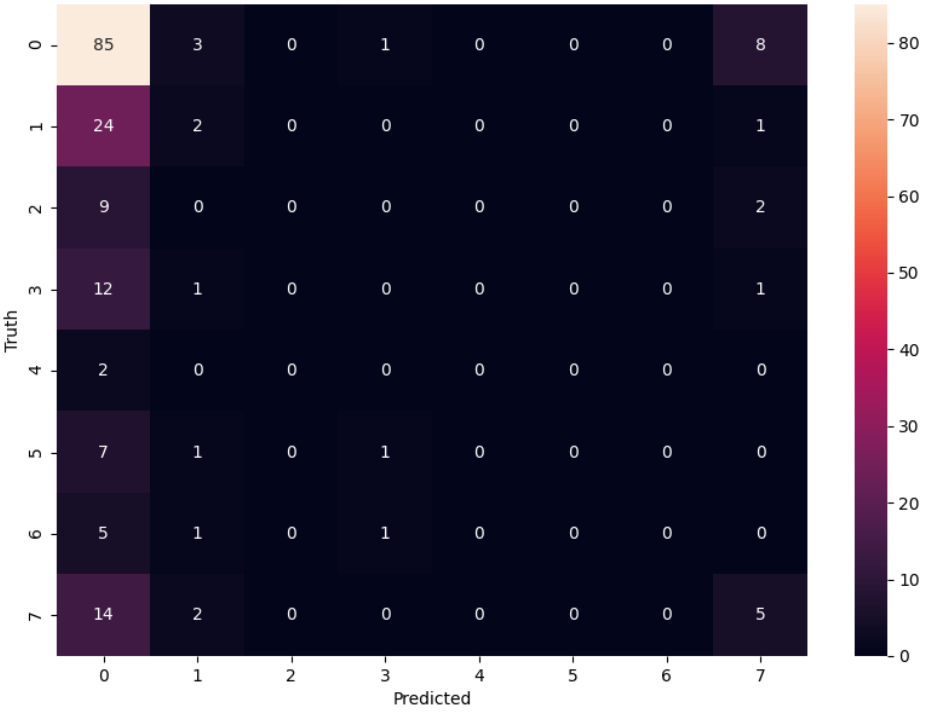


Figure 46: confusion matrix of PAT_nS

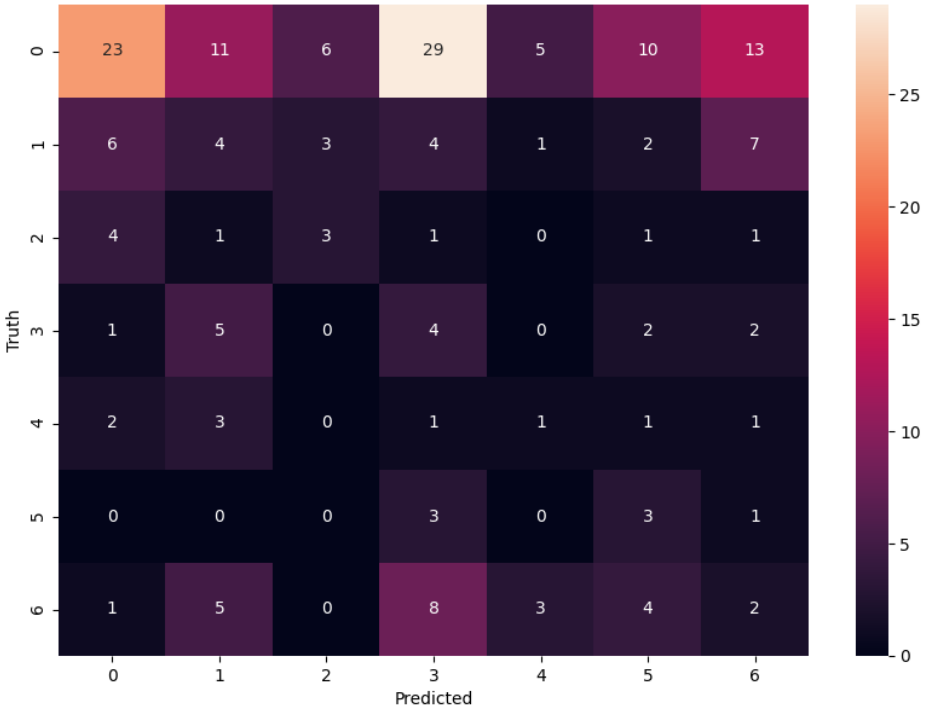


Figure 47: confusion matrix of PAT2_S

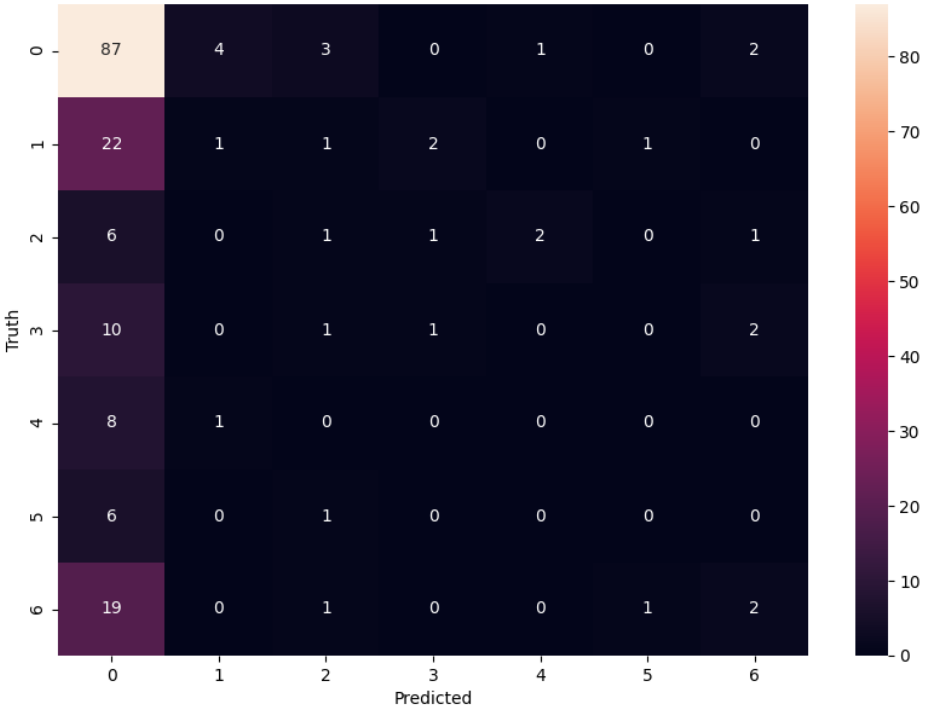


Figure 48: confusion matrix of PAT2_nS

9.4.3 RF

TOX: confusion matrix:
[[61 36]
 [31 117]]

TOX2: confusion matrix:
[[68 29]
 [22 69]]

PAT_S: confusion matrix:
[[78 9 1 1 1 4 1 2]
 [18 3 0 3 0 2 0 1]
 [7 0 2 1 0 0 1 0]
 [5 2 0 5 0 0 0 2]
 [1 0 0 0 0 0 1 0]
 [5 2 1 0 0 1 0 0]
 [4 1 0 0 0 0 1 1]
 [9 2 2 2 0 0 0 6]]

PAT_nS: confusion matrix:
[[88 6 0 0 0 2 1 0]
 [23 3 0 0 0 0 0 1]
 [10 0 1 0 0 0 0 0]
 [9 0 0 4 0 0 0 1]
 [1 0 0 0 0 0 1 0]
 [6 2 0 0 0 1 0 0]
 [5 0 0 1 0 0 1 0]
 [14 2 1 1 0 0 0 3]]

PAT2_S: confusion matrix:
[[82 6 1 0 4 2 2]
 [16 4 0 3 3 0 1]
 [6 0 1 0 0 1 3]
 [6 2 2 3 0 0 1]
 [5 1 1 0 1 0 1]
 [5 1 0 0 0 1 0]
 [11 2 0 1 2 2 5]]

PAT2_nS: confusion matrix:
[[89 5 0 0 2 1 0]
 [23 2 0 0 0 1 1]
 [8 0 0 0 0 1 2]
 [11 0 1 1 0 0 1]
 [7 0 1 0 1 0 0]
 [5 1 0 0 0 1 0]
 [14 2 1 0 0 1 5]]

