**DMLL Group 3 Lab Report**

**Members**:-

Gowresh (H00413256)

Yehia (H00409854)

Sheetal (H00412798)

Gunjan (H00419339)

Nasir (H00416471)

# Table of Contents

# Lab 1:-

**Date:** 22-09-22

**Summary of Discussion: -**

**Forest Fire Dataset**

The dataset chosen for our lab work is obtained from satellites Tegra and Aqua using an instrument called MODIS based on hotspots of forest fire in Turkey. There are 36,011 records/instances in the dataset with each row pertaining to the satellite readings within a specific time interval. The columns represent 15 attributes/features. The target feature is the 'confidence' which helps to gauge the quality of the hotspot/fire. The 'confidence ' attribute is logged in terms of percentage, which can be represented as a nominal data with three categories Low, Nominal/Medium and High. This dataset can be used to train a model to predict whether the confidence of the forest fire is Low, Nominal or High.

The following are the each attribute names with their data types and attribute types:

Latitude (meters)

float (ratio)

Longitude (meters)

float(ratio)

brightness temperature (Kelvin)

float(ordinal)

scan (meters)

float(ratio)

track (meters)

float(ratio)

acq_date

string(interval)

acq_time

int(interval)

satellite

String(nominal)

instrument

String(nominal)

confidence (%)

int(can be represented as nominal)

version

float(nominal)

brightness 31 channel temperature (Kelvin)

float(ordinal)

frp (MW)

float(ratio)

type (inferred hotspot type)

int (can be represented as nominal)

day night

string(nominal)

This dataset has a good combination of categorical and ordinal data that will help us to best understand ML Practices. There are no missing values in the dataset which may benefit us with accuracy of prediction. The end outcome we are expecting from using this dataset is to predict the confidence levels of the hotspot given a set of attributes. The target feature can be used to create categorical data that will help us to understand the normalization and binning concepts effectively.

**Other Datasets considered:**

Mobile Robot Floor Prediction -

A dataset containing 500,000 records containing quaternion, acceleration, and velocity features that all are ratio attributes and a target feature 'floor type' that is nominal

Hand Gesture Detection -

A dataset containing 778 training images of hand gestures and 6 different classes to predict. Additionally, labels are provided defining the bound boxes of the classes in each image.

# Lab 2:

**Date:** 29-09-22

**Summary of Discussion: -**

A Github repository was created for the collaboration purpose of this portfolio[i]

There are totally 15 attributes in the dataset. The latitude and longitude attributes refer to the geographic location of the hotspots detected by the satellites. The brightness attribute is the pixel

measurement of the temperature in Kelvin. Scan and Track attributes refers to the pixel measurement of the scanned area. acq_date refers to the date of acquisition of data while acq_time refers to the time spent during the acquisition of data. Satellite attribute refers to which satellite was used for acquiring the data while instrument refers to the instrument used by the satellites for measurement. Confidence refers to the quality of the hostpot in the specified region of interest. This parameter can be used to predict whether a hotspot is a fired area or not. This will be our target feature (Values greater than 60% or 70% usually mean the hotspot is actually a fire spot). Version attribute refers to the collection of sources. bright_t31 is the pixel measurement of temperature in the 3-1 Channel. frp attribute depicts the pixel-integrated fire radiative power in MW (Mega Watts). Daynight attribute refers to when the instance was obtained, whether day or night. Finally, the type attribute refers to the type of environment of the instance.

There were no missing values in the dataset previously. However, to demonstrate the process of data cleaning, we have manually removed the 5 values pertaining to acq_time attribute.

The value_count() displays the number of values in each category of an attribute across the dataset. Here there are 28,203 values indicating that the respective measurements were taken during daytime while 7,808 values indicating that those measurements were taken during nighttime. The satellite dataset has two categories indicating the model of satellite that was used for measurement. It is observed that the instrument attribute has only one category across the entire dataset, which clearly indicates that the attribute can dropped from the dataset when processing.

The histogram plots help us to summarize between discrete and continuous data that are measured on an interval scale. Through the plot we can identify that frp attribute has an outlier. Similarly, it is also evident that the attributes version and type do not have any effect on the target feature.

While the acq_date is not categorical data, we must consider the activity of changing the data type.

The first plot does not help us in finding any pattern. The second plot is comparatively better in representing the density of areas where the measurements was taken. The third plot is more indicative as we can understand the different confidence levels of hotspot across the areas.

The most correlated attributes with the target attribute confidence are the brightness and the frp such that they have strong positive correlation. According to the definitions of these two attributes, it makes sense that the brightness, which is the temperature of hotspot region, and the frp, which is the radiative power of the hotspot, can affect the confidence of the hotspot more than the other features. The scan, track and acq_time attributes have negative correlation with the confidence attribute. This may be since the more time the satellite spends on a particular region of interest, the less likely it can be that that region has low confidence of hotspot.

Due to the upward trend, it can be concluded that the correlation is strong. There are straight horizontal lines at around 100, 90, 85, 75 and even around 0. This needs to be considered to prevent data quirks.

We have represented the categorical values in numbers. Generally, in ML, there may be a possibility that nearby values are more similar than distant values. However as far as this dataset is concerned the categorical values have only binary classification

Since this dataset does not have null values, we are inserting null values to some records to demonstrate the Data Cleaning Process. The three different options of Dealing with Null Data were tried on the dataset: -

(i) Removing the records related to null values

(ii) Removing the entire attribute related to the null values

(iii) Filling the mean values of the attribute to the null values

Of these three options, we have decided to use the third option as it will help us in avoiding the elimination of the records in the dataset.

Normalisation or Standardisation need to be done on the dataset in order to scale the values. For eg, brightness attribute has a min and max values of 300.000000 and 504.4 while frp attribute has a min and max values of 0 and 3679.5. To avoid the numerical values from having such different scales, we perform normalisation or standardisation. To best eliminate outliers in the data set, we are going to use the standardised dataset for further processing.

# Lab 3:

**Date:** 06-10-22

We dropped the attributes containing the same values for all instances in the dataset. The dropped attributes are instrument and version as they contained the same values for all instances.

The two attributes daynight and satellite are categorical in nature. So, we converted these categorical data into numerical data by mapping the classes with 0s and 1s.

We then found out the correlation of the remaining attributes with respect to the class attribute confidence.

Additionally, we tried out the following feature selection algorithms[ii]:

1. Univariate Selection
2. Feature Importance
3. Correlation Matrix with Heatmap

We tried to find the best features using Univariate Selection using the reference link: -

Due to using chi square method in Univariate Selection[iii], it does not take negative values (Like Latitude and Longitude). For the purpose of demonstrating the Univariate Selection Method, we are dropping the latitude and longitude attributes. However due to this limitation the results cannot be considered for our dataset

We found the best features using Feature Importance using the same link. According to the results of this method, the top attributes were brightness, frp, latitude, bright_t31 and longitude.

We finally found out the best features using Correlation Matrix with Heatmap using the same reference link. According to the results of this method, the top attributes were identified as brightness, frp and bright_t31.

Univariate Selection had equivalent results with the correlation matrix but it is not a fit for our dataset because it does not take negative values so attributes with negative values had to be dropped before implementation.

Feature importance also showed comparable results with the correlation matrix algorithm and works well with the dataset provided and makes it easier to see the most correlation features using the histogram.

Correlation matrix with heat map is a visualised form of correlation matrix which also makes it easier to identify the most correlating features.

Consolidating all the results, excluding Univariate Selection for reasons stated above, the top features are brightness, frp and bright_t31.

We performed the binning method to use our dataset as a multi-class problem. Using the binning method, the target feature was classified into 3 classes and the classes stored as numerical data from 0-2

70-100 – High (2)

40-70 – Medium (1)

-1-40 – Low (0)

***Note:*** *We had to use -1 as the start point for Low bin as using 0 caused assignment of NaN values to the instances having 0 confidence.*

As per the lab sheet, to find the top features per each class, we divided the datasets into three as per the following criteria: -

- dataset _1 contains the output labels of class low as 0 while other classes will be denoted as 1
- dataset _2 contains the output label of class medium as 1 while other classes will be denoted as 0
- dataset _3 contains the output label of class high as 2 while other classes will be denoted as 0

On each of the three datasets, we used the heatmap correlation matrix to find out the top correlating features. The observations are as follows:

***Note:*** *We chose top 2, top 5 and top 8 features per each class as our dataset contained only 11 features. Choosing top 10 features yielded in a dataset that was the same as the original dataset.*

| Features | Low Class | Medium Class | High Class |
|---|---|---|---|
| **Top 2** | Brightness, Brightness_t31 | Brightness, FRP | Brightness, FRP |
| **Top** | Brightness, Brightness_t31, Daynight, FRP, Acq_time | Brightness, FRP, Daynight, Brightness_t31, Track | Brightness, FRP, Brightness_t31, Track, Scan |
| **Top 8** | Brightness, Brightness_t31, Daynight, FRP, Acq_time, Scan, Track, Longitude | Brightness, FRP, Daynight, Brightness_t31, Track, Scan, Acq_time, Latitude | Brightness, FRP, Brightness_t31, Track, Scan, Acq_time, Daynight, Latitude |

Three datasets were created containing the top 2, top 5 and top 8 correlating features of each of the 3 classes and were named dataset_1, dataset_2 and dataset_3, respectively.

Then using SciKit learn, we implement the PCA (Principal Component Analysis) to reduce the dimensions of our original dataset.

We first tried to reduce the entire dataset into 2 dimensions. We found out that the explained variance of the resulting principal components was less (0.27705587, 0.19335908). To explore further we tried to reduce the dimensions to a value such that the variance is preferred to 95%. The result was that 11 features were reduced to 8 principal components with variances (0.27705587, 0.19335908, 0.14166932, 0.1151124, 0.09112534, 0.08817307, 0.03785807, 0.02530086)

Using a simple classifier (Decision Tree Classifier) we trained and tested the accuracy of our model for our original dataset as well as the three subsets.

The following were the recorded accuracies by running Decision Tree Classifier on the different datasets: -

| **Dataset Type** | **Prediction Accuracy(%)** |
|---|---|
| Original | 67.4 |
| Dataset_1 containing Top 2 features per each class | 61 |
| Dataset_2 containing Top 5 features per each class | 66.4 |
| Dataset_3 containing Top 8 features per each class | 67.2 |

It is our observation that for our dataset, creating subsets had not increased the accuracy of the classifier model. But it is also clear that the accuracy of the dataset containing top 8 features is the same as the accuracy of the original dataset containing the 11 features. It was also observed that running the PCA on our original dataset reduced the dimensions from 11 to 8.

# Lab 4:

**Date: 13-10-22**

In this lab, we used three different classifiers on our dataset and evaluated each classifier with accuracy, precision, recall, F1 score, confusion matrix and ROC.

To begin with we had to adapt our dataset so that it can be used for binary classification. As ROCs can be computed only for binary classification models, we decided to use binning method to convert our target feature, confidence, into two different classes. As in the previous lab, we used binning method but with two labels. The label 1 classifies the confidence values greater than 70%(High Confidence) while label 0 classifies the confidence values less that 70% (Low Confidence).

For evaluation purposes, we split our dataset into a training set and a test set. We used the sciket-learn's train_test_split function to split the dataset such that 75% of the original dataset was in training set while the remaining 25% was in test set.

As recommended by the Python Tutorial Module 2, we further changed the target features test set and train set into an array of True or False, True referring to High Class of Confidence and False referring to Low class of Confidence.

We fitted three different classifiers on the training dataset. The classifiers are: -

1. Stochastic Gradient Descent Classifier
2. Random Forest Classifier
3. K-Nearest Neighbour Classifier

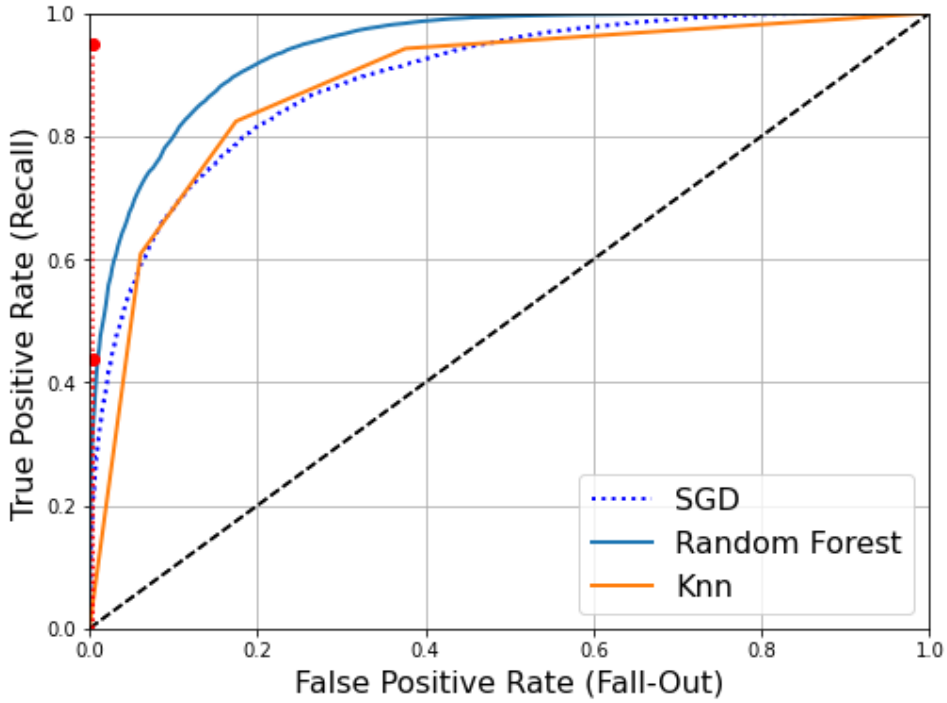
With the test set, the following are the results obtained using SGD Classification

| **Metrics** | **SGD Classifier** |
|---|---|
| **Accuracy** | 69.6% |
| **Precision** | 76.3% |
| **Recall** | 74.3% |
| **F1 Score** | 74.08% |
| **Confusion Matrix** | ([[12025,1713],<br>[5140, 8130]]) |
| **ROC Area** | 89.17% |

After the two other classifiers were used, the Receiver Operating Curve (ROC) was plotted and the region under the curve was found out for each of the classifiers. The results are tabulated as follows: -

| **Metrics** | **SGD Classifier** | **RF Classifier** | **KNN Classifier** |
|---|---|---|---|
| **Precision** | 76.3% | 85.4% | 82.0% |

| | | | |
|---|---|---|---|
| **Recall** | 74.3% | 86.3% | 82.4% |
| **ROC Area** | 89.17% | 94.18% | 88.3% |
| **True Positives** | 8130 | 11462 | 10936 |
| **False Negatives** | 5140 | 1808 | 2334 |
| **True Negatives** | 12025 | 11786 | 11341 |
| **False Positives** | 1713 | 1952 | 2397 |




From the above plot and the tabulated results, it is evident that the Random Forest Classifier is most suitable for this dataset as its area under the curve is higher than the other classifiers.

