# Dataset 1 - Mice Protein Expression

**Dataset Link** - https://www.kaggle.com/datasets/ruslankl/mice-protein-expression

## Introduction to dataset

The data set consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction of cortex. There are 38 control mice and 34 trisomic mice (Down syndrome), for a total of 72 mice. In the experiments, 15 measurements were registered of each protein per sample/mouse. Therefore, for control mice, there are 38x15, or 570 measurements, and for trisomic mice, there are 34x15, or 510 measurements. The dataset contains a total of 1080 measurements per protein. Each measurement can be considered as an independent sample/mouse.

The first target label in the dataset is **Genotype**, which is divided into two categories: Control and Ts65Dn. Other labels that could be used for binary classification include **Treatment**, which has two categories: saline and memantine, as well as **Behaviour**, in which some mice were stimulated to learn (C/S) while others were not (S/Ct). Furthermore, considering genotype, treatment, and behaviour, mice are classified into eight groups, which are as follows: -

- c-CS-s: control mice, stimulated to learn, injected with saline (9 mice)
- c-CS-m: control mice, stimulated to learn, injected with memantine (10 mice)
- c-SC-s: control mice, not stimulated to learn, injected with saline (9 mice)
- c-SC-m: control mice, not stimulated to learn, injected with memantine (10 mice)
- t-CS-s: trisomy mice, stimulated to learn, injected with saline (7 mice)
- t-CS-m: trisomy mice, stimulated to learn, injected with memantine (9 mice)
- t-SC-s: trisomy mice, not stimulated to learn, injected with saline (9 mice)
- t-SC-m: trisomy mice, not stimulated to learn, injected with memantine (9 mice)

## Goal

With the use of this dataset, we plan to accurately categorise the mice depending on their genotype while reducing the amount of proteins to a minimum.

# Dataset 1 - Method Overview

**Preprocessing**

- Load the dataset.
- Check for null values.
    - In columns
    - In rows
- Remove null values.
    - In column, that have 100% Null Values
    - In rows, that have >50% Null Values
- Impute the null values using mean of observations in that column.
- Transform the categorical features (For ex. Genotype [‘Control’, ‘Ts65Dn’] to [0, 1]).
- Drop other target classes.

**Feature Selection**

The input features in our dataset are numeric, and our output variable is categorical. In this instance, specific feature selection algorithms get the best results, such as: -

- Kendall's Correlation Coefficient is a statistic that is used to determine the ordinal relationship between two measured parameters.

- Anova is a technique for determining the significance of an independent characteristic on the target class. Based on this, one might choose the k-best features that have a large influence on the target class.

- Forward selection utilising random forest, an Embedded approach for selecting the best k features by computing how much each feature reduces impurity. The more a feature reduces impurity, the more essential the feature is (impurity is measured by either the Gini impurity or the information gain/entropy).

We first start with Kendall’s method to reduce all the features that are co-related to each other by more than 70%. Further, we have used Forward feature selection and Anova to select the best features with different values of k.

- **Results & Testing**

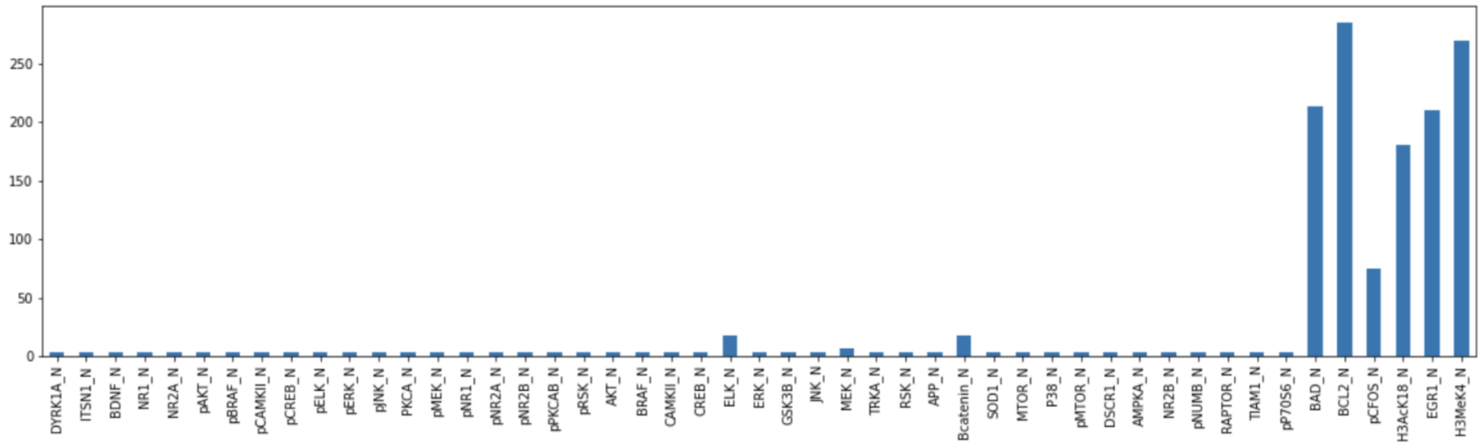
We alter our dataset based on the best features collected after each cycle of feature selection. We train our chosen machine learning models on this new dataset and store the training accuracy, 10 fold cross validation score, and AUC score in a table to evaluate the performance of different feature selection strategies.

