Report_work

Contributors

Shyama Goel (2021492)
Vinit Kumar Kushwah (2021501)
Shivam (2021489)

Github link:

https://github.com/cosylabiiit/Winter24_toxic_molecule_prediction

Drive link:

https://drive.google.com/drive/folders/1D9N-AoiLN-wuvImy6TY-LQ-oN0PdxIb1

Introduction-

The application of machine learning (ML) models in predicting compound toxicity has gained significant attention in recent years. This paper explores the effectiveness of various ML algorithms in predicting the toxicity of chemical compounds. By utilizing vast amounts of chemical data, these models aim to accurately predict compound toxicity, thus aiding in the early identification of potentially hazardous compounds and facilitating safer drug development processes.

Literature review-

The study on predicting compound toxicity through machine learning (ML) models provides a significant contribution to the field of cheminformatics, particularly in the early identification of hazardous compounds and enhancing the safety of drug development. This research harnesses an extensive dataset comprising 7,722 molecules (3,619 positive and 4,103 negative), sourced from multiple research papers and databases such as the Toxicity Database and the OpenFoodTox database. This dataset is rich in molecular descriptors and features, totalling 13,089, including PubChem IDs, SMILES, and classes.

Feature generation is handled through RDKit and PaDEL, where 211 and 2,758 types of columns are created respectively. These features encompass a broad range of molecular descriptors like atomic counts, molecular connectivity indices, autocorrelation descriptors, and more advanced metrics like log P values and molecular volumes. This extensive profiling is critical for understanding molecular behaviour and interactions.

Feature selection is streamlined using methods like the Variance Threshold method and Pearson correlation coefficient analysis, reducing the features to a more manageable number that retains essential information while eliminating redundancy. For instance, after applying variance thresholding and Pearson correlation analysis, the dataset dimensions were reduced significantly.

This literature review summarizes the various machine learning techniques used in our study to predict the toxicity of chemical compounds, each chosen for their specific strengths and suitability for handling complex datasets involved in toxicological predictions.

1. Logistic Regression

Logistic regression is a straightforward statistical method used for binary classification, which means it's used to distinguish between two possible outcomes (like toxic vs. non-toxic). In this study, it helps estimate the likelihood of a compound being toxic based on several influencing factors.

2. Gaussian Naive Bayes

Gaussian Naive Bayes is a simple yet effective algorithm particularly good for large datasets with many features, as it assumes that all features follow a normal distribution. This approach is efficient and quick to implement, making it ideal for our extensive dataset of molecular descriptors.

3. Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a robust algorithm that finds the best boundary (hyperplane) that separates data into two categories. It's highly accurate in many scientific fields, especially useful here for its precision in dealing with complex datasets.

4. Random Forest Classifier

The Random Forest classifier builds multiple decision trees and merges them together to get more accurate and stable predictions. It's great for handling large datasets with high dimensionality, making it easier to figure out which features are most important for predicting toxicity.

5. Decision Tree Classifier

A Decision Tree Classifier makes decisions by splitting data into branches at different levels based on feature values. It's very intuitive and easy to visualize, which helps in understanding how decisions are made, useful in preliminary data analysis.

6. Neural Networks (Keras)

Neural networks, specifically those implemented using the Keras library, are designed to mimic how human brains operate, making them excellent at recognizing patterns in large, complex datasets. They're used in this project to tackle the non-linear relationships and intricate patterns among the data features.

7. AdaBoost

AdaBoost, or Adaptive Boosting, is a technique that combines several weak classifiers to form a strong classifier. The method focuses on difficult cases by adjusting the importance of misclassified instances, thereby improving the model's accuracy with each iteration.

8. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is an easy-to-understand, effective classification technique that classifies new examples based on the most common class among its nearest neighbours. It's a straightforward, non-parametric method that's been widely used for pattern recognition.

9. XGBoost

XGBoost stands for Extreme Gradient Boosting and is known for its performance and speed in building models. It's particularly effective for large-scale machine learning tasks, due to its ability to handle sparse data efficiently.

10. Hyperparameter Tuning

Hyperparameter tuning involves choosing a set of optimal parameters for a learning algorithm. Techniques like grid search or random search are commonly used to find the best settings, enhancing the overall performance of the models.

This literature review encapsulates the diversified techniques and methodologies implemented in this project, underlining the vast potential of machine learning in enhancing predictive accuracy in toxicological assessments. Each method has its strengths and limitations, which are leveraged according to the specific requirements of our predictive modelling tasks.

Materials and Methods

Dataset:

Write about the nature of the dataset, the source from where you collected the dataset and the count of molecules.

Data collection :

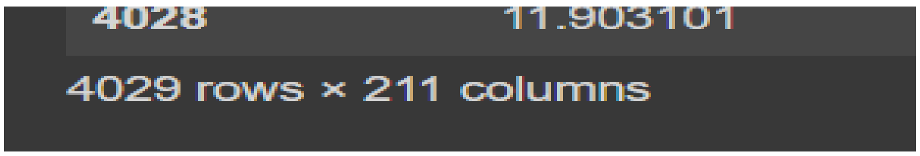
We searched for several research papers, and articles in order to find the toxic molecules in food compounds.