The Confusion Matrix compares the decisions made by the model with the actual decisions. This will give an understanding of the level of accuracy of the model in terms of how well the model will perform on new previously unseen data. It is useful to compare the performance as measured using the validation and the testing dataset with the performance as measured using the training dataset. In our case of prediction of the forest fire confidence, the Random Forest Classifier has the highest True Positives, which is essential for the prediction of whether a given hotspot is a forest fire or not.

ROC provides more visual clarity on where to make the cut off when evaluating a continuous measure. It gives an idea about the range of behaviours between the true positives and the false positives rate. It is plotted with the two metrics against each other, TPR on y axis and FPR on x axis.

Implementation of Pipeline

The pipeline in Python is used to automate the flow of code in modelling for machine learning. The pipeline was implemented to automate the flow of transformation from categorical to numerical data.

# Lab 5

Date: 20-10-2022

Bayesian Learning and Bayes Nets

In this week, we performed the evaluation steps with the Naive Bayes Classifier namely

1.Multinomial Naive Bayes

2. Gaussian Naive Bayes

3. Complement Naive Bayes

4. Bernoulli Naive Bayes

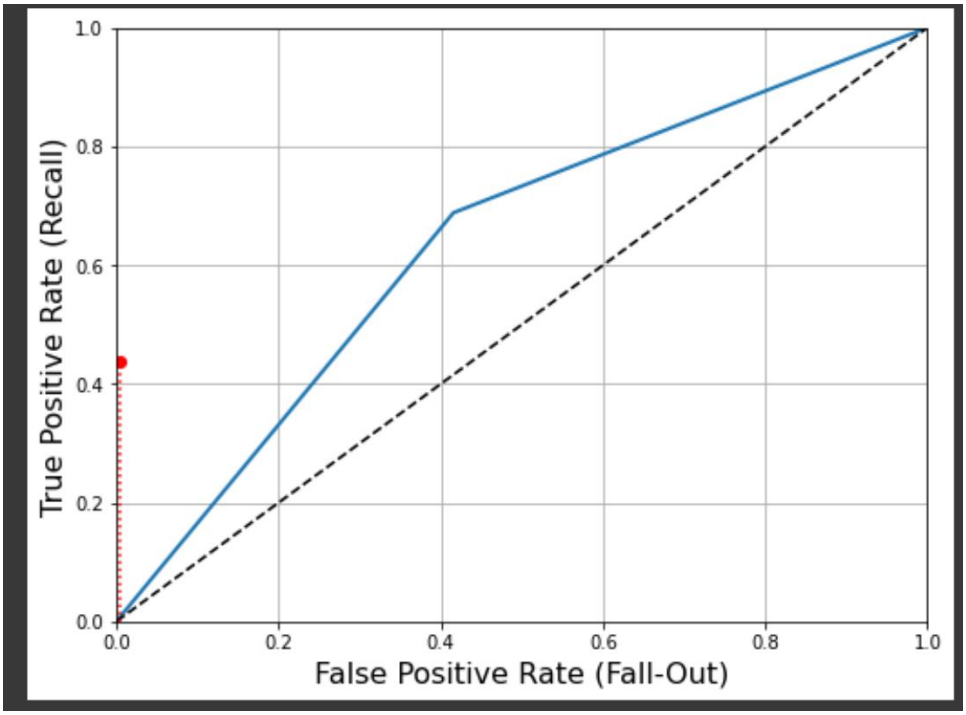
5. Categorical Naive Bayes

We predict that Accuracy would be the best evaluation metric to find the best Bayes Net implementation. The Gaussian Naïve Bayes implementation would provide the best solution as it is assumed to be in a normal distribution.

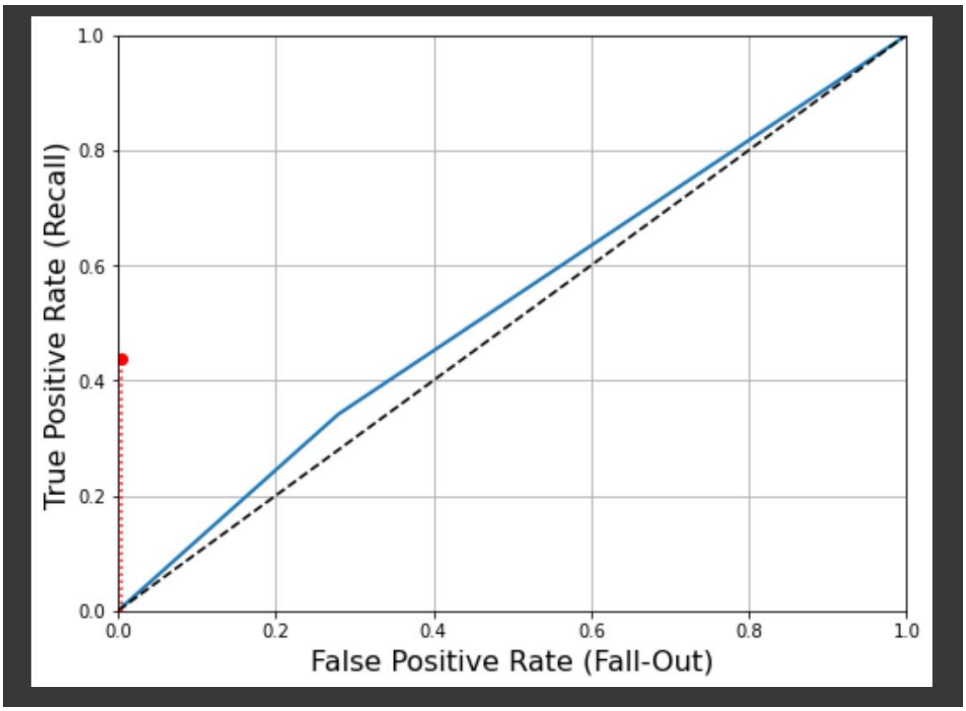
The results are tabulated below:

| Naïve Bayes Algorithm | Accuracy | TP | FP | TN | FN | Sensitivity | Specificity | Precision | Recall | Area under RoC Curve |
|---|---|---|---|---|---|---|---|---|---|---|
| 1. Multinomial Naive Bayes | 64.80 % | 8229 | 5041 | 9274 | 4464 | 64.83 % | 64.78 % | 64.80 % | 64.75 % | 64.75% |
| 2. Gaussian Naive Bayes | 74.82 % | 7829 | 5441 | 12830 | 1358 | 85.21 % | 69.46 % | 77.34 % | 74.55 % | 74.55% |
| 3. Compl | 63.54 % | 9132 | 4138 | 8030 | 5708 | 61.53 % | 65.99 % | 63.76 % | 63.63 % | 63.63% |

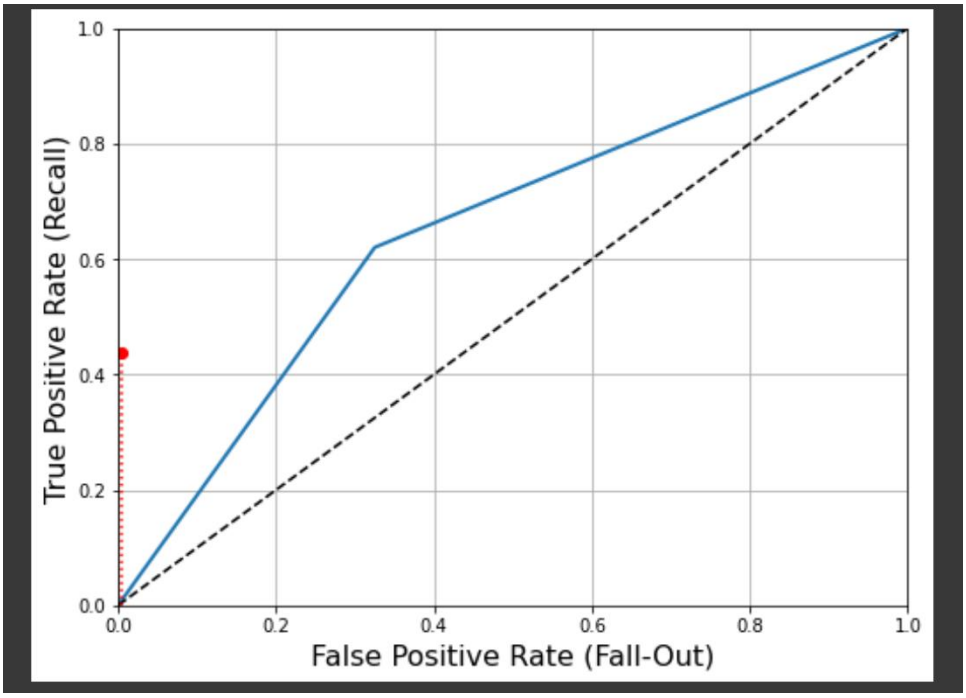
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ement Naive Bayes | | | | | | | | | | |
| 4. Bernoulli Naive Bayes | 53.42 % | 8229 | 5041 | 9274 | 4464 | 85.21 % | 69.46 % | 53.61 % | 53.09 % | 53.09% |
| 5. Categorical Naive Bayes | 74.82 % | 7829 | 5441 | 12830 | 1358 | 85.21 % | 69.46 % | 77.34 % | 74.55 % | 74.55% |
| | | | | | | | | | | |



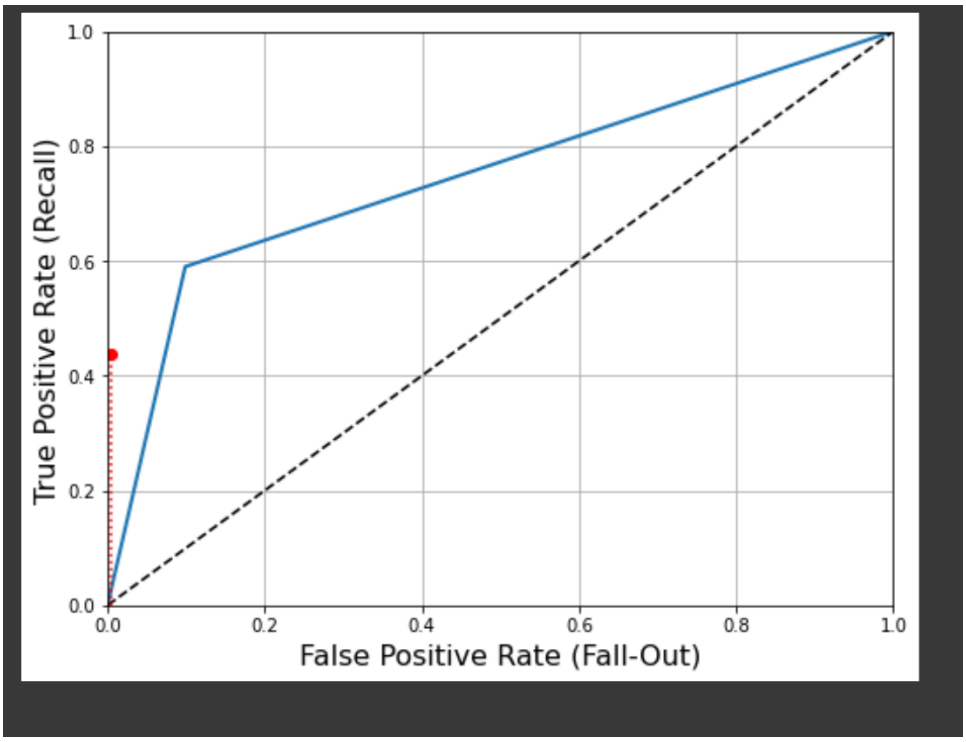

Complement Naïve Bayes Algorithm



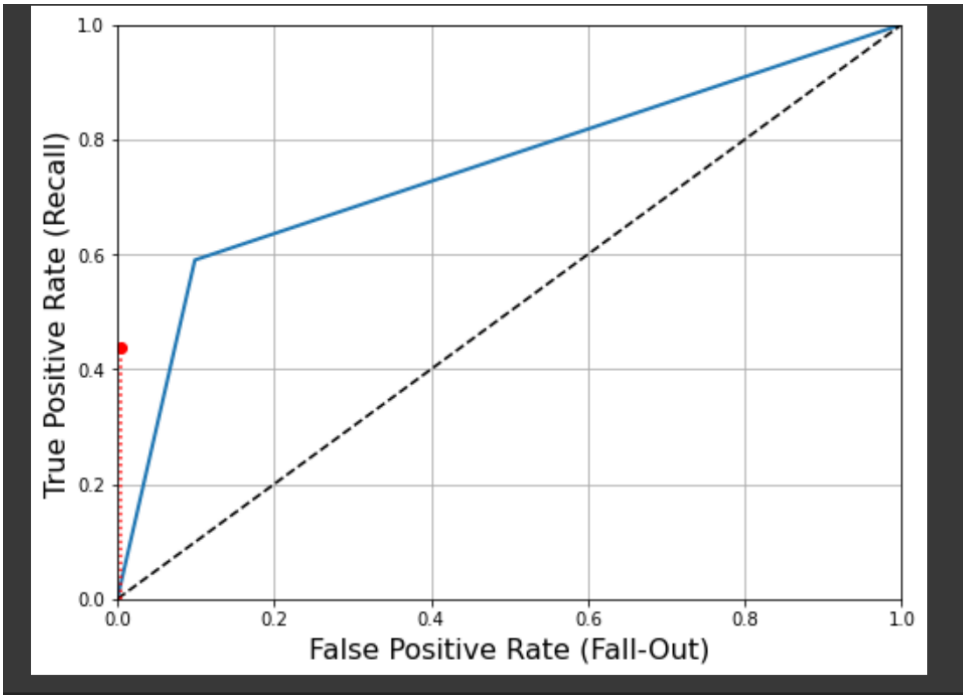

Bernoulli Naïve Bayes Algorithm




Multinomial Naïve Bayes Algorithm




Gaussian Naïve Bayes Algorithm




Categorical Naïve Bayes Algorithm

The TP evaluation metric is the suitable one to select the optimum algorithm. Based on our results, Complement Naïve Bayes algorithm is the most appropriate.