Furthermore, once we've identified our top features, we run all machine learning models through hyperparameter tuning tests.

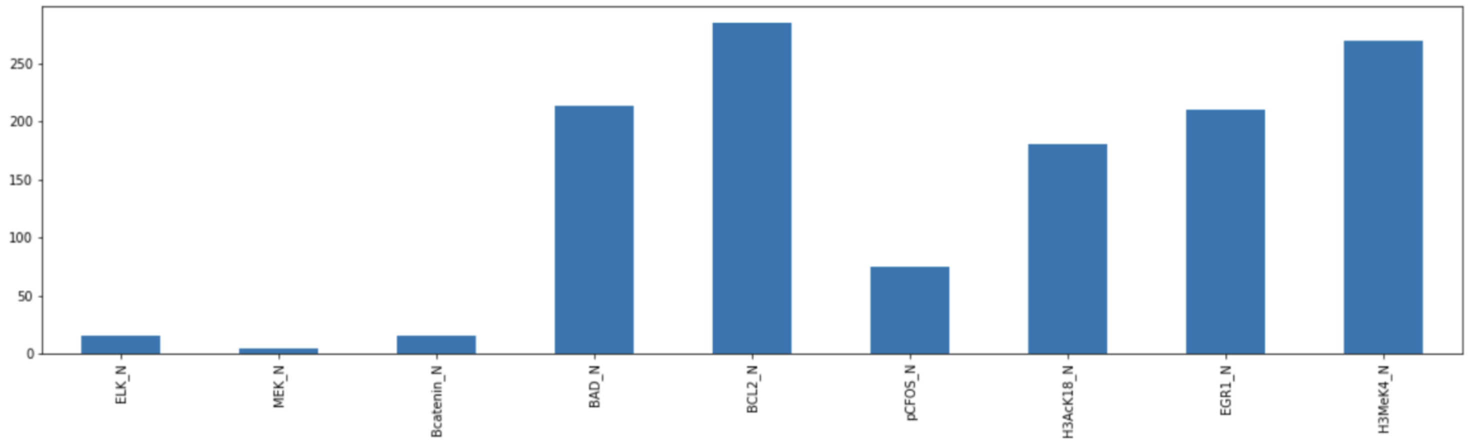
Finally, we utilise our fine-tuned machine learning models and the best features chosen to categorise the mice into eight distinct groups and provide the classification report.

## Dataset 1 - Preprocessing Results

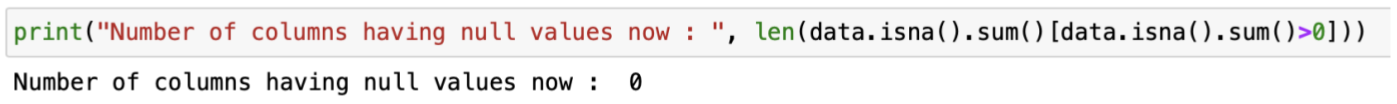
The initial dataset has 77 numerical (float values) and 4 categorical variables. The bar graph below depicts the distribution of null values across our numerical features.



