St. Francis Xavier University

Department of Computer Science

CSCI 555: Data Mining & Machine Learning

FINAL REPORT

THE TOXICITY PREDICTION CHALLENGE II

Sai Teja Reddy Palam (x2021gpy)

1. INTRODUCTION:

The Toxicity Prediction Challenge II is a machine learning competition hosted on the Kaggle platform. Its purpose is to develop a model that can predict which chemicals are toxic. The competition can be accessed through this link:
https://www.kaggle.com/competitions/the-toxicity-prediction-challenge-ii/overview

There are two datasets provided for the competition: the training dataset and the testing dataset. The training dataset consists of two columns as follows:

- Id: This column contains Chemical_Id and Assay_Id
- Expected: This column indicates whether the chemical is toxic or not

On the other hand, the testing dataset only contains one column "x", which includes Chemical_Id and Assay_Id. The task is to predict whether the given element is toxic or not.

2. DATA PREPARATION:

2.1 Splitting the ID into Chemical_ID and Assay_ID:

- The initial stage of preparing the data for this competition involves splitting the "Id" column in the training dataset. The resulting split strings are then assigned to two new columns named "SMILES" (Chemical_Id) and "Assay_Id".
- Similarly, in the testing dataset, the "x" column is split, and the resulting split strings are assigned to the "SMILES" (Chemical_Id) and "Assay_Id" columns. The following are the columns that are generated after splitting:

```
#Splitting the ID into Chemical_ID and Assay_ID
train[['SMILES', 'Assay_Id']] = train['Id'].str.split(';', expand=True)
test[['SMILES', 'Assay_Id']] = test['x'].str.split(';', expand=True)

#printing the columns of the testing and training datasets
print('Train Columns:', ', '.join(train.columns))
print('Test Columns:', ', '.join(test.columns))
```

✓ 0.3s Python

```
Train Columns: Id, Expected, SMILES, Assay_Id
Test Columns: x, SMILES, Assay_Id
```

2.2 Molecules Generation:

- A new column named "Molecule" is created in both the training and testing dataset by applying the "Chem.MolFromSmiles()" function to the "SMILES" column. This function converts a SMILES string into an RDKit molecule object.

- After that, filtered out any rows in the training and testing dataset where the “Molecule” column is null.

2.3 Feature Generation:

- Feature generation starts by creating a list of callable methods using the “dir()” function from the “Descriptors” module.
- Inside the loop, each method is applied to the SMILES strings of the newly synthesized chemicals, using the “Chem.MolFromSmiles()” function to convert each SMILES string to an RDKit molecule object. The resulting molecular weight of each molecule is then calculated using the method and stored in a new column in the training and testing datasets. The new column is named after the applied method.

2.4 Feature Selection:

- The RFECV (Recursive feature elimination with cross-validation) technique and BorutaPy feature selector are utilized to identify the optimal set of features for classification using the LightGBM Classifier model.
- The result of this process is the identification of approximately 300 important features, which are further refined using BorutaPy with an LGBMClassifier as the base estimator to narrow down to the best features.
- The final set of features includes 33 important features that are stored as separate files, namely **rfe_features.pkl** and **boruta_features.pkl**.
- These features are Assay_Id, AUTOCORR2D_102, AUTOCORR2D_115, AUTOCORR2D_129, AUTOCORR2D_131, AUTOCORR2D_137, AUTOCORR2D_139, AUTOCORR2D_144, AUTOCORR2D_145, AUTOCORR2D_163, AUTOCORR2D_43, AUTOCORR2D_49, AUTOCORR2D_56, BCUT2D_CHGLO, BCUT2D_LOGPHI, BCUT2D_LOGPLOW, BCUT2D_MRLOW, BCUT2D_MWLOW, EState_VSA8, FpDensityMorgan1, HeavyAtomMolWt, Kappa3, MaxAbsPartialCharge, MinPartialCharge, MolLogP, MolMR, PEOE_VSA14, SlogP_VSA5, VSA_EState3, VSA_EState4, VSA_EState5, VSA_EState9, qed.
- By utilizing these features, the model can effectively classify chemical toxicity with high accuracy.