Total count of molecules: 7722
Positive molecules: 3619
Negative molecules: 4103
Total features: 1389 (included pubchem_id,smile,class)

Feature Generation

Rdkit:

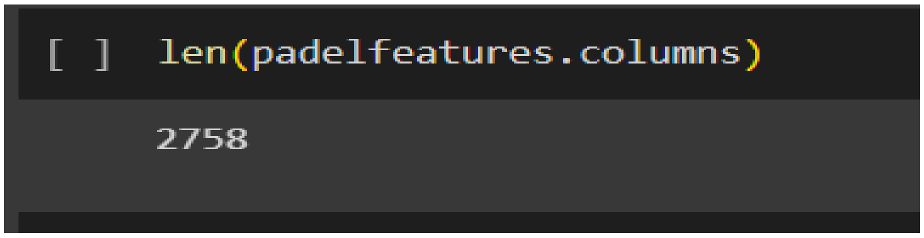
We have generated a total of 211.



```
4028                11.903101
4029 rows × 211 columns
```

Padel:

We have generated a total of 2758 columns



```
[ ]  len(padelfeatures.columns)

     2758
```

Total no of features-1389

```
(7722, 1389)
```

The dataset comprises an extensive array of molecular descriptors and PubChem fingerprints tailored for cheminformatics. Features include basic atomic properties (e.g., counts of various atom types like hydrogen, and carbon), molecular connectivity indices, and autocorrelation descriptors which reflect the molecular structure's depth and complexity. Additionally, the dataset incorporates more advanced descriptors such as logP values, molecular volume, and topological features, essential for predicting molecular behaviour and interactions. The inclusion of over 881 binary PubChem fingerprints provides detailed insights into the presence of specific substructures within the molecules, crucial for tasks such as drug design and toxicity prediction.

Feature Selection:

We have implemented two types of feature selection methods:

i) variance threshold method:

In this we used the VarianceThreshold method from scikit-learn for feature selection. It drops specific columns ('pcid', 'smile', 'class') from the DataFrame, removes rows containing infinite values, and then applies the VarianceThreshold method with a threshold of 0.02. Finally, it transforms the data frame to include only the selected features.
After this, we have 876 columns.

ii) Pearson correlation coefficient:

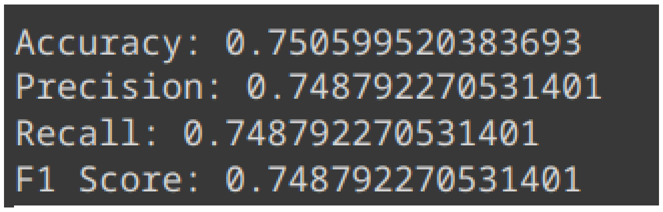
The Pearson correlation coefficient for each pair of features in the DataFrame `cleaned_df` and visualizes the correlation matrix using a heatmap. It identifies highly correlated features (with a correlation coefficient greater than 0.98) and stores them in the list `columns_to_drop`, which are potential candidates for removal to reduce multicollinearity.
After applying this we have 7722 rows × 712 columns in our dataset.

iii) PCA(principal component analysis):

After applying the PCA we have finally left 7721 rows × 250 columns in our dataset. We have chosen 250 principle components in this technique.

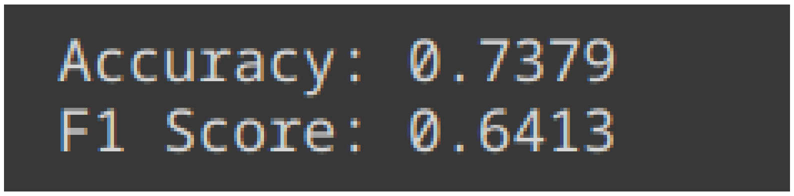
Model Implementation and Results

1. Logistic regression



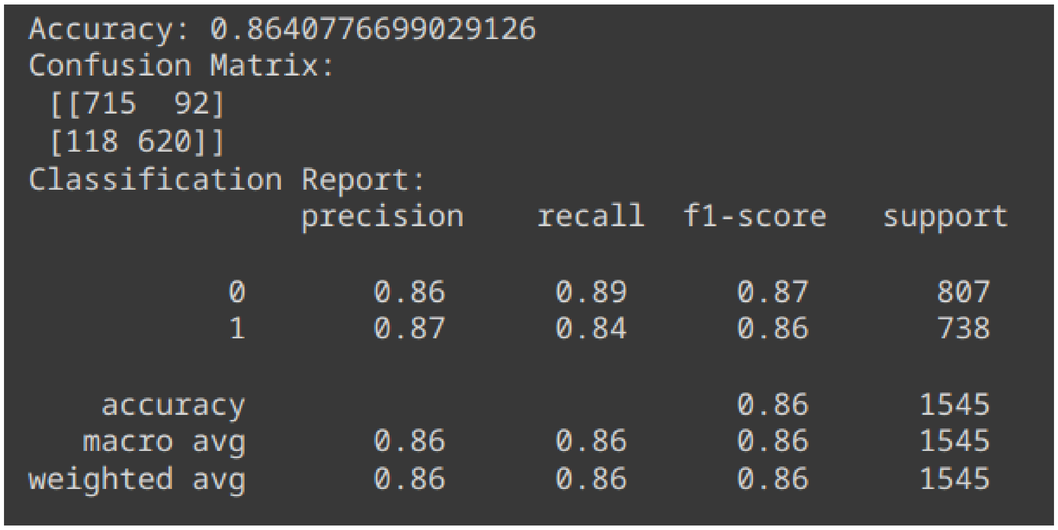
```
Accuracy: 0.750599520383693
Precision: 0.748792270531401
Recall: 0.748792270531401
F1 Score: 0.748792270531401
```