9.4.4 SVM

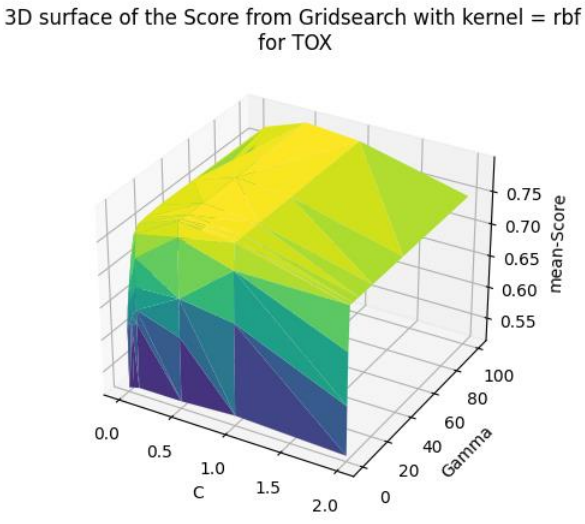


Figure 49: GridsearchCV 3D surface plot representation of TOX

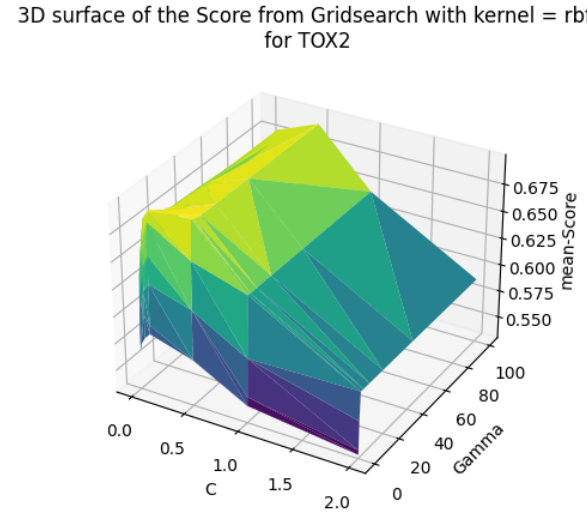


Figure 50: GridsearchCV 3D surface plot representation of TOX2

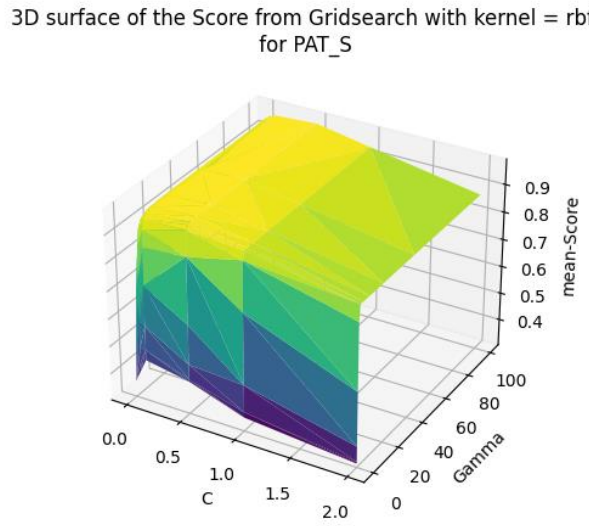


Figure 51: GridsearchCV 3D surface plot representation of PAT_S

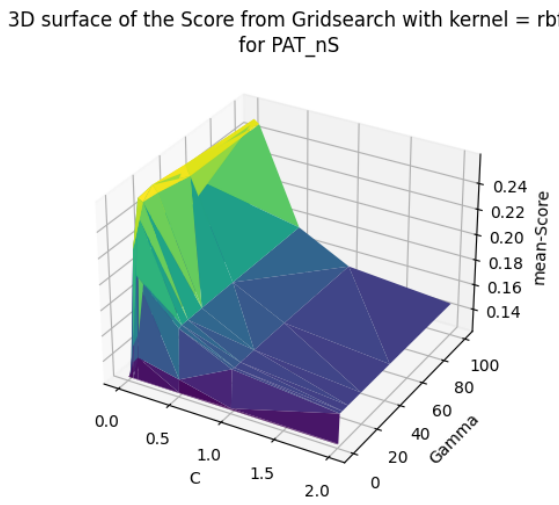


Figure 52: GridsearchCV 3D surface plot representation of PAT_nS

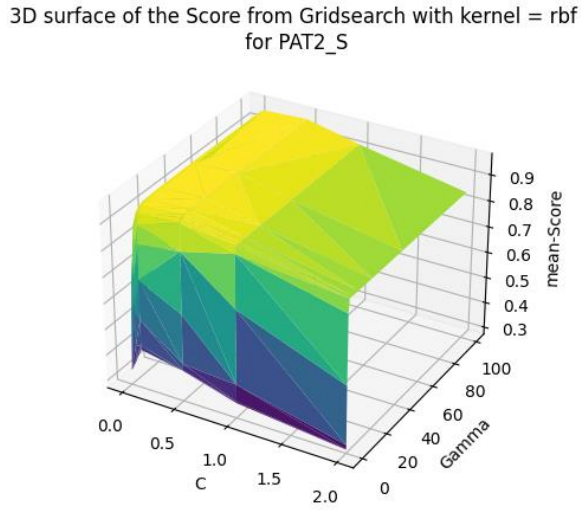


Figure 53: GridsearchCV 3D surface plot representation of PAT2_S

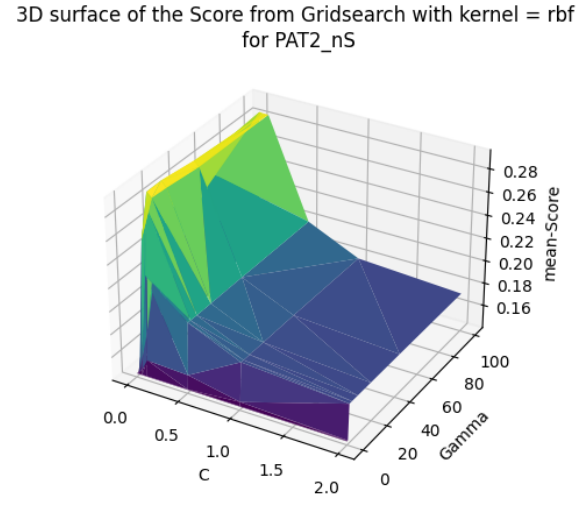


Figure 54: GridsearchCV 3D surface plot representation of PAT2_nS

TOX: confusion matrix:
[[39 58]
 [23 125]]

TOX2: confusion matrix:
[[77 20]
 [48 43]]

PAT_S: confusion matrix:
[[89 4 1 0 0 2 1 0]
 [24 1 0 2 0 0 0 0]
 [10 0 1 0 0 0 0 0]
 [11 0 0 2 0 0 0 1]
 [2 0 0 0 0 0 0 0]
 [9 0 0 0 0 0 0 0]
 [6 0 0 0 0 0 1 0]
 [16 0 1 1 0 0 0 3]]

PAT_nS: confusion matrix:
[[78 9 3 1 1 3 2 0]
 [17 5 0 4 0 1 0 0]
 [9 0 1 0 0 0 0 1]
 [4 3 1 4 0 0 0 2]
 [0 1 0 0 0 0 1 0]
 [7 1 0 0 0 0 0 1]
 [4 0 0 1 0 0 1 1]
 [10 3 0 1 0 0 1 6]]

PAT2_S: confusion matrix:
[[90 3 1 0 2 1 0]
 [25 1 0 1 0 0 0]
 [11 0 0 0 0 0 0]
 [11 1 1 0 0 0 1]
 [9 0 0 0 0 0 0]
 [6 0 0 0 0 1 0]
 [21 0 0 0 0 0 2]]

PAT2_nS: confusion matrix:
[[77 11 2 0 3 2 2]
 [14 7 0 4 1 0 1]
 [7 0 0 0 0 1 3]
 [2 3 2 4 0 0 3]
 [5 1 1 0 0 0 2]
 [4 0 0 1 0 1 1]
 [9 3 1 0 0 2 8]]

9.4.5 kNN

Only the best figures are represented per model (confusion matrix and two best feature visualizations).

