**IST707 Applied Machine Learning**

**GROUP 1 - CHECKPOINT 3**

**MACHINE LEARNING**

**FOR**

**DRUG DISCOVERY IN DOPAMINE RECEPTORS**

**Gughapriyaa Elango**

**Rohith Sai Kanchi**

**Sravan Kumar Mangalagiri**

**Peiyuan Feng**

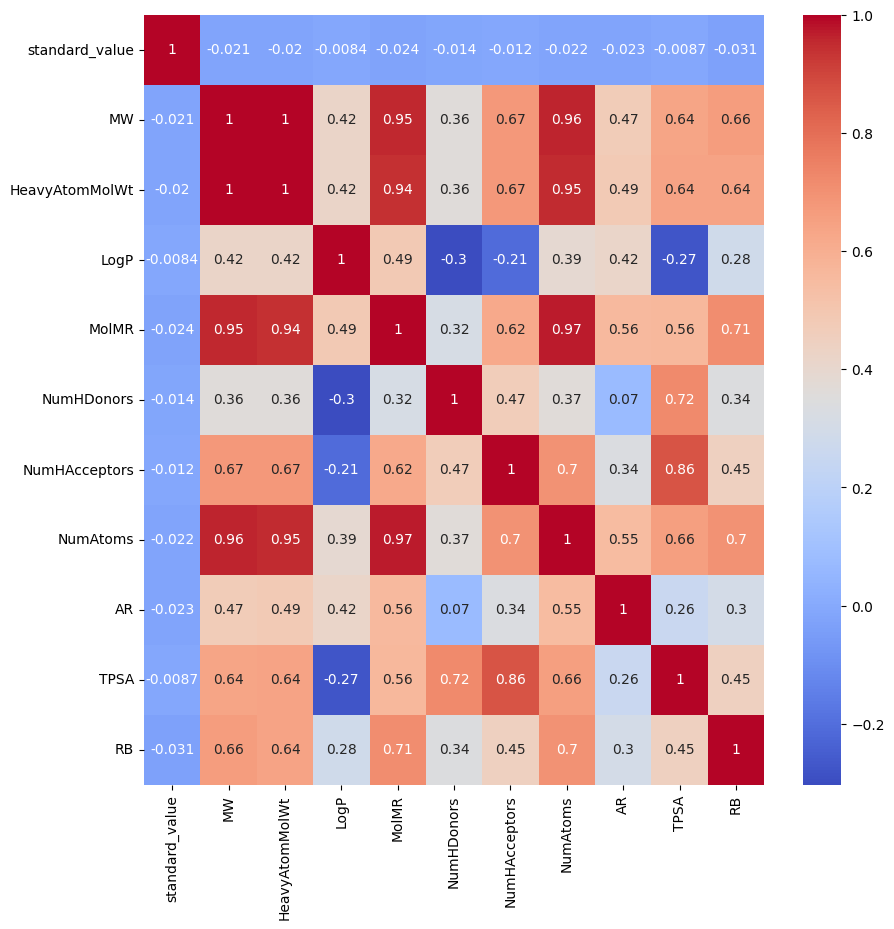
**Date: 23rd April 2023**

“I certify that this assignment represents my work. I have not used any unauthorized or unacknowledged assistance or sources in completing it, including free or commercial systems or services offered on the internet.”

**DATA PREPROCESSING:**

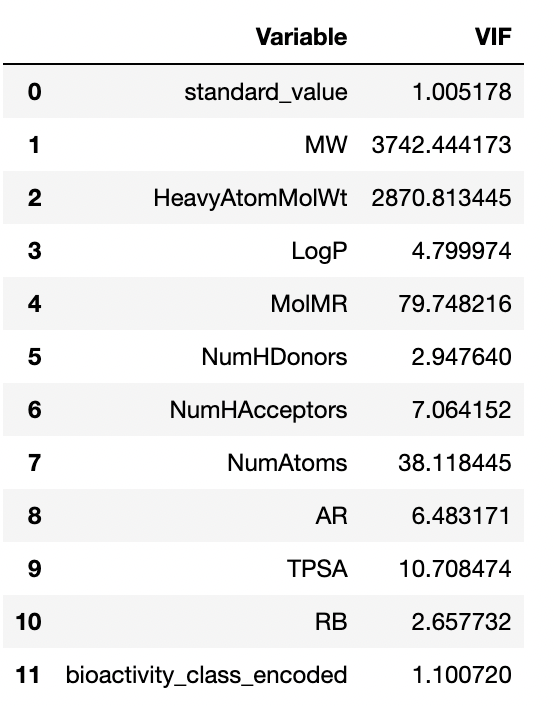
**REMOVING CORRELATED FEATURES:**

**Before:**

****

**Checking the VIF - Variance Inflation Factors:**

The Variance Inflation Factor (VIF) quantifies the degree to which independent variables in a regression model are correlated with one another. With VIF we can identify multicollinearity and assess the degree of association between the independent variables. Having highly correlated features will result in less significant results of predictions.

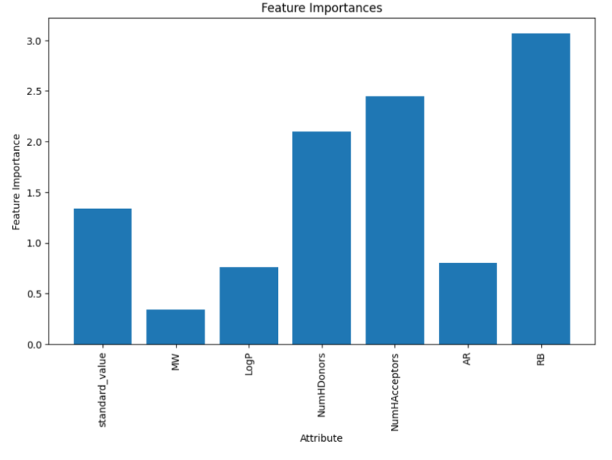
****

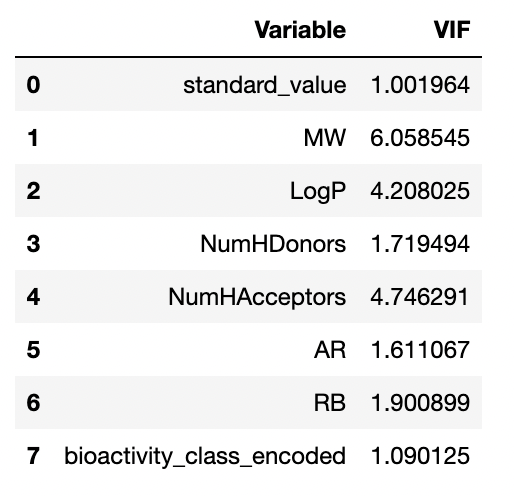
Here we can see very high values of VIF especially in MW, HeavyAtomMolWt, MolMR, NumAtoms, TPSA.

**After:**

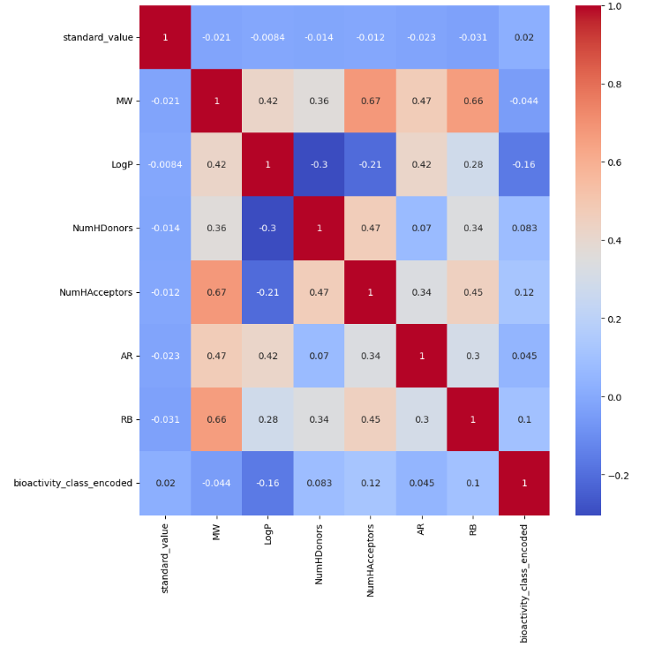
By combining the correlated features together, and eliminating one variable, correlation between the features have drastically come down. After removing multicollinearity:

Finding feature importance using SelectKbest:





When comparing the current data to the historical data, we see that all of the VIF values, as well as the number of variables and their correlation, have decreased. The high correlations that were visible earlier have been adjusted, as can be seen in the heatmap below.



