CSCI – 555 Data Mining & Machine Learning

Project Report

The Toxicity Prediction Challenge II

ST. FRANCIS XAVIER UNIVERSITY

Sheel Bhardwaj

202205090

1. INTRODUCTION

To guarantee that freshly synthesized compounds are used safely in diverse sectors, it is essential to forecast their toxicity. Traditional animal testing in laboratories is not only costly and time-consuming, but it also presents ethical questions. The employment of in-silico techniques, such as machine learning, has grown in popularity as a solution to these problems. With the use of a dataset of over 9,000 compounds and more than 1500 high-throughput test endpoints, we want to build a machine learning model to predict the toxicity of chemicals in this research. Toxicity prediction of newly synthesized chemicals is a crucial task that requires significant time and resources. Traditional laboratory animal testing is not only expensive and time-consuming but also raises ethical concerns. In this context, in-silico techniques, such as machine learning, have become increasingly popular. The challenge presented in this project aims to predict the toxicity of chemicals using machine learning models. The training data is provided in a CSV file that contains the chemical id, assay id, and label. The data needs to be parsed to extract the relevant features and prepare a dataset for each assay. The chemical id is provided as SMILES structures that need to be processed using python libraries. In this competition, I've used RDKit library for the same. Once the training dataset was prepared, as recommended, I performed perform feature selection to avoid using irrelevant features for training the model. All the feature generation techniques using RDKit, and feature selection techniques used through out this competition will be discussed further.

2. DATA PREPARATION

A. Data Parsing

To separate the "ID" column into two new columns called "Chemical_Id" and "Assay_Id", thus used the String split() method, which divides a string at a specified separator and returns a list of the resulting strings. We can use the delimiters specified in the original dataset to perform the splitting process.

B. Preparing Dataset

The dataset given was not complete. We need to generate features from the given SMILES which can be done using RDKit library. Specifically for this competition I have generated Molecular descriptors and Morgan fingerprints using RDKit library. After generating all the features I merged them with the original dataset.

C. Data Pre-processing

The features generated for training and testing datasets contained null values. To handle these null values, I used the fillna() method to replace them with the mean values. Additionally, I used the astype() function to convert the data type of the Assay_Id column to int. I have also used LableEncoder to transform unique values of Chemical_Id in both train and test datasets to map them with their corresponding integer values.

D. Feature Selection

First, I split the target variable which is "Expected" in train dataset and store it in a variable named train_Y, and then I dropped the Expected column from train dataset and stored it in train_X. I used these train_X and train_Y for feature selection. Following algorithms were used for feature selection:

- **Variance Threshold:** This technique involves eliminating features that have low variance. By default, it removes all features with zero variance. The threshold value can be adjusted to determine the optimal number of features to keep. In this project, various threshold values were used to identify the ideal one that produces the best accuracy for the model.

  Ultimately, a variance threshold value of 0.25 was selected for the best model. This value allowed for 129 features to be retained, resulting in the most accurate model possible.

- **Sequential Feature Selector:** The input data is split into features (X) and target variable (y) using the 129 features which we got from variance threshold and 'train_Y' variables. The dataset is then split into training and testing sets using the train_test_split function from the sklearn.model_selection module.

The DecisionTreeClassifier model is initialized and passed as an argument to the SequentialFeatureSelector function from the mlxtend.feature_selection library. The function is set up to select the top 50 features using forward selection, and the f1_macro scoring metric is used to evaluate the model. The CV parameter sets the number of cross-validation folds used to evaluate the model.

The StandardScaler function from the sklearn.preprocessing module is used to scale the training dataset, and the SequentialFeatureSelection algorithm is then applied to the scaled dataset.

The output of the feature selection algorithm is printed to the console, including the best accuracy score, the index of the best subset of features, and the corresponding feature names. The subsets_ attribute of the SFS object is used to generate a pandas DataFrame containing the selected feature subsets and their corresponding scores.

E. Splitting of dataset

I created a list of all the selected final 50 features and then this list was used to split the dataset and select only these 50 features from both test and train datasets. The subset of train dataset containing these 50 features was named X_train and similarly the subset of test dataset was named X_test. Remember, we have the target variable in train_Y.

F. Scaling

The MinMaxScaler class is used to scale the features of a dataset to a given range. This is particularly useful when working with machine learning algorithms that are sensitive to the scale of the input data. Scaling the features can help improve the performance and accuracy of these algorithms by ensuring that all features are on a similar scale. In this case, the data in X_train and X_test is scaled to a range of (0, 1) using the MinMaxScaler object.

3. MODEL FITTING

In the following competition I have used various models for training my dataset such RandomForestClassifier, BaggingClassifier, XGBClassifier, CatBoostClassifier and VotingClassifier. The best accuracy was achieved using VotingClassifier. The VotingClassifier aggregates the results of multiple classifiers and uses them to make a final prediction, potentially improving the model's performance. By combining the results of the XGBClassifier and LGBMClassifier, the VotingClassifier is able to leverage the strengths of both models and create a more robust and accurate model for this competition.

Achieved **80.28** accuracy in internal evaluation and **82.21** accuracy in private leaderboard.

4. INTERNAL EVALUATION

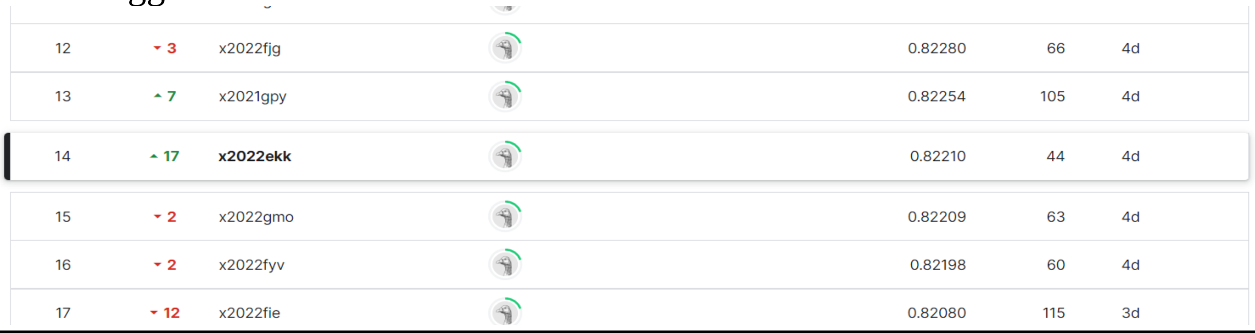
- **Cross-Validation Score**: We calculate cross-validation score by using “cross_val_score” function which is imported from sklearn.model_selection in python.
- **F1-Score:** The cross_val_score function is used while keeping scoring parameter to be f1_macro.

5. PREDICTIONS

A new variable **val_predictions** is created and is appended to the Predicted column of the Output file which is a CSV file created by using .to_csv() function.

6. LEADERBOARD SCORE

For this Toxicity prediction challenge, which is hosted on Kaggle, I did 44 submissions. My Kaggle submission with highest accuracy was positioned at 14th rank on the private leaderboard with a score of 0.82210 and 31st rank on the public leaderboard with a score of 0.80654. The screenshot from the Kaggle leaderboard is seen below.



12	▾ 3	x2022fjg	0.82280	66	4d
13	▴ 7	x2021gpy	0.82254	105	4d
14	▴ 17	**x2022ekk**	0.82210	44	4d
15	▾ 2	x2022gmo	0.82209	63	4d
16	▾ 2	x2022fyv	0.82198	60	4d
17	▾ 12	x2022fie	0.82080	115	3d