TOX

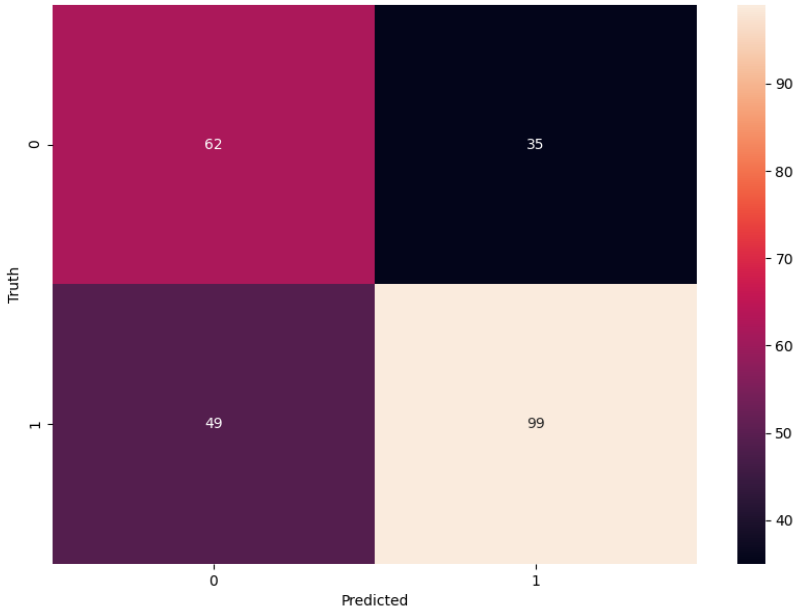


Figure 55: confusion matrix for TOX

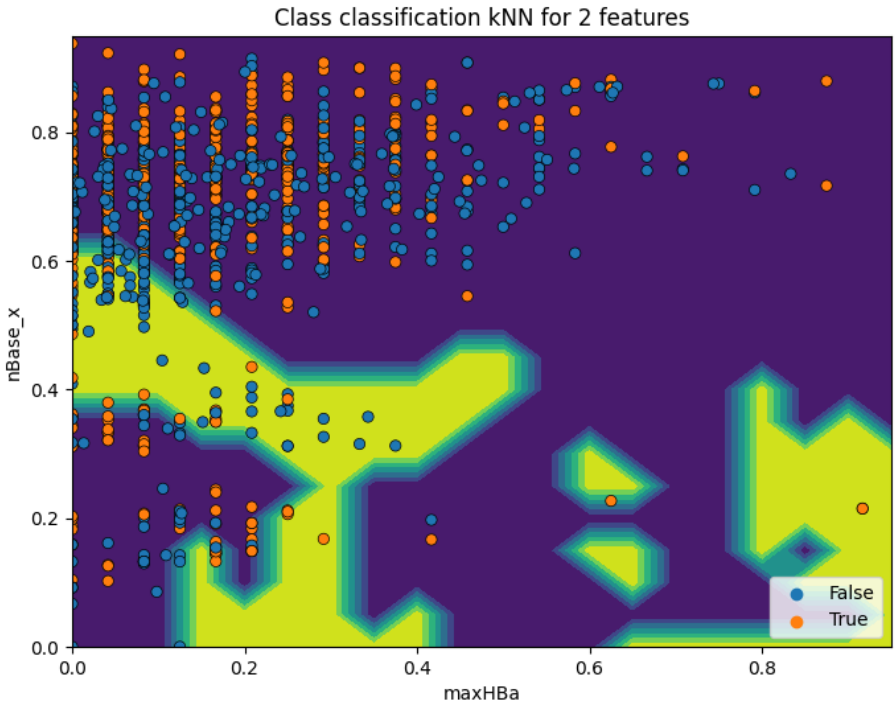


Figure 56: Best Features visualization of TOX

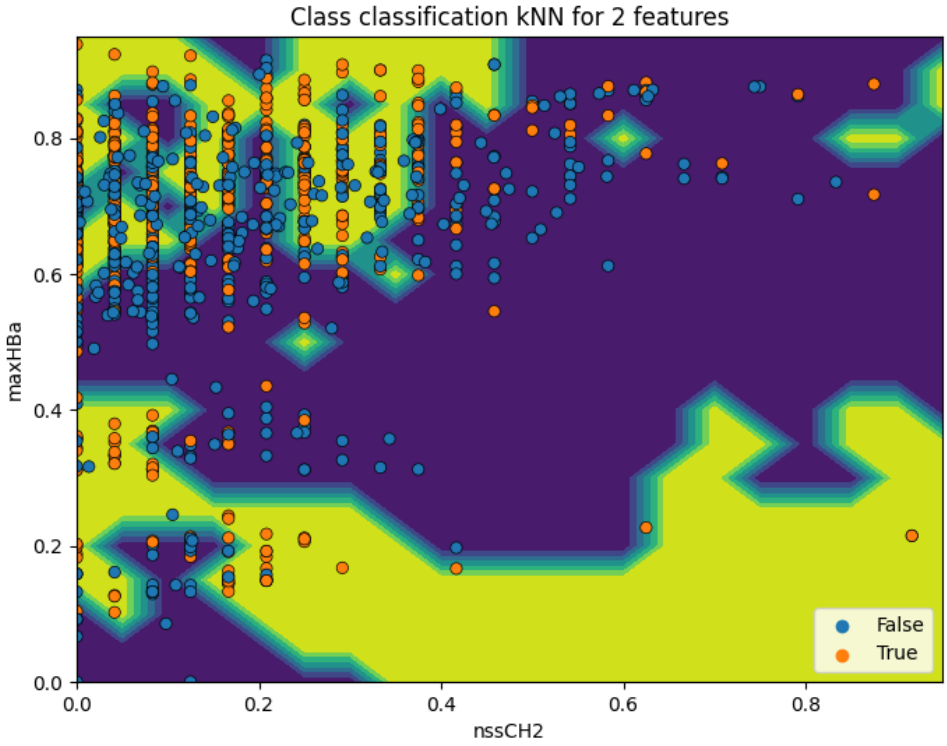


Figure 57: Best Features visualization of TOX

TOX2

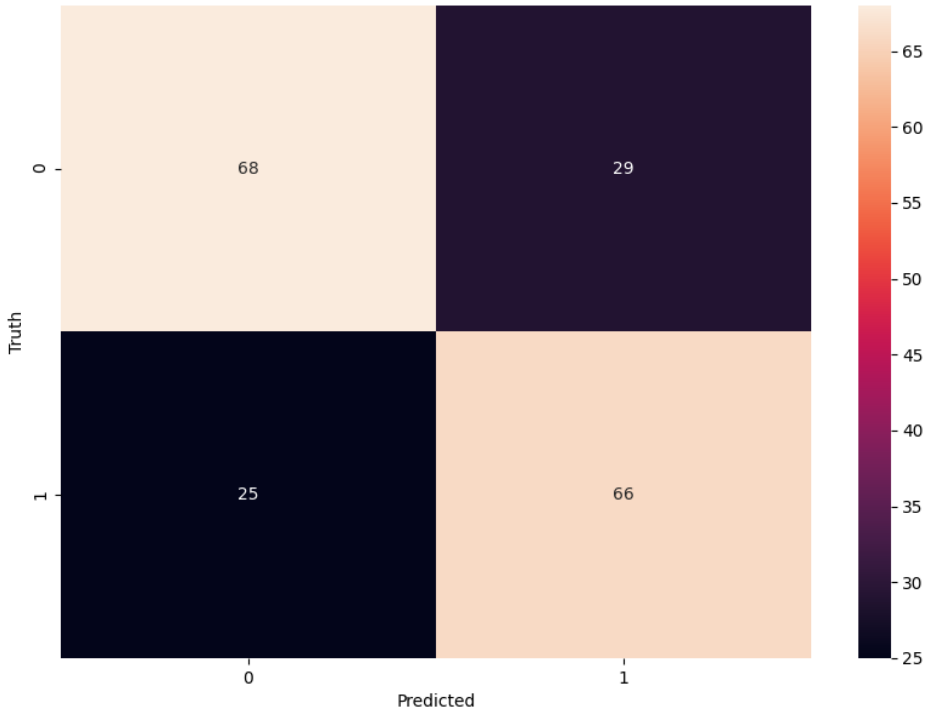


Figure 58: confusion matrix for TOX2

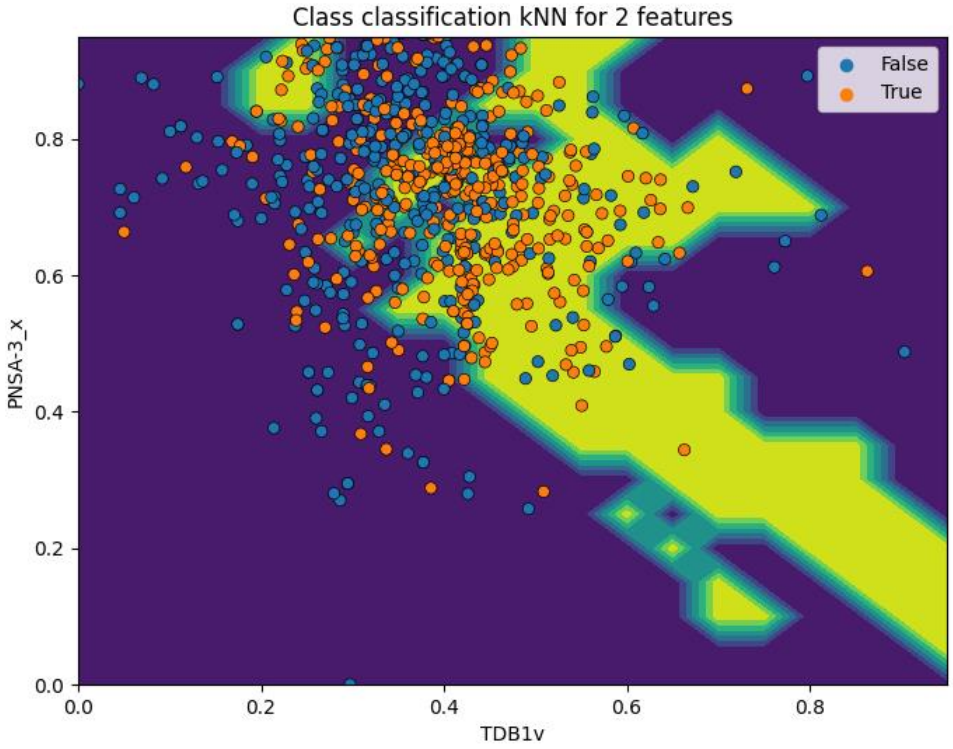


Figure 59: Best Features visualization of TOX2

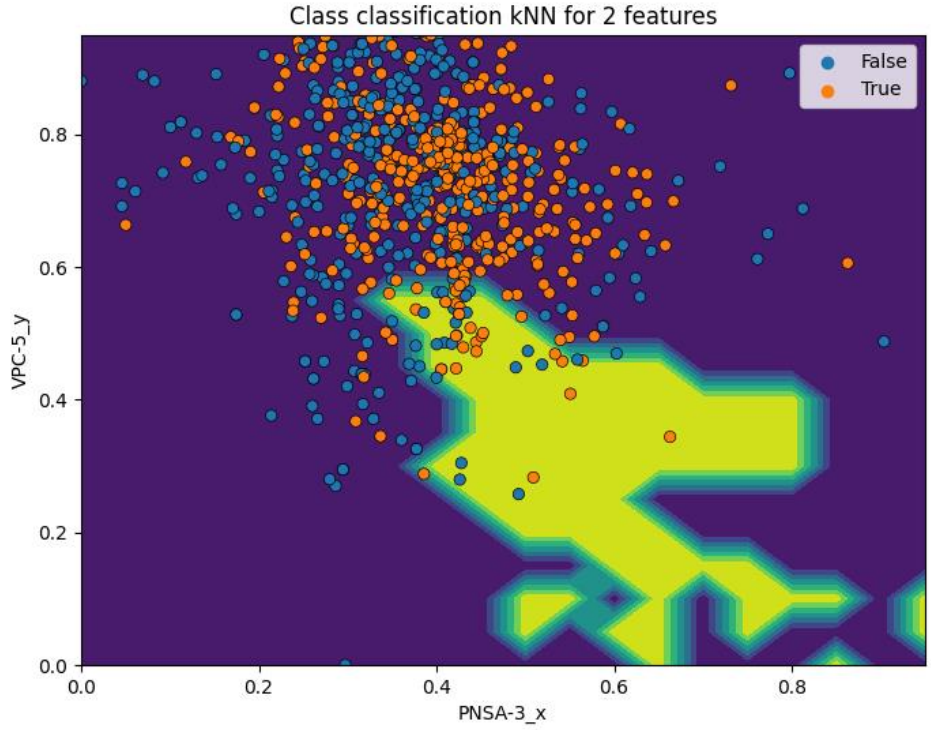


Figure 60: Best Features visualization of TOX2

PAT_S

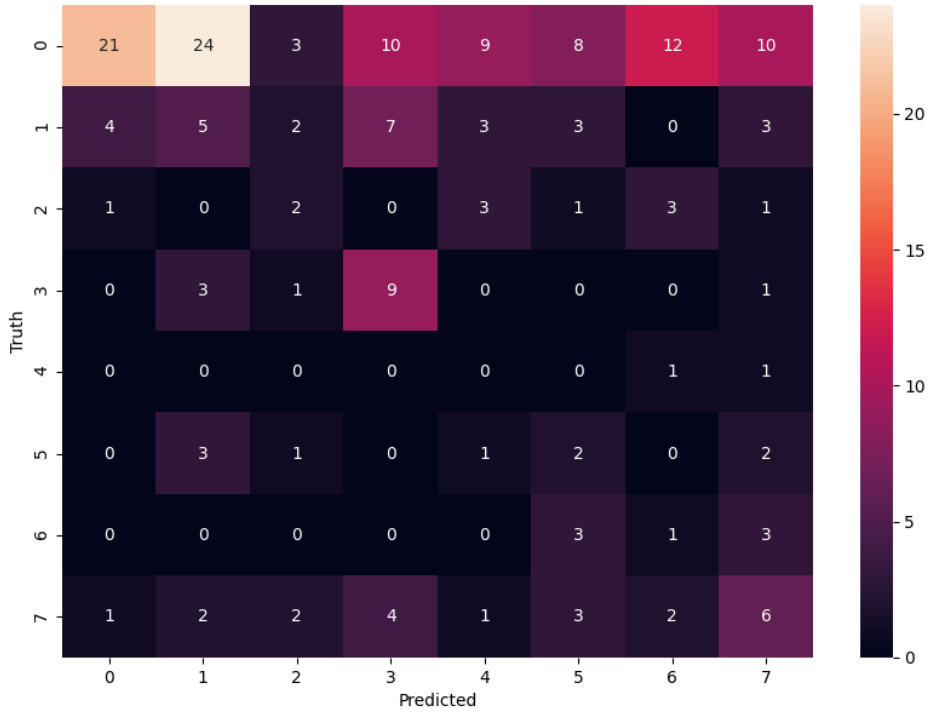


Figure 61: confusion matrix for PAT_S

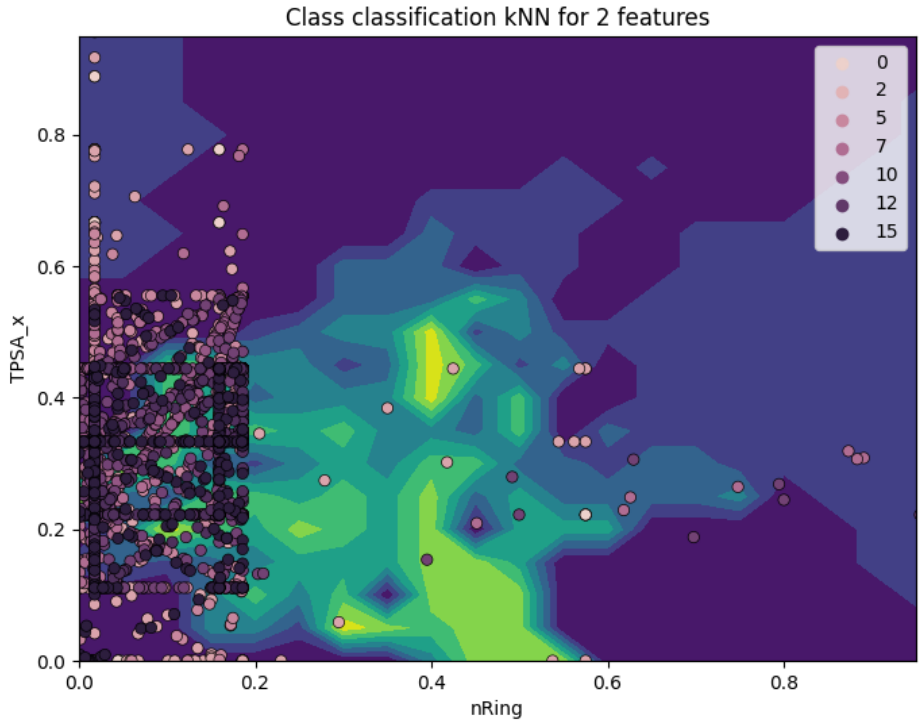


Figure 62: Best Features visualization of PAT_S

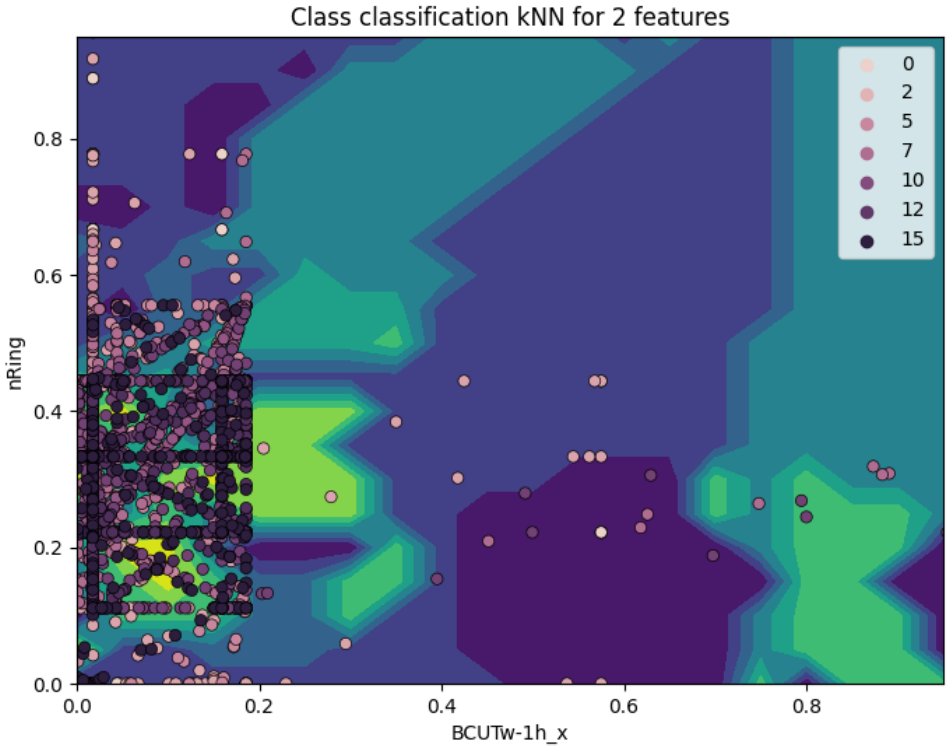


Figure 63: Best Features visualization of PAT_S

PAT_nS

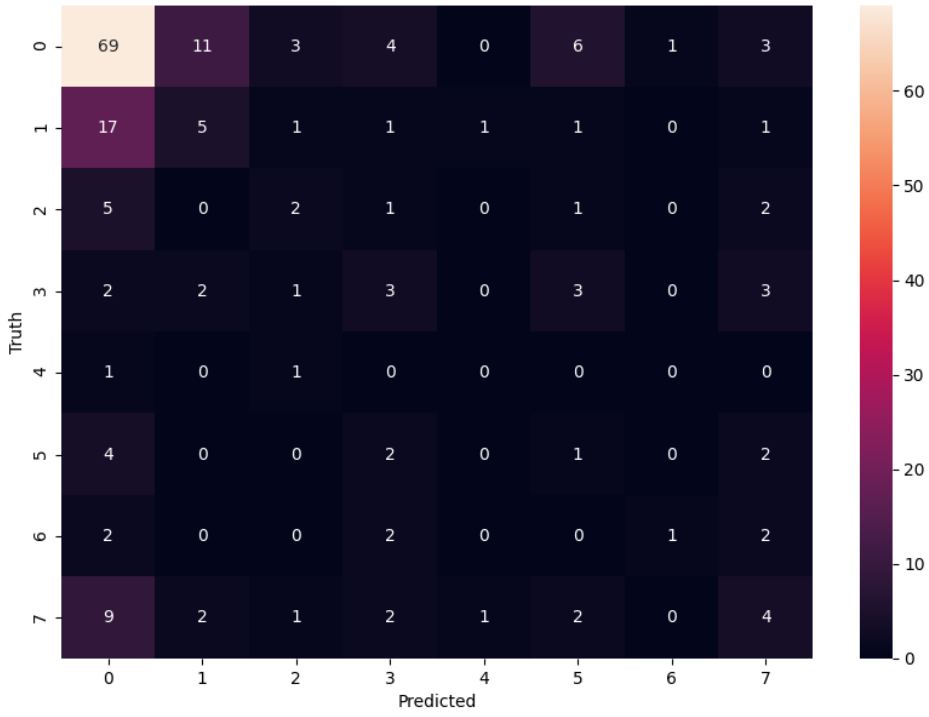


Figure 64: confusion matrix for PAT_nS

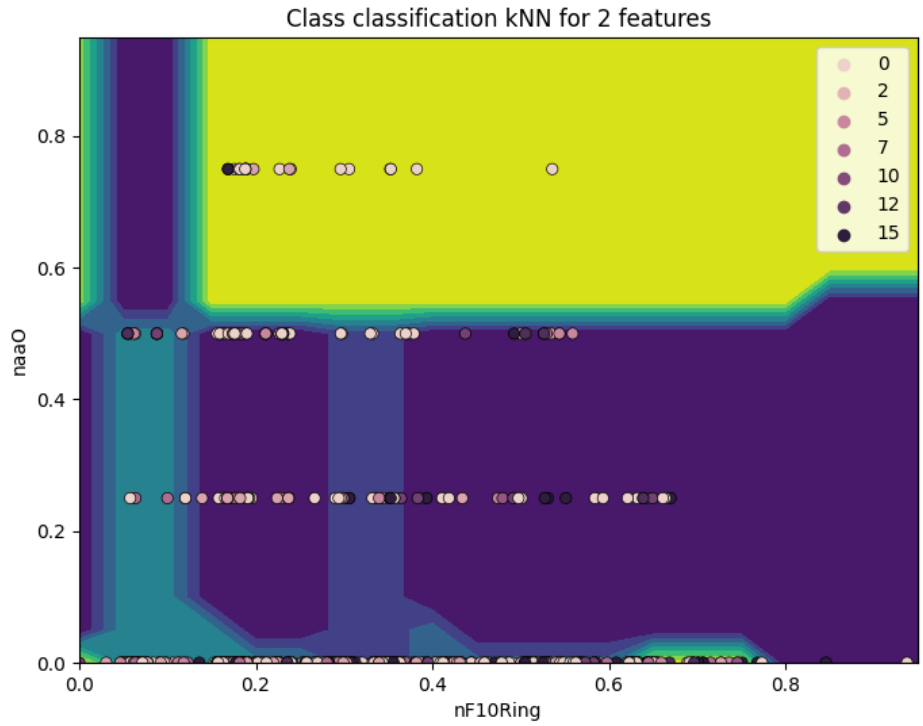


Figure 65: Best Features visualization of PAT_nS

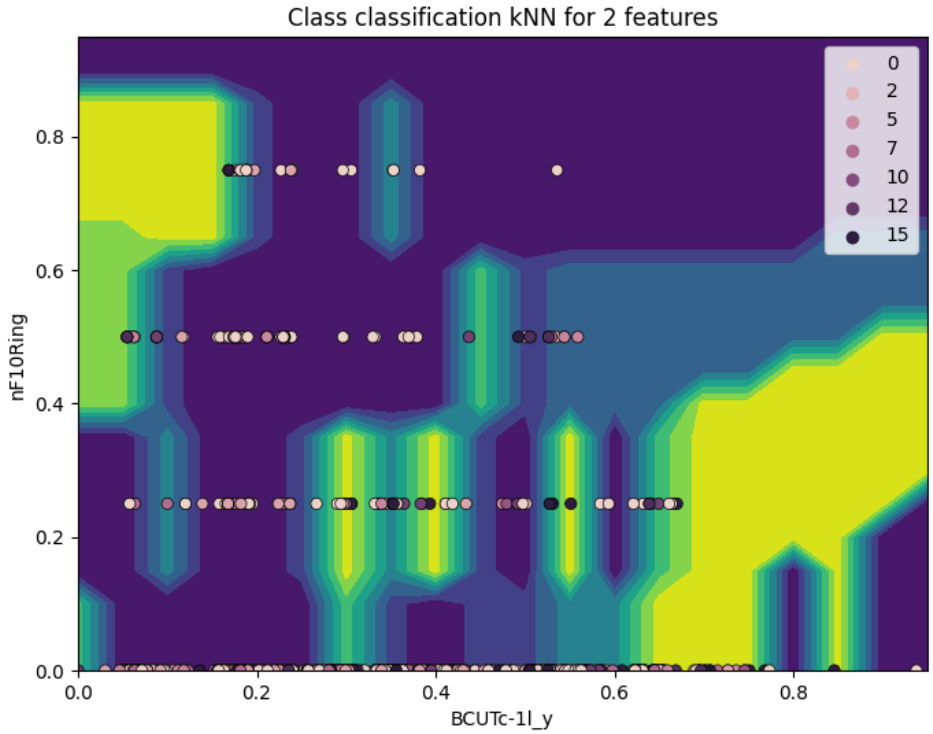


Figure 66: Best Features visualization of PAT_nS

PAT2_S

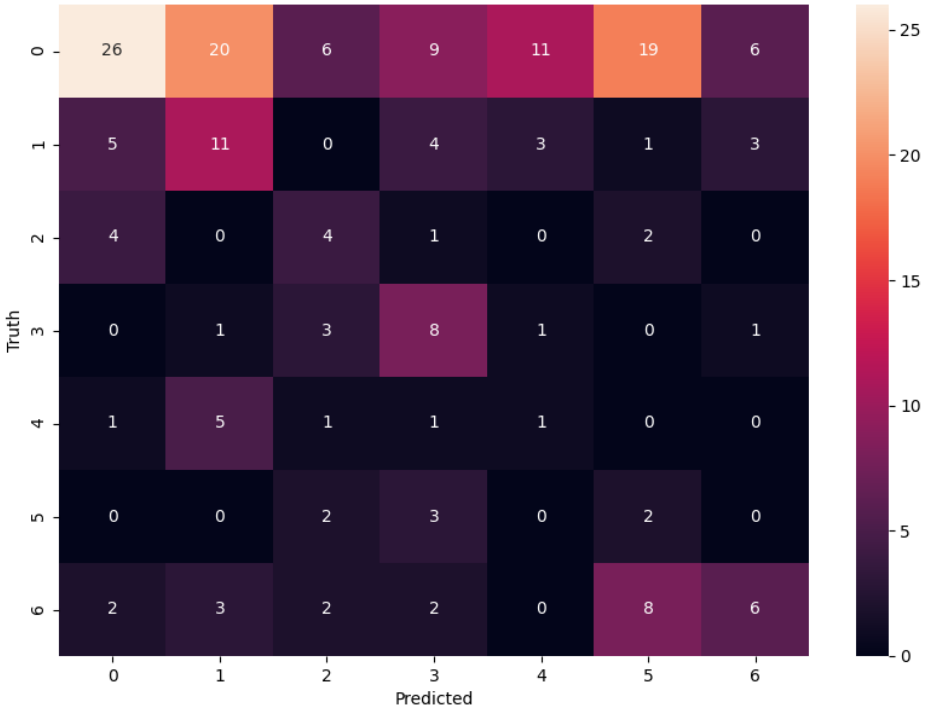


Figure 67: confusion matrix for PAT2_S

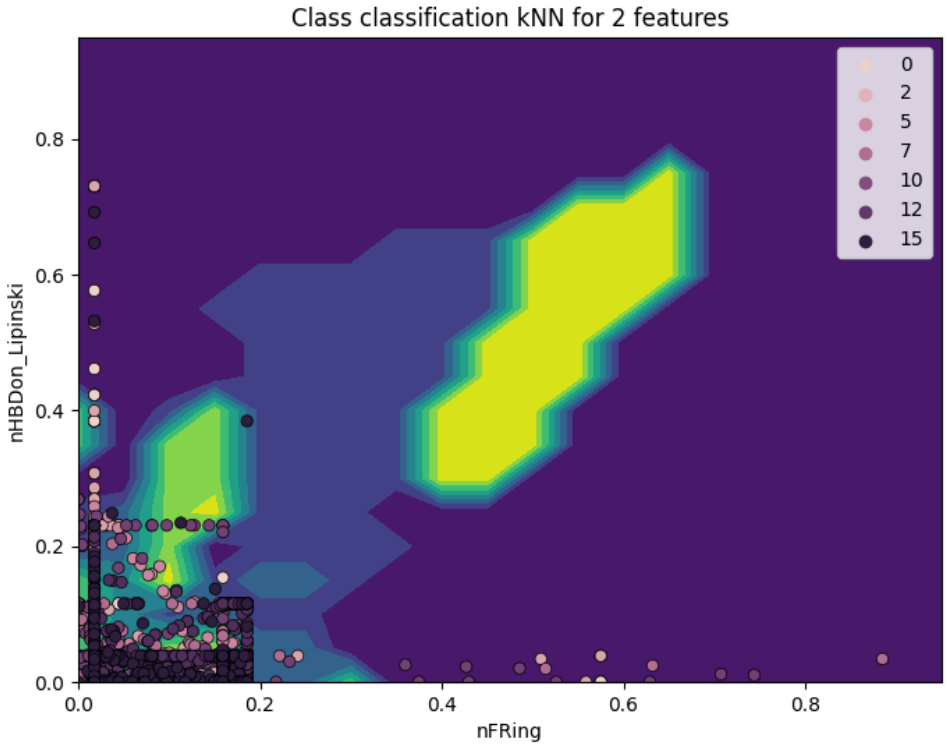


Figure 68: Best Features visualization of PAT2_S

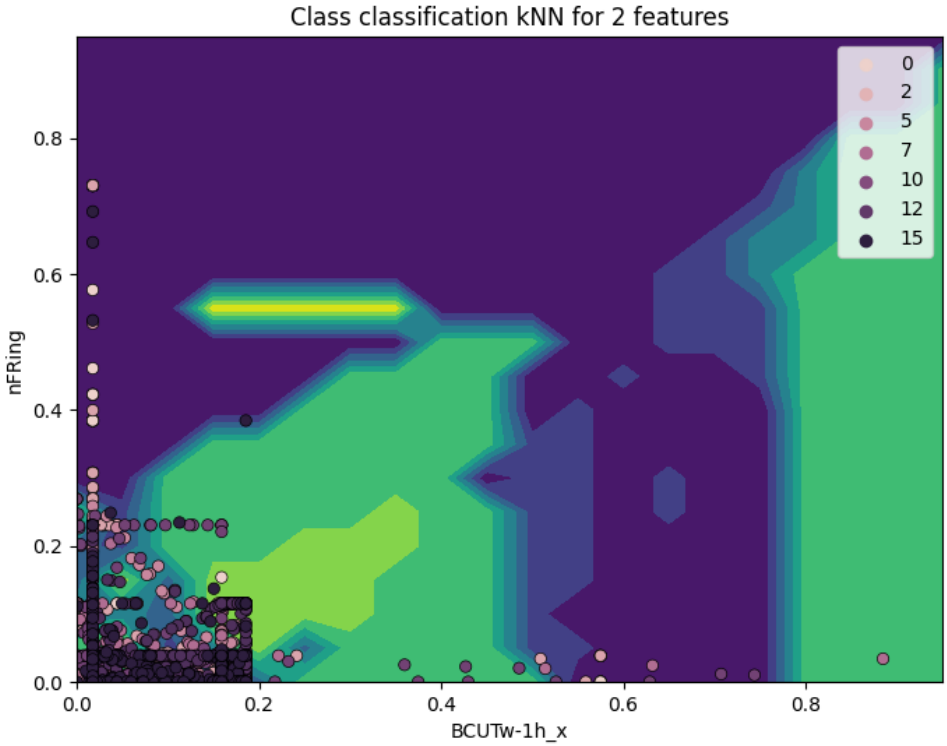


Figure 69: Best Features visualization of PAT2_S

PAT2_nS

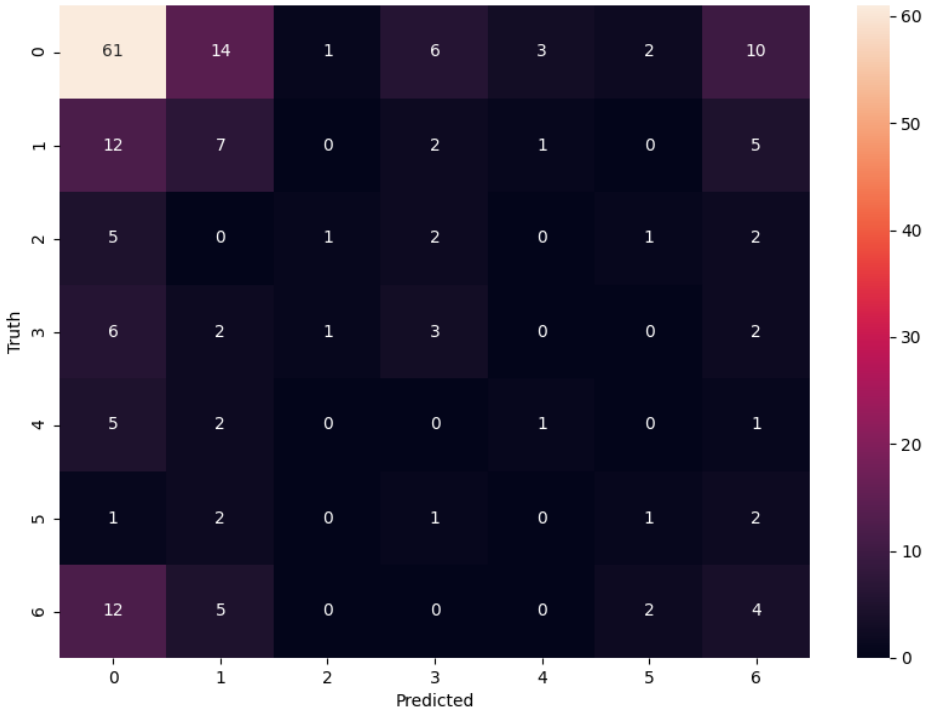


Figure 70: confusion matrix for PAT2_nS

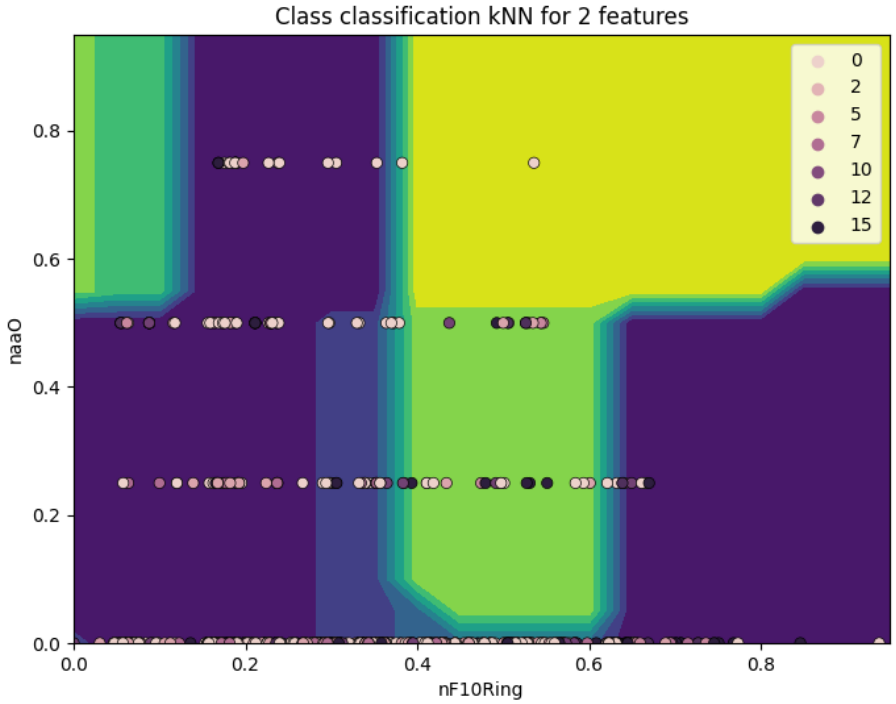


Figure 71: Best Features visualization of PAT2_nS

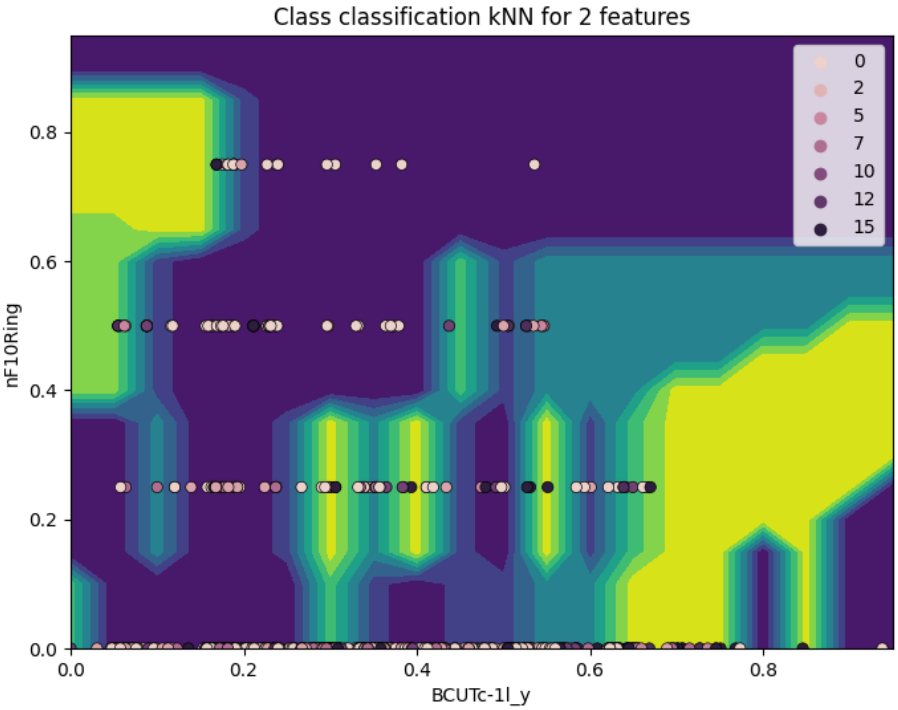


Figure 72: Best Features visualization of PAT2_nS

9.5 Desriptors

Descriptor type	Num-ber	Descriptor	Class
Acidic group count	1	nAcid	2D
ALOGP	3	ALogP, ALogp2, AMR	2D
APol	1	apol	2D
Aromatic atoms count	1	naAromAtom	2D