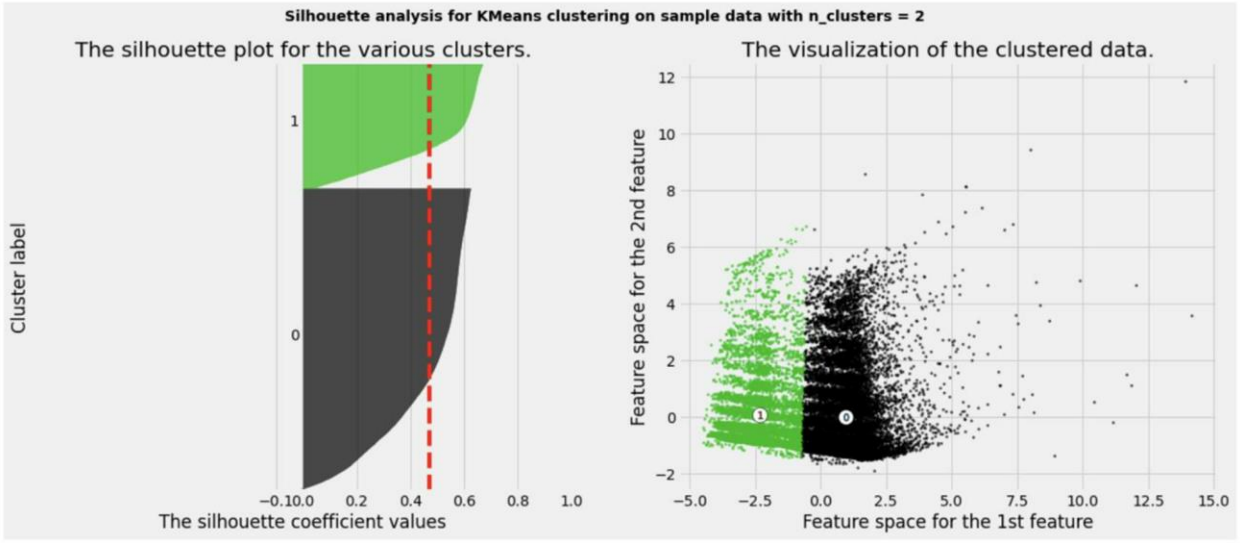
It also holds a good accuracy of 63.54%.

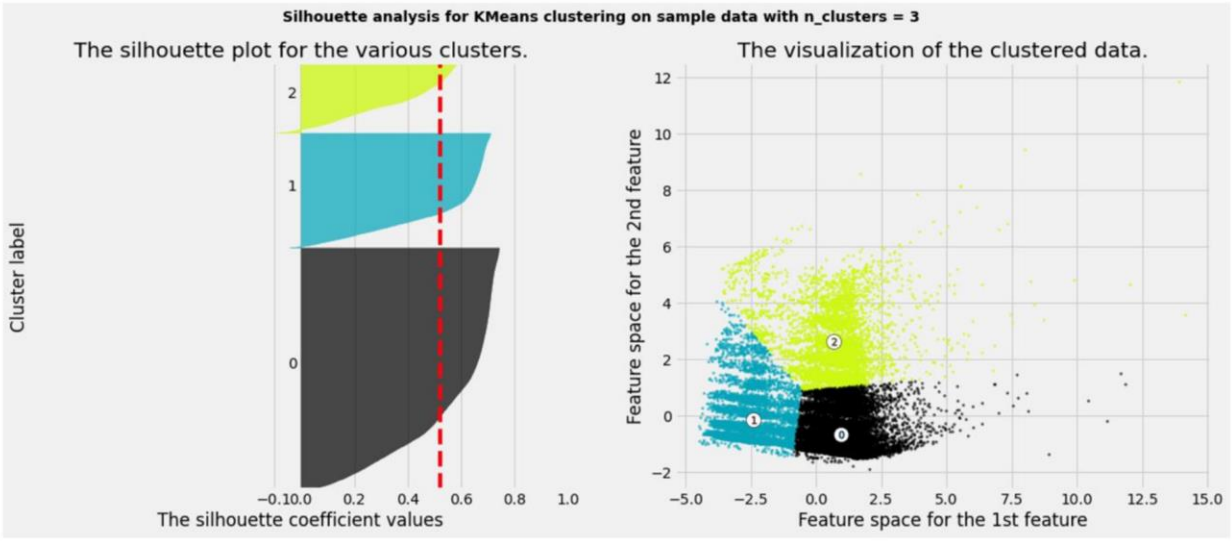
# Lab 7

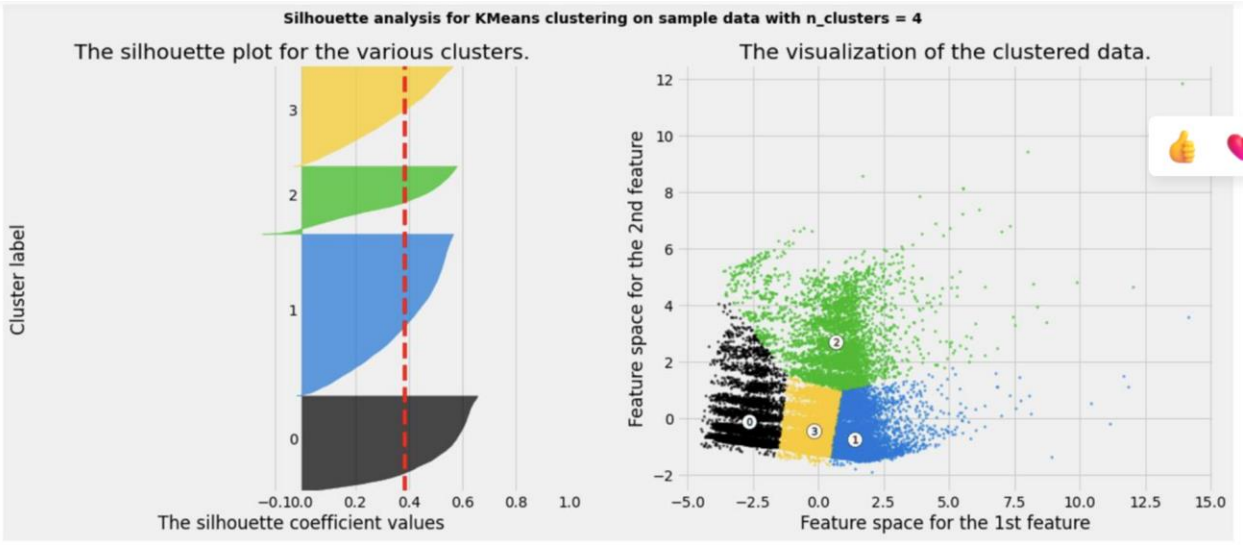
Clustering is the grouping of specific objects based on their characteristics and their similarities. As for data mining, this methodology divides the data that is best suited to the desired analysis using a special join algorithm.

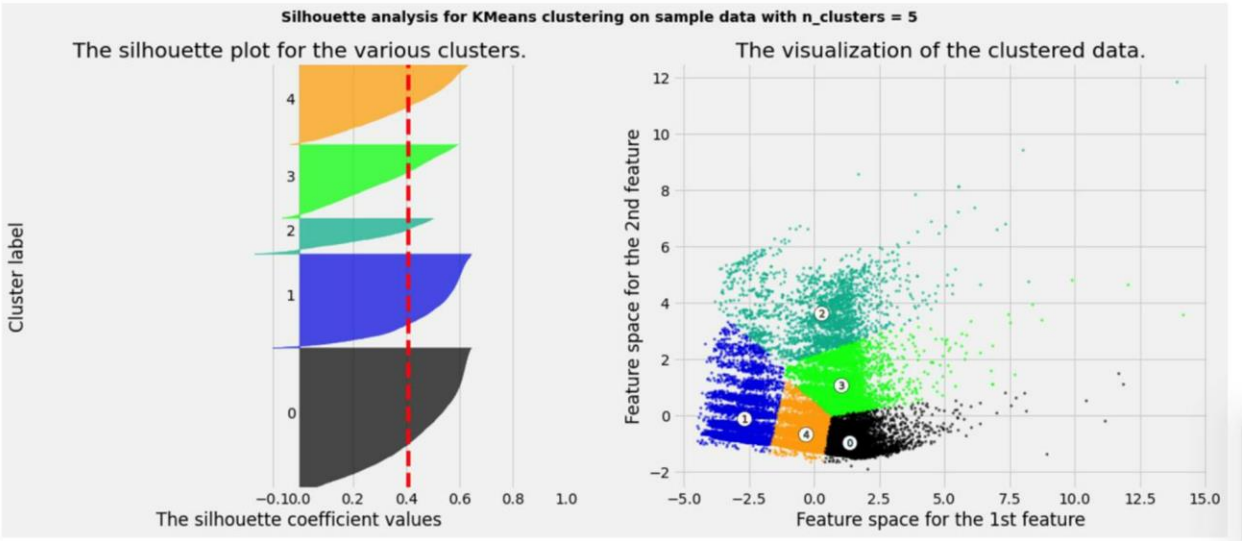
K-means clustering is a method used for clustering analysis, especially in data mining and statistics. It aims to partition a set of observations into several clusters (k), resulting in the partitioning of the data.

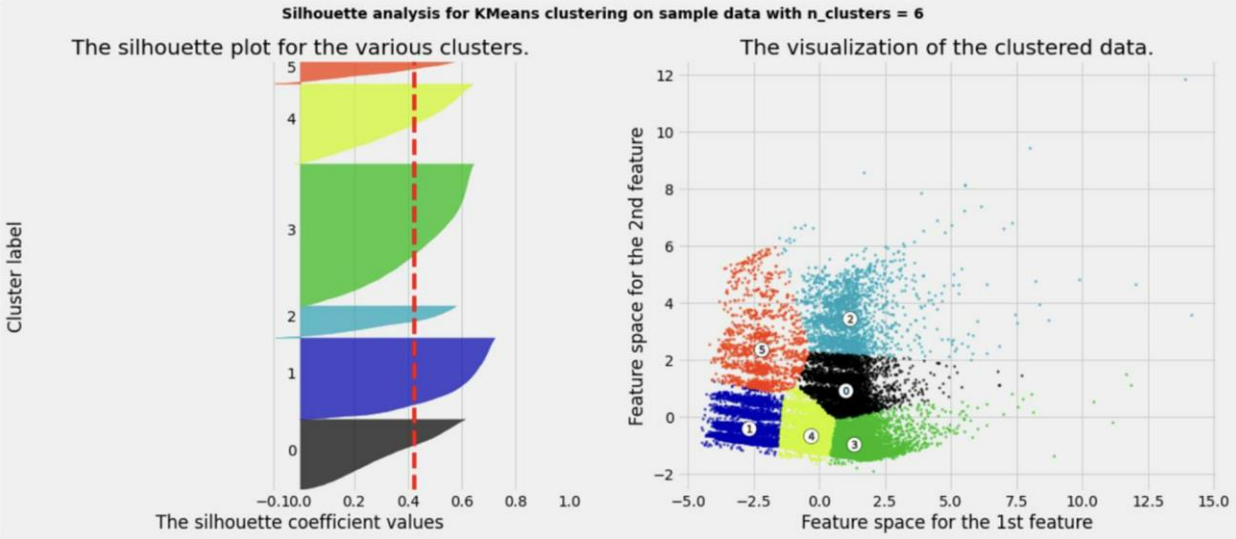
The number of the clusters varied, and the Silhouette score for K-means clustering was recorded.