The heatmap shows the correlation between the different features in the dataset. The darker the color, the stronger the correlation.

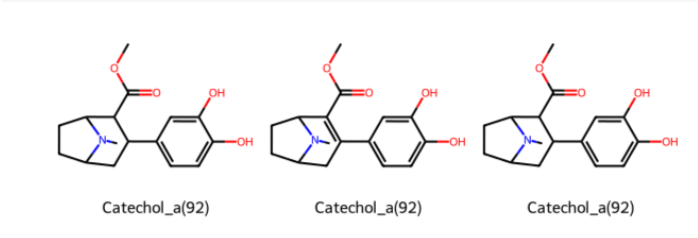
The most strongly correlated features are:

**NumHDonors and NumHAcceptors:** The number of hydrogen bond donors and acceptors are strongly correlated, as expected. This is because hydrogen bonds are typically formed between a hydrogen bond donor and a hydrogen bond acceptor.

**LogP and MW:** The octanol-water partition coefficient (LogP) and molecular weight (MW) are also strongly correlated. This is because LogP is a measure of how lipophilic a molecule is, and MW is a measure of how large a molecule is. Lipophilic molecules tend to have a higher MW, and vice versa.

The heatmap also shows some interesting correlations between the bioactivity class and other features. For example, active molecules tend to have a higher LogP and a lower RB. This suggests that active molecules are more lipophilic and have fewer rotatable bonds.

**REMOVING PAINS MOLECULES:**



PAINS (Pan-Assay Interference Compounds) are a set of substructures that are commonly found in molecules that interfere with biological assays, but not through specific interaction with the assay target. They often cause **false positives in prediction** and can lead to erroneous results in drug discovery.

Removing PAINS molecules from a dataset can improve the quality of the dataset by reducing the number of false positives and improving the reliability of the assay results. Hence, using FilterCatalogParams library, we removed about 125 PAINS molecules. Few examples of the PAINS molecules are visualised above.

**PREDICTION MODELS:**

We started building different models for classification. The evaluation metrics used were accuracy, F1 score, AUC score, recall score, Precision score, Confusion matrix and speed of training are all taken into consideration.

After preprocessing, we split the data into train and test sets at 80-20 ratio.

**Decision tree model:**

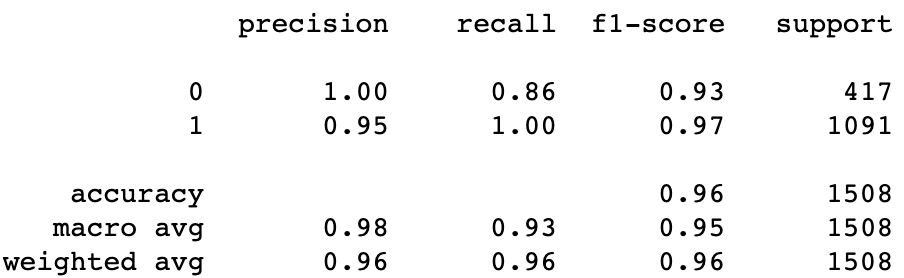
Running the decision tree classifier model with max depth of 2, we get a test score of 0.96.

Confusion matrix:

[[ 360 57]

[ 0 1091]]

The f1 score for the model is 0.96.



**Trying different iterations with decision trees:**

min\_samples\_leaf=4,max\_depth=4:

Accuracy: 0.9998341350140986

min\_samples\_leaf=4:

Accuracy: 0.9998341350140986

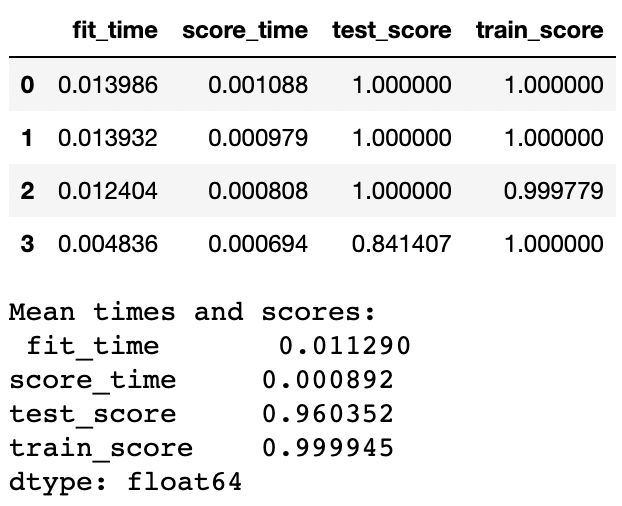
min\_samples\_leaf=1,max\_depth=1:

Accuracy: 0.9998341350140986

Using cross-validation with cv = 4:

Cross-validation scores: [1. 1. 1. 0.84140677]

Average cross-validation score: 0.96



Predicting after cross-validation:

Accuracy: 0.9998341350140986

Predicting after stratified cross-validation:

Average cross-validation score: 0.95

Accuracy: 0.9560457787361089

Fine-tuning hyperparameters using gridsearch cv:

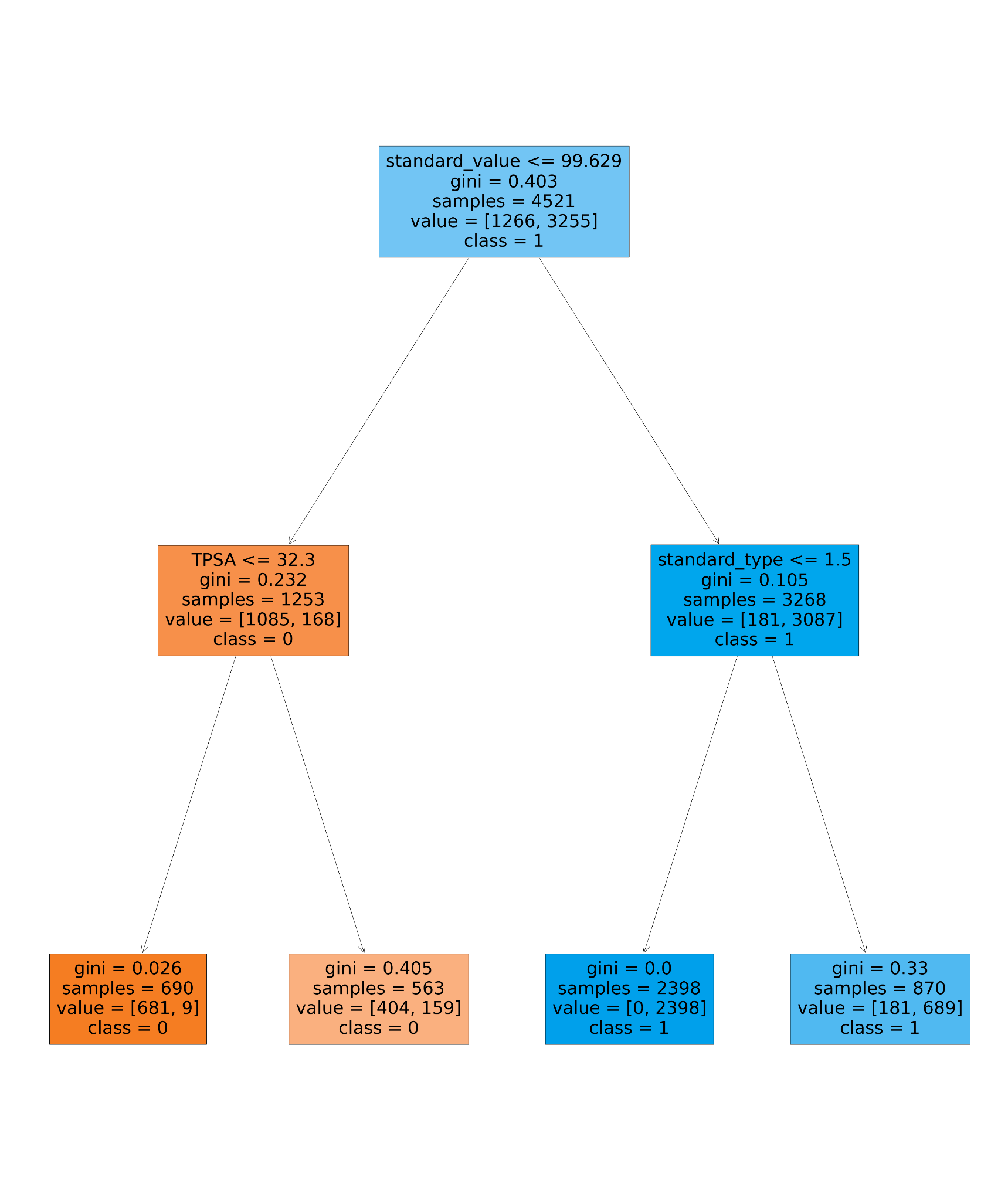
Fitting 5 folds for each of 40 candidates, totalling 200 fits

Accuary: 0.9212558059942306

F1 score: 0.9680111265646731

AUC score: 0.9852829010851814

**Best parameters: {'max\_depth': 2, 'min\_samples\_leaf': 0.1}**



We see that the decision tree model is overfitting to the training data.

