Customer Churn Prediction

Anusha Garg, Mishty Verma

Abstract-"Customer Churn" refers to the situation where a client stops doing business with a company. Loss of clients can have a direct impact on a company's ability to grow.. Therefore, predicting the possibility of customer churn can be important for businesses. Different machine learning models can be constructed to predict if the customer would churn or not. In this project, we used a variety of machine learning approaches to construct a model to predict customer churn. Our strategy included a few key steps:

1. ***Feature Selection and Data Analysis: We identified and selected relevant features that impacted the probability of customer churn.***
2. ***Feature Engineering and Scaling: We encoded categorical data into numerical format to facilitate model training.***
3. ***Class Balancing: To address the imbalance in the dataset, we applied random oversampling to ensure a close to equal number of churn and no churn instances.***
4. ***Data Splitting: The dataset was divided into training and testing sets with an 80-20 split.***
5. ***Model Selection: We implemented and compared multiple models,namely linear regression, logistic regression, k-nearest neighbors (KNN),decision tree, and random forest.***
6. ***Model Evaluation: We assessed the performance of each model based on accuracy to determine the most effective approach for predicting customer churn.***

The accuracy of the random forest model was the highest, at about 90%, whereas the other models had a significantly poorer performance of about 70%-80%.

Keywords- Churn prediction, Machine Learning, Feature Selection, Linear Regression, Logistic Regression, K-nearest neighbors, Random forest, Model Evaluation, Accuracy.

Introduction

Customer churn, the phenomenon of a customer switching to another company or stopping business with a company, can directly affect the growth of a business. High churn rates indicate poor services and/or low quality products . Figuring out the causes of high churn rate can directly affect the business growth positively and help the business retain more customers.

Several machine learning algorithms can be applied to predict customer churn. The process of building the models was divided into the following phases:

1. Data Collection: For our study, we utilized the Telco Customer Churn dataset from Kaggle (https://www.kaggle.com/datasets/blastchar/telco-customer-churn). This dataset includes information about a telecom company's customers.

The columns in the dataset are as follows:

- customerID: unique value
- Gender: Male or Female
- SeniorCitizen: yes or no
- Partner: yes o no
- Dependents: yes or no
- Tenure: in months

- PhoneService: Yes or no
- MultipleLines: No phone service , yes or no
- InternetService: DSL FiberOptics or no service
- OnlineSecurity: Yes or no or no internet service
- OnlineBackup: Yes or no No internet
- DeviceProtection: Yes or no or no internets service
- TechSupport: Yes or no or no internet service
- StreamingTV: Yes or no or no internet service
- StreamingMovies: Yes or no or no internet service
- Contract: month-to-month, one year, two year
- PaperlessBilling: Yes or No
- PaymentMethod: electronic, mailcheck, bank transfer, credit card
- MonthlyCharges: Dollar
- TotalCharges: Dollar
- Churn: yes or no

2. Feature Selection and Data Analysis: Irrelevant attributes were deleted, and the correlation between different attributes and churn was analyzed through bar graphs.
3. Feature Engineering and Scaling: All binary values (yes/no) were converted to 0 and 1, and for attributes with more than two categories, one-hot encoding was applied.
4. Class Balancing: The initial dataset showed a significant imbalance, with far more instances of non-churn compared to churn. To balance the classes, random oversampling was applied trying to achieve an equal representation of both churn and non-churn instances. This step was essential for preventing model bias towards the majority class.
5. Data Splitting: All null values were appropriately handled, and a randomness seed of 42 was used to ensure consistency in the final predictions. An 80-20 train-test split was performed on the dataset, with 80% of the data used for training and 20% for testing. This split was chosen because our dataset contained fewer than 10,000 values, and an 80-20 split is generally more efficient for smaller datasets.
6. Model Selection: Multiple machine learning models were implemented and compared, namely linear regression, logistic regression, k-nearest neighbors (KNN), and random forest. Each model brought unique strengths to the prediction task.
7. Model Evaluation: The performance of each model was assessed based on accuracy and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). AUC provides a comprehensive measure of a model's ability to distinguish between classes, offering insight into the trade-offs between sensitivity and specificity.

Proposed method

Various machine learning algorithms, namely Linear Regression, Logistic Regression, K-Nearest Neighbors, Decision Tree, and Random Forest, have been compared to determine the best algorithm for binary classification. First linear regression was used to get the probability of churning, followed by the classification algorithms. The accuracy of every model has been evaluated.Furthermore, the AUC-ROC curve for each algorithm is used to evaluate its performance.