2. Gaussian Naive Bayes



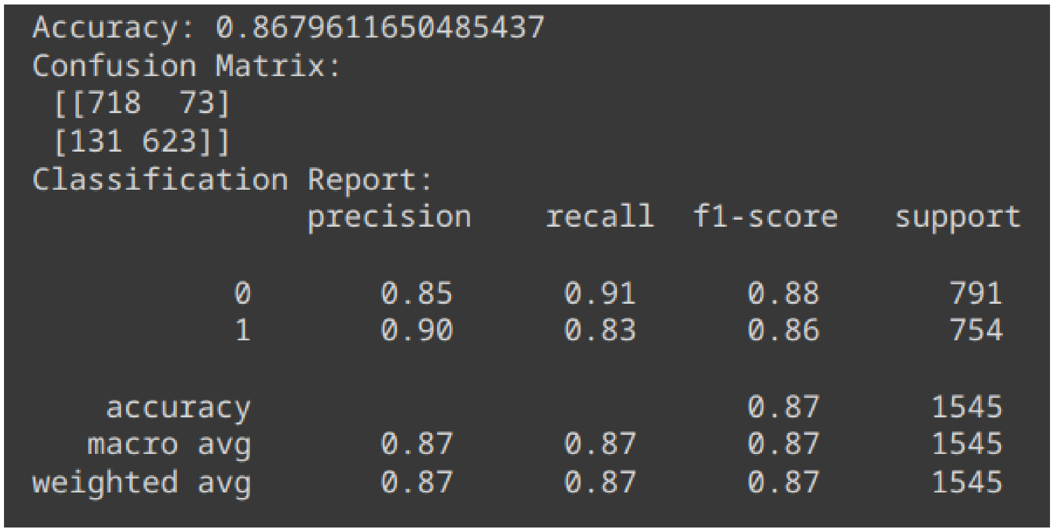
```
Accuracy: 0.7379
F1 Score: 0.6413
```

3. Svm (support vector machine)



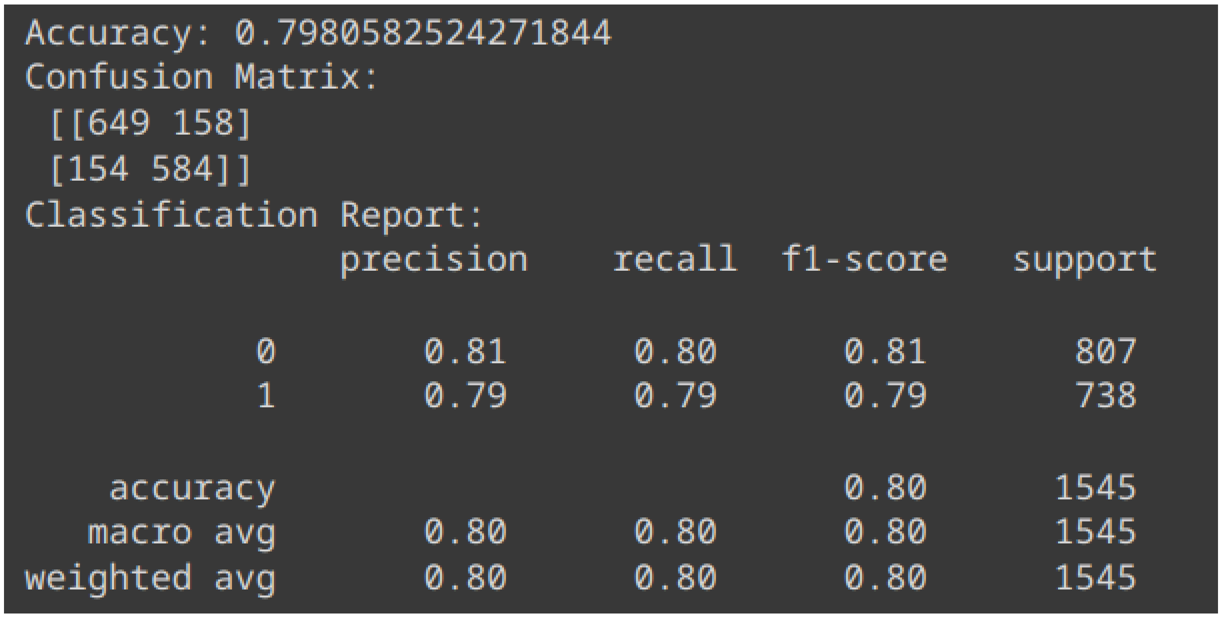
```
Accuracy: 0.8640776699029126
Confusion Matrix:
 [[715  92]
 [118 620]]
Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.89      0.87       807
           1       0.87      0.84      0.86       738

    accuracy                           0.86      1545
   macro avg       0.86      0.86      0.86      1545
weighted avg       0.86      0.86      0.86      1545
```