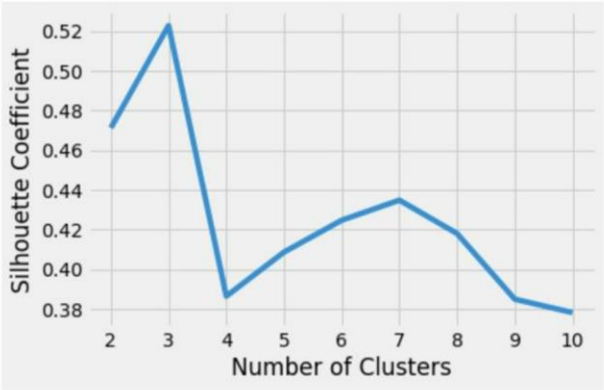



Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 5
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature



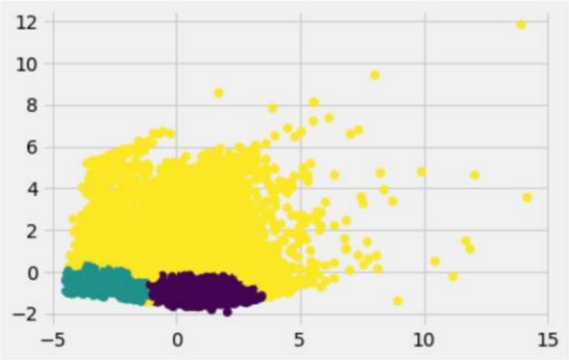

```
For n_clusters = 2 The average silhouette_score is : 0.4712119391675152
For n_clusters = 3 The average silhouette_score is : 0.5227382589467306
For n_clusters = 4 The average silhouette_score is : 0.3862670769794484
For n_clusters = 5 The average silhouette_score is : 0.4072661192587642
For n_clusters = 6 The average silhouette_score is : 0.42490472374729293
```

| Number of clusters | Average Silhouette score/Accuracy |
|---|---|
| 2 | 47.12% |
| 3 | 52.27% |
| 4 | 38.62% |
| 5 | 40.72% |
| 6 | 42.49% |

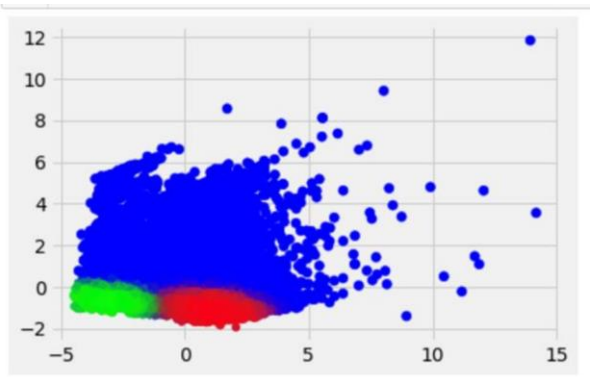
**Conclusion:** The optimal number of clusters is 3 as it has the highest Silhouette score of 52.27%

**Gaussian Mixture Model**

The number of clusters is 3 when clustered using Gaussian Mixture Model.

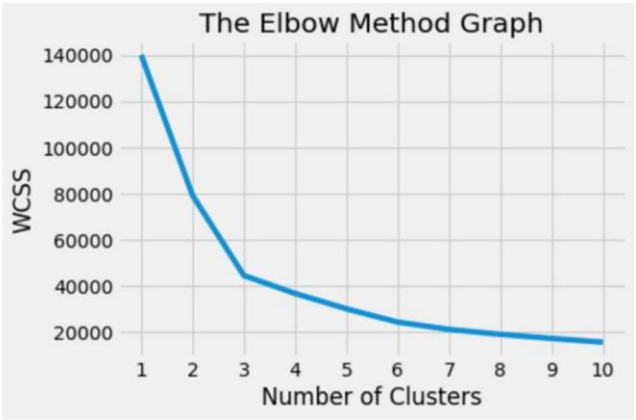



GMM Hard Clustering- where each item belongs to only one cluster




GMM Soft Clustering- where each item belongs to more than one cluster.

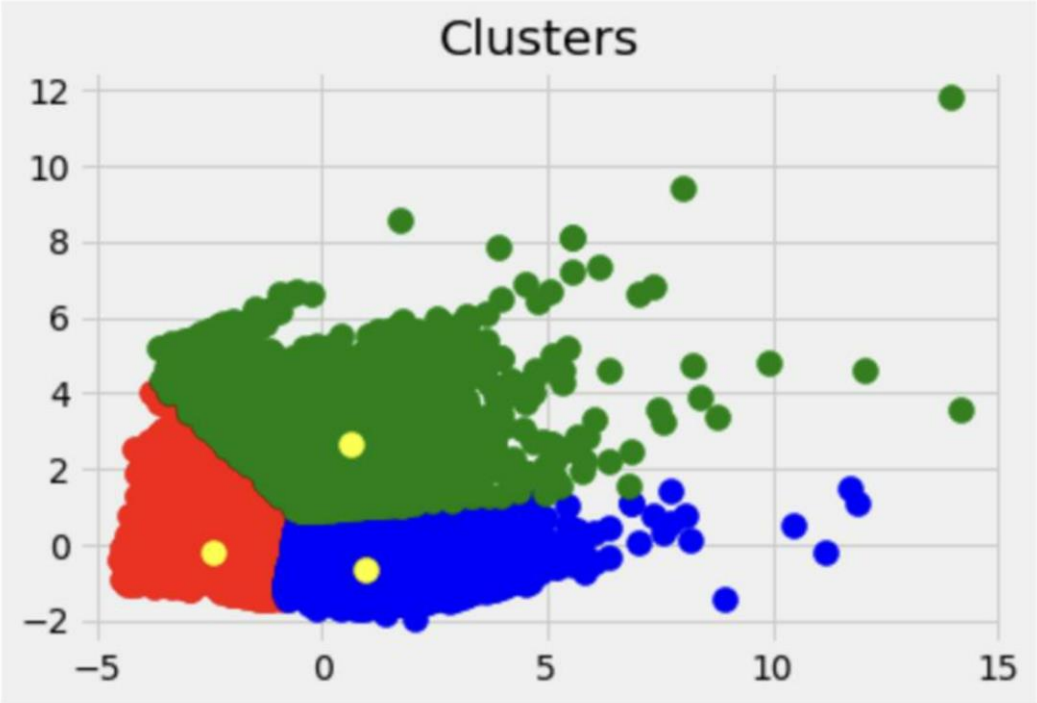
Note- Hard clustering is method to grouping the data items such that each item is only assigned to one cluster, K-Means is one of them. While Soft clustering is method to grouping the data items such that an item can exist in multiple clusters



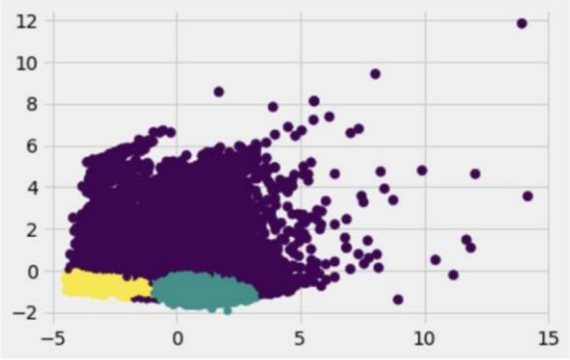

The Elbow method is a representation of the number of clusters, K against the Sum Squared error (within cluster or otherwise). It shows the optimal number of clusters is 3. The reason behind this conclusion is

the point in the curve (at the number of clusters,3) where the inertia stops dropping and stabilizes there on. This gives the elbow shape.

The library, Kneed has a function KneeLocator shows the point on the elbow shape. It can be used as a source of cross verification.




K-Means Plot




BGM Pred Plot

The Elbow method shows the optimal number of clusters is 3. The reason behind this conclusion is the kink at the number of clusters,3 which gives the elbow shape.

| Cluster Algorithm | Optimum number of clusters | BIC | AIC | Inertia | Bgm score |
|---|---|---|---|---|---|
| K-Means | 3 | - | - | 8040 | |
| Gaussian Mixture Model | 3 | -1118645.29 | -1120556.80 | - | |
| BGM predict | 3 | | | | 17.59 |

**Result:**

The three algorithms of clustering showed that the optimal number of clusters is 3. K-means is a hard clustering algorithm and BGM Predict is a soft clustering algorithm while Gaussian Mixture model is a combination of both hard and soft clustering.

In comparison, The Bayesian classification methods yielded better accuracy than the clustering algorithm. Basically, the clustering algorithm is used to segregate the data into the optimal number of clusters.

Pros and Cons of Hard and Soft Clustering

| Pros | Cons |
|---|---|
| Soft clustering allows overlapping whereas in hard clustering clusters are exclusive. | Soft Clustering is slower than the hard clustering process as there are more values to compute. |
| Soft clustering works better than the traditional hard clustering. | Soft clustering requires the data to be in Euclidean |

# LAB 8

In the previous lab session, we had practiced various algorithms for hard and soft clustering, like K-means algorithm, BGM Predict and Gaussian Mixture Model. In addition to that, the need and pros and cons of using various clustering algorithms were discussed.

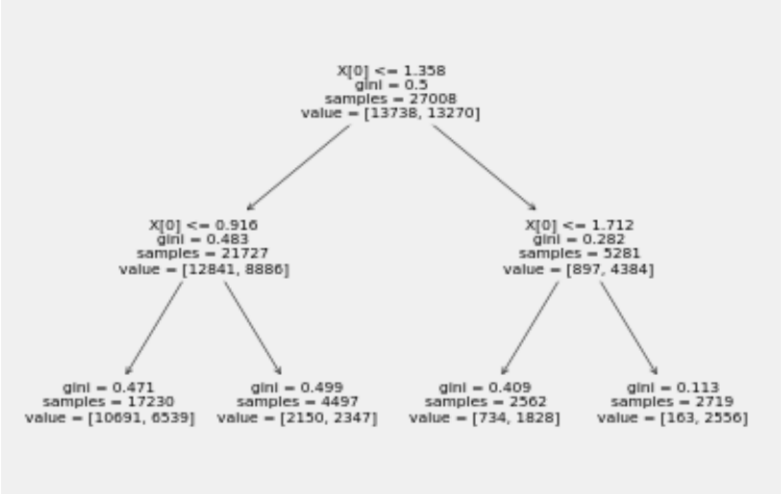
**Algorithm for building and using Decision Trees**

The purpose of this lab is to implement the algorithm for building and using Decision Trees under the various conditions.

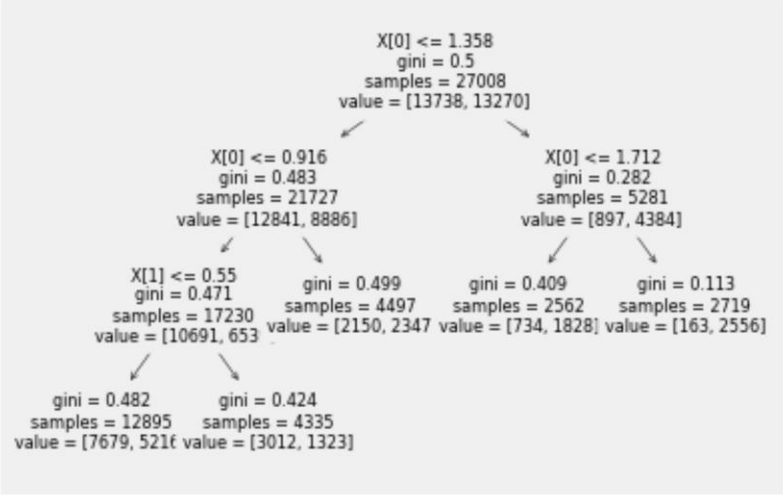
Using PCA analysis, the large dataset is analyzed by visualization of large number of features without affecting the necessary information contained in them. The target feature is converted into three classes using the binning method.

**Tuning the different hyperparameters and analyzing the affects on the Decision Tree**

**Experimentation with Different Depth Parameter**

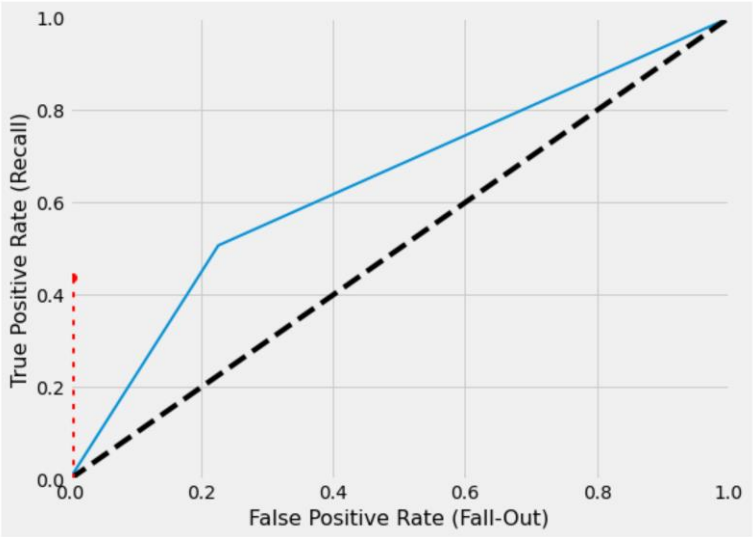



Depth Parameter=2

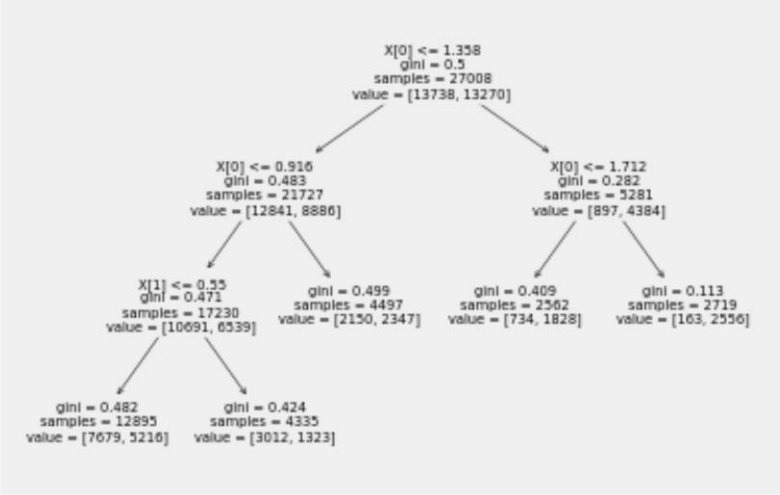



Depth parameter = 5

| Depth Parameter | TN | FN | FP | TP | Accuracy | 10-fold cross validation score | Specificity | Sensitivity | Precision |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 10636 | 3102 | 6541 | 6729 | 64.29% | 63% | 61.92% | 68.44% | 65.18% |
| 5 | 10636 | 3102 | 6541 | 6729 | 64.29% | 63% | 61.92% | 68.44% | 64.06% |