Aromatic bonds count	1	nAromBond	2D
Atom count	14	nAtom, nHeavyAtom, nH, nB, nC, nN, nO, nS, nP, nF, nCl, nBr, nI, nX	2D
Autocorrelation	346	ATS0m, ATS1m, ATS2m, ATS3m, ATS4m, ATS5m, ATS6m, ATS7m, ATS8m, ATS0v, ATS1v, ATS2v, ATS3v, ATS4v, ATS5v, ATS6v, ATS7v, ATS8v, ATS0e, ATS1e, ATS2e, ATS3e, ATS4e, ATS5e, ATS6e, ATS7e, ATS8e, ATS0p, ATS1p, ATS2p, ATS3p, ATS4p, ATS5p, ATS6p, ATS7p, ATS8p, ATS0i, ATS1i, ATS2i, ATS3i, ATS4i, ATS5i, ATS6i, ATS7i, ATS8i, ATS0s, ATS1s, ATS2s, ATS3s, ATS4s, ATS5s, ATS6s, ATS7s, ATS8s, AATS0m, AATS1m, AATS2m, AATS3m, AATS4m, AATS5m, AATS6m, AATS7m, AATS8m, AATS0v, AATS1v, AATS2v, AATS3v, AATS4v, AATS5v, AATS6v, AATS7v, AATS8v, AATS0e, AATS1e, AATS2e, AATS3e, AATS4e, AATS5e, AATS6e, AATS7e, AATS8e, AATS0p, AATS1p, AATS2p, AATS3p, AATS4p, AATS5p, AATS6p, AATS7p, AATS8p, AATS0i, AATS1i, AATS2i, AATS3i, AATS4i, AATS5i, AATS6i, AATS7i, AATS8i, AATS0s, AATS1s, AATS2s, AATS3s, AATS4s, AATS5s, AATS6s, AATS7s, AATS8s, ATSC0c, ATSC1c, ATSC2c, ATSC3c, ATSC4c, ATSC5c, ATSC6c, ATSC7c, ATSC8c, ATSC0m, ATSC1m, ATSC2m, ATSC3m, ATSC4m, ATSC5m, ATSC6m, ATSC7m, ATSC8m, ATSC0v, ATSC1v, ATSC2v, ATSC3v, ATSC4v, ATSC5v, ATSC6v, ATSC7v, ATSC8v, ATSC0e, ATSC1e, ATSC2e, ATSC3e, ATSC4e, ATSC5e, ATSC6e, ATSC7e, ATSC8e, ATSC0p, ATSC1p, ATSC2p, ATSC3p, ATSC4p, ATSC5p, ATSC6p, ATSC7p, ATSC8p, ATSC0i, ATSC1i, ATSC2i, ATSC3i, ATSC4i, ATSC5i, ATSC6i, ATSC7i, ATSC8i, ATSC0s, ATSC1s, ATSC2s, ATSC3s, ATSC4s, ATSC5s, ATSC6s, ATSC7s, ATSC8s, AATSC0c, AATSC1c, AATSC2c, AATSC3c, AATSC4c, AATSC5c, AATSC6c, AATSC7c, AATSC8c, AATSC0m, AATSC1m, AATSC2m, AATSC3m, AATSC4m, AATSC5m, AATSC6m, AATSC7m, AATSC8m, AATSC0v, AATSC1v, AATSC2v, AATSC3v, AATSC4v, AATSC5v, AATSC6v, AATSC7v, AATSC8v, AATSC0e, AATSC1e, AATSC2e, AATSC3e, AATSC4e, AATSC5e, AATSC6e, AATSC7e, AATSC8e, AATSC0p, AATSC