4. Random forest classifier

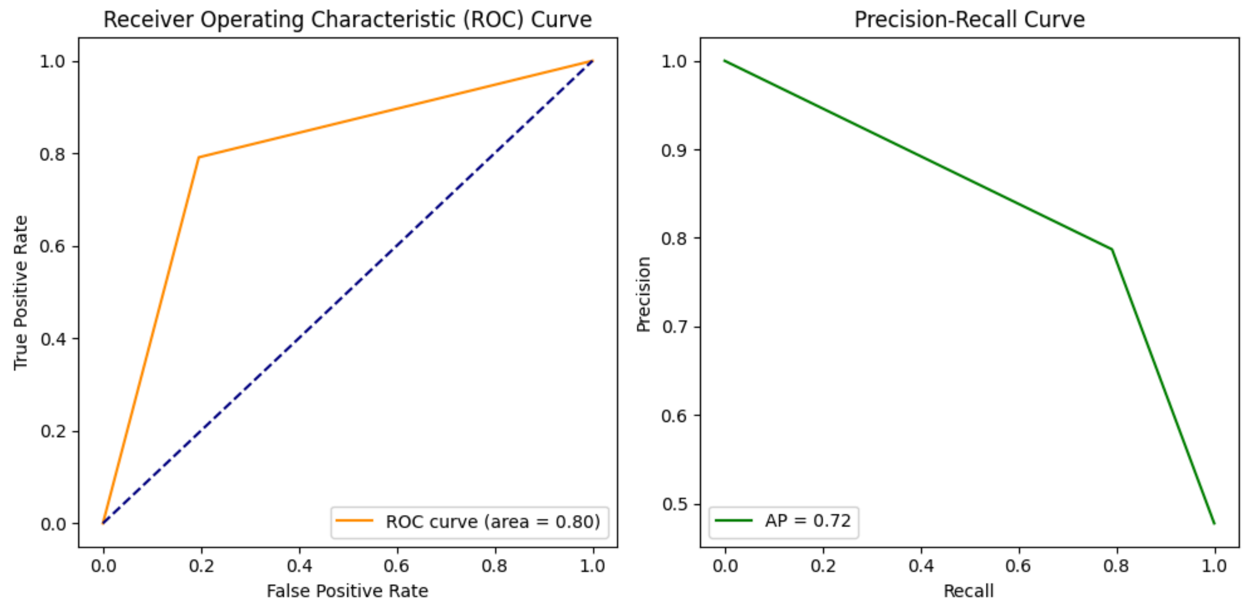


```
Accuracy: 0.8679611650485437
Confusion Matrix:
 [[718  73]
 [131 623]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       791
           1       0.90      0.83      0.86       754

    accuracy                           0.87      1545
   macro avg       0.87      0.87      0.87      1545
weighted avg       0.87      0.87      0.87      1545
```

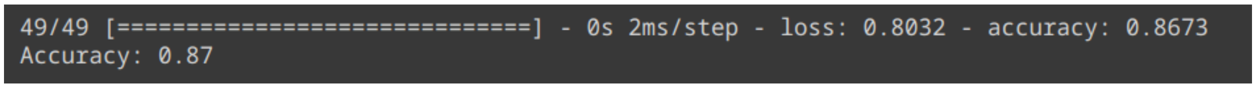
5. Decision tree classifier



```
Accuracy: 0.7980582524271844
Confusion Matrix:
 [[649 158]
 [154 584]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.80      0.81       807
           1       0.79      0.79      0.79       738

    accuracy                           0.80      1545
   macro avg       0.80      0.80      0.80      1545
weighted avg       0.80      0.80      0.80      1545
```