## Minimal number of instances permissible per leaf




Minimum samples per leaf = 5

## Overfitting and Underfitting in Machine Learning

### Overfitting

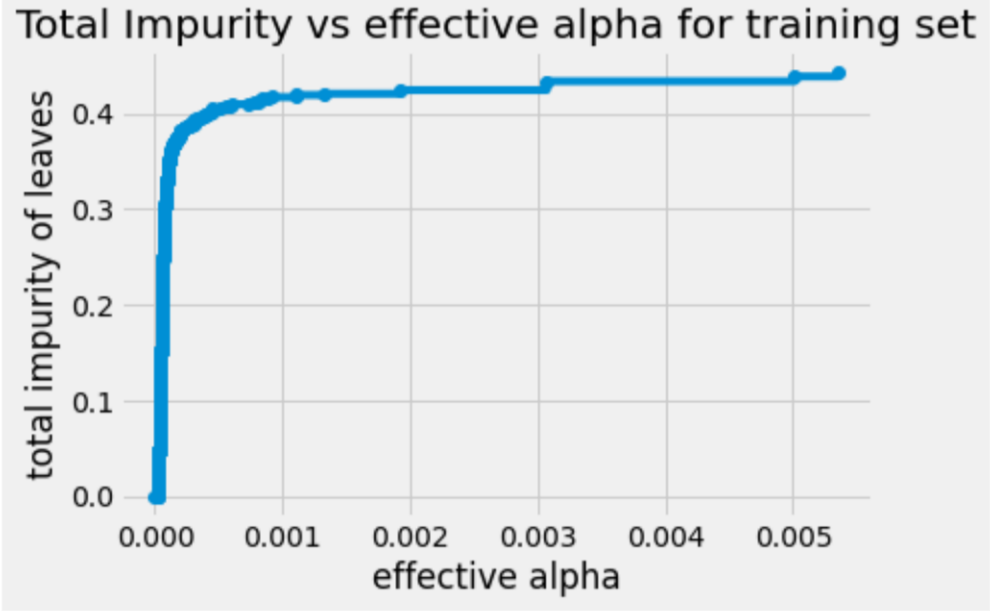
Overfitting is a condition where the machine earning model achieves a good accuracy score over the training set while the performance over the test det is poor. Subsequently, the model cannot understand and generalize the data. The cause of underfitting could be the dataset being too complex with noise and the size of the training set used to train the model being too small that the dataset couldn't make any significant learning.

**Underfitting**

Underfitting is a condition where the Machine learning model doesn't exhibit better performance on either the training or the test set. The cause of underfitting could be the dataset being too simple and the size of the training set used to train the model being too small that the dataset couldn't make any significant learning.

**Pruning**

Pruning is the process of cutting down the decision tree in case it grows into complex structure due to overfitting. The hyperparameters such as minimum samples per leaf and maximum depth are tuned to avoid overfitting.




**Effect of Pruning**

The cost complexity parameter, ccp_alpha, used in pruning technique regularizes the trees. The greater values of ccp_alpha increases the number of nodes pruned which increases the impurities of its leaves.

**GridsearchCV for hyperparameter tuning**

In order to search the hyper parameter space for the best cross validation score, The Grid search CV method is used where all the combinations of parameters are considered. Based on the evaluation, the best combination of the parameters is retained.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
params = {'max_depth': list(range(2, 10)), 'min_samples_leaf': [2, 3, 4, 5]}
grid_search_cv = GridSearchCV(DecisionTreeClassifier(random_state=42), params, verbose=1, cv=3)
grid_search_cv.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 32 candidates, totalling 96 fits
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(random_state=42),
             param_grid={'max_depth': [2, 3, 4, 5, 6, 7, 8, 9],
                         'min_samples_leaf': [2, 3, 4, 5]},
             verbose=1)
```

