ID2214 - Programming for Data Science

Project Report
Group 5

Charlotte Jacquet
Lukas Olenborg
Isabella Rositi

15.12.2022

Contents

1 Introduction

In this report, we implement and review a Machine Learning (ML) model for predicting biological activity in chemical compounds. We are given a large training dataset with 156 258 chemical compounds as SMILES strings. SMILES is a string representation of a chemical molecule or compound, and it even accurately describes the structure of complex two or three-dimensional models of compounds, as explained by EPA. Additionally, this training dataset has class labels for activity (1) or no biological activity (0).

The first task is to formulate a suitable representation of this problem for a Machine Learning model, as using SMILES directly to predict biological activity is infeasible. From SMILES representations, we can extract properties of chemical compounds using the RDKit library. These properties can consist of counts of different structures or atoms e.g., the Lipinski parameters. Additionally, molecular fingerprints can be extracted from SMILES, representing the compound as a binary array, e.g., the Morgan fingerprints.

We approach feature selection with the black box method, meaning that we do not handpick suitable features based on knowledge in chemistry but instead find them empirically. We use the wrapper method, i.e., finding the best feature subset from a large feature pool and verifying the results with cross-validation. We compare these subsets by running a base model and calculating the AUC. Once the optimal features are selected, we conduct hyperparameter analysis to finetune the models. Finally, we choose the best combination of representation and model to maximize the AUC. We use this model to make predictions on a test dataset with 52 086 chemical compounds with unknown class labels.

We begin by describing the methodology implemented in this project in Section 2, then we continue discussing data preparation in Section 3. Then, Section 4 explores the methods used for feature selection, and in Section 5 we go over the methods used in model selection. Finally, we conclude the report with a discussion on the optimal representation and model in Section 6.

2 Methodology

In order to perform the task of this project, we will implement the methodology as follows.

We are given a training set with class labels and a test set without class labels. Our goal is, thus, to predict a set of labels for the test set and provide an *estimated AUC* for these. In order to calculate this AUC, we first need to train and validate a number of models on the training set, and later choose the optimal model to use on the test set.

Therefore, we start by further dividing the given training set into a *train* and *validation set*. Moreover, in order to perform feature selection and tune the hyperparameters of the models, we implemented a k-fold cross-validation. We took care of implementing a stratified cross-validation since the data presents a highly imbalanced subdivision of class labels (99% inactives to 1% actives).

First, we apply cross-validation on the obtained *train set* to choose the optimal subset of features to include in the models. The set of features is selected based on their AUC achieved by implementing the ML algorithms with their default parameters, as explained in more detail in Section 3. Once we choose the optimal set of features, we employ on the *train set* a grid search combined with a stratified k-fold cross validation to tune the hyperparameters of the chosen models in order to test whether we can improve the performances obtained with the default parameters of the models.

We later validate the models with the best combination of hyperparameters on the *validation set* in order to provide an *estimated* AUC that can be as unbiased as possible. For the final step, we use the better performing model on the *test set* in order to compute the predictions.

3 Data Preparation

We handpicked 18 atomic features from the RDKit library without any prior knowledge of the topic. Among the selected features thare are the weight of the molecules, the count of heavy atoms, and most of them are counts of specific atom structures in the molecules. Thus, these features are all numerical data.

In addition, we also took into consideration Morgan Fingerprints, which are arrays of 124 bits, each encoding a specific fragment of the molecule: 0 if the fragment is absent on the molecule and 1 if it is present, as reported by Landrum [2012]. Thus, Morgan Fingerprints can be seen as binary data.

First, we computed some statistical descriptive analysis. Out of the 156 258 instances in the training set, 99% have 'ACTIVE' class equal to 0, and only 1% have 'ACTIVE' class equal to 1. This class imbalance will be taken into consideration as we proceed with the analysis since a poor management of this issue can contribute to predictions that do not take into account the smaller class.

Secondly, we made sure that there were no missing values in the data set. There was indeed no need to apply imputation on our training set.