We begin by removing rows with total null values greater than 50%. By doing so, we can observe that there are only 9 columns remaining with null values. The bar graph below demonstrates this.



We then impute these null values using the sklearn.imputer function SimpleImputer(strategy ='mean'). Following that, we get a dataset that has no null values.



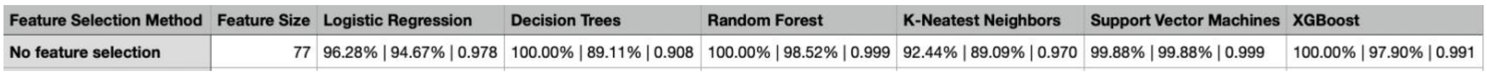
```
print("Number of columns having null values now : ", len(data.isna().sum()[data.isna().sum()>0]))
```

```
Number of columns having null values now :  0
```

## Dataset 1 - Feature Selection & Results

### Before Feature Selection

- We wanted to examine how the present dataset with all 77 features performed with our chosen machine learning models before doing feature selection.
- The following findings might be shown as follows: on the x axis are machine learning models, and on the y axis is a feature selection approach.
- A string comprising,
  < **Training Accuracy % | Cross Validation Score (10 Fold) % | AUC Score** >
  is represented by the grid representing the intersection of x and y.



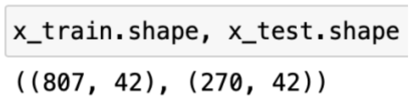
| Feature Selection Method | Feature Size | Logistic Regression | Decision Trees | Random Forest | K-Neatest Neighbors | Support Vector Machines | XGBoost |
|---|---|---|---|---|---|---|---|
| No feature selection | 77 | 96.28% \| 94.67% \| 0.978 | 100.00% \| 89.11% \| 0.908 | 100.00% \| 98.52% \| 0.999 | 92.44% \| 89.09% \| 0.970 | 99.88% \| 99.88% \| 0.999 | 100.00% \| 97.90% \| 0.991 |

### Kendall's Correlation

- We used Kendall's correlation to see how closely the numerical characteristics are related to one another. We set the threshold value to 70% and then look for attributes that are more than 70% associated with one another.
- On performing this we found out the following **35 features** that are highly related to each other: -
- {'JNK_N', 'ITSN1_N', 'TIAM1_N', 'GSK3B_N', 'pMEK_N', 'pELK_N', 'NR1_N', 'AMPKA_N', 'NR2A_N', 'AKT_N', 'P38_N', 'Tau_N', 'pBRAF_N', 'ELK_N', 'pS6_N', 'CAMKII_N', 'RAPTOR_N', 'MEK_N', 'PKCA_N', 'pPKCAB_N', 'ERK_N', 'RSK_N', 'CREB_N', 'pNUMB_N', 'NUMB_N', 'BRAF_N', 'TRKA_N', 'NR2B_N', 'Bcatenin_N', 'pJNK_N', 'pMTOR_N', 'DSCR1_N', 'pERK_N', 'pNR1_N', 'pNR2B_N'}
- We then go and remove these features from our dataset.

### Train-Test Split

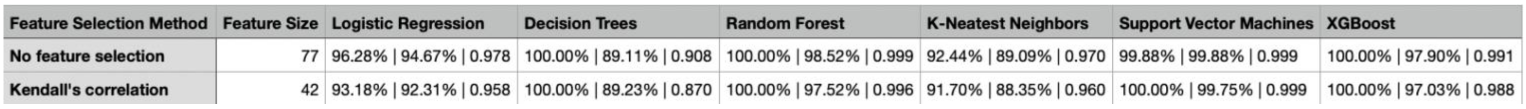
- After Kendall's correlation, we divide our dataset into train & test with a ratio of 3:1 (75% training examples and 25% testing data). The shape of the training features dataset (x_train) and testing features dataset (x_test) are shown below.



```
x_train.shape, x_test.shape
((807, 42), (270, 42))
```