**Naive Bayes model:**

Trying *multinomial* Naive Bayes, tuning hyperparameters:

Parameter grid:

{'alpha': [0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]}

Accuary: 0.4328726837138806

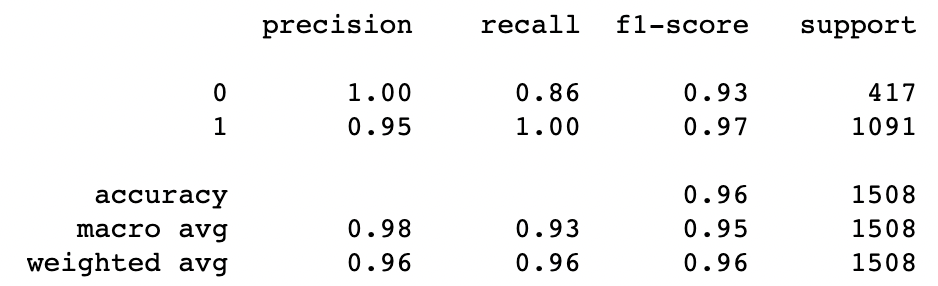
Metrics after hyperparameter tuning:

Accuracy: 0.4270557029177719

Best parameters: {'alpha': 0.001}

F1 score: 0.3501393463389917

AUC score: 0.6063244729605866



**K Nearest Neighbours classification:**

Fitting the model to find the best hyperparameters:

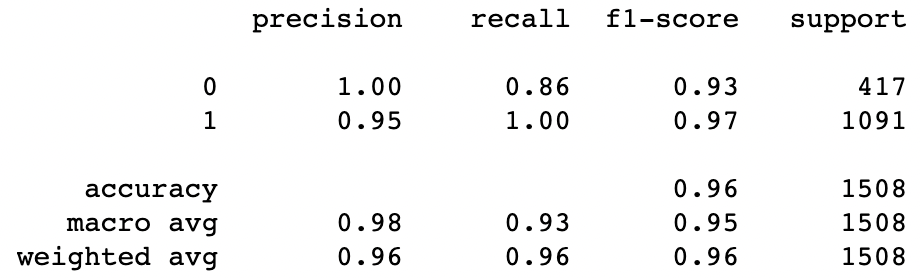
Best parameters: {'metric': 'manhattan', 'n\_neighbors': 5}

Best cross-validation score: 0.96

accuracy: 0.9542440318302388

F1 score: 0.9680111265646731

AUC score: 0.9852829010851814



**Support Vector Machine:**

Trying preprocessing techniques with SVM, minmaxscaler was tried to standardize the data. Predicting the test file after running the SVM model with cross validation:

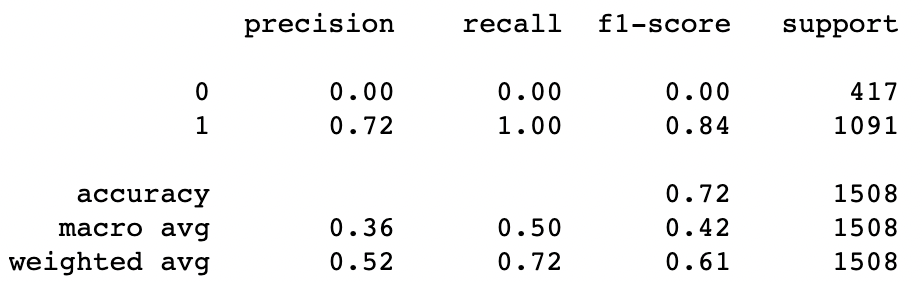
Accuracy: 0.7234748010610079

Average cross-validation score: 0.72

accuracy: 0.7234748010610079

F1 score: 0.8395536744901885

AUC score: 0.5



**Random Forest:**

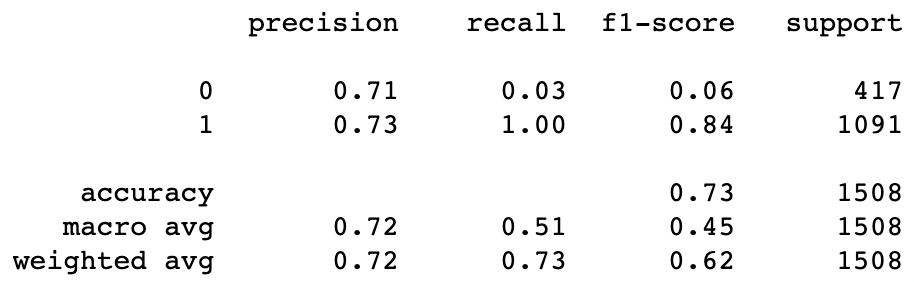
Accuracy: 0.7281167108753316

Average cross-validation score: 0.96

accuracy: 0.7281167108753316

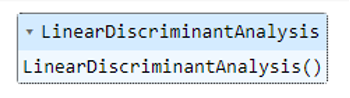
F1 score: 0.8412083656080558

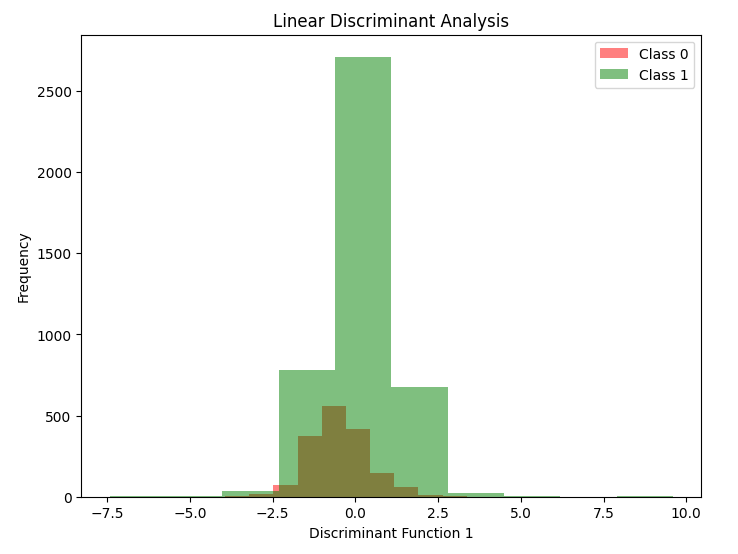
AUC score: 0.5120970134982756



**Linear Discriminant Analysis model:**

Trying *LinearDiscriminantAnalysi,* tuning hyperparameters:

****

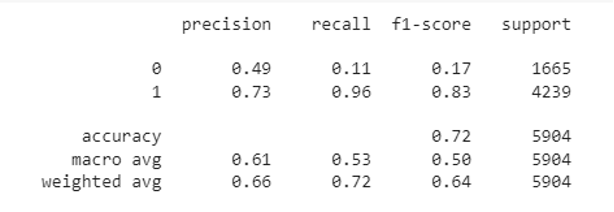
****

**Metrics after hyperparameter tuning:**

The total accuracy is 0.72 and the total f1 score is 0.83.

Precision for category 0 is 0.49 and for category 1 is 0.73.

Recall for category 0 is 0.11 and for category 1 is 0.96.

****

The LDA model has not performed well with our data, it hasn't been able to split the categories properly.