3. MODELING:

An ensemble method is used for modeling that combines 3 LGBMClassifier models as base models in order to produce one optimal predictive model. Each model is trained on different parts of the training dataset where one model is trained with only Toxicity Expected value “1” and some “2”s, while another model is trained with only “2”s and some “1”s and the last model is trained with all of them. A new Logistic Regression model is trained using the predicted probabilities from these three models, and this model is then used to make the final predictions. Each model is described as follows:

3.1 Model-1:

- Selects a subset of training data with only the Toxicity Expected value "1" and another subset with only the Toxicity Expected value "2". Concatenates the two subsets and splits the data into training and testing sets with a 90:10 ratio while maintaining the same proportion of the target variable in both sets.
- Initialized a LightGBM model which is a gradient-boosting framework that uses tree-based learning algorithms. Here are some of the parameters that are specified for this model:
    - boosting_type: This parameter specifies the type of boosting algorithm to be used. In this model, ‘goss’ is used, which stands for Gradient-based One-Side Sampling.
    - n_estimators: This parameter specifies the number of boosting iterations to be performed. In this model, ‘10000’ is used.
    - class_weight: This parameter is used to handle class imbalance by giving more weight to underrepresented classes. In this model, ‘balanced’ is specified, which means that the weight of each class is adjusted inversely proportional to its frequency.
    - max_depth: This parameter specifies the maximum depth of each decision tree. In this model, the value ‘30’ is used.
    - min_split_gain: This parameter specifies the minimum loss reduction required to make a further partition on a leaf node where ‘0.6’ is used in this model.

- importance_type: This parameter specifies the method used to calculate feature importance. In this model, ‘shap’ is used.
- reg_lambda: This parameter specifies the L2 regularization strength to prevent overfitting. In this model, ‘0.2’ is used.
- num_leaves: This parameter specifies the maximum number of leaves in each decision tree. In this model, the value ‘50’ is used.
- random_state: This parameter specifies the random seed used for initializing the random number generator. This ensures that the results are reproducible.

- Subsequently, the LGBMClassifier is fitted to the training data. Once the model is trained, it is used to predict outcomes on the test data, allowing for the evaluation of the model’s performance.

3.2 Model-2 and Model-3:

- For Model-2, selected a subset of training data with only the Toxicity Expected value “1” and another subset with only some Toxicity Expected value “2”.
- For Model-3, selected a subset of training data with only some Toxicity Expected value “1” and another subset with only Toxicity Expected value “2”.
- LGBMClassifier object is initialized for both Model-2 and Model-3 using the specified hyperparameters as in Model-1. Subsequently, the LGBMClassifier is fitted to the training data. Once the model is trained, it is used to predict outcomes on the test data, allowing for the evaluation of the model’s performance.

4. RESULT:

After running all three models, ensembling is performed which refers to the process of combining the predictions of multiple models to obtain a final prediction. To make a final prediction, the Soft Voting technique is used where the predicted probabilities of these models are combined. Basically, this technique takes the confidence of each model in its prediction, unlike hard voting where the final prediction is based on the majority vote.

The first step is making predictions on the test data using Model-1, Model-2, and Model-3. The predict_proba method is used to predict the probabilities of the test data, and the first column of predicted probabilities is stored in separate variables. These predictions are then added to a DataFrame with columns named after the respective models. Next, a Logistic Regression model is fitted on the data in the DataFrame, using all columns except "original" as features and "original" as the target variable where it contains labels.

The fitted model is then used to make predictions on the test data, and the predicted probabilities are stored in a new DataFrame. Finally, a submission DataFrame is created with the predicted values and written to a CSV file. In conclusion, Soft Voting is a powerful technique for ensembling that allows us to combine the predictions of multiple models. By using the predicted probabilities of different models, we can obtain a final prediction that takes into account the confidence of each model in its prediction.

5. LEADERBOARD SCORE:

6. RESOURCES:

You can find all the resources for this competition, including the code and instructions on how to run it, in the GitHub repository at:
https://github.com/palamsaiteja333/Toxicity-Prediction-Challenge

Note:

The above model may not yield the same result as seen in the leaderboard. This is because I did not save the features generated by the "rfe" and "borutapy" feature selectors which resulted in the above score.