

REPORT ON TOXICITY PREDICTION

INTRODUCTION

In the toxicity prediction challenge, the main goal is to build a machine learning prediction model which detects if a chemical is toxic (i.e., 1) or non-toxic (i.e., 2) using the features associated with that chemical. For these predictions, four data files for the chemical compounds are provided i.e., the training data, testing data, features, and the name of the features.

DATA PREPARATION

STEP 1: DATA PARSING

- Renamed the “x” column in the test dataset to “Id” by using **rename()** function to keep both datasets in consistent format.
- Splitting the “ID” column into two different columns namely “chemical id, Assay id” by considering the delimiters mentioned in the original dataset. The splitting is done by using the String split () method which breaks up a string at the specified separator and returns a list of strings.

STEP 2: PREPARING DATASET

- As the training and testing dataset is not in a complete format, to get a complete dataset with all the information, we merged both the datasets with the feamat dataset. Using the merge function from the pandas library, we merged the 2 datasets to give a final dataset with all chemicals along with their features.

STEP 3: DATA PRE-PROCESSING

- The training and testing datasets had infinity values and there were no null or missing values in the dataset. To deal with infinity values, we used **replace()** method which takes 2 positional arguments. First is the list of the values to replace and second with which values you want to replace the values. So, here we replaced all the infinity values with null values.
- To deal with the null values we replaced them with the median values using the **fillna()** method.
- Used **astype()** function to cast the Assay Id column’s datatype to float.

STEP 4: SPLITTING THE DATASET

Before the model fitting, we split the test and train datasets as follows:

- X_train - Contains all the columns of training data except columns “Expected” and “Chemical ID”. These columns are dropped from train dataset.
- Y_train - Contains only the column “Expected” from train dataset.
- X_test - Contains all the columns from test data except column “Chemical Id”. The column “Chemical Id” is dropped from the test dataset.

STEP 5: FEATURE SELECTION:

We have used a different feature selection method to select the optimal features for each model.

1. Variance Threshold:

- It's a technique for removing features whose variance isn't high enough. It removes all Zero-variance functionality by default.
- By changing the threshold value, we can choose the best number of features.
- We used various threshold values in this project and chose the right one that suits the model and gives us the best accuracy.
- For our best model, we have used a variance Threshold value of 0.21 which gave us 149 best features.

2. SelectKbest:

- It's a feature selection process that takes Score function like chi2, mutual_info_classif as a parameter, and returns K features with the highest scores.
- We used SelectKbest for feature selection in our project, and we choose a few best features by setting K values in the range of (30-500). But it didn't help us increase our score in the end, so we stopped doing it.

STEP 6: SAMPLING:

- We found Imbalanced Data in our dataset so to handle this we have used Oversampling techniques known as “SMOTE”.
- SMOTE is a technique implemented by using an imbalanced-learn python library.
- For our final model, we have used SMOTE techniques which help us to improve our score.

STEP 7: SCALING

To scale the datasets, we have used two different types of scaling such as

1. Standard Scaling:

- To standardize the data, we used the Standard Scaling function **StandardScaler()**, which was applied to all of the selected optimal features.

2. Normalization

- We have used the **MinMaxScaler()** function to standardize the data on all chosen optimum functions.
- But for our final best model, we used the **StandardScaler()** function, which gave us the best ranking.

MODEL FITTING:

The Different models and their results are tabulated below:

Sr.No	Model	Features	F1/CV Score	Parameters	Private leader board score	Public Leaderboard Score
1	XGboost	149	F1 score=0.93	max_depth=8, n_estimators=665	0.80243	0.81550
2	XGboost	149	F1 score=0.80	max_depth=8, n_estimators=400	0.79957	0.81601
3	Gradient Boosting	300	F1 score=0.75	n_estimators=200, learning_rate=0.35, max_features=46, min_samples_leaf=0.116329, max_depth=7, min_samples_split=0.1	0.74245	0.76223

4	Bagging classifier	225	F1 score=0.78	base_estimator=estimator, n_estimators=10, random_state=0	0.76462	0.78669
5	Extra tree classifier	All features	CV score= 83.06	n_estimators=10	0.65503	0.67456
6	Decision tree classifier	All features	CV score= 88.012	criterion = 'entropy', random_state = 11	0.74422	0.76752
7	k-nearest Neighbors	All features	CV score= 85.74	n_neighbors = 5, random_state = 11	0.67547	0.67328
8	Random forest classifier	All features	CV score=83.36	n_estimators = 200, random_state = 11	0.64547	0.66832

1. Random Forest Classifier

The first model which we have used is the Random forest classifier which gave us the cross-validation score of 83.36. The parameters selected for this model are n_estimators = 200, random_state=11. The confusion matrix for this model is:

[4235 6683]

[6183 60312]

After submission on Kaggle score achieved was 66.83%. Further, we tried processing the random forest with different parameters, but the score was not much affected.