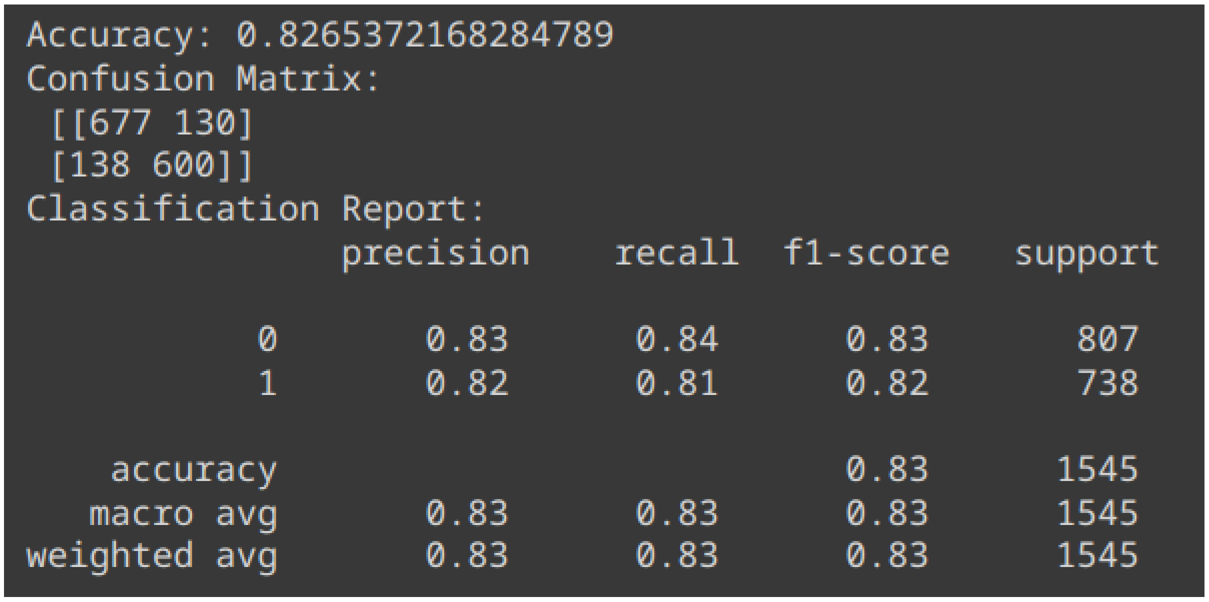


6. Neural network Keras



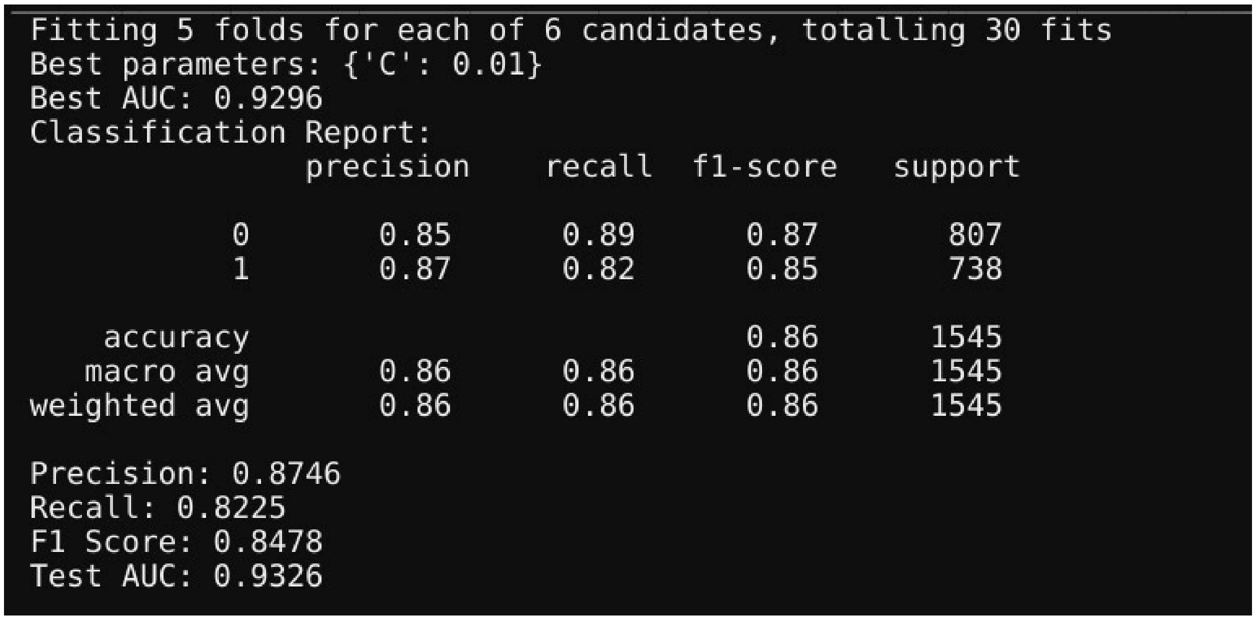
```
49/49 [==============================] - 0s 2ms/step - loss: 0.8032 - accuracy: 0.8673
Accuracy: 0.87
```

7. Ada Boost



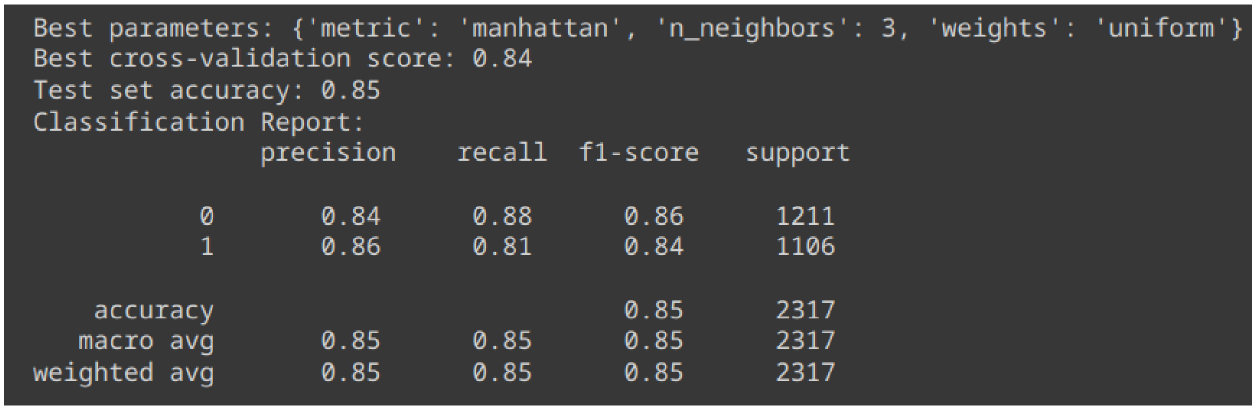
```
Accuracy: 0.8265372168284789
Confusion Matrix:
 [[677 130]
 [138 600]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       807
           1       0.82      0.81      0.82       738

    accuracy                           0.83      1545
   macro avg       0.83      0.83      0.83      1545
weighted avg       0.83      0.83      0.83      1545
```