### Results after Kendall's Correlation

- We discovered that deleting the correlated features had no significant effect on test and train accuracy or cross validation. The outcomes are depicted below: -



| Feature Selection Method | Feature Size | Logistic Regression | Decision Trees | Random Forest | K-Neatest Neighbors | Support Vector Machines | XGBoost |
|---|---|---|---|---|---|---|---|
| No feature selection | 77 | 96.28% \| 94.67% \| 0.978 | 100.00% \| 89.11% \| 0.908 | 100.00% \| 98.52% \| 0.999 | 92.44% \| 89.09% \| 0.970 | 99.88% \| 99.88% \| 0.999 | 100.00% \| 97.90% \| 0.991 |
| Kendall's correlation | 42 | 93.18% \| 92.31% \| 0.958 | 100.00% \| 89.23% \| 0.870 | 100.00% \| 97.52% \| 0.996 | 91.70% \| 88.35% \| 0.960 | 100.00% \| 99.75% \| 0.999 | 100.00% \| 97.03% \| 0.988 |

**Forward Feature Selection with random forest**

- We utilised the embedded feature selection approach to identify k-best features with k values ranging from 42 to 30, then 20, 15, and 10.
- When we kept the maximum feature size at 42, our approach discovered 30 top features. Further, with max features size of 30, 20, 15 & 10 our approach found 30, 19, 15 & 10 best features respectively.
- We ran the new dataset with updated features on our machine learning models after each round of feature selection and recorded the results.

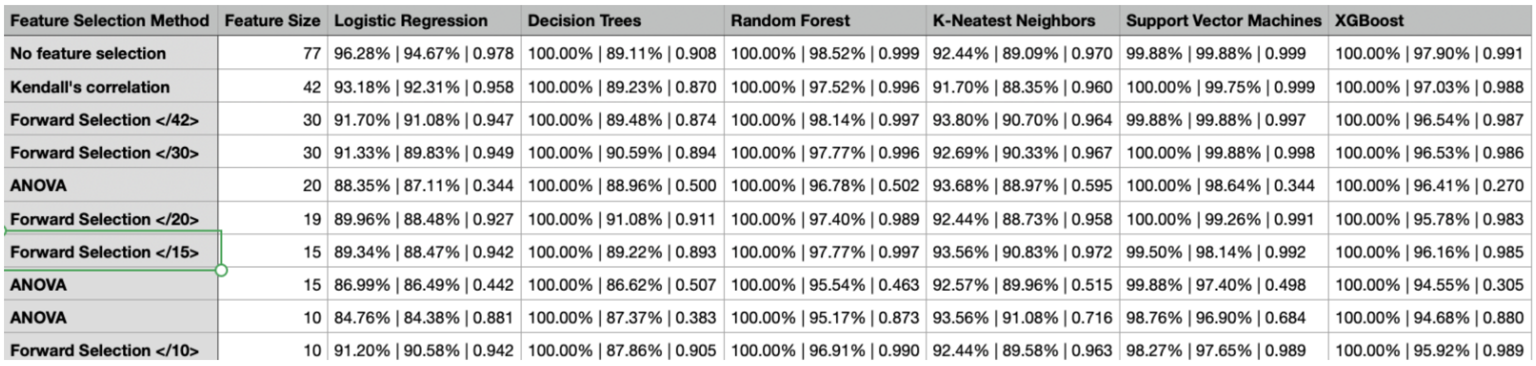
**Anova based feature selection**

- Using the statistical method ANOVA, we selected k-best features with k-values ranging from 20, 15 & 10.
- Similarly here, we ran the new dataset with updated features on our machine learning models after each round of feature selection and recorded the results.

**Results after feature selection**

- The table below shows the results we obtained after training the models on the x-axis using various feature selection methods.

**NOTE** - The intersection of x & y, let's say for **y = No feature selection** and **x = Logistic Regression**, the value **<96.28% | 94.67% | 0.978>** == **< Training Accuracy % | Cross Validation Score (10 Fold) % | AUC Score >**