2. K-nearest Neighbors

We tried using the KNN model to check if we can improve the accuracy of the model. It was found that with KNN, the model was not giving much accuracy as compared to the random forest classifier. So further to increase the model accuracy we decided to try scaling and used a robust scaler for scaling the features. Post scaling, the model accuracy was improved to 85.74% however the Kaggle score was Improved by just 1% i.e., 67.328%. The parameters selected for this model are n_neighbors=5, random_state=11. The confusion matrix is:

[6317 4601]

[4679 61816]

3. Decision Tree Classifier

First, we used the decision tree classifier with default parameters, which resulted in a cross-validation score of 87.47 percent and a leaderboard score of 74.65 percent. Then, using the criteria "entropy" and the parameters random state=11, we obtained a cross-validation score of 88.012 percent and a leaderboard score of 76.75 percent. It was observed that the decision tree classifier gave the highest accuracy as compared to the other models. The confusion matrix is as shown below:

[6317 4601]

[4679 61816]

4. Extra Tree Classifier

We used Extra Tree Classifier with the `n_estimators=10` parameter and picked all features, resulting in a leaderboard score of 67.456. The cross-validation score resulted in 83.06. We decided not to do this in future work because it was not an improvement compared to the previous score. The confusion matrix is:

```
[4657    6261]
```

```
[6852    59643]
```

5. Bagging Classifier

We used Bagging Classifier on the optimal number of features selected by the feature selection method and base estimator as a Decision tree. As a result, it gave us a cross-validation score of 88.72% and a leaderboard score of 75.10%. Then we changed the variance threshold values to a range of [0.1 to 0.5], and 225 features were chosen at a threshold value of 0.17. We increased our score by 2.68 percent to 78.669 percent. We began evaluating our internal score using the F1 score, which was 0.78. The confusion matrix is:

```
[59956    6539]
```

```
[13825    52670]
```

6. Ada Boost Classifier

We used the AdaBoost classifier with the parameters `n_estimators=100`, `random_state=11`, and various thresholds for this model, the best being threshold value 0.2, which gives 179 features, but the Leader Board score was 66.07 percent, which was very poor compared to all previous scores. The cross-validation score resulted in 87.16. The confusion matrix is:

```
[1843    9075]
```

```
[863     65632]
```

7. Gradient Boosting classifier

Next, we used a Gradient Boosting classifier with various parameters chosen by `RandomizedSearchCv`, and we tested different features using different feature selection methods, eventually settling on 300 best features as the best feature for this classifier, giving us an f1 score of 0.76 and a leader board score of 76.72. The cross-validation score resulted in 86.54. As a result, we stopped using this model because it did not increase our ranking. The confusion matrix is:

```
[901     10017]
```

```
[399     66096]
```

8. XGboost Classifier

To increase the leaderboard ranking, we used the XGboost classifier with a set of parameters chosen by `RandomizedSearchCv`, but only two of them were used: `max_depth=8` and `n_estimators=400`. For feature selection, we used the Variance threshold, changing the threshold values to 0.17, which gave 149 features with the best score. The leader board score was 0.8059, and the f1 score was 0.80. Then we used scaling and oversampling strategies on the same code, resulting in an F1 score of 0.93 and a leaderboard score of 0.81601. The confusion matrix is:

[60290 6205]

[3064 63431]

We changed the `n_estimators` parameter to 665, which resulted in a f1 score of 0.94 and a leaderboard score of 0.81550, which was chosen for the private leaderboard score evaluation with a private leaderboard score of 0.80243.

PARAMETER SELECTION:

We used **RandomizedSearchCV** to find the best parameters for our models. It is a parameter selection process in sklearn that can be imported from the python library sklearn model selection.

INTERNAL EVALUATION:

For internal evaluation to ensure which model will give the highest score on Kaggle we used

1. Cross-Validation Score:

- We calculate cross-validation score by using “`cross_val_score`” function. Which is imported from `sklearn.model_selection` in python.

2. Confusion Matrix:

- We used a confusion matrix for our internal evaluation to see an increase in true positive and true negative values.
- The confusion matrix is calculated using `cross_val_predict()` function to get the predictions and `confusion_matrix()` function to get the result.

3. F1-Score:





- To calculate F1-score we used `Classification_report()` function with parameters true predictions values and the values which are predicted using `cross_val_predict()` function.

PREDICTIONS

- For predictions, we have created a new variable “prediction” and stored the data using `pd.DataFrame()` pandas function.
- To get the prediction file we concatenated the “Chemical Id” and “Assay Id” from the test dataset using `str.cat()` function and stored the concatenated value to “ID”.
- The output file is generated by converting the Prediction file into the comma-separated format by using `.to_csv()` function.

LEADERBOARD SCORE:

For this Toxicity prediction challenge, which is hosted on Kaggle, we did 52 submissions. Our final Kaggle submission earned us 4th place on the private leaderboard with a score of 0.80243 and 5th place on the public leaderboard with a score of 0.81601. The screenshot from the Kaggle leaderboard is seen below.

#	△pub	Team Name	Notebook	Team Members	Score 📊	Entries	Last
1	▲ 2	NextGen			0.80775	43	5d
2	▲ 5	x2020fvt			0.80478	29	8d
3	▼ 2	Maple Squad			0.80294	74	5d
4	▲ 1	Orion			0.80243	52	5d