8. LogisticRegression(using grid search and hyperparameter tuning)



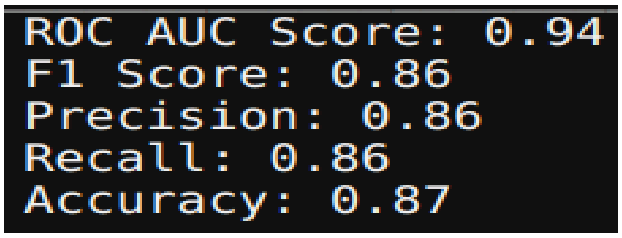
```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters: {'C': 0.01}
Best AUC: 0.9296
Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.89      0.87       807
           1       0.87      0.82      0.85       738

    accuracy                           0.86      1545
   macro avg       0.86      0.86      0.86      1545
weighted avg       0.86      0.86      0.86      1545

Precision: 0.8746
Recall: 0.8225
F1 Score: 0.8478
Test AUC: 0.9326
```

9. KNN (K nearest neighbours) using grid search



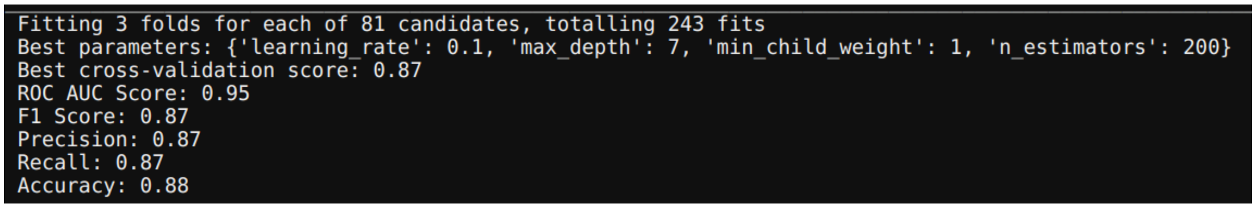
```
Best parameters: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'uniform'}
Best cross-validation score: 0.84
Test set accuracy: 0.85
Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.88      0.86      1211
           1       0.86      0.81      0.84      1106

    accuracy                           0.85      2317
   macro avg       0.85      0.85      0.85      2317
weighted avg       0.85      0.85      0.85      2317
```

10. XGBoost



```
ROC AUC Score: 0.94
F1 Score: 0.86
Precision: 0.86
Recall: 0.86
Accuracy: 0.87
```

11. XGBoost (hyperparameter tuning)

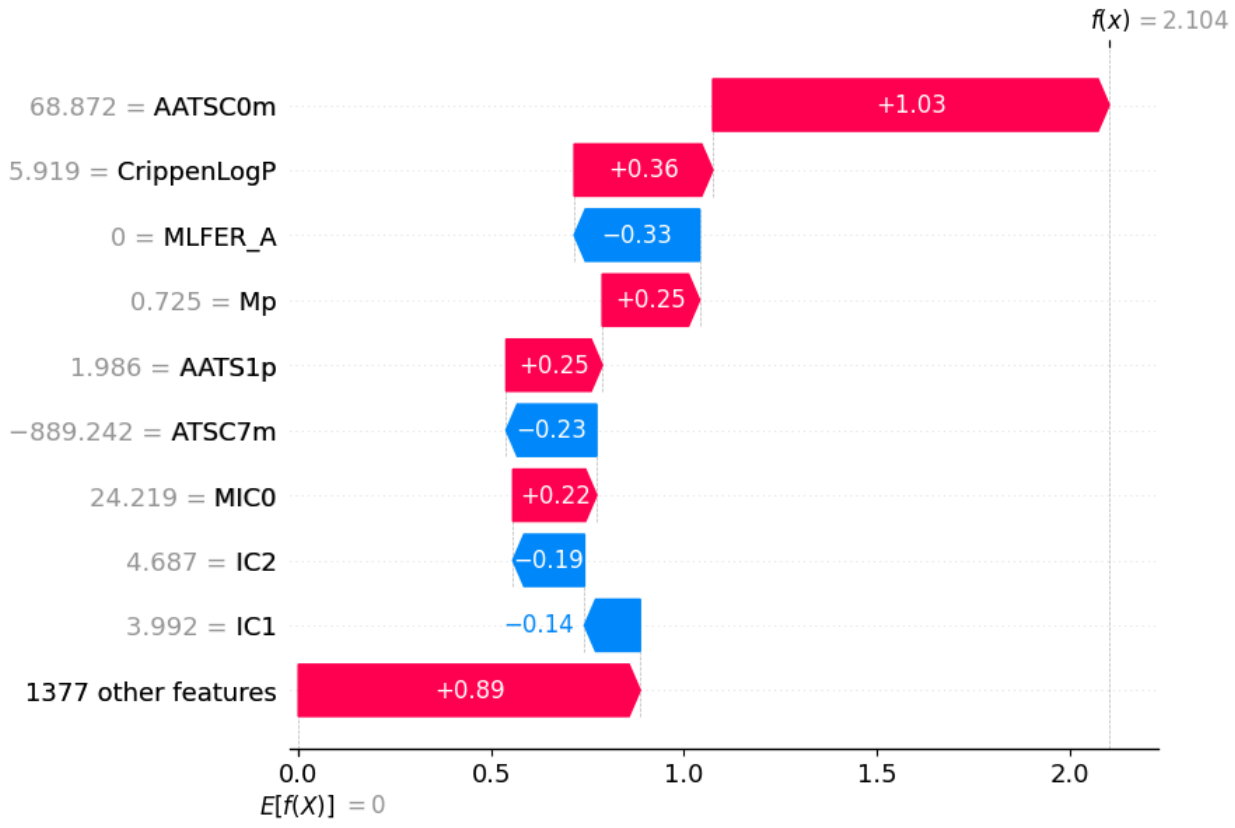


```
Fitting 3 folds for each of 81 candidates, totalling 243 fits
Best parameters: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1, 'n_estimators': 200}
Best cross-validation score: 0.87
ROC AUC Score: 0.95
F1 Score: 0.87
Precision: 0.87
Recall: 0.87
Accuracy: 0.88
```