1p, AATSC2p, AATSC3p, AATSC4p, AATSC5p, AATSC6p, AATSC7p, AATSC8p, AATSC0i, AATSC1i, AATSC2i, AATSC3i, AATSC4i, AATSC5i, AATSC6i, AATSC7i, AATSC8i, AATSC0s, AATSC1s, AATSC2s, AATSC3s, AATSC4s, AATSC5s, AATSC6s, AATSC7s, AATSC8s, MATS1c, MATS2c, MATS3c, MATS4c, MATS5c, MATS6c, MATS7c, MATS8c, MATS1m, MATS2m, MATS3m, MATS4m, MATS5m, MATS6m, MATS7m, MATS8m, MATS1v, MATS2v, MATS3v, MATS4v, MATS5v, MATS6v, MATS7v, MATS8v, MATS1e, MATS2e, MATS3e, MATS4e, MATS5e, MATS6e, MATS7e, MATS8e, MATS1p, MATS2p, MATS3p, MATS4p, MATS5p, MATS6p, MATS7p, MATS8p, MATS1i, MATS2i, MATS3i, MATS4i, MATS5i, MATS6i, MATS7i, MATS8i, MATS1s, MATS2s, MATS3s, MATS4s, MATS5s, MATS6s, MATS7s, MATS8s, GATS1c, GATS2c, GATS3c, GATS4c, GATS5c, GATS6c, GATS7c, GATS8c, GATS1m, GATS2m, GATS3m, GATS4m, GATS5m, GATS6m, GATS7m, GATS8m, GATS1v, GATS2v, GATS3v, GATS4v, GATS5v, GATS6v, GATS7v, GATS8v,	2D

		GATS1e, GATS2e, GATS3e, GATS4e, GATS5e, GATS6e, GATS7e, GATS8e, GATS1p, GATS2p, GATS3p, GATS4p, GATS5p, GATS6p, GATS7p, GATS8p, GATS1i, GATS2i, GATS3i, GATS4i, GATS5i, GATS6i, GATS7i, GATS8i, GATS1s, GATS2s, GATS3s, GATS4s, GATS5s, GATS6s, GATS7s, GATS8s	
Barysz matrix	91	SpAbs_DzZ, SpMax_DzZ, SpDiam_DzZ, SpAD_DzZ, SpMAD_DzZ, EE_DzZ, SM1_DzZ, VE1_DzZ, VE2_DzZ, VE3_DzZ, VR1_DzZ, VR2_DzZ, VR3_DzZ, SpAbs_Dzm, SpMax_Dzm, SpDiam_Dzm, SpAD_Dzm, SpMAD_Dzm, EE_Dzm, SM1_Dzm, VE1_Dzm, VE2_Dzm, VE3_Dzm, VR1_Dzm, VR2_Dzm, VR3_Dzm, SpAbs_Dzv, SpMax_Dzv, SpDiam_Dzv, SpAD_Dzv, SpMAD_Dzv, EE_Dzv, SM1_Dzv, VE1_Dzv, VE2_Dzv, VE3_Dzv, VR1_Dzv, VR2_Dzv, VR3_Dzv, SpAbs_Dze, SpMax_Dze, SpDiam_Dze, SpAD_Dze, SpMAD_Dze, EE_Dze, SM1_Dze, VE1_Dze, VE2_Dze, VE3_Dze, VR1_Dze, VR2_Dze, VR3_Dze, SpAbs_Dzp, SpMax_Dzp, SpDiam_Dzp, SpAD_Dzp, SpMAD_Dzp, EE_Dzp, SM1_Dzp, VE1_Dzp, VE2_Dzp, VE3_Dzp, VR1_Dzp, VR2_Dzp, VR3_Dzp, SpAbs_Dzi, SpMax_Dzi, SpDiam_Dzi, SpAD_Dzi, SpMAD_Dzi, EE_Dzi, SM1_Dzi, VE1_Dzi, VE2_Dzi, VE3_Dzi, VR1_Dzi, VR2_Dzi, VR3_Dzi, SpAbs_Dzs, SpMax_Dzs, SpDiam_Dzs, SpAD_Dzs, SpMAD_Dzs, EE_Dzs, SM1_Dzs, VE1_Dzs, VE2_Dzs, VE3_Dzs, VR1_Dzs, VR2_Dzs, VR3_Dzs	2D