Lastly, we decided not to apply any type of normalization to our data for a variety of reasons. First of all, normalization is specifically needed when certain ML algorithms are implemented. These include PCA, KNN with Euclidean distance, SVM or Ridge, and Lasso Regression. However, we are not implementing any model that requires normalized data as input. Indeed for most models, normalization makes no difference in their predictive power. We performed all our analysis on non-normalized data and to empirically prove that there wasn't any significant difference with analysis conducted using normalized data, we also ran the models on data that were processed using Min-Max normalization. We applied this normalization only on the numerical features. Indeed, as shown in Table 1, the difference is irrelevant.

	Logistic Regression	Neural Network	Random Forest
Non-normalized	0.7629	0.6815	0.7125
Normalized	0.7622	0.6909	0.7093

Table 1: Difference in AUC in models ran with non-normalized data and normalized data

4 Feature Selection

We approached feature selection with the black box method. As we are not experts in chemistry, this approach allows us to consider a large number of features and find the optimal covariates. Using feature selection methods and algorithms presents advantages, as it can reduce model complexity and increase learning efficiency.

In each representation discussed below, we started by running a feature selection algorithm based on a filter method. We chose correlation-based feature selection (CFS), an algorithm that evaluates feature subsets solely based on the correlation in the data. We aim to find a subset with a low correlation between the different selected features. The chosen algorithm estimates the merit of a subset s

$$Merit_s = \frac{k\bar{r}_{cf}}{\sqrt{k + (k-1)\bar{r}_{ff}}} \tag{1}$$

with k number of features subset s. Here, $\bar{r}_{cf}$ is the average feature-to-class correlation, and $\bar{r}_{ff}$ is the average feature-to-feature correlation. Therefore, a subset with low correlation between the selected features has high merit. To find the best subset, the algorithm uses the merit heuristic approach developed by Hall [2000]. Unfortunately, this method did not result in promising AUCs. We then proceeded to use the straightforward wrapper method instead, coupled with cross-validation.

In the first representation, we use various atomic properties from the RDKit library to find an optimal subset of features. In the second representation, we instead attempt to use Morgan's fingerprints as features, and in the final representation, we combine these feature sets.

4.1 First Representation: Atomic Features

This representation is given by atomic features, mainly Lipinski properties from RDKit. In the wrapper method, we run our models (introduced later in Section 5) several times with different feature subsets to find the best-performing features in terms of AUC. We initially tried three different feature subset sizes, namely 5, 8, and 12 and we run each ML algorithm 50 times, each time selecting the features randomly from the existing pool of 18 features. We then repeat another 50 runs where we have increased the number of folds, i.e. the number of training-validation splits from 5 to 10. Each time, we collect the optimal set of features in terms of the AUC. The results for the Logistic Regression can be seen in Table 2 below.

Number of Features	Number of Folds	AUC
5	5	0.7094
5	10	0.7094
8	5	0.7145
8	10	0.7154
12	5	0.7145
12	10	0.7160
18	**5**	**0.7629**
18	10	0.7629

Table 2: Optimal feature subsets in terms of AUC with logistic regression.

Based on these results, it seems that having more features is beneficial. Therefore, we decide to implement for our further analysis the representation including all **18 atomic features**.

An interesting finding is that the optimal features chosen in the subsets varied depending on the number of folds. As an example, the optimal features found with logistic regression with a subset of size 8 and both 5 and 10 folds are shown in table 3 and 4 below. One thing to note, though we are not chemists, these features sound quite similar and the issue of potential correlation between features should be taken into consideration.

Description	Type
Number of saturated rings	*int*
Number of rings	*int*
Number of aromatic heterocycles	*int*
Number of heavy atoms	*int*
Number of Nitrogens and Oxygens	*int*
Number of Heteroatoms	*int*
Number of Hydrogen bond donors	*int*
Number of saturated heterocycles	*int*

Table 3: Optimal features for logistic regression and a subset of size 8 features, with 5 folds.