```
[240] grid_search_cv.best_estimator_
```

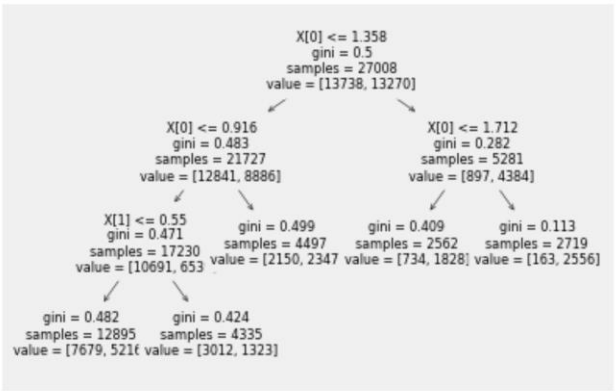
```
DecisionTreeClassifier(max_depth=9, min_samples_leaf=5, random_state=42)
```

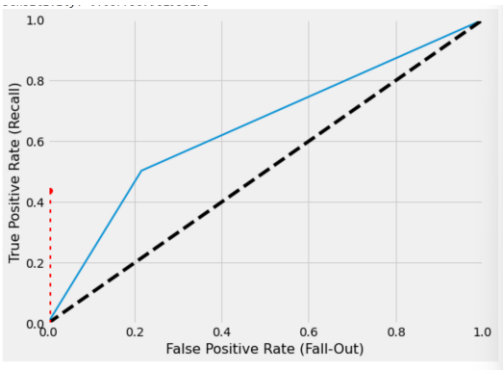
**Decision Trees**

A decision tree is a map showing the possible outcomes of a series of related choices.

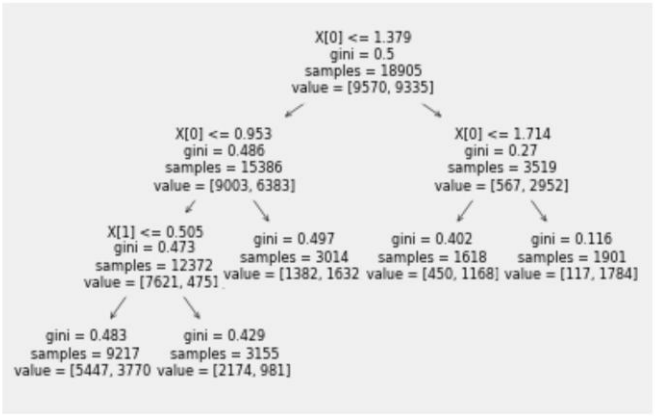
Using the best working parameters obtained as shown above, the Decision Tree algorithm is used on the dataset and the accuracy is measured. The recordings are tabulated as below:

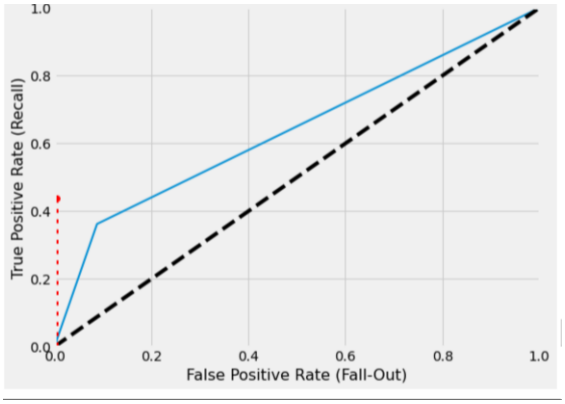
| Training Set | TN | FN | FP | TP | Accuracy | Specificity | Sensitivity |
|---|---|---|---|---|---|---|---|
| Training set | 3521 | 1011 | 2275 | 2196 | 63.50% | 60.74% | 68.47% |
| 30% of the training set into testing | 3269 | 899 | 1955 | 1980 | 64.77% | 62.57% | 68.77% |
| 60% of the training set into testing | 6193 | 2060 | 3737 | 4215 | 64.22% | 62.36% | 67.17% |



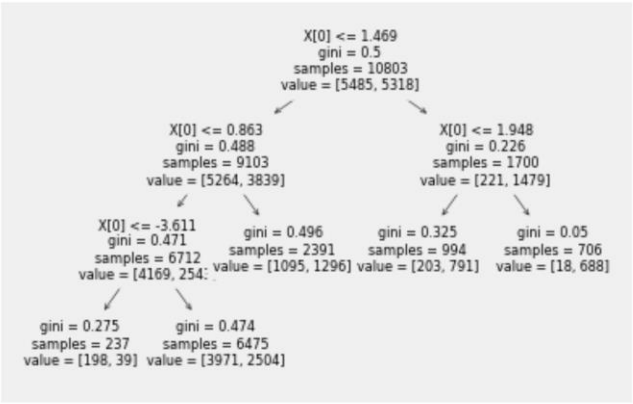



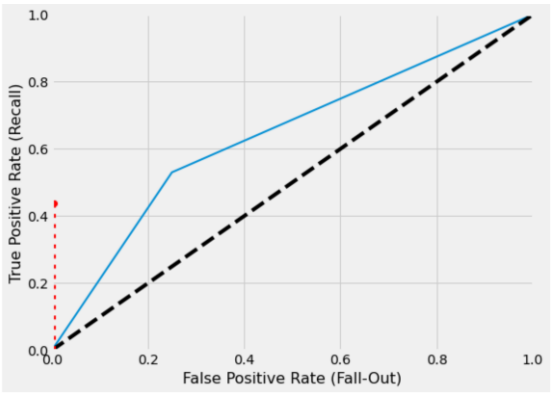

Decision Tree 1 using the testing set on J48 classifier and the corresponding ROC curve




Decision Tree 2 using the 30%of the original training set into the testing set and the corresponding ROC curve

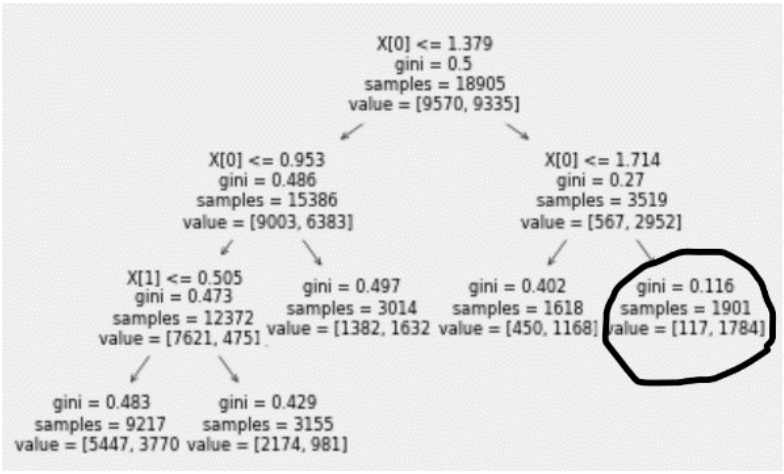



Decision Tree 3 using the 60% of the original training set into the testing set and the corresponding ROC curve

The Gini value shows how impure the information of a node is. It ranges from 0 to 0.5. Gini value equals to 0 means it is the purest node and it is a leaf.

Predicting the class of a new target value is simple using a decision tree. Based on out experiments with the decision trees, we understand that the tree using the 30% of the original dataset is the one with the greatest accuracy. By analyzing the Gini values at each node of Tree 2, the highlighted node has the least Gini value of 0.116 which indicates least impurity at that node. So, a new sample value arriving at that node has a higher probability of belonging to the class with 1784 training samples (indicated in the value) at the node.

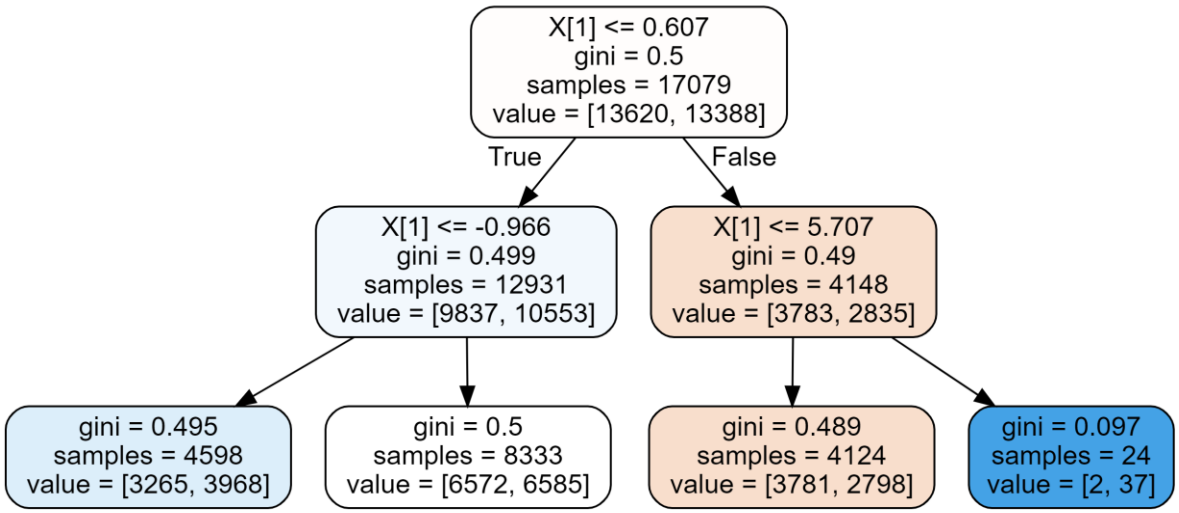


### Random Forest

A random forest is a collection of decision trees whose results are aggregated into a final result. After performing the random forest, the performance is tabulated as below:

| TN | FN | FP | TP | Accuracy | Specificity | Sensitivity |
|---|---|---|---|---|---|---|
| 4134 | 398 | 2852 | 1619 | 63.90% | 59.17% | 80.26% |

A high sensitivity percentage when compared to that of the specificity indicates that there are few false negative results.



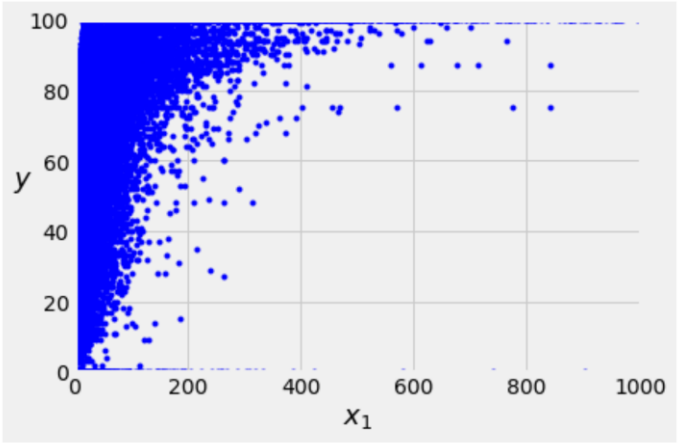

The node with the least Gini value; the node with the least impurity.

## Lab 9

In the previous lab, we implemented the decision trees with the three classifiers and the Random Forest. In Lab 9, the linear regression was implemented and tested.

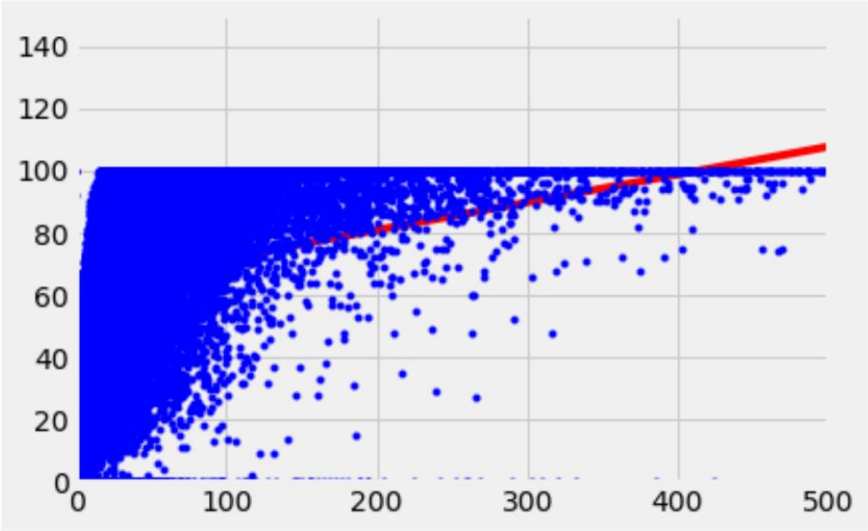
A linear regression is used to predict the value of a variable based on another variable. It will try to fist the line in these values. In our dataset ‘fire’, the ‘frp’ is selected as the value based on which the target, ‘confidence’ has to be predicted.

Then the feature and target are plotted.



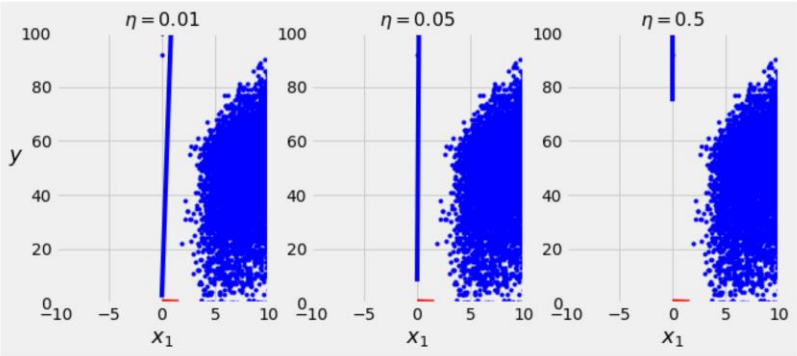

A new training set is created where the x0 = 1 for all the instances.

Accordingly, the weight, theta_best is generated and X_new and y_predict are plotted.

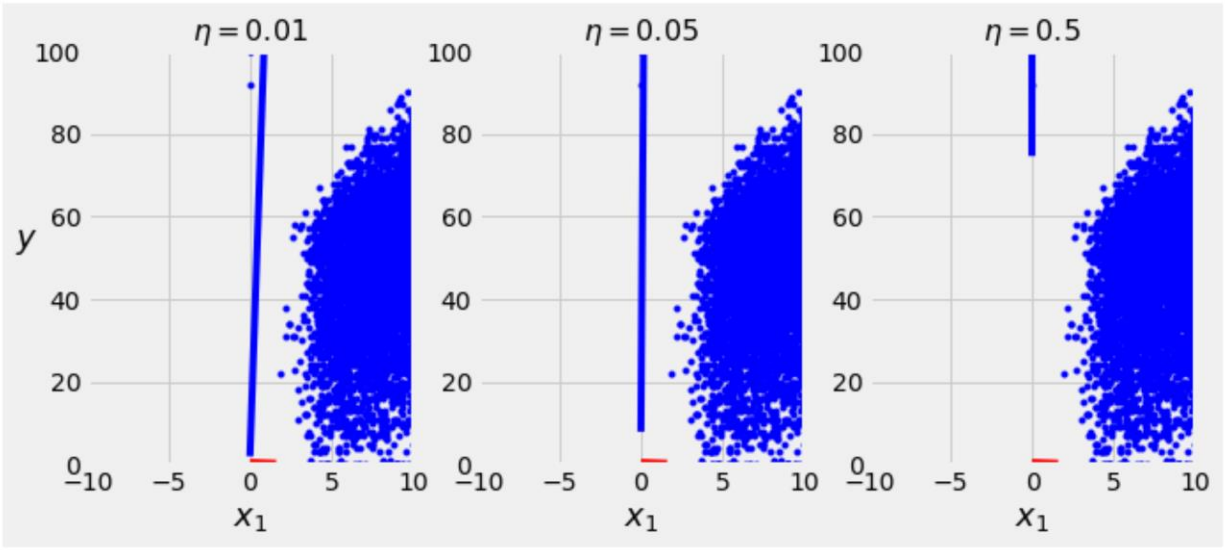


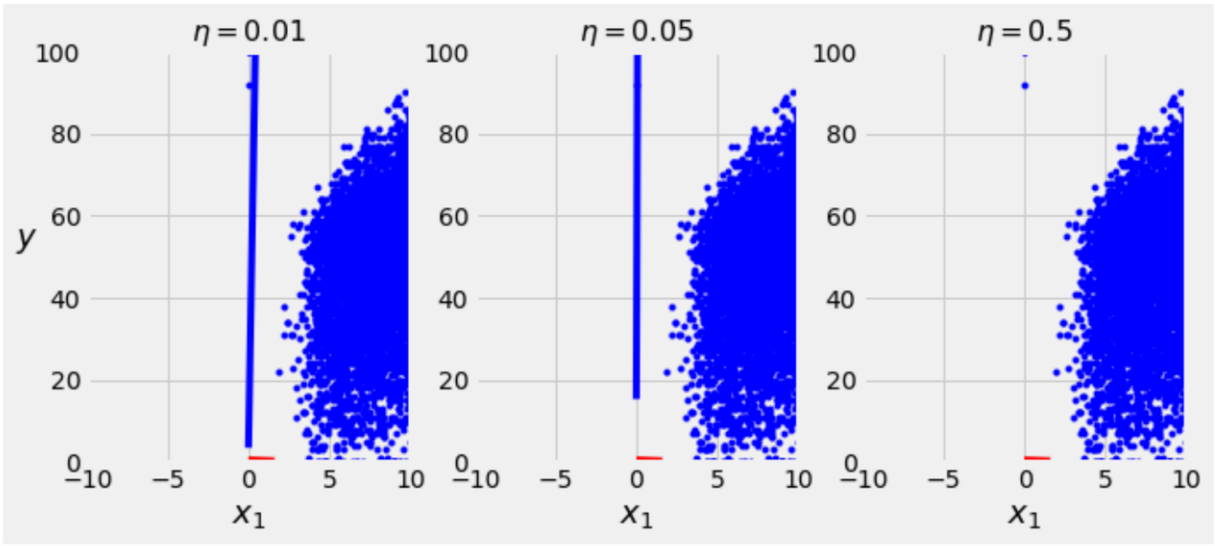

DIFFERENT REGRESSION PARAMETERS

Regression is experimented with different parameters that control the regression and results are tabulated.



A linear model classifier is created using the linear_model class. The different regression parameters like eta (learning rate), n_iterations(number of iterations), and m(batch size) are defined. A random set of weights are generated, and the experiment is repeated with different parameters. The results are tabulated.




**Batch Size = 18006, Number of Iterations = 100**

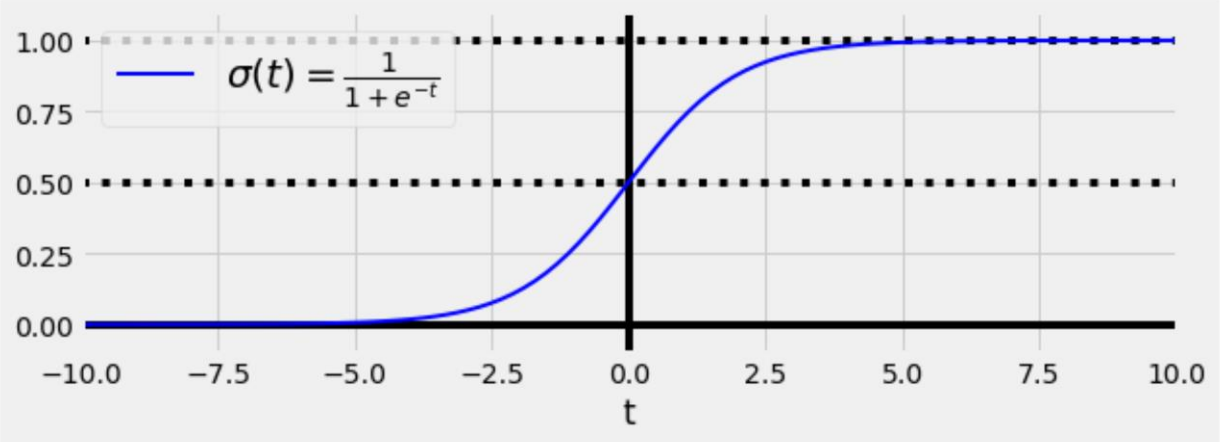
**The configuration tested is as below:**

| Batch Size | Number of Iterations | Learning Rate |
|---|---|---|
| 36011 | 100 | 0.01,0.05,0.5 |
| | 1000 | |
| 18006 | 100 | 0.01,0.05,0.5 |

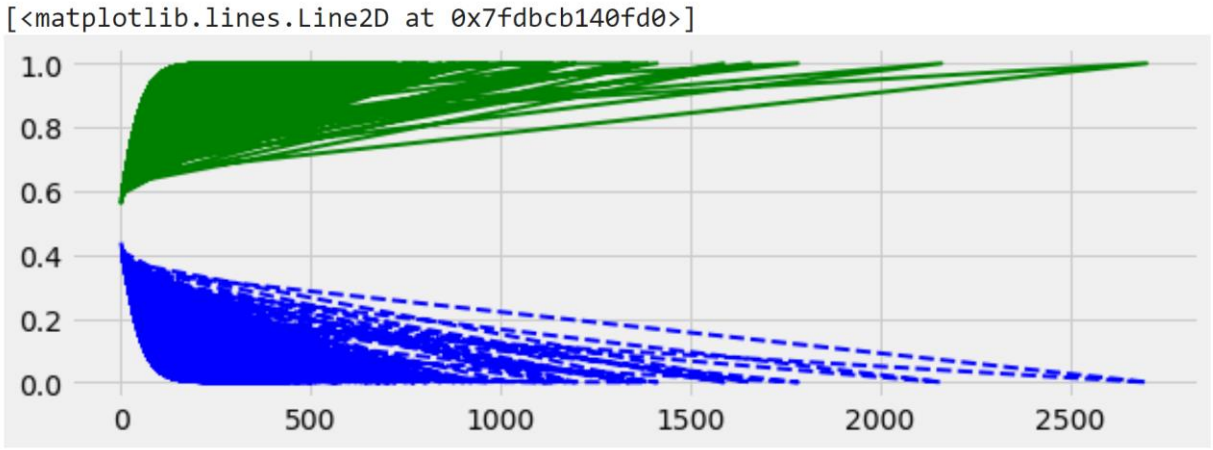
LOGISTIC REGRESSION

The logistic regression is used to solve classification problems. It achieves this by predictiong categorical outcomes.

The ideal plot which is of the sigmoid function is as given below:




The logistic regression is plotted against X_test and y_proba as shown below.

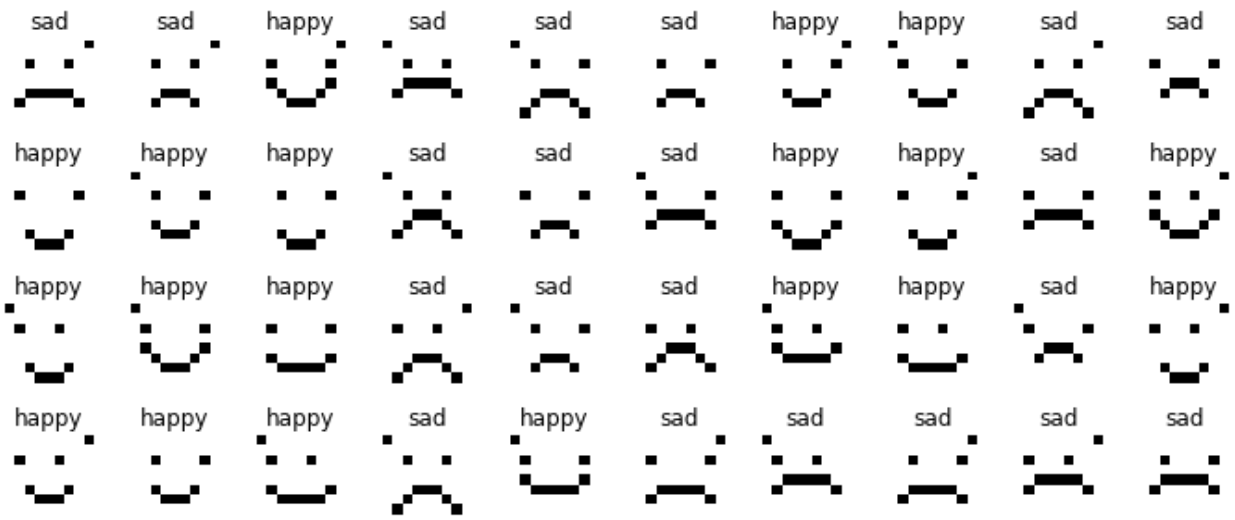



The accuracy score for the logistic regression without and with regularization parameter is 76.44

# Lab 10

For Lab 10, to demonstrate the performance of Multi Layered Perceptron, we tried to implement the MLP on our chosen Forest Fire Dataset. We first installed and imported the tensorflow python libraries. From the previous labs, we used the dataset where the target feature was converted into nominal data of binary classification. For the features, we chose our two most correlating features ‘brightness’ and ‘frp’. We split the dataset into train and test set.

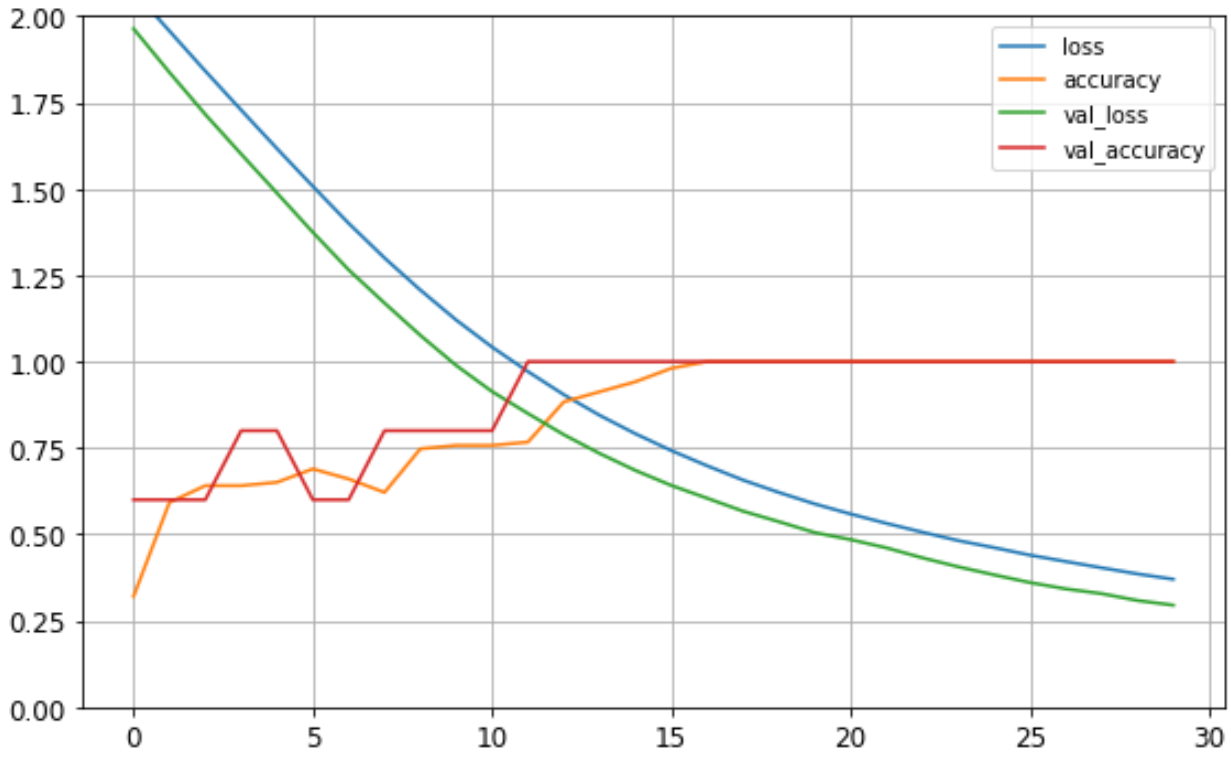
Inorder to demonstrate the performance of MLP with an image dataset, we chose the smiley dataset provided in canvas. We mapped the images with the labels and we got the following plot:




For our first run, we chose Sequential model with the activation function as ‘relu’, 3 layers and optimizer as ‘sgd’.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 81)                0
_________________________________________________________________
dense (Dense)                (None, 300)               24600
_________________________________________________________________
dense_1 (Dense)              (None, 100)               30100
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1010
=================================================================
Total params: 55,710
Trainable params: 55,710
Non-trainable params: 0
```

