**The Toxicity Prediction Challenge II**

**CSCI 555 course – Prof. Othman Soufan**

**Report by Laxman Uppal (202003782)**

Introduction

Before starting the competition on Kaggle platform, there are few steps to go through to form a reproducible code as the steps below :

1. Understand the problem statement: Read and understood the problem statement of the competition to identify the key objectives and standards that need to be achieved.
2. Analyze the data: Analyzing the data provided to depict the features and target variables. Explored the data and the constraints related to it by going through the documentation.
3. Preprocess the data: Preprocess the data to make it suitable for training the machine learning models. This includes handling missing values to normalize the data.
4. Train the models: Train multiple machine learning models on the preprocessed data. Experimented with different algorithms like RandomForest classifier, Deep Neural Networks, GradientBoost classifier and Catboost classifier. Along with the algorithms, I have been studying about the hyperparameters to achieve the best performance through Google and Chat-GPT in understanding the best parameters to go with to achieve maximum toxicity prediction.
5. Evaluate the models: Evaluate the performance of the models using the different algorithms and measurers provided by the competition to identify the models that perform the best on the test set.
6. Submit the predictions: Generate predictions on the test set using the best-performing model. Submit the predictions to the competition and obtain the score.
7. Best choice of submissions: Talking about achieving maximum score, it was tricky with the private leaderboard hidden. So, I had to choose different submissions as my best choices among which one of them was my highest achievable score on public leaderboard.

Below, I have included the description of my code for the best submission. Here's a brief documentation of the code to get the required prediction file:

* The first line of code imports the pandas library for data manipulation, hashlib for hashing functions, rdkit for molecular chemistry functions, and CatBoostClassifier for gradient boosting.
* The next two lines of code load the training and test datasets from CSV files using pandas.
* The following two lines of code split the 'Id' column in the training dataset and 'x' column in the test dataset into separate columns for 'chemical\_id' and 'assay\_id'.
* The next two lines of code use rdkit's Chem.MolFromSmiles() function to convert the SMILES strings in the 'chemical\_id' columns of the training and test datasets into molecular objects.
* The following two lines of code remove rows with invalid SMILES strings from the training and test datasets.
* The next eight lines of code calculate eight molecular descriptors (MolLogP, TPSA, Chi2v, MolMR, Chi1v, Chi3v, fr\_benzene, Chi4v) for the training dataset using the molecular objects obtained from the SMILES strings.
* The next eight lines of code calculate the same eight molecular descriptors for the test dataset.
* The following line of code uses CatBoostClassifier with the following hyperparameters: n\_estimators=1000, learning\_rate=0.3, max\_depth=10, subsample=0.75, random\_state=42. I have selected these features by trying out different combination of features along with the help of Chat-GPT which has been providing me a good understanding of the different descriptors which contribute for maximum accuracy.
* The next line of code trains the classifier on the training dataset using the eight molecular descriptors as features and the 'Expected' column as the target.
* The next line of code predicts the outcomes for the test set using the same eight molecular descriptors as features.
* The following line of code creates a pandas DataFrame with the predicted outcomes and the corresponding 'x' values from the test dataset.
* The next line of code writes the predicted outcomes to a CSV file.
* Overall, this code is using CatBoostClassifier to predict the outcomes of some chemical assays based on molecular descriptors calculated from SMILES strings.

Explanation of internal evaluation code :

* Line 1-2: Comments indicating that the code is related to trials and cross-validation.
* Line 4: Importing the pandas library as pd.
* Line 5-6: Importing hashlib library and Chem module from rdkit library.
* Line 7: Importing the Descriptors module from rdkit.Chem library.
* Line 8-9: Importing KFold and metrics from sklearn library.
* Line 10: Importing CatBoostClassifier from catboost library.
* Line 13: Reading a CSV file ("train\_II.csv") using pandas read\_csv() function and storing it in the data variable.
* Line 16-18: Splitting the 'Id' column into 'chemical\_id' and 'assay\_id' columns by applying the lambda function to the 'Id' column and storing the output in data variable.
* Line 21-23: Converting SMILES strings into molecule objects by applying the lambda function to the 'chemical\_id' column and storing the output in the 'Molecule' column of data variable.
* Line 26: Removing the rows with invalid SMILES strings by checking if the 'Molecule' column contains null values and storing the output in the data variable.
* Line 29-37: Creating new features based on the molecule data by applying the lambda function to the 'Molecule' column and storing the output in the data variable.
* Line 40-41: Separating the features and labels from the data variable and storing them in the features and labels variables respectively.
* Line 44-46: Setting up the k-fold cross-validation by defining the number of splits, the split method (shuffle), and the random state.
* Line 49-61: Training and evaluating the CatBoostClassifier model using k-fold cross-validation. The model is trained on each fold of the data and its accuracy is evaluated on the test set. The accuracies of each fold are stored in the accuracies list.
* Line 64: Computing the mean accuracy across all folds and printing it.
* Line 65: Computing the F1 score using the f1\_score() function from sklearn.metrics library and printing it.

Leaderboard screenshot :

Graphical user interface, application

Description automatically generated

But my best submission results me to give a prediction of 82.302% after using the suggestions from our fellow mates in the last class of the course.

Graphical user interface, website

Description automatically generated