**Bagging model:**

Trying *BaggingClassifier*, tuning hyperparameters:

Parameter grid:

{

'n\_estimators': [5, 10, 15, 20],

'max\_samples': [0.5, 0.7, 0.8, 1.0],

'max\_features': [0.5, 0.7, 0.8, 1.0]

}

Best Hyperparameters:

{'max\_features': 1.0, 'max\_samples': 0.7, 'n\_estimators': 10}

**Metrics after hyperparameter tuning:**

The total accuracy is 0.97 and the total f1 score is 0.97.

Precision for category 0 is 0.92 and for category 1 is 0.99.

Recall for category 0 is 0.98 and for category 1 is 0.96.

Bagging model has performed very well, and we can see that with hyper parameter tuning, metrics have increased well.

**XGBoost model:**

Trying *XGBClassifier*, tuning hyperparameters:

Parameter grid:

{

'max\_depth': [3, 5, 7],

'learning\_rate': [0.1, 0.01, 0.001],

'n\_estimators': [100],

'gamma': [0, 0.1, 0.2],

}

Best Hyperparameters:

{'gamma': 0.1, 'learning\_rate': 0.1, 'max\_depth': 7, 'n\_estimators': 100}

**Metrics after hyperparameter tuning:**

The total accuracy is 0.97 and the total f1 score is 0.97.

Precision for category 0 is 0.93 and for category 1 is 0.99.

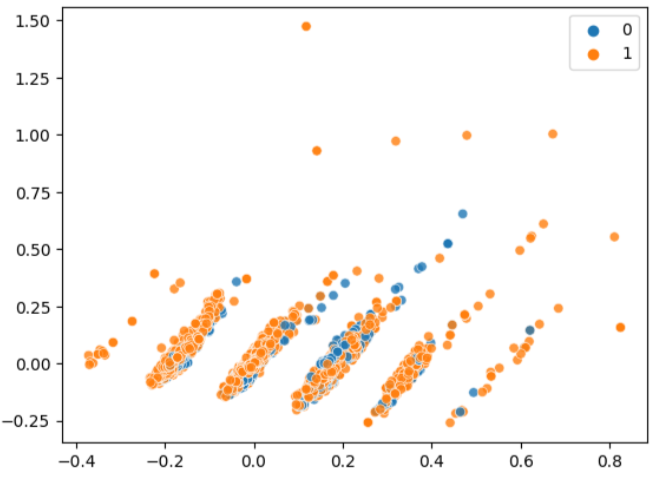
Recall for category 0 is 0.97 and for category 1 is 0.97

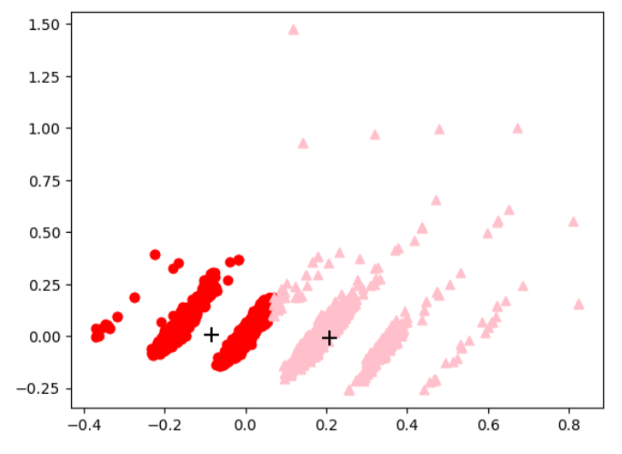
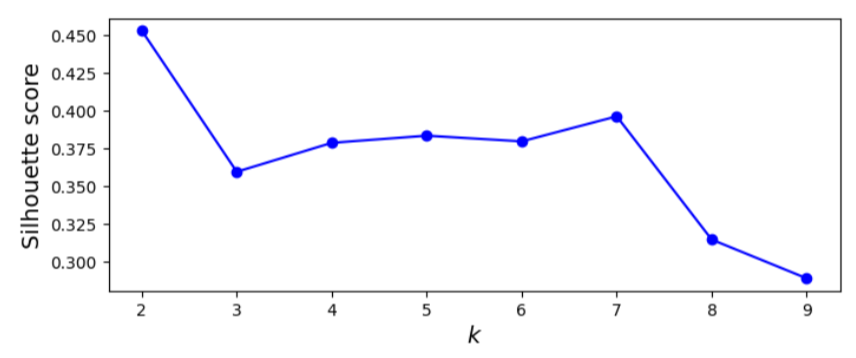
Boosting model has performed very well, and we can see that with hyper parameter tuning, metrics have increased well.

**CLUSTERING ALGORITHMS:**

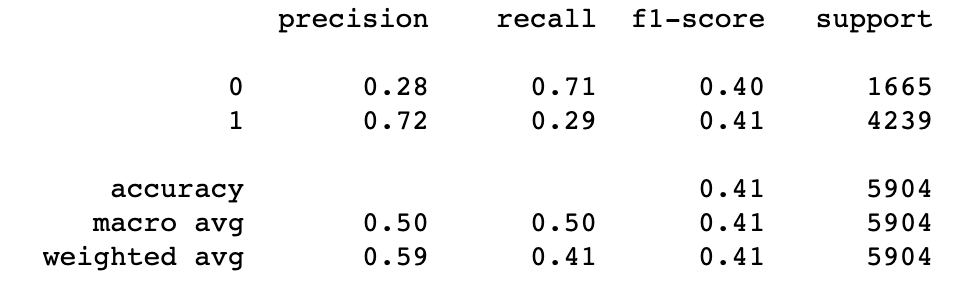
Here we dropped the target variable of bio-activity class to try clustering-based analysis.

**K Means Clustering:**

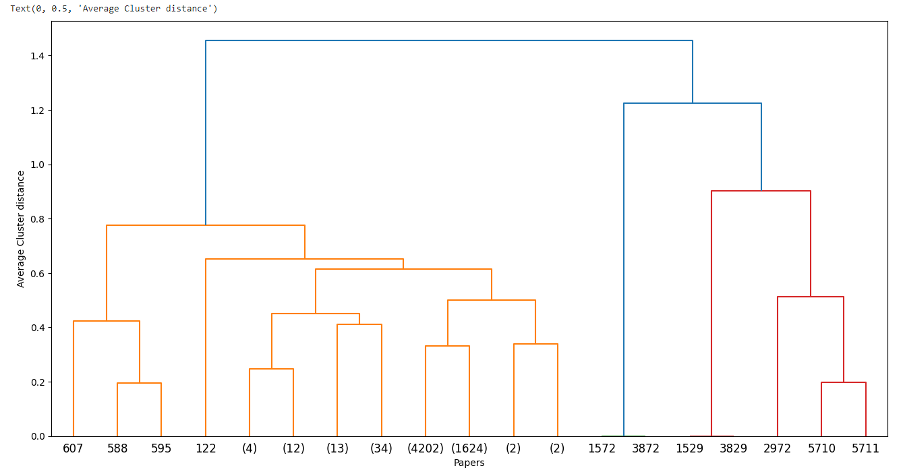




Even after choosing the best k, the data points do not cluster so well and the centroids are not located accurately. Clustering does not give us good insights here.



**Hierarchical Agglomerative clustering:**

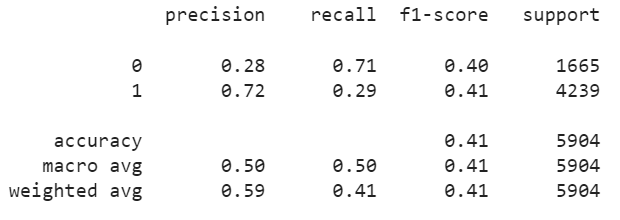


**Metrics:**

The total accuracy is 0.41 and the total f1 score is 0.41.

Precision for category 0 is 0.28 and for category 1 is 0.72.

Recall for category 0 is 0.71 and for category 1 is 0.29.



In both the clustering models, the models do not give powerful insights. Similarly, with LDA, we can see that the data does not cluster well with the attributes present.