Basic group count	1	nBase	2D
BCUT	6	BCUTw-1l, BCUTw-1h, BCUTc-1l, BCUTc-1h, BCUTp-1l, BCUTp-1h	2D
Bond count	10	nBonds, nBonds2, nBondsS, nBondsS2, nBondsS3, nBondsD, nBondsD2, nBondsT, nBondsQ, nBondsM	2D
BPol	1	bpol	2D
Burden modified eigenvalues	96	SpMax1_Bhm, SpMax2_Bhm, SpMax3_Bhm, SpMax4_Bhm, SpMax5_Bhm, SpMax6_Bhm, SpMax7_Bhm, SpMax8_Bhm, SpMin1_Bhm, SpMin2_Bhm, SpMin3_Bhm, SpMin4_Bhm, SpMin5_Bhm, SpMin6_Bhm, SpMin7_Bhm, SpMin8_Bhm, SpMax1_Bhv, SpMax2_Bhv, SpMax3_Bhv, SpMax4_Bhv, SpMax5_Bhv, SpMax6_Bhv, SpMax7_Bhv, SpMax8_Bhv, SpMin1_Bhv, SpMin2_Bhv, SpMin3_Bhv, SpMin4_Bhv, SpMin5_Bhv, SpMin6_Bhv, SpMin7_Bhv, SpMin8_Bhv, SpMax1_Bhe, SpMax2_Bhe, SpMax3_Bhe, SpMax4_Bhe, SpMax5_Bhe, SpMax6_Bhe, SpMax7_Bhe, SpMax8_Bhe, SpMin1_Bhe, SpMin2_Bhe, SpMin3_Bhe, SpMin4_Bhe, SpMin5_Bhe, SpMin6_Bhe, SpMin7_Bhe, SpMin8_Bhe, SpMax1_Bhp, SpMax2_Bhp, SpMax3_Bhp, SpMax4_Bhp, SpMax5_Bhp, SpMax6_Bhp, SpMax7_Bhp, SpMax8_Bhp, SpMin1_Bhp, SpMin2_Bhp, SpMin3_Bhp, SpMin4_Bhp, SpMin5_Bhp, SpMin6_Bhp, SpMin7_Bhp, SpMin8_Bhp, SpMax1_Bhi, SpMax2_Bhi, SpMax3_Bhi, SpMax4_Bhi, SpMax5_Bhi, SpMax6_Bhi, SpMax7_Bhi, SpMax8_Bhi, SpMin1_Bhi, SpMin2_Bhi, SpMin3_Bhi, SpMin4_Bhi, SpMin5_Bhi, SpMin6_Bhi, SpMin7_Bhi, SpMin8_Bhi, SpMax1_Bhs, SpMax2_Bhs, SpMax3_Bhs, SpMax4_Bhs, SpMax5_Bhs, SpMax6_Bhs, SpMax7_Bhs, SpMax8_Bhs, SpMin1_Bhs, SpMin2_Bhs, SpMin3_Bhs, SpMin4_Bhs, SpMin5_Bhs, SpMin6_Bhs, SpMin7_Bhs, SpMin8_Bhs	2D
Carbon types	9	C1SP1, C2SP1, C1SP2, C2SP2, C3SP2, C1SP3, C2SP3, C3SP3, C4SP3	2D
Chi chain	10	SCH-3, SCH-4, SCH-5, SCH-6, SCH-7, VCH-3, VCH-4, VCH-5, VCH-6, VCH-7	2D