| Feature Selection Method | Feature Size | Logistic Regression | Decision Trees | Random Forest | K-Neatest Neighbors | Support Vector Machines | XGBoost |
|---|---|---|---|---|---|---|---|
| No feature selection | 77 | 96.28% \| 94.67% \| 0.978 | 100.00% \| 89.11% \| 0.908 | 100.00% \| 98.52% \| 0.999 | 92.44% \| 89.09% \| 0.970 | 99.88% \| 99.88% \| 0.999 | 100.00% \| 97.90% \| 0.991 |
| Kendall's correlation | 42 | 93.18% \| 92.31% \| 0.958 | 100.00% \| 89.23% \| 0.870 | 100.00% \| 97.52% \| 0.996 | 91.70% \| 88.35% \| 0.960 | 100.00% \| 99.75% \| 0.999 | 100.00% \| 97.03% \| 0.988 |
| Forward Selection </42> | 30 | 91.70% \| 91.08% \| 0.947 | 100.00% \| 89.48% \| 0.874 | 100.00% \| 98.14% \| 0.997 | 93.80% \| 90.70% \| 0.964 | 99.88% \| 99.88% \| 0.997 | 100.00% \| 96.54% \| 0.987 |
| Forward Selection </30> | 30 | 91.33% \| 89.83% \| 0.949 | 100.00% \| 90.59% \| 0.894 | 100.00% \| 97.77% \| 0.996 | 92.69% \| 90.33% \| 0.967 | 100.00% \| 99.88% \| 0.998 | 100.00% \| 96.53% \| 0.986 |
| ANOVA | 20 | 88.35% \| 87.11% \| 0.344 | 100.00% \| 88.96% \| 0.500 | 100.00% \| 96.78% \| 0.502 | 93.68% \| 88.97% \| 0.595 | 100.00% \| 98.64% \| 0.344 | 100.00% \| 96.41% \| 0.270 |
| Forward Selection </20> | 19 | 89.96% \| 88.48% \| 0.927 | 100.00% \| 91.08% \| 0.911 | 100.00% \| 97.40% \| 0.989 | 92.44% \| 88.73% \| 0.958 | 100.00% \| 99.26% \| 0.991 | 100.00% \| 95.78% \| 0.983 |
| Forward Selection </15> | 15 | 89.34% \| 88.47% \| 0.942 | 100.00% \| 89.22% \| 0.893 | 100.00% \| 97.77% \| 0.997 | 93.56% \| 90.83% \| 0.972 | 99.50% \| 98.14% \| 0.992 | 100.00% \| 96.16% \| 0.985 |
| ANOVA | 15 | 86.99% \| 86.49% \| 0.442 | 100.00% \| 86.62% \| 0.507 | 100.00% \| 95.54% \| 0.463 | 92.57% \| 89.96% \| 0.515 | 99.88% \| 97.40% \| 0.498 | 100.00% \| 94.55% \| 0.305 |
| ANOVA | 10 | 84.76% \| 84.38% \| 0.881 | 100.00% \| 87.37% \| 0.383 | 100.00% \| 95.17% \| 0.873 | 93.56% \| 91.08% \| 0.716 | 98.76% \| 96.90% \| 0.684 | 100.00% \| 94.68% \| 0.880 |
| Forward Selection </10> | 10 | 91.20% \| 90.58% \| 0.942 | 100.00% \| 87.86% \| 0.905 | 100.00% \| 96.91% \| 0.990 | 92.44% \| 89.58% \| 0.963 | 98.27% \| 97.65% \| 0.989 | 100.00% \| 95.92% \| 0.989 |

**Interpretation of the above table**

- Close inspection of the table above reveals that the method Forward Selection </10> (LAST ROW) on the y-axis has picked the best 10 features, and the results corresponding to that are not considerably different from the results obtained with all 77 characteristics (FIRST ROW). * Examine the AUC values.
- As a result, we select the top ten features and use them to build our new dataset.
- These are the top ten features chosen: -
- ['DYRK1A_N', 'BDNF_N', 'APP_N', 'MTOR_N', 'pGSK3B_N', 'pPKCG_N', 'AcetylH3K9_N', 'IL1B_N', 'pGSK3B_Tyr216_N', 'CaNA_N']

## Dataset 1 - Further Testing & Results

On testing all our models on the new dataset the following classification report is produced.