**COMPARISON OF METRICS OF ALL THE MODELS:**

Comparing the 5 models created above, let’s compare their evaluation metrics:

|  | Parameters tuned | Accuracy | F1 | Precision | Recall |
| --- | --- | --- | --- | --- | --- |
| LDA | k=2, alpha=auto,beta=auto | 0.660 | 0.720 | 0.610 | 0.535 |
| Decision tree | Max\_depth = 3,min\_samples\_leaf = 0.01 | 1.0 | 1.0 | 1.0 | 1.0 |
| Naive bayes | alpha=0.001 | 0.42 | 0.35 | 0.60 | 0.21 |
| kNN | Metric = manhattan,n\_neighbors=5 | 0.95 | 0.96 | 0.95 | 0.95 |
| SVM |  |  |  |  |  |
| KMeans | n=2,max\_iter=100 | 0.590 | 0.405 | 0.500 | 0.500 |
| HAC | n=2,linkage=’average’ | 0.590 | 0.405 | 0.500 | 0.500 |
| Random Forest |  |  |  |  |  |
| Bagging | 'max\_features': 1.0, 'max\_samples': 0.7, 'n\_estimators': 10 | 0.970 | 0.970 | 0.955 | 0.970 |
| XGBoost | 'gamma': 0.1, 'learning\_rate': 0.1, 'max\_depth': 7, 'n\_estimators': 100 | 0.970 | 0.970 | 0.960 | 0.970 |

In drug discovery, the consequences of false positives (i.e., identifying non-drug candidates as drugs) and false negatives (i.e., missing true drug candidates) can have different implications.

Precision is the proportion of true positives out of the total predicted positives. It indicates the model's ability to correctly identify true drug candidates among the predicted positive samples. High precision means that the model is less likely to produce false positives, which can reduce the risk of wasting resources on non-promising drug candidates, and testing the unnecessary drugs on humans.

On the other hand, recall is the proportion of true positives out of the total actual positives. It is a measure of the model's ability to correctly identify all the positive samples, including true drug candidates. High recall means that the model is less likely to miss true drug candidates which can be important in ensuring that potential drug candidates are not overlooked.

**NEURAL NETWORKS TO PREDICT BIOACTIVITY CLASS:**

Neural Networks are potent machine learning models that can effectively model the intricate relationships between the input variables and the objective variable. In the case of predicting the Bioactivity class, a large number of variables are likely to contribute to the classification, making it difficult to identify the most significant variables and their interactions. Neural Networks are capable of autonomously identifying and capturing these nonlinear relationships.

In bioactivity prediction, where there are thousands of compounds and numerous molecular descriptors for each compound, Neural Networks have demonstrated their ability to effectively manage large volumes of data. Neural Networks are a viable option for predicting the Bioactivity class due to their capacity to manage large quantities of data and recognize complex relationships.

**TYPES OF MODELS TRIED:**

**ANN** - Artificial Neural Network

**CNN** - Convolutional Neural Network

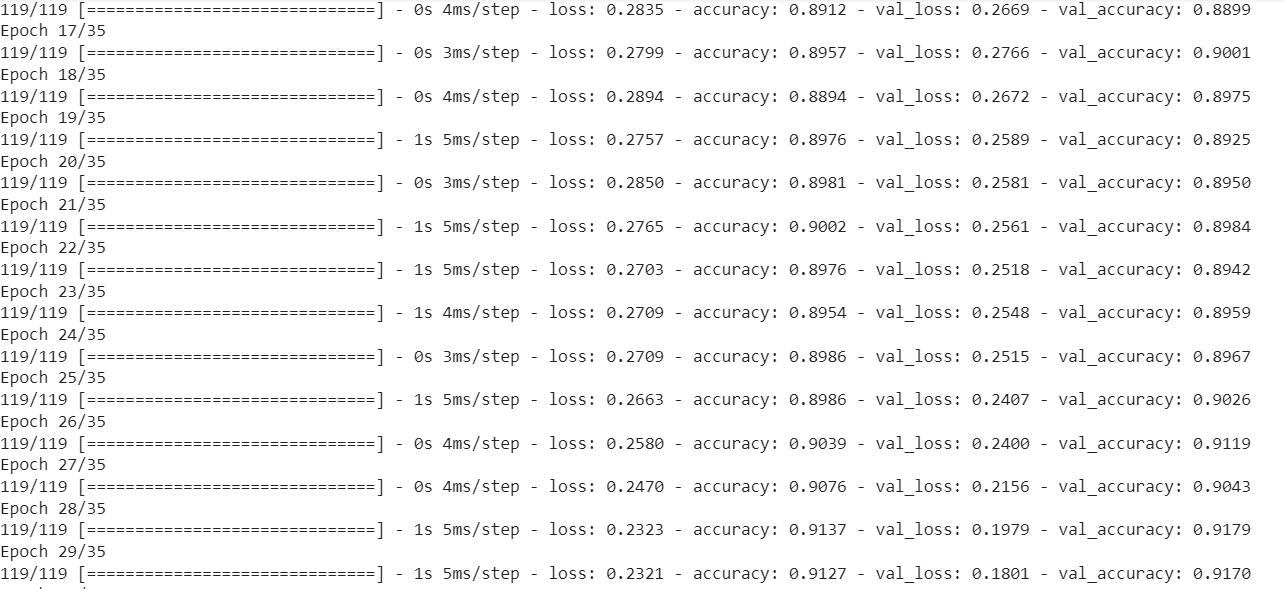
**RNN** - Recurrent Neural Network

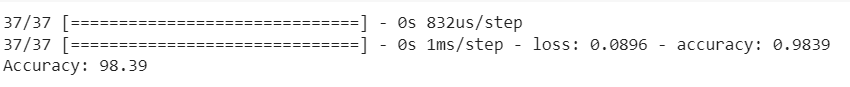
**ANN**

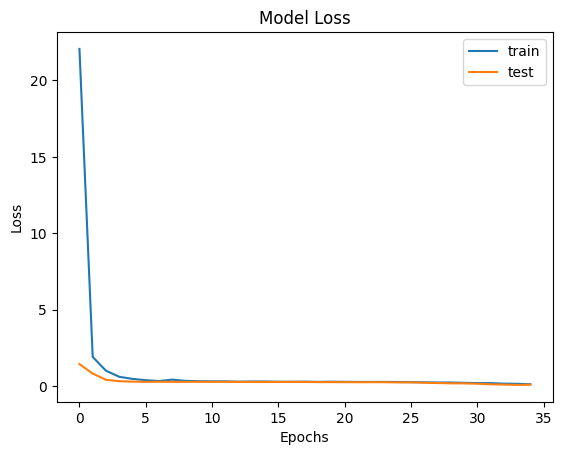
Artificial Neural Network (ANN). Each node conducts a mathematical operation on its inputs and passes the output to the next tier.

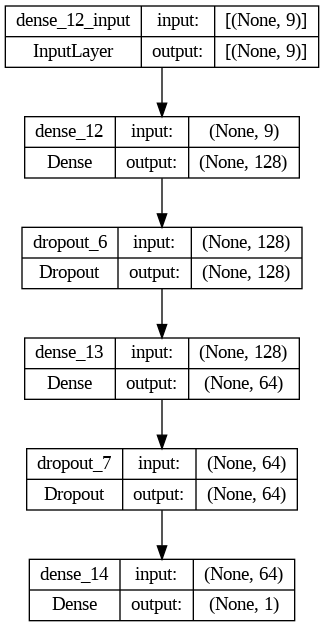
Based on Lipinski descriptors, an ANN may estimate the bioactivity class of a molecule using the available data. The ANN can predict compound classes in the test set by training on a subset of data with known bioactivity classes. The ANN recognizes patterns in input data and generates a probability distribution for each class.

epochs=35









ANNs are fully connected and make no data structure assumptions, unlike CNNs and RNNs.