Chi cluster	8	SC-3, SC-4, SC-5, SC-6, VC-3, VC-4, VC-5, VC-6	2D
Chi path cluster	6	SPC-4, SPC-5, SPC-6, VPC-4, VPC-5, VPC-6	2D
Chi path	32	SP-0, SP-1, SP-2, SP-3, SP-4, SP-5, SP-6, SP-7, ASP-0, ASP-1, ASP-2, ASP-3, ASP-4, ASP-5, ASP-6, ASP-7, VP-0, VP-1, VP-2, VP-3, VP-4, VP-5, VP-6, VP-7, AVP-0, AVP-1, AVP-2, AVP-3, AVP-4, AVP-5, AVP-6, AVP-7	2D
Constitutional	12	Sv, Sse, Spe, Sare, Sp, Si, Mv, Mse, Mpe, Mare, Mp, Mi	2D
Crippen logP and MR	2	CrippenLogP, CrippenMR	2D
Detour matrix	11	SpMax_Dt, SpDiam_Dt, SpAD_Dt, SpMAD_Dt, EE_Dt, VE1_Dt, VE2_Dt, VE3_Dt, VR1_Dt, VR2_Dt, VR3_Dt	2D
Eccentric connectivity index	1	ECCEN	2D
Atom type electrotopological state	489	nHBd, nwHBd, nHBa, nwHBa, nHBint2, nHBint3, nHBint4, nHBint5, nHBint6, nHBint7, nHBint8, nHBint9, nHBint10, nHsOH, nHdNH, nHsSH, nHsNH2, nHssNH, nHaaNH, nHsNH3p, nHssNH2p, nHsssNHp, nHtCH, nHdCH2, nHdsCH, nHaaCH, nHCHnX, nHCsats, nHCsatu, nHAvin, nHother, nHmisc, nsLi, nssBe, nssssBem, nsBH2, nssBH, nsssB, nssssBm, nsCH3, ndCH2, nssCH2, ntCH, ndsCH, naaCH, nsssCH, nddC, ntsC, ndssC, naasC, naaaC, nssssC, nsNH3p, nsNH2, nssNH2p, ndNH, nssNH, naaNH, ntN, nsssNHp, ndsN, naaN, nsssN, nddsN, naasN, nssssNp, nsOH, ndO, nssO, naaO, naOm, nsOm, nsF, nsSiH3, nssSiH2, nsssSiH, nssssSi, nsPH2, nssPH, nsssP, ndsssP, nddsP, nssssssP, nsSH, ndS, nssS, naaS, ndssS, nddssS, nssssssS, nSm, nsCl, nsGeH3, nssGeH2, nsssGeH, nssssGe, nsAsH2, nssAsH, nsssAs, ndsssAs, nddsAs, nssssssAs, nsSeH, ndSe, nssSe, naaSe, ndssSe, nssssssSe, nddssSe, nsBr, nsSnH3, nssSnH2, nsssSnH, nssssSn, nsI, nsPbH3, nssPbH2, nsssPbH, nssssPb, SHBd, SwHBd, SHBa, SwHBa, SHBint2, SHBint3, SHBint4, SHBint5, SHBint6, SHBint7, SHBint8, SHBint9, SHBint10, SHsOH, SHdNH, SHsSH, SHsNH2, SHssNH, SHaaNH, SHsNH3p, SHssNH2p, SHsssNHp, SHtCH, SHdCH2, SHdsCH, SHaaCH, SHCHnX, SHCsats, SHCsatu, SHAvin, SHother, SHmisc, SsLi, SssBe, SssssBem, SsBH2, SssBH, SsssB, SssssBm, SsCH3, SdCH2, SssCH2, StCH, SdsCH, SaaCH, SsssCH, SddC, StsC, SdssC, SaasC, SaaaC, SssssC, SsNH3p, SsNH2, SssNH2p, SdNH, SssNH, SaaNH, StN, SsssNHp, SdsN, SaaN, SsssN, SddsN, SaasN, SssssNp, SsOH, SdO, SssO, SaaO, SaOm, SsOm, SsF, SsSiH3, SssSiH2, SsssSiH, SssssSi, SsPH2, SssPH, SsssP, SdsssP, SddsP, SssssssP, SsSH, SdS, SssS, SaaS, SdssS, SddssS, SssssssS, SSm, SsCl, SsGeH3, SssGeH2, SsssGeH, SssssGe, SsAsH2, SssAsH, SsssAs, SdsssAs, SddsAs, SssssssAs, SsSeH, SdSe, SssSe, SaaSe, SdssSe, SssssssSe, SddssSe, SsBr, SsSnH3, SssSnH2, SsssSnH, SssssSn, SsI, SsPbH3, SssPbH2, SsssPbH, SssssPb, minHBd, minwHBd, minHBa, minwHBa, minHBint2, minHBint3, minHBint4, minHBint5, minHBint6, minHBint7, minHBint8, minHBint9, minHBint10, minHsOH, minHdNH, minHsSH, minHsNH2, minHssNH, minHaaNH, minHsNH3p, minHssNH2p, minHsssNHp, minHtCH, minHdCH2, minHdsCH, minHaaCH, minHCHnX, minHCsats, minHCsatu, minHAvin, minHother, minHmisc, minsLi, minssBe, minsssBem, minsBH2, minssBH, minsssB, minssssBm, minsCH3, mindCH2, minssCH2, mintCH, mindsCH, minaaCH, minsssCH, minddC, mintsC, mindssC, minaasC, minaaaC, minssssC, minsNH3p, minsNH2, minssNH2p, mindNH, minssNH, minaaNH, mintN, minsssNHp, mindsN, minaaN, minsssN, minddsN, minaasN, minssssNp, minsOH, mindO, minssO, minaaO, minaOm, minsOm, minsF, minsSiH3, minssSiH2, minsssSiH, minssssSi, minsPH2, minssPH,	2D

		minsssP, mindsssP, minddsP, minssssssP, minsSH, mindS, minssS, minaaS, mindssS, minddssS, minsssssssS, minSm, minsCl, minsGeH3, minssGeH2, minsssGeH, minssssGe, minsAsH2, minssAsH, minsssAs, mindsssAs, minddsAs, minssssssAs, minsSeH, mindSe, minssSe, minaaSe, mindssSe, minsssssssSe, minddssSe, minsBr, minsSnH3, minssSnH2, minsssSnH, minssssSn, minsI, minsPbH3, minssPbH2, minsssPbH, minssssPb, maxHBd, maxwHBd, maxHBa, maxwHBa, maxHBint2, maxHBint3, maxHBint4, maxHBint5, maxHBint6, maxHBint7, maxHBint8, maxHBint9, maxHBint10, maxHsOH, maxHdNH, maxHsSH, maxHsNH2, maxHssNH, maxHaaNH, maxHsNH3p, maxHssNH2p, maxHsssNHp, maxHtCH, maxHdCH2, maxHdsCH, maxHaaCH, maxHCHnX, maxHCsats, maxHCsatu, maxHAvin, maxHother, maxHmisc, maxsLi, maxssBe, maxssssBem, maxsBH2, maxssBH, maxsssB, maxssssBm, maxsCH3, maxdCH2, maxssCH2, maxtCH, maxdsCH, maxaaCH, maxsssCH, maxddC, maxtsC, maxdssC, maxaasC, maxaaaC, maxssssC, maxsNH3p, maxsNH2, maxssNH2p, maxdNH, maxssNH, maxaaNH, maxtN, maxsssNHp, maxdsN, maxaaN, maxsssN, maxddsN, maxaasN, maxssssNp, maxsOH, maxdO, maxssO, maxaaO, maxaOm, maxsOm, maxsF, maxsSiH3, maxssSiH2, maxsssSiH, maxssssSi, maxsPH2, maxssPH, maxsssP, maxdsssP, maxddsP, maxssssssP, maxsSH, maxdS, maxssS, maxaaS, maxdssS, maxddssS, maxsssssssS, maxSm, maxsCl, maxsGeH3, maxssGeH2, maxsssGeH, maxssssGe, maxsAsH2, maxssAsH, maxsssAs, maxdsssAs, maxddsAs, maxssssssAs, maxsSeH, maxdSe, maxssSe, maxaaSe, maxdssSe, maxsssssssSe, maxddssSe, maxsBr, maxsSnH3, maxssSnH2, maxsssSnH, maxssssSn, maxsI, maxsPbH3, maxssPbH2, maxsssPbH, maxssssPb, sumI, meanI, hmax, gmax, hmin, gmin, LipoaffinityIndex, MAXDN, MAXDP, DELS, MAXDN2, MAXDP2, DELS2	
Extended topochemical atom	43	ETA_Alpha, ETA_AlphaP, ETA_dAlpha_A, ETA_dAlpha_B, ETA_Epsilon_1, ETA_Epsilon_2, ETA_Epsilon_3, ETA_Epsilon_4, ETA_Epsilon_5, ETA_dEpsilon_A, ETA_dEpsilon_B, ETA_dEpsilon_C, ETA_dEpsilon_D, ETA_Psi_1, ETA_dPsi_A, ETA_dPsi_B, ETA_Shape_P, ETA_Shape_Y, ETA_Shape_X, ETA_Beta, ETA_BetaP, ETA_Beta_s, ETA_BetaP_s, ETA_Beta_ns, ETA_BetaP_ns, ETA_dBeta, ETA_dBetaP, ETA_Beta_ns_d, ETA_BetaP_ns_d, ETA_Eta, ETA_EtaP, ETA_Eta_R, ETA_Eta_F, ETA_EtaP_F, ETA_Eta_L, ETA_EtaP_L, ETA_Eta_R_L, ETA_Eta_F_L, ETA_EtaP_F_L, ETA_Eta_B, ETA_EtaP_B, ETA_Eta_B_RC, ETA_EtaP_B_RC	2D
FMFDescriptor	1	FMF	2D
Fragment complexity	1	fragC	2D
Hbond acceptor count	4	nHBAcc, nHBAcc2, nHBAcc3, nHBAcc_Lipinski	2D
Hbond donor count	2	nHBDon, nHBDon_Lipinski	2D
Hybridization ratio	1	HybRatio	2D
Information content	42	IC0, IC1, IC2, IC3, IC4, IC5, TIC0, TIC1, TIC2, TIC3, TIC4, TIC5, SIC0, SIC1, SIC2, SIC3, SIC4, SIC5, CIC0, CIC1, CIC2, CIC3, CIC4, CIC5, BIC0, BIC1, BIC2, BIC3, BIC4, BIC5, MIC0, MIC1, MIC2, MIC3, MIC4, MIC5, ZMIC0, ZMIC1, ZMIC2, ZMIC3, ZMIC4, ZMIC5	2D
Kappa shape indices	3	Kier1, Kier2, Kier3	2D
Largest chain	1	nAtomLC	2D