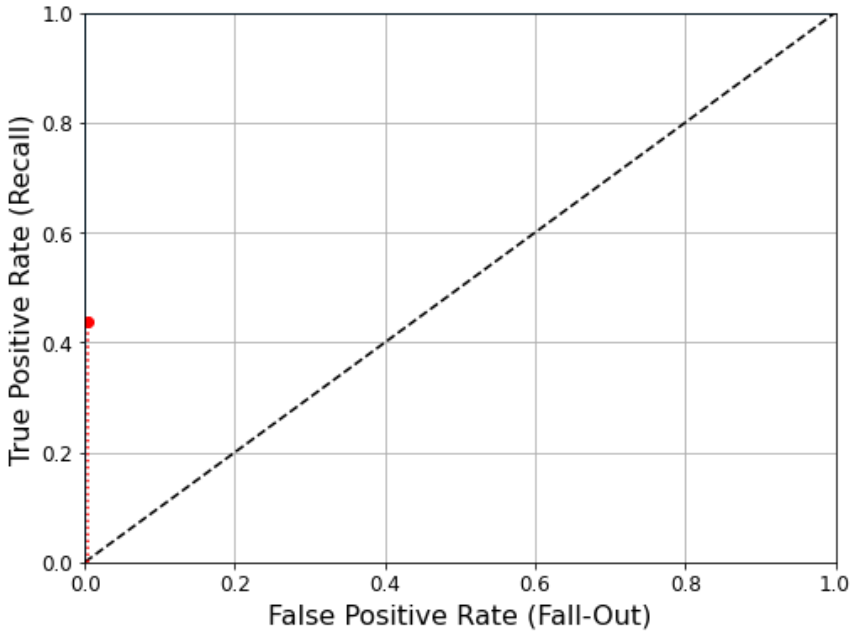
For the above configuration, the following was the learning curves plot that was obtained. The accuracy for this model was found to be 100% with an entropy loss as 35%.



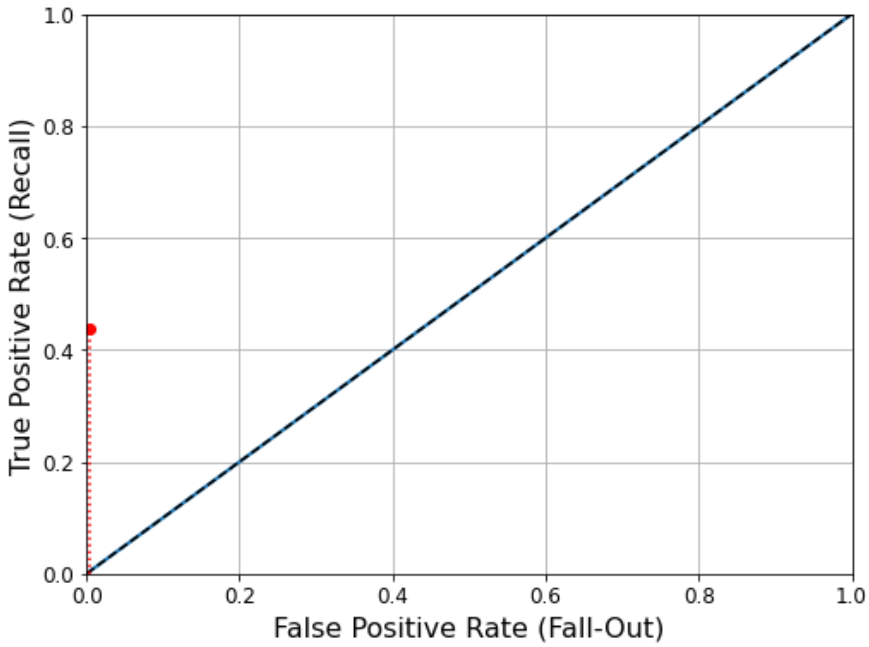

By varying the different parameters of the model, the following table was formed after repeated experimenting

| MLP | Accuracy | TP | FP | TN | FN | Sensitivity | Specificity | Precision | Recall |
|---|---|---|---|---|---|---|---|---|---|
| Architecture1 RELU, SGD, 300 batch | 100% | 18 | 0 | 18 | 0 | 1.0 | 1.0 | 1 | 1 |
| Architecture2 Sigmoid, SGD, 300 batch | 50% | 6 | 12 | 12 | 6 | 0.5 | 0.5 | 0.5 | 0.3 |
| Architecture3 Sigmoid, SGD, 500 batch | 44% | 3 | 15 | 13 | 5 | 0.37 | 0.46 | 0.37 | 0.16 |
| Arvhitecture4 Sigmoid, adam, 300 batch | 80.5% | 17 | 1 | 12 | 6 | 0.73 | 0.92 | 0.73 | 0.05 |

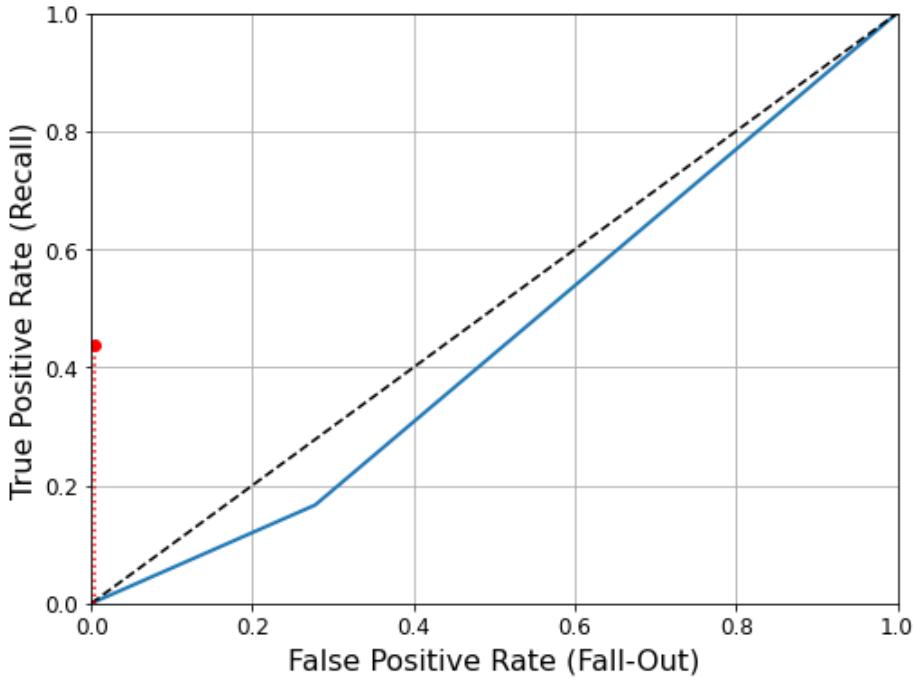
Architecture 1 ROC



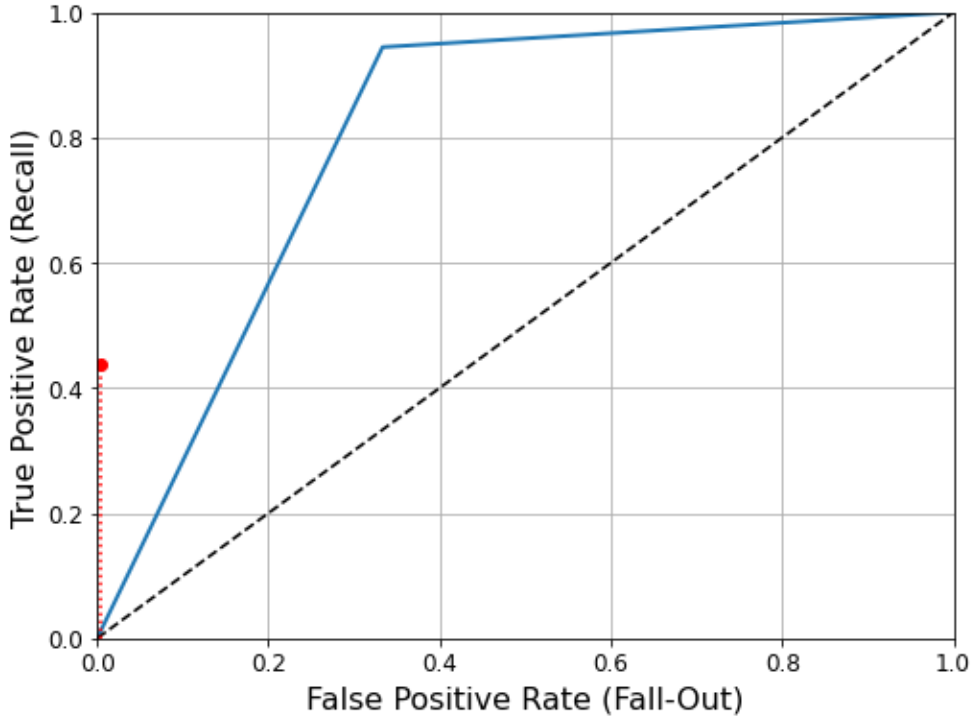

Architecture 2 ROC



Architecture 3 ROC




Architecture 4 ROC

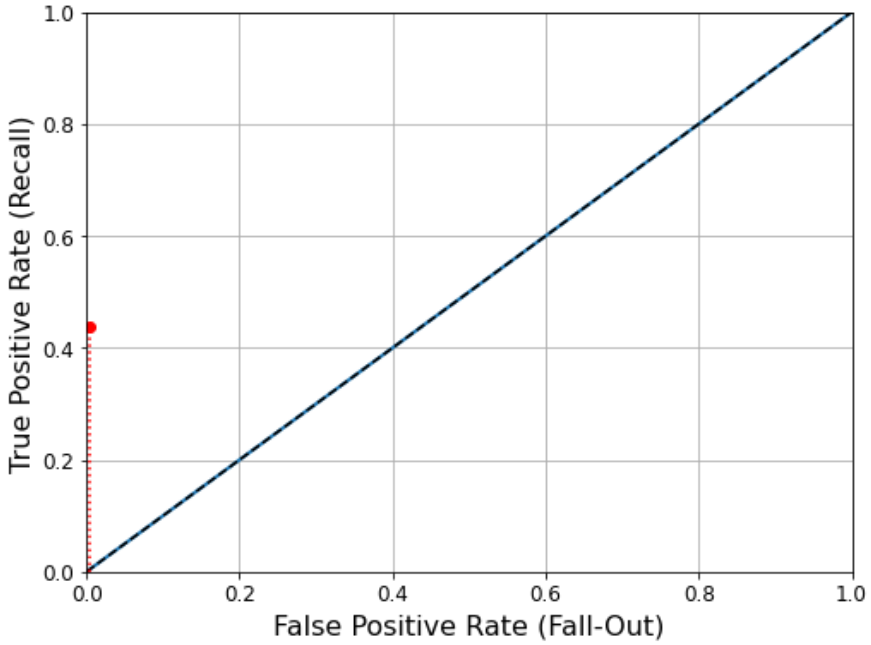


From the above table it is evident that the Architecture 1 is the best performing MLP with 100% Accuracy.