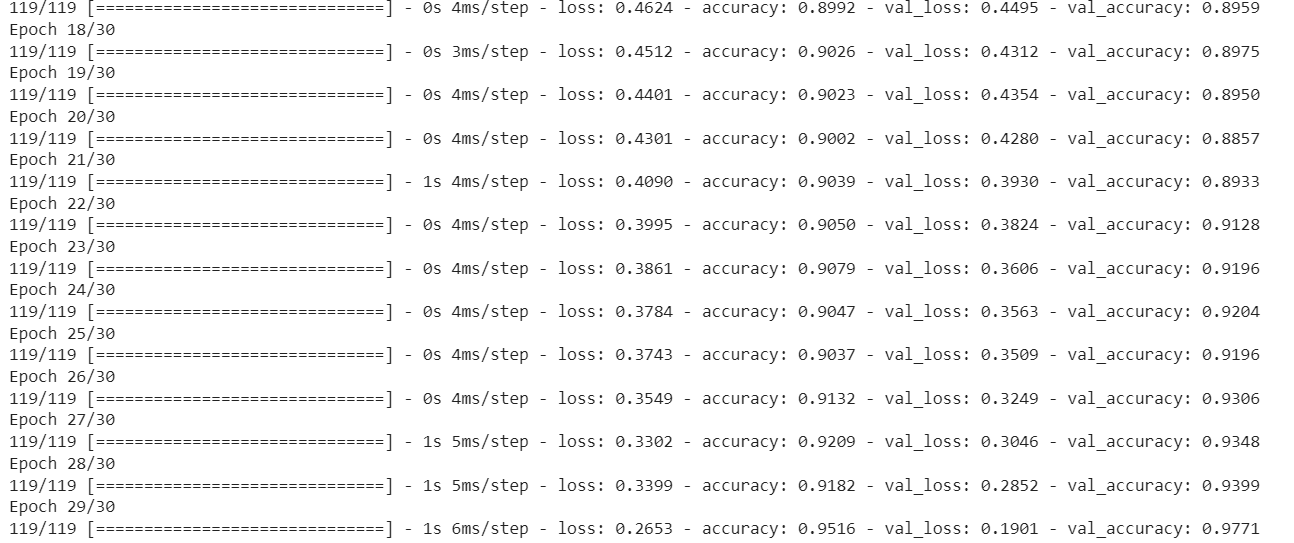
This ANN architecture uses a rectified linear unit (ReLU) activation function and consists of a 9-node input layer followed by two 128-node hidden layers. For binary classification, the output layer just contains a single node with a sigmoid activation function.

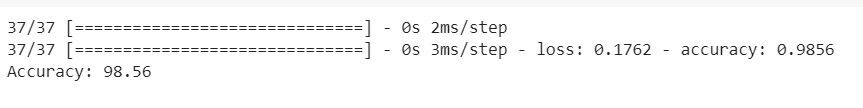
The loss function in this model is binary cross-entropy, and the batch size is 32. During backpropagation, the weights are updated using the Adam optimizer and a learning rate of 0.001.

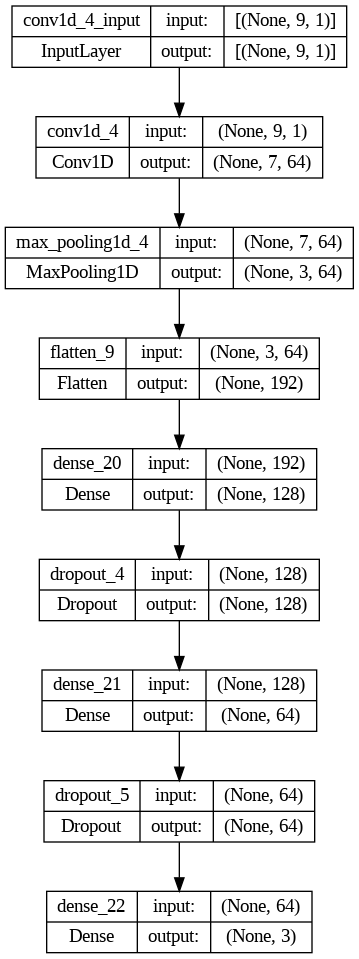
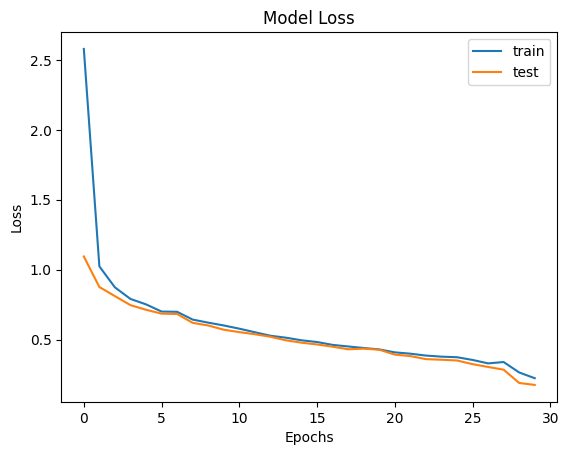
In the ANN architecture, dropout is implemented after the first and second hidden layers with a dropout rate of 0.25. Early stopping is also implemented using a validation set, where training stops when the validation loss does not decrease for five epochs. These techniques help prevent overfitting and improve the generalization performance of the network.

**CNN**

Deep learning algorithm CNN processes images and signals. Convolutional and pooling layers recognize patterns and features in pictures and signals. The convolutional layers filter each input image to create a feature map that highlights particular features. Pooling layers then minimize feature map size by taking the maximum or average value of a group of pixels. This reduces noise and improves calculation.



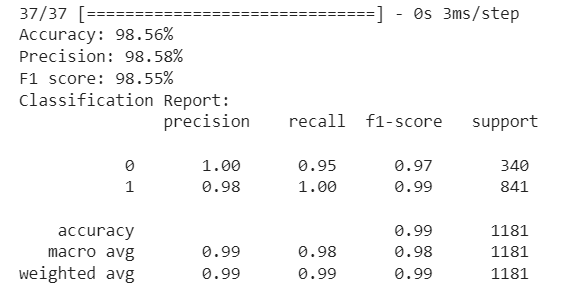




Padding, stride, and many layers with distinct activation functions are all features of the CNN design. The activation function of ReLU is used in the input layer, whereas the activation function of Softmax is used in the output layer.

The stride and filter sizes vary across the convolutional layers. The max pooling layers shrink the feature maps to a more manageable size. Classification is performed on the dense layers with dropout regularization to prevent overfitting.

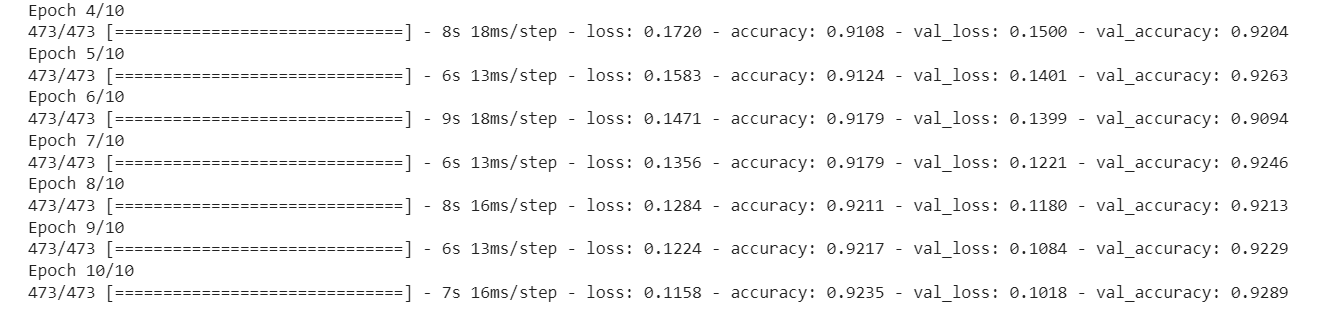
The model employs the Adam optimizer and the loss function of categorical cross entropy.

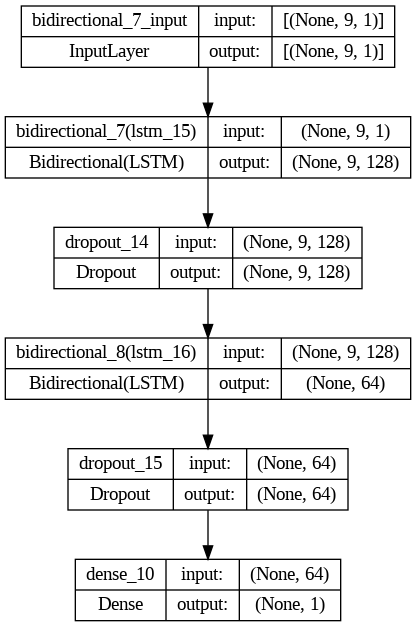


**RNN**

Recurrent neural networks (RNNs) can process variable-length sequential data and output related outputs. RNNs process sequential data like time-series, natural language processing, speech recognition, and music composition.