SHAP analysis

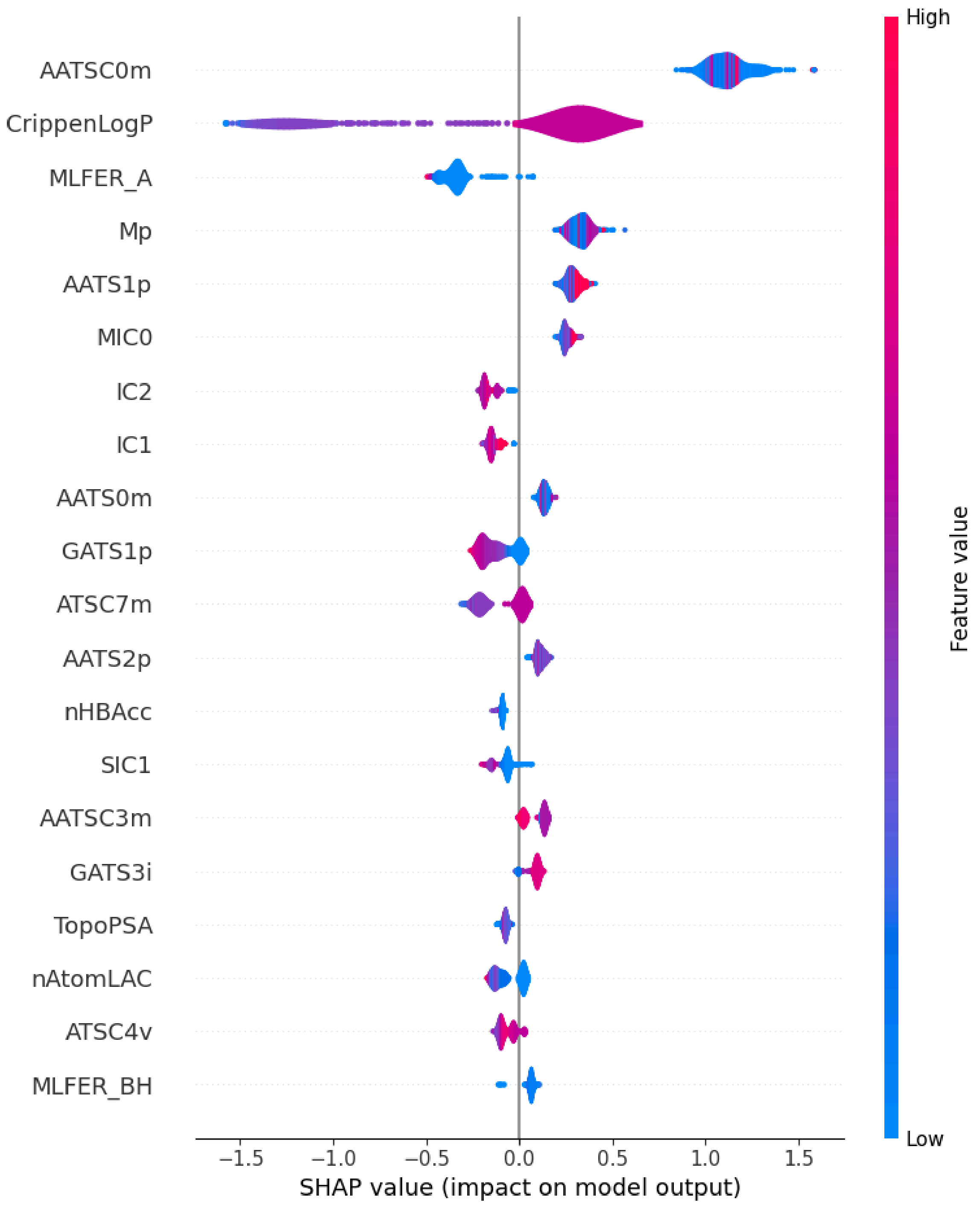
Waterfall Plot Image

The waterfall plot offers a detailed breakdown of the contribution of each feature to the model's prediction for a specific instance. It visualizes how the combination of features leads to the final output, highlighting the direction and magnitude of each feature's effect.