Largest Pi system	1	nAtomP	2D
Longest aliphatic chain	1	nAtomLAC	2D
Mannhold LogP	1	MLogP	2D
McGowan volume	1	McGowan_Volume	2D
Molecular distance edge	19	MDEC-11, MDEC-12, MDEC-13, MDEC-14, MDEC-22, MDEC-23, MDEC-24, MDEC-33, MDEC-34, MDEC-44, MDEO-11, MDEO-12, MDEO-22, MDEN-11, MDEN-12, MDEN-13, MDEN-22, MDEN-23, MDEN-33	2D
Molecular linear free energy relation	6	MLFER_A, MLFER_BH, MLFER_BO, MLFER_S, MLFER_E, MLFER_L	2D
Path counts	22	MPC2, MPC3, MPC4, MPC5, MPC6, MPC7, MPC8, MPC9, MPC10, TPC, piPC1, piPC2, piPC3, piPC4, piPC5, piPC6, piPC7, piPC8, piPC9, piPC10, TpiPC, R_TpiPCTPC	2D
Petitjean number	1	PetitjeanNumber	2D
Ring count	68	nRing, n3Ring, n4Ring, n5Ring, n6Ring, n7Ring, n8Ring, n9Ring, n10Ring, n11Ring, n12Ring, nG12Ring, nFRing, nF4Ring, nF5Ring, nF6Ring, nF7Ring, nF8Ring, nF9Ring, nF10Ring, nF11Ring, nF12Ring, nFG12Ring, nHeteroRing, n3HeteroRing, n4HeteroRing, n5HeteroRing, n6HeteroRing, n7HeteroRing, n8HeteroRing, n9HeteroRing, n10HeteroRing, n11HeteroRing, n12HeteroRing, nG12HeteroRing, nFHeteroRing, nF4HeteroRing, nF5HeteroRing, nF6HeteroRing, nF7HeteroRing, nF8HeteroRing, nF9HeteroRing, nF10HeteroRing, nF11HeteroRing, nF12HeteroRing, nFG12HeteroRing, nTHeteroRing, nT4HeteroRing, nT5HeteroRing, nT6HeteroRing, nT7HeteroRing, nT8HeteroRing, nT9HeteroRing, nT10HeteroRing, nT11HeteroRing, nT12HeteroRing, nTG12HeteroRing	2D
Rotatable bonds count	4	nRotB, RotBFrac, nRotBt, RotBtFrac	2D
Rule of five	1	LipinskiFailures	2D
Topological	3	topoRadius, topoDiameter, topoShape	2D
Topological charge	21	GGI1, GGI2, GGI3, GGI4, GGI5, GGI6, GGI7, GGI8, GGI9, GGI10, JGI1, JGI2, JGI3, JGI4, JGI5, JGI6, JGI7, JGI8, JGI9, JGI10, JGT	2D
Topological distance matrix	11	SpMax_D, SpDiam_D, SpAD_D, SpMAD_D, EE_D, VE1_D, VE2_D, VE3_D, VR1_D, VR2_D, VR3_D	2D
Topological polar surface area	1	TopoPSA	2D
Van der Waals volume	1	VABC	2D
Vertex adjacency information (magnitude)	1	vAdjMat	2D

Walk counts	20	MWC2, MWC3, MWC4, MWC5, MWC6, MWC7, MWC8, MWC9, MWC10, TWC, SRW2, SRW3, SRW4, SRW5, SRW6, SRW7, SRW8, SRW9, SRW10, TSRW	2D
Weight	2	MW, AMW	2D
Weighted path	5	WTPT-1, WTPT-2, WTPT-3, WTPT-4, WTPT-5	2D
Wiener numbers	2	WPATH, WPOL	2D
XLogP	1	XLogP	2D
Zagreb index	1	Zagreb	2D
3D autocorrelation	80	TDB1u, TDB2u, TDB3u, TDB4u, TDB5u, TDB6u, TDB7u, TDB8u, TDB9u, TDB10u, TDB1m, TDB2m, TDB3m, TDB4m, TDB5m, TDB6m, TDB7m, TDB8m, TDB9m, TDB10m, TDB1v, TDB2v, TDB3v, TDB4v, TDB5v, TDB6v, TDB7v, TDB8v, TDB9v, TDB10v, TDB1e, TDB2e, TDB3e, TDB4e, TDB5e, TDB6e, TDB7e, TDB8e, TDB9e, TDB10e, TDB1p, TDB2p, TDB3p, TDB4p, TDB5p, TDB6p, TDB7p, TDB8p, TDB9p, TDB10p, TDB1i, TDB2i, TDB3i, TDB4i, TDB5i, TDB6i, TDB7i, TDB8i, TDB9i, TDB10i, TDB1s, TDB2s, TDB3s, TDB4s, TDB5s, TDB6s, TDB7s, TDB8s, TDB9s, TDB10s, TDB1r, TDB2r, TDB3r, TDB4r, TDB5r, TDB6r, TDB7r, TDB8r, TDB9r, TDB10r	3D
Charged partial surface area	29	PPSA-1, PPSA-2, PPSA-3, PNSA-1, PNSA-2, PNSA-3, DPSA-1, DPSA-2, DPSA-3, FPSA-1, FPSA-2, FPSA-3, FNSA-1, FNSA-2, FNSA-3, WPSA-1, WPSA-2, WPSA-3, WNSA-1, WNSA-2, WNSA-3, RPCG, RNCG, RPCS, RNCS, THSA, TPSA, RHSA, RPSA	3D
Gravitational index	9	GRAV-1, GRAV-2, GRAV-3, GRAVH-1, GRAVH-2, GRAVH-3, GRAV-4, GRAV-5, GRAV-6	3D
Length over breadth	2	LOBMAX, LOBMIN	3D
Moment of inertia	7	MOMI-X, MOMI-Y, MOMI-Z, MOMI-XY, MOMI-XZ, MOMI-YZ, MOMI-R	3D
Petitjean shape index	3	geomRadius, geomDiameter, geomShape	3D
RDF	210	RDF10u, RDF15u, RDF20u, RDF25u, RDF30u, RDF35u, RDF40u, RDF45u, RDF50u, RDF55u, RDF60u, RDF65u, RDF70u, RDF75u, RDF80u, RDF85u, RDF90u, RDF95u, RDF100u, RDF105u, RDF110u, RDF115u, RDF120u, RDF125u, RDF130u, RDF135u, RDF140u, RDF145u, RDF150u, RDF155u, RDF10m, RDF15m, RDF20m, RDF25m, RDF30m, RDF35m, RDF40m, RDF45m, RDF50m, RDF55m, RDF60m, RDF65m, RDF70m, RDF75m, RDF80m, RDF85m, RDF90m, RDF95m, RDF100m, RDF105m, RDF110m, RDF115m, RDF120m, RDF125m, RDF130m, RDF135m, RDF140m, RDF145m, RDF150m, RDF155m, RDF10v, RDF15v, RDF20v, RDF25v, RDF30v, RDF35v, RDF40v, RDF45v, RDF50v, RDF55v, RDF60v, RDF65v, RDF70v, RDF75v, RDF80v, RDF85v, RDF90v, RDF95v, RDF100v, RDF105v, RDF110v, RDF115v, RDF120v, RDF125v, RDF130v, RDF135v, RDF140v, RDF145v, RDF150v, RDF155v, RDF10e, RDF15e, RDF20e, RDF25e, RDF30e, RDF35e, RDF40e, RDF45e, RDF50e, RDF55e, RDF60e, RDF65e, RDF70e, RDF75e, RDF80e, RDF85e, RDF90e, RDF95e, RDF100e, RDF105e, RDF110e, RDF115e, RDF120e, RDF125e, RDF130e, RDF135e, RDF140e, RDF145e, RDF150e, RDF155e, RDF10p, RDF15p, RDF20p, RDF25p, RDF30p, RDF35p, RDF40p, RDF45p,	3D

		RDF50p, RDF55p, RDF60p, RDF65p, RDF70p, RDF75p, RDF80p, RDF85p, RDF90p, RDF95p, RDF100p, RDF105p, RDF110p, RDF115p, RDF120p, RDF125p, RDF130p, RDF135p, RDF140p, RDF145p, RDF150p, RDF155p, RDF10i, RDF15i, RDF20i, RDF25i, RDF30i, RDF35i, RDF40i, RDF45i, RDF50i, RDF55i, RDF60i, RDF65i, RDF70i, RDF75i, RDF80i, RDF85i, RDF90i, RDF95i, RDF100i, RDF105i, RDF110i, RDF115i, RDF120i, RDF125i, RDF130i, RDF135i, RDF140i, RDF145i, RDF150i, RDF155i, RDF10s, RDF15s, RDF20s, RDF25s, RDF30s, RDF35s, RDF40s, RDF45s, RDF50s, RDF55s, RDF60s, RDF65s, RDF70s, RDF75s, RDF80s, RDF85s, RDF90s, RDF95s, RDF100s, RDF105s, RDF110s, RDF115s, RDF120s, RDF125s, RDF130s, RDF135s, RDF140s, RDF145s, RDF150s, RDF155s	
WHIM	91	L1u, L2u, L3u, P1u, P2u, E1u, E2u, E3u, Tu, Au, Vu, Ku, Du, L1m, L2m, L3m, P1m, P2m, E1m, E2m, E3m, Tm, Am, Vm, Km, Dm, L1v, L2v, L3v, P1v, P2v, E1v, E2v, E3v, Tv, Av, Vv, Kv, Dv, L1e, L2e, L3e, P1e, P2e, E1e, E2e, E3e, Te, Ae, Ve, Ke, De, L1p, L2p, L3p, P1p, P2p, E1p, E2p, E3p, Tp, Ap, Vp, Kp, Dp, L1i, L2i, L3i, P1i, P2i, E1i, E2i, E3i, Ti, Ai, Vi, Ki, Di, L1s, L2s, L3s, P1s, P2s, E1s, E2s, E3s, Ts, As, Vs, Ks, Ds	3D

Erklärung

«Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche kenntlich gemacht. Mir ist bekannt, dass andernfalls der Senat gemäss dem Gesetz über die Universität Bern zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

Für die Zwecke der Begutachtung und der Überprüfung der Einhaltung der Selbständigkeitserklärung bzw. der Reglemente betreffend Plagiate erteile ich der Universität Bern das Recht, die dazu erforderlichen Personendaten zu bearbeiten und Nutzungshandlungen vorzunehmen, insbesondere die schriftliche Arbeit zu vervielfältigen und dauerhaft in einer Datenbank zu speichern sowie diese zur Überprüfung von Arbeiten Dritter zu verwenden oder hierzu zur Verfügung zu stellen.»

Datum und Unterschrift der Studierenden:

10.7.2021

Arlene Günter

Arlene Günter