Description	Type
Number of saturated rings	*int*
Number of rings	*int*
Number of aromatic heterocycles	*int*
Number of heavy atoms	*int*
Number of Nitrogens and Oxygens	*int*
Number of Heteroatoms	*int*
Number of aliphatic carbocycles	*int*
Number of aromatic carbocycles	*int*

Table 4: Optimal features for logistic regression and a subset of size 8 features, with 10 folds.

4.2 Second Representation: Morgan Fingerprints

In this representation, we initially use 124 Morgan fingerprints as features. As discussed at the beginning of this section (4), we first attempt to use the feature subset recommended by the filter method CFS. In this case, the filter method returns a set of 29 fingerprints.

We then test feature subset sizes of 50 and 100 with the wrapper method, as in Section 4.1. Although the AUC was higher in comparison to using the 29 features mentioned above, we found the best results for every ML algorithm when using all 124 fingerprints as features. Naturally, this number could be further increased, but due to computational challenges, we decided to implement only 124 bits. The results for the Logistic Regression are shown in table 6 below.

Number of Features	Number of Folds	AUC
29	10	0.6381
50	10	0.6391
100	10	0.6606
124	**10**	**0.7209**

Table 5: Optimal feature subsets from Morgan fingerprints in terms of AUC with logistic regression.

Hence, we decide to implement for our further analysis the representation including all **124 Morgan fingerprints**.

4.3 Third Representation: Combination of Features

In this representation, we combine the Morgan fingerprints and the atomic features (Lipinski properties). Based on the findings in the previous section 4.2, we use all 124 Morgan fingerprints as features and attempt to add different subsets of the basic features on top of that.

Number of Features	Number of Folds	AUC
124 + 5	5	0.7278
124 + 8	5	0.7360
124 + 12	5	0.7611
124 + 18	**5**	**0.8073**

Table 6: Optimal feature subsets from a combination of Morgan fingerprints and subsets of atomic features in terms of AUC with logistic regression.

As can be observed by Table 6 specifically for Logistic Regression, once again the optimal model performance in terms of AUC is provided by the complete model: **124 morgan fingerprints combined with 18 atomic features**. This is true for every ML algorithm we implement.

5 Model Selection

The first model we take into consideration is **Logistic Regression** since it is a relatively simple generalized linear model suited for binary classification. It is also much faster computationally than many other models, which is essential, as discussed in Section 4. Due to the class imbalance od the data, in the implementation we use the parameter *class_weight = 'balanced'* that takes care of the issue and provides more accurate results.

The second model we consider is **Random Forest**. This model can handle large datasets easily and it combines two techniques: bagging, consisting of selecting a random sample of data in a training set with replacement, and the random subspace method, consisting of picking random samples of features. These techniques significantly decrease the model's variance, making Random Forest a valuable model to implement, especially when handling numerous covariates. Like for the Logistic regression, also the Random Forest Classifier in sklearn offers the implementation of the parameter *class_weight = 'balanced'* to take care of the class imbalance and achieve more accurate results.

The third model we consider is **MultiLayerPerceptron**, a Neural Network. This model is well known for its predictive power, especially when carefully fine-tuned. Nowadays, it is the go-to ML algorithm when dealing with predictions, so we decided to test it ourselves.

We implement these models using the scikit-learn library [2011].

Representation	Logistic Regression	Random Forests	Neural Network
Atomic	0.7623	0.7042	0.6815
MF	0.7309	0.7308	0.7344
Atomic + MF	**0.8071**	**0.7881**	**0.800**

Table 7: AUCs of baseline models ran with default parameters with the three different representations.

In table 7, we show the AUCs of the baseline models run with default parameters for each of the three representations. It is noticeable that for each model, the best representation is the combined one, which includes both the atomic features and the fingerprints. The largest improvement is seen in the Neural Network model. Hence, we will tune the hyperparameters of the models using this third representation.