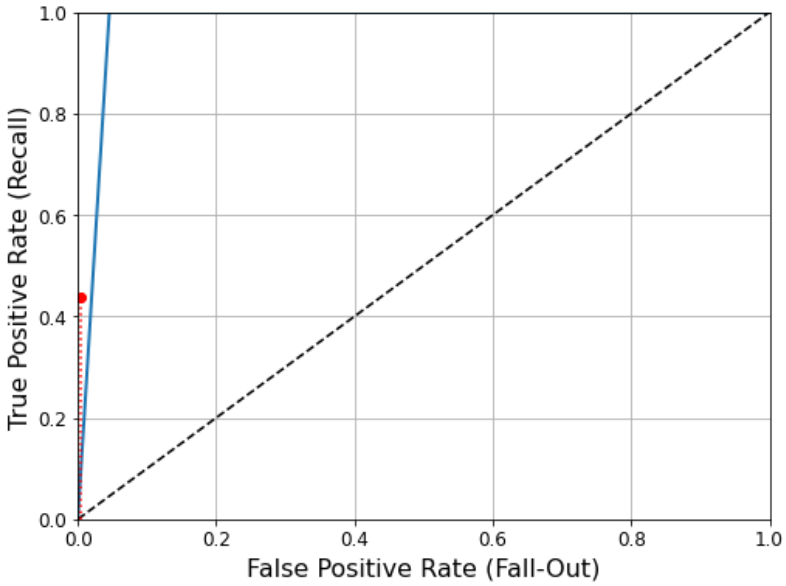
Based on the training set and test set allocation, the following table was formed for different classifiers

| MLP | Accuracy | TP | FP | TN | FN | Sensitivity | Specificity | Precision | Recall |
|---|---|---|---|---|---|---|---|---|---|
| Classifier 1 | 100% | 18 | 0 | 18 | 0 | 1.0 | 1.0 | 1 | 1 |
| Classifier 2 | 97% | 22 | 0 | 21 | 1 | 0.95 | 1.0 | 0.95 | 1 |
| Classifier 3 | 96% | 45 | 0 | 39 | 3 | 0.93 | 1.0 | 0.93 | 1 |

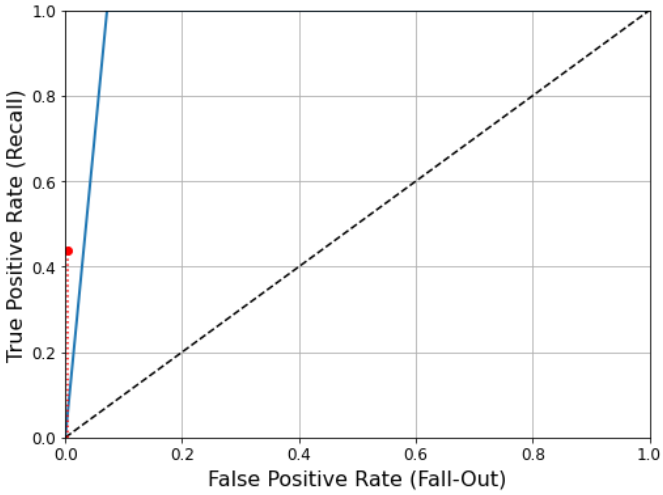
Classifier 1 ROC



Classifier 2 ROC




Classifier 3 ROC

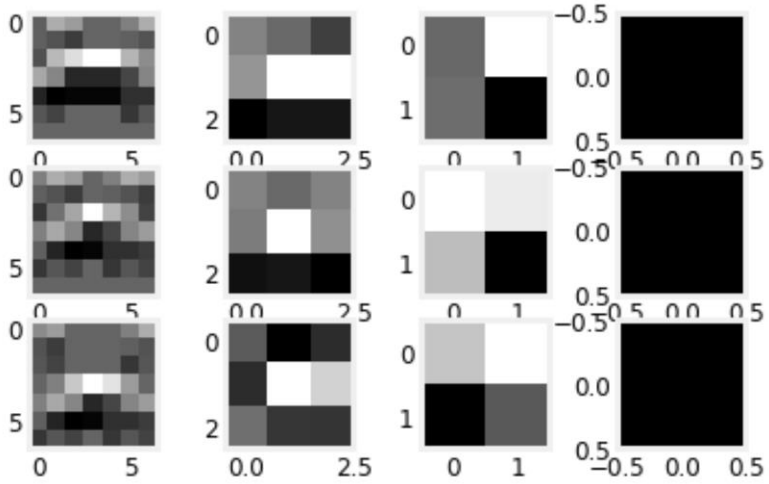



Changing the activation function from 'relu' to 'sigmoid' has a undesirable effect on the accuracy. However, with sigmoid as activation function and 'adam' as optimizer, the accuracy increased considerably. The different training sets has a minimum effect on the overall accuracy of the models when trained and fitted multiple times.

# LAB 11

CNN is a deep learning neural network which is used for processing structured images in arrays. It can run directly on the input image without the need of any preprocessing. It consists of many convolutional layers assembeled over one another which can identify the different shapes in the images.

Visualizing in CNN through different layers



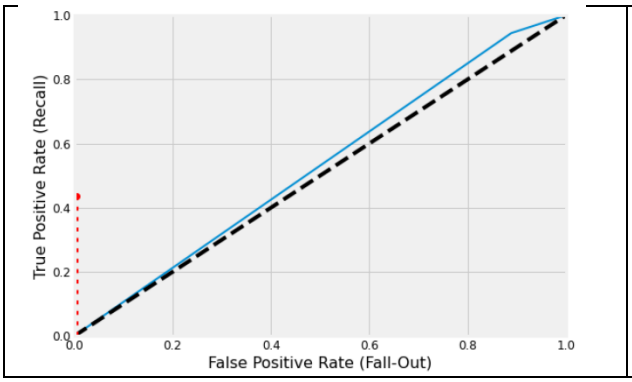
**INVESTIGATION ON DIFFERENT ARCHITECTURES OF CNN**

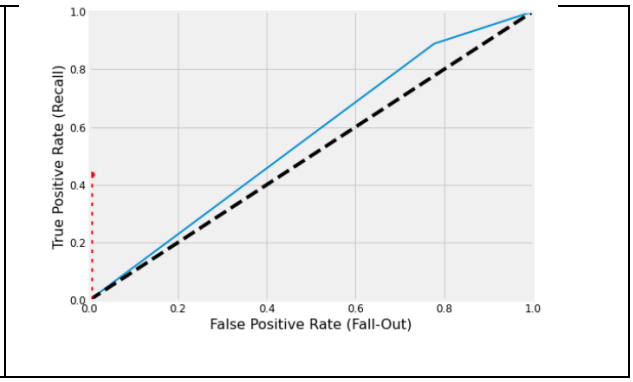
The dataset used here is the smiley consisting of happy and sad as the labels. The input nodes, output nodes and the pooling attribute have been unchanged throughout the experiment. The kernel size and the number of filters has been changed in order to create different architectures of the CNN. The architectures are tested based on the metrics of evaluation, namely, accuracy, TP, FP, TN,FN, Precision, Recall, Sensitivity, specificity and ROC cure and the results are tabulated.

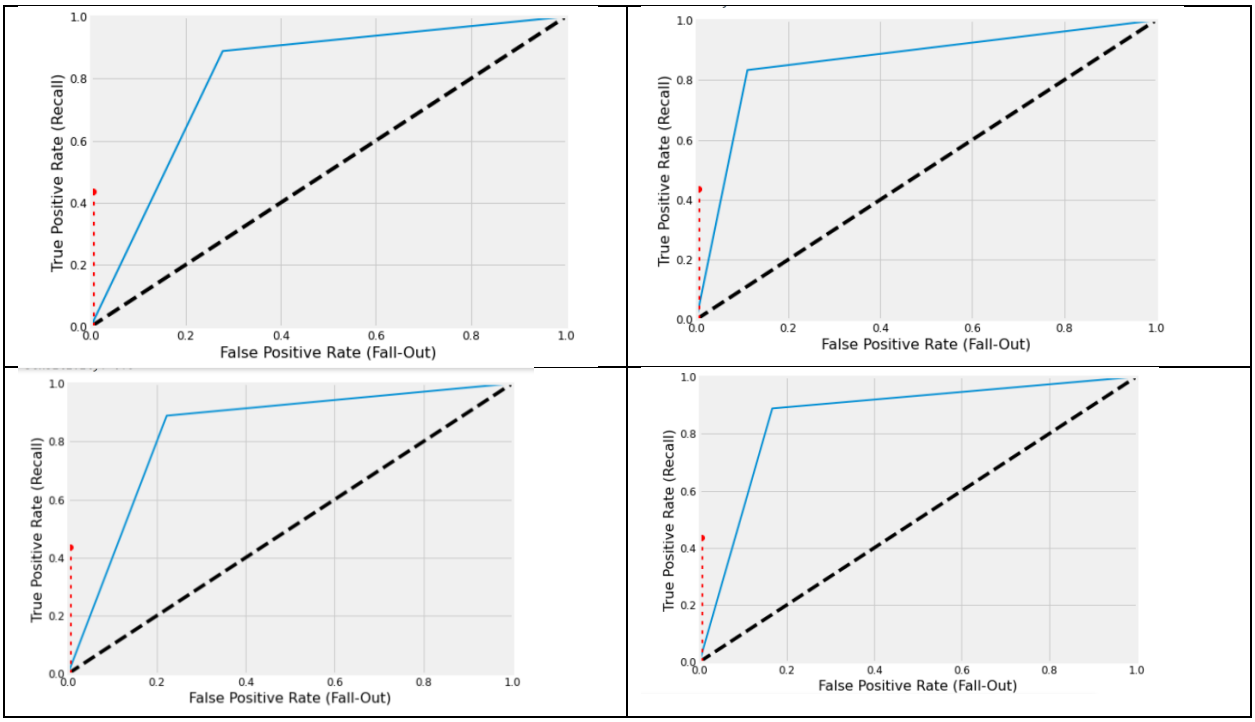
| CNN | Accuracy | TP | FP | TN | FN | Sensitivity | Specificity | Precision | Recall | Area under RoC Curve |
|---|---|---|---|---|---|---|---|---|---|---|
| Architecture 1;1st conv layer 64 filters, 3X3, 2nd conv layer, 64 filters 1X1 | 52.77 % | 17 | 1 | 2 | 16 | 51.51 % | 66.66 % | 52.77 % | 52.77 % | 52.77% |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Architecture 2;1st conv layer 64 filters, 3X3, 2nd conv layer, 32 filters, 1X1 | 55.55 % | 16 | 2 | 4 | 14 | 53.33 % | 66.66 % | 60% | 55.55 % | 55.55% |
| Architecture 3;1st conv layer 128 filters, 5X5, 2nd conv layer, 64 filters, 1X1 | 80.55 % | 16 | 2 | 13 | 5 | 76.19 % | 86.66 % | 81.42 % | 80.55 % | 80.55% |
| **Architecture 4;1st conv layer 128 filters, 5X5, 2nd conv layer, 128 filters, 1X1** | **86.11 %** | **15** | **3** | **16** | **2** | **88.23 %** | **84.21 %** | **86.22 %** | **86.11 %** | **86.11%** |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Architecture 5;1st conv layer 64 filters, 5X5, 2nd conv layer, 64 filters, 1X1 | 83.33 % | 16 | 2 | 14 | 4 | 80% | 87.5% | 83.75 % | 83.33 % | 83.33% |
| Architecture 6;1st conv layer 64 filters, 5X5, 2nd conv layer, 32 filters, 1X1 | 86.11 % | 16 | 2 | 15 | 3 | 88.23 % | 84.21 % | 86.22 % | 86.11 % | 86.11% |






**RESULTS**

From the above observation, although Architecture 6 and Architecture 4 have the same accuracy of 86.11%, architecture 6 has a higher TPR when compared to FPR making it the best architecture. Architecture 6 has the first convolution layer composed of 64 filters with a kernel size of 5 X 5 and the second layer of 32 filter with a kernel size of 1X1.

# Reference


[i] https://github.com/Gowresh-HW/DMLL-F21DL

[ii] https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

[iii] https://stackoverflow.com/questions/25792012/feature-selection-using-scikit-learn

[1] https://scikit-learn.org/stable/user_guide.html

[2] Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow , Aurelien Geron ( 2nd edition), 2019

[3] https://github.com/amitanalyste/aurelienGeron