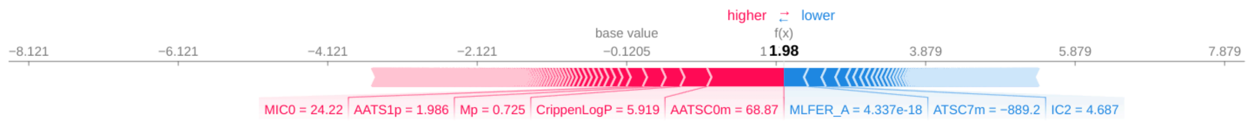
Summary Plot Image

The summary plot provides a comprehensive overview of feature importance. It presents a violin plot for each feature, showing the distribution of SHAP values across the dataset. Features with wider violin plots indicate higher importance in influencing model predictions.

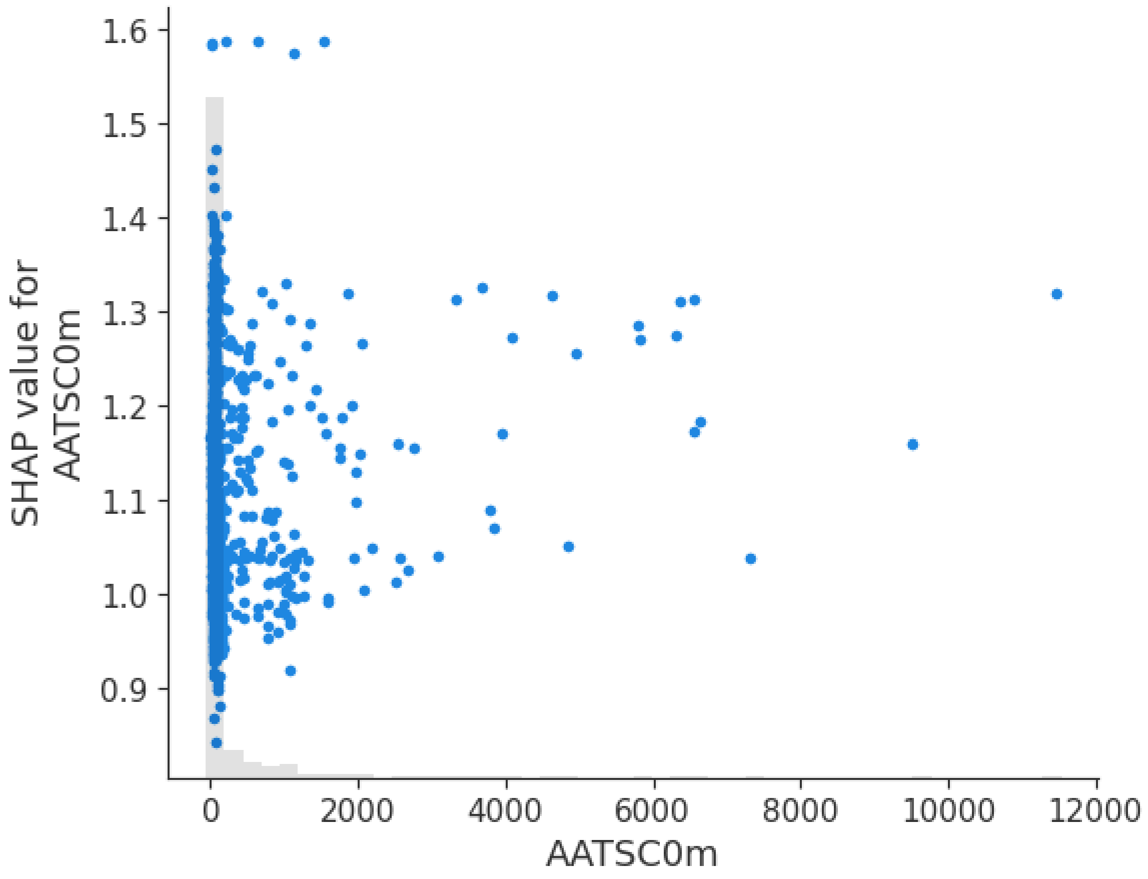


Force Plot Image

We also visualized individual SHAP values for specific predictions using force plots. These plots illustrate how each feature contributes to the model's output for an individual sample. This helps in understanding why the model made a certain prediction for that instance.



SHAP value for AATSC0m



Top 10 Features by Importance

After computing feature importances, we identified the top 10 features based on their SHAP values. The following list presents these features along with their corresponding importance scores and normalized values:

AATSC0m -> 1.1216 (softmax = 0.0022)
CrippenLogP -> 0.5525 (softmax = 0.0012)
MLFER_A -> 0.3534 (softmax = 0.0010)
Mp -> 0.3230 (softmax = 0.0010)
AATS1p -> 0.2857 (softmax = 0.0010)
MIC0 -> 0.2522 (softmax = 0.0009)
IC2 -> 0.1685 (softmax = 0.0008)
IC1 -> 0.1440 (softmax = 0.0008)
AATS0m -> 0.1313 (softmax = 0.0008)
GATS1p -> 0.1234 (softmax = 0.0008)

References and research paper:

➢ ToxiM: A Toxicity Prediction Tool for Small Molecules Developed Using Machine Learning and Chemoinformatics Approaches
➢ T3DB: Downloads
➢ Chemical Hazards Database (OpenFoodTox) | EFSA