RNN can assess the dataset's consecutive molecules. RNN can forecast a molecule's bioactivity class based on its chemical structure and its predecessors' bioactivity levels. Drug discovery and lead optimization employ this to find new molecules with high bioactivity based on known chemicals. Sequence-to-sequence mapping uses RNN to create a series of structurally similar molecules.



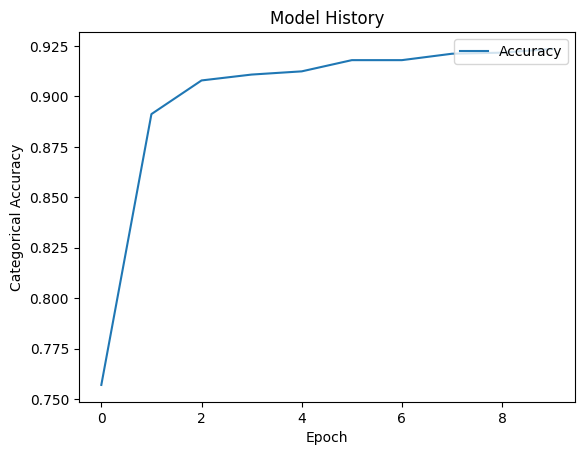


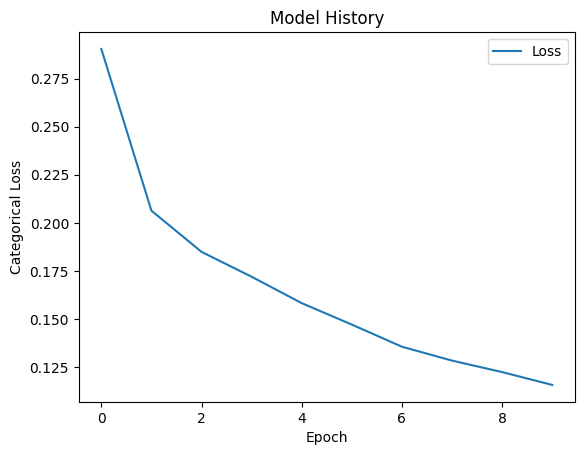
· An embedding layer receives the input sequence and uses it to learn a very detailed representation of the input language.

· A 64-unit, single-layer RNN with linear activation function is fed the embedded sequence. The RNN layer generates a stream of unobservable states.

· The output of the RNN is fully connected and uses the ReLU activation function on its 32 units.

· In the end, a regularization L1 L2 layers applied to a single output unit coupled to the fully connected layer to get a binary classification.





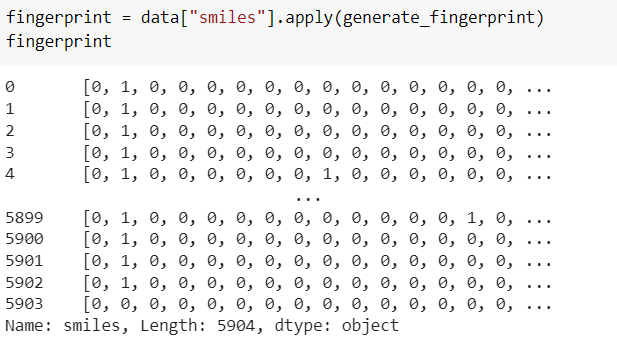
The depicted RNN architecture can be enhanced by incorporating techniques such as Bidirectional LSTM to capture past and future context, Dropout to prevent overfitting, and Early stopping to monitor validation loss and cease training when validation loss stops improving. Adding a Bidirectional LSTM layer before or after the existing LSTM layers, Dropout layers after each LSTM layer, and the EarlyStopping callback in Keras can help the RNN architecture to better understand the sequence, prevent overfitting, and improve generalization performance by stopping the training process when the validation loss stops improving.

**NEURAL NETWORKS TO PREDICT CHEMICAL FINGERPRINTS:**

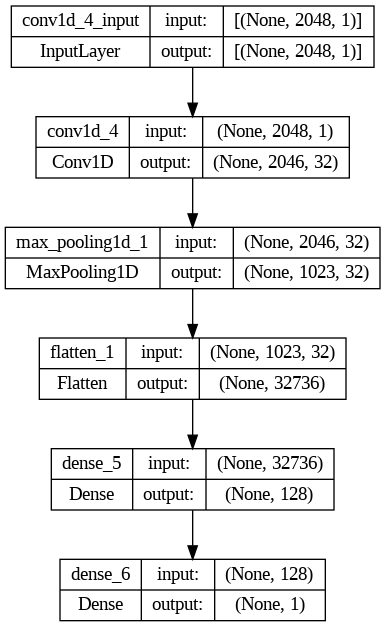
**USING MORGAN FINGERPRINT AND CNN:**

Morgan fingerprints are a type of molecular fingerprint that encode the topology of a molecule using circular fingerprints. They are commonly used in cheminformatics and machine learning applications to represent molecules and compare their similarity. Morgan fingerprint is a binary vector that encodes the presence or absence of each circular substructure in the molecule, and its length depends on the chosen fingerprint radius and bit length.

FINAL SPARSE DATA USING 2048 MORGAN FINGERPRINT MOLECULE:



A sequential convolutional neural network is built with the following layers:



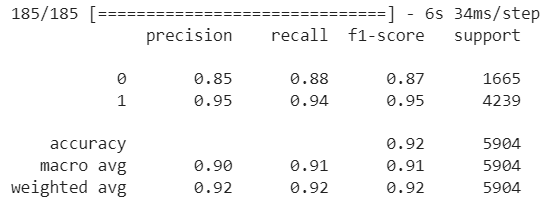
Trained for:

Epochs: 50

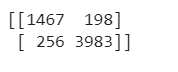
185/185 [==============================] - 4s 22ms/step - loss: 0.2155 - accuracy: 0.9231

Test accuracy: 0.9231029748916626

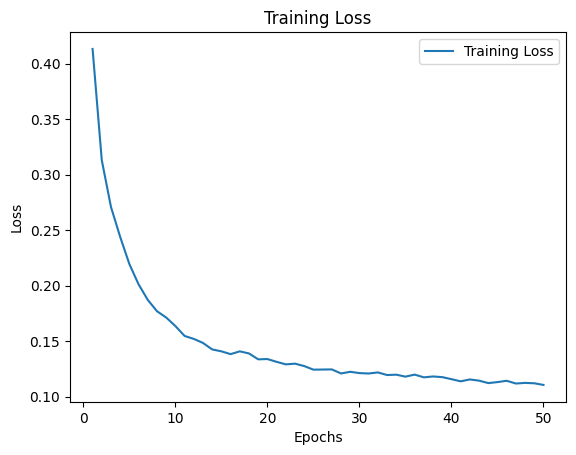
Metrics:



Confusion matrix:



Plotting the training loss:



**USING ATOM TOKENIZATION AND RNN:**

CNN can process SMILES-stringed molecular structures using the input data. SMILES strings can be translated into 2D graphics with pixels for atoms and lines for bonds. A CNN model can learn bioactivity-related patterns and characteristics from these photos. The model may classify molecules as active or inactive.

**ATOM TOKENIZATION:**

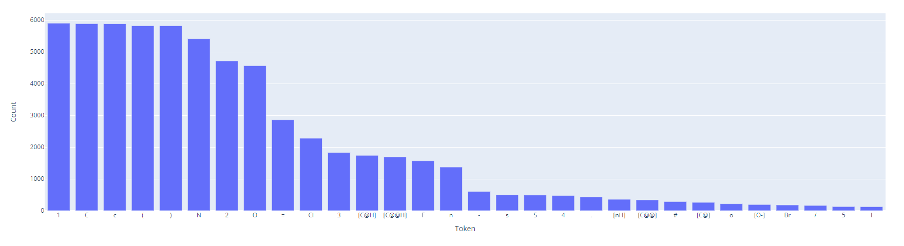
From the chemical canonical SMILES, we are tokenizing them into respective atoms using SmilesPE package.

SmilesPE is a tokenizer designed specifically for chemical structures encoded in the Simplified Molecular Input Line Entry System (SMILES) notation. It is a byte pair encoding (BPE) tokenizer that uses a vocabulary of SMILES-specific tokens to split SMILES strings into subtokens.