5.1 Logistic Regression

In order to improve the performance of Logistic Regression, we implement a grid search including the following hyperparameters (where the ones in *italics* are the default parameters):

$$\{\text{penalty} : [\text{'L1'}, \textit{'L2'}, \text{'none'}],$$
$$\text{fit\_intercept} : [\textit{True}, \text{False}]\}$$

It turned out that the best model is the baseline model, providing an AUC of **0.8071** on the *validation set*:

$$\{\text{penalty} : \text{'L2'}, \text{fit\_intercept} : \text{True}\}$$

Indeed, there is little to no difference between the different 'penalty' parameters, however, fitting the intercept provides significantly higher performance than setting 'fit_intercept' = False.

5.2 Random Forest

In order to improve the performance of Random Forest, we implement a grid search including the following hyperparameters (where the ones in *italics* are the default parameters):

{n_estimators: [*100*, 200, 300, 400],
max_depth: [*'none'*, 30, 50] }

The best model provides an AUC of **0.8543** on the *validation set* with parameters:

{n_estimators: 400, max_depth: 30}

Indeed, we noticed that the performance improves as the number of trees and depth increase.

5.3 Neural Network

In order to improve the performance of the Neural Network, we implement a grid search including the following hyperparameters (where the ones in *italics* are the default parameters):

{hidden_layer_sizes: [*(100,)*, (150,)],
solver: [*'adam'*, 'sgd'],
alpha: [*0.0001*, 0.005] }

The best model provides an AUC of **0.8054** on the *validation set*, with parameters:

{hidden_layer_sizes: (150,), solver: 'adam', alpha: 0.0001}

This model performs better than the MLP model with the default parameters. Indeed, we noticed that as the number of nodes in the hidden layer increased, so did the performance, and that the stochastic gradient ('sgd') definitely works worse than the solver 'adam', which is a sgd variant that updates the learning rates individually for different parameters.

6 Final Model

In Table 8, we gather the performance of the best models for each ML algorithm, which are the result of a process of hyperparameter tuning on the *train set.* The AUCs provided in the table are computed on the *validation set.*

Logistic Regression	Random Forests	Neural Network
0.8071	**0.8543**	0.8054

Table 8: AUCs calculated on the combined *validation set* using the best combination of hyperparameters for each model.

From the table, we can conclude that the best performing model is Random Forest, with an *estimated* AUC of **0.8543**. This means that we use this Random Forest model with the best parameters to predict the test set's labels.

We can observe that the performance of the Neural Network model is not better than the Logistic Regression in this setting. This could be due to only implementing MLP with one hidden layer because of the high computational cost of adding multiple hidden layers. A further implementation of this work could include venturing into the Deep Learning realm and testing MLP with multiple hidden layers.

Moreover, future implementations of this project could also see the participation of a chemical specialist that can help handpick specific features relevant to the target variable. Further, we did not explore ML algorithms and methodologies explicitly dedicated to the chemical field, which is not our area of expertise, but that could help reach better results since being targeted towards this type of implementation.

7 Conclusion

In this report, we implemented a Machine Learning model to predict biological activity in chemical elements. We first extracted the properties from chemical compounds. Next, we conducted feature selection using the wrapper method to find the best feature subsets, verified with the cross-validation technique. Then, we performed a grid search to conduct a hyperparameter analysis for each model and chose the best combination of hyperparameters, i.e. one that maximizes the AUC. For this problem, the best model based on the AUC metric was Random Forest. Therefore, we used the Random Forest with the best parameters found, to predict the labels of the test set.

References

EPA. SMILES tutorial. URL https://archive.epa.gov/med/med_archive_03/web/html/smiles.html.

Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 359–366, 2000.

Gregory Landrum. Fingerprints in the RDKit, 2012. URL https://www.rdkit.org/UGM/2012/Landrum_RDKit_UGM.Fingerprints.Final.pptx.pdf.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

RDKit. Rdkit: Open-source cheminformatics. URL http://www.rdkit.org.