* Note - The confusion matrix is (1, 0) vs (1, 0) which is **Ts65Dn as 1, Control as 0**

<<<Executing all ML models for 10 features>>>

**Model Name : Random Forest**
**Parameters : {'criterion': 'gini'}**
Training Accuracy : 100.00%
Testing Accuracy : 95.19%
Cross Validation : 96.53%
Confusion Matrix :

Classification Report: -

```
[[136   5]
 [  8 121]]
```

**Model Name : Support Vector Machines**
**Parameters : {'C': 10, 'kernel': 'rbf', 'probability': True}**
Training Accuracy : 97.40%
Testing Accuracy : 93.33%
Cross Validation : 95.67%
Confusion Matrix :

Classification Report: -

```
[[131  10]
 [  8 121]]
```

**Model Name : XGBoost**
**Parameters : {}**
Training Accuracy : 100.00%
Testing Accuracy : 96.30%
Cross Validation : 95.54%
Confusion Matrix :

Classification Report: -

```
[[136   5]
 [  5 124]]
```

**Model Name : K-Nearest Neighbors**

**Parameters : {'n_neighbors': 20}**
Training Accuracy : 93.93%
Testing Accuracy : 89.26%
Cross Validation : 91.70%
Confusion Matrix :

Classification Report: -
```
[[132   9]
 [ 20 109]]
```

**Model Name : Decision Trees**
**Parameters : {'criterion': 'gini'}**
Training Accuracy : 100.00%
Testing Accuracy : 90.74%
Cross Validation : 90.10%
Confusion Matrix :

Classification Report: -
```
[[126  15]
 [ 13 116]]
```

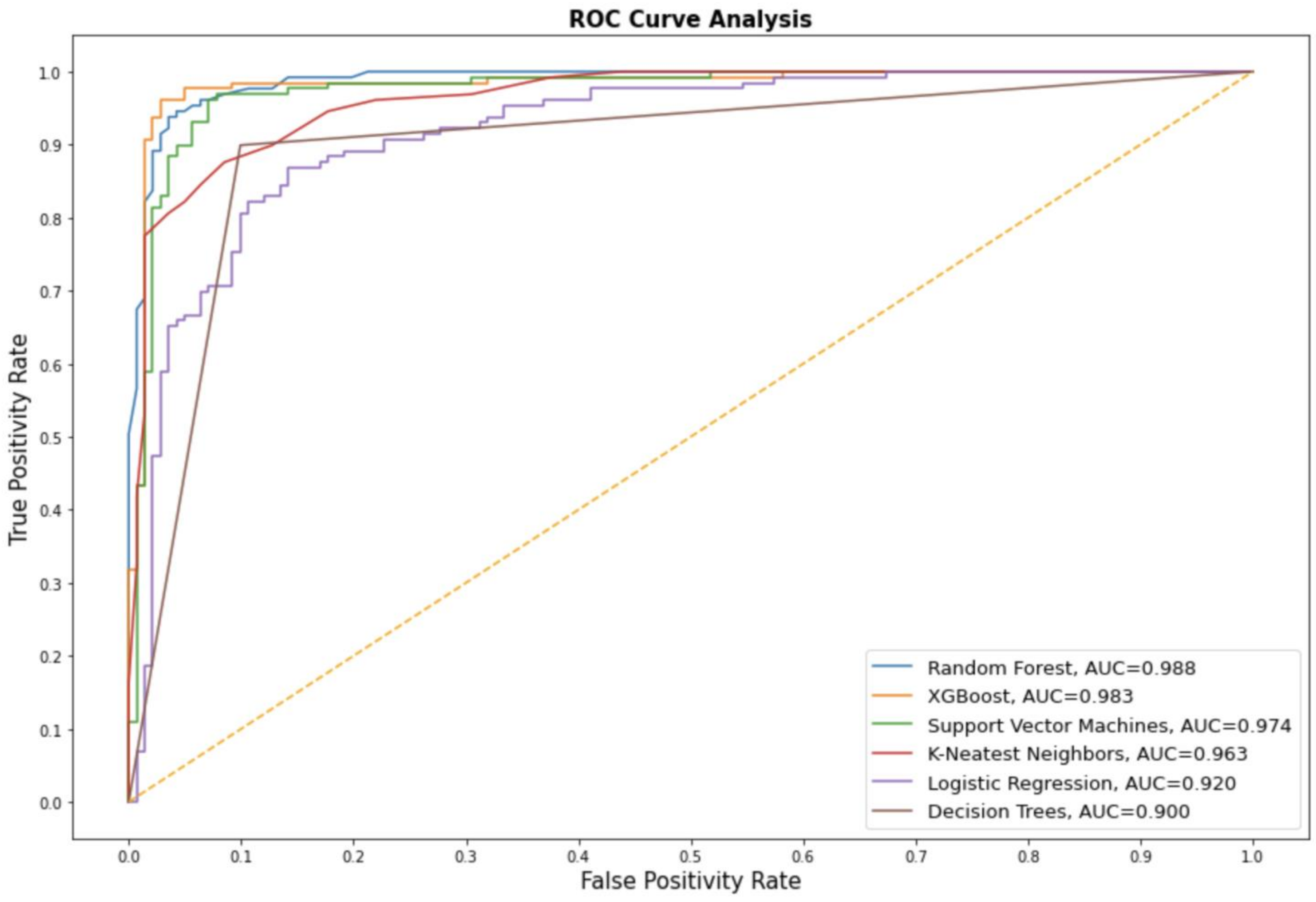
**Model Name : Logistic Regression**
**Parameters : {'C': 1, 'max_iter': 1000}**
Training Accuracy : 89.59%
Testing Accuracy : 85.56%
Cross Validation : 88.10%
Confusion Matrix :