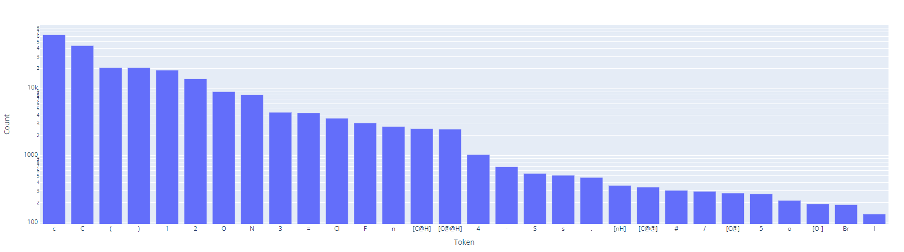
When SmilesPE.tokenizer processes a SMILES string, it first converts the string into a sequence of characters. It then iteratively replaces pairs of characters with new tokens from the vocabulary, based on the frequency of the pairs in the input data. This process continues until the maximum vocabulary size is reached or no more frequent pairs are found.

In addition to splitting the SMILES string into atoms, SmilesPE.tokenizer also adds special start-of-sequence and end-of-sequence tokens to the beginning and end of the tokenized sequence, respectively. This allows the tokenizer to encode the SMILES string as a sequence of tokens that can be processed by a machine learning model.

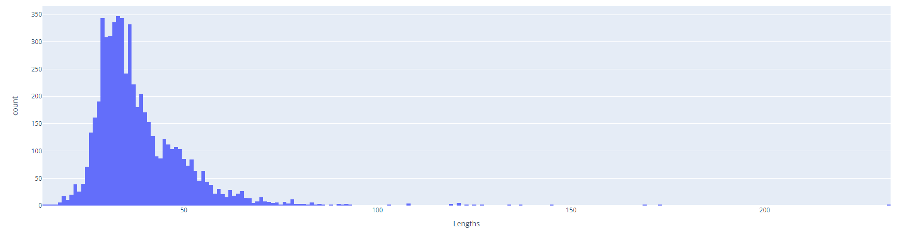
**Token frequency of tokens that appears once:**

****

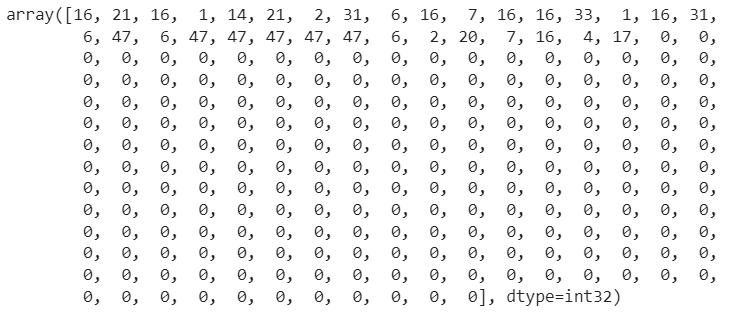
**Token frequency of all tokens:**

****

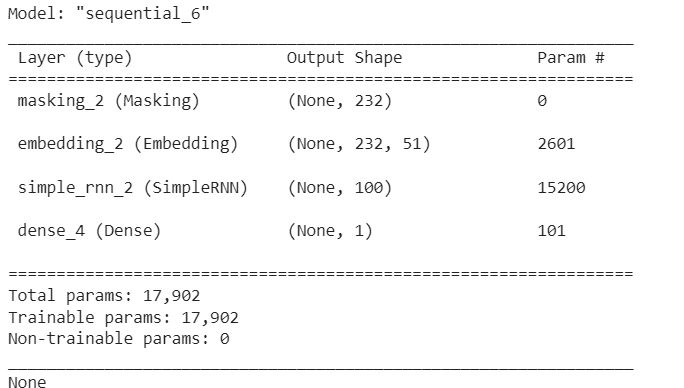
**Histogram of length of the chemical fingerprint sequence:**

****

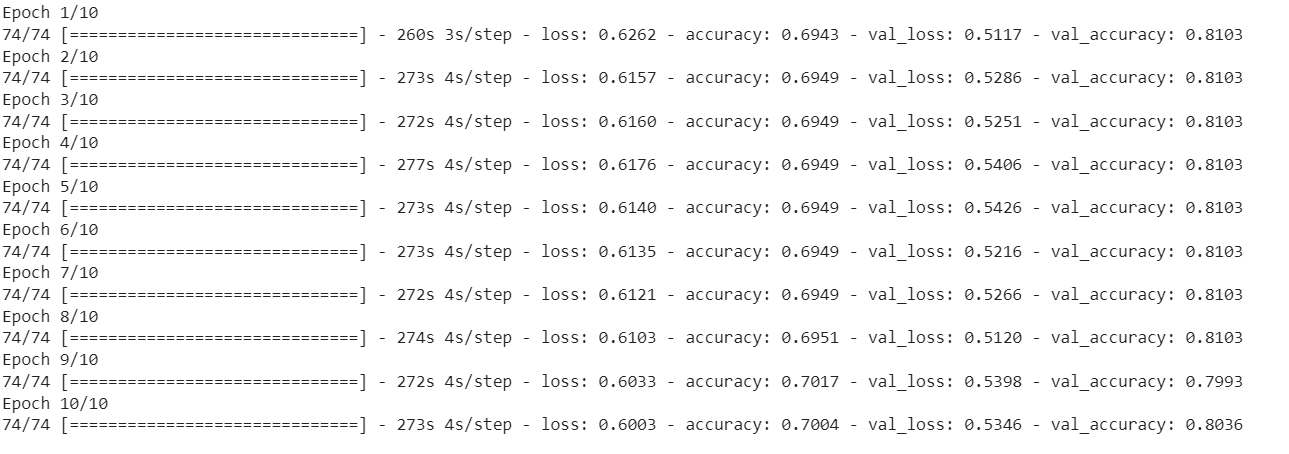
**Example final padded sequence of a compound:**

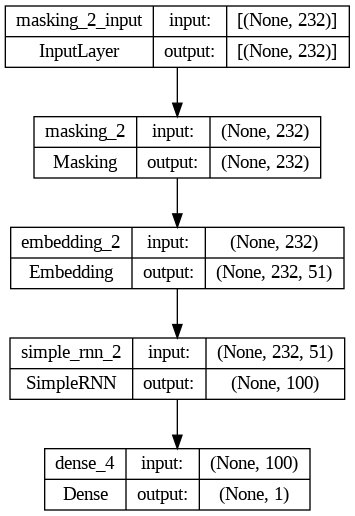
****

**Simple RNN as baseline:**

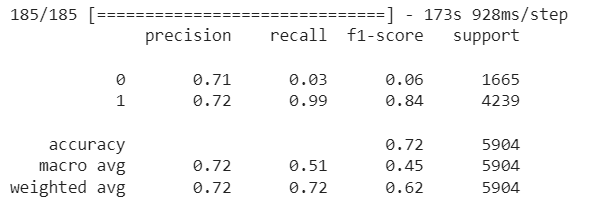


Epochs: 10

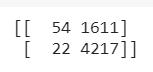




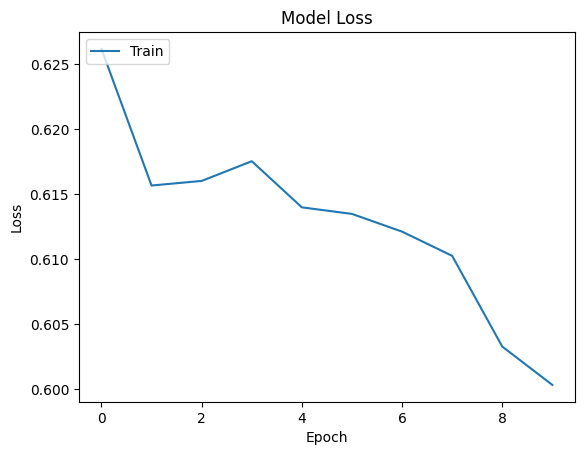
**Metrics:**



Confusion matrix:



Plotting training loss:



**Conclusion**

Using machine learning and neural network models, were were able to predict the bioactivity classes of chemical compounds and their chemical structures useful in inhibiting dopamine receptors. Among machine learning models, ensemble models worked well in predicting, and in neural networks, RNN models performed the best. In predicting chemical fingerprints, RNNs with Bi-LSTM achieved about 72% accuracy.

**References**

<https://github.com/Molecular-Exploration/toxicity-classification/blob/main/Colab_Notebooks/SMILES_ensemble.ipynb> - referred for molecular exploration and tokenization