Classification Report: -
```
[[127  14]
 [ 25 104]]
```

—-----------------------------------------------------------------------------------------------------------------

According to the aforementioned classification report, and taking the issue of overfitting into consideration, I can state that **Logistic Regression** is the best ML model for our classification use case.

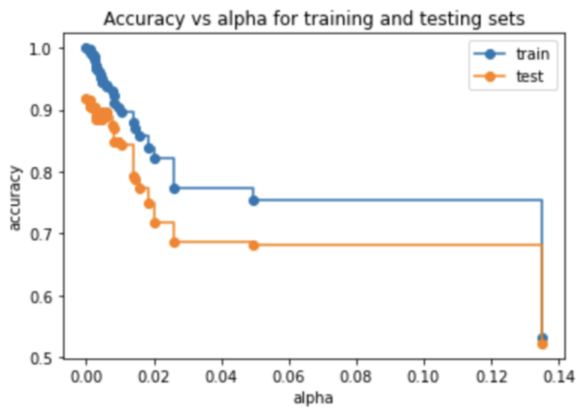
**ROC-AUC Curve**



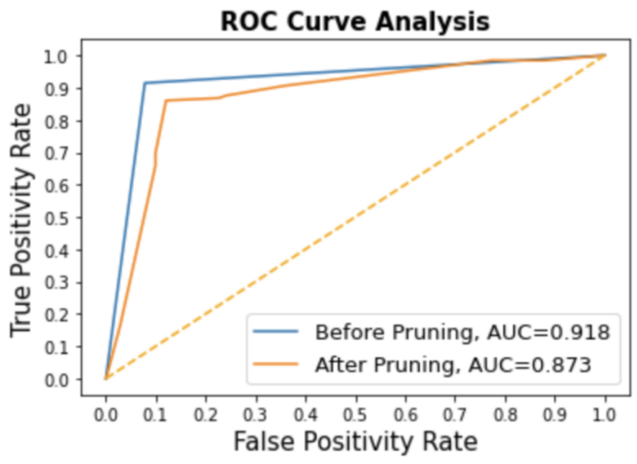
- Using ROC curve analysis, we can observe that **Random Forest**, which has an AUC value of 0.988, is our best model.

**An attempt towards reducing overfitting in decision trees using pruning**

- The figure that shows the training and testing precision of a decision tree model developed on a new dataset with 10 features is shown below.
- We can observe that the training and testing accuracy of the model is closest around alpha value 0.01, which results in reduced overfitting.



- We retrained our decision tree with an alpha value of 0.0084 and plotted the roc-curve. However, we discovered that the newly trained model, which exhibits less overfitting, did not perform much better than the previous one.



## Dataset 1 - Improvement Needed & Future Plans

- Our efforts were limited in optimising the machine learning methods we deployed since feature reduction remained our major emphasis.
- To explore the capabilities of the current model, further hyper-parameter tuning may have been used.
- Different approaches may have been used to achieve more notable gains in decreasing the overfitting, which is quite obvious in some models, such as decision trees.
- Additionally, the findings may have been visualised more effectively using the abundance of Python tools.