Linear regression:

Linear regression is a simple statistical method for modeling the relationship between a dependent variable and one or more independent variables. In a linear regression mode, we try to minimize the difference between the expected and actual values by fitting a linear equation to the observed data. Linear regression predicts continuous outcomes. The of equation linear regression model for linear regression is:

$$Y= B+a_1x_1+a_2x_2+.....a_nx_n +E_0$$

Where,Y is the dependent variable, E_0 is the error,$x_1,x_2,\ldots x_n$ are the independent variables, B is the intercept,and $a_1,a_2\ldots a_n$ are weights.The value of the dependent variable is calculated using the value of these independent variables. Weights represent the expected change in the dependent variable (Y) for a one-unit increase in the corresponding independent variable ($x_1,x_2,\ldots x_n$), holding all other variables constant.

Mean Squared Error (MSE)

Mean Squared Error (MSE) is a measure of the average squared difference between the observed actual outcomes and the outcomes predicted by the model. It is used to evaluate the accuracy of a regression model. The formula for MSE is:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where, n is the number of observations, y_i is the actual value of dependent variable ,

$\hat{y}_i$ is the predicted value of the dependent variable, $\bar{y}$ is the mean of actual values.

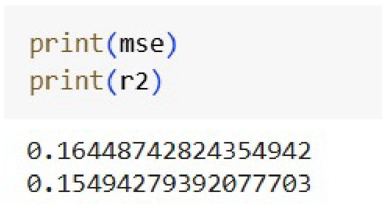
A lower MSE indicates a better fit of the model to the data.

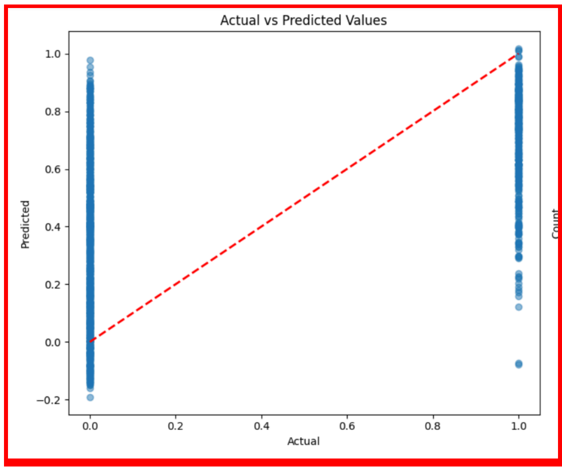
Coefficient of determination

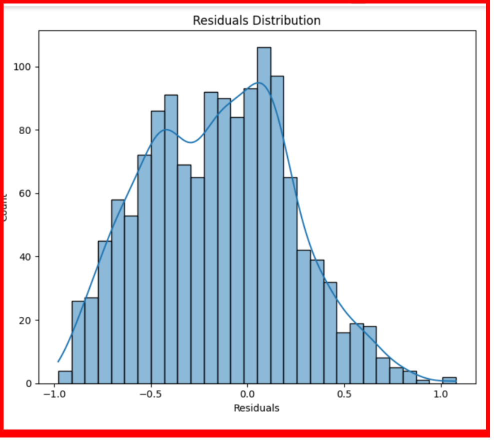
Coefficient of determination(R^2) is a statistical measure in linear regression that assesses how well the regression model explains the variability of the dependent variable. It provides a goodness of fit of the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (yi - \hat{y}i)^2}{\sum_{i=1}^{n} (yi - \bar{y})^2}$$

For our linear regression model, the values of MSE and R^2 were as follows:


```
print(mse)
print(r2)
```
```
0.16448742824354942
0.15494279392077703
```





For classification, the following were evaluated to evaluate the performance on the models.

Precision

- **Definition**: Precision measures the proportion of true positive predictions among all positive predictions made by the model. It indicates how many of the predicted positive cases are actually positive. Precision is important when the cost of false positives is high.

Recall

- **Definition**: Recall measures the proportion of true positive predictions among all actual positive cases. It reflects how well the model identifies positive cases. Recall is crucial when the cost of false negatives is high, meaning you want to capture as many positive cases as possible.

F1-Score

- **Definition**: The F1-score is the harmonic mean of precision and recall. It balances the trade-off between precision and recall, providing a single metric that considers both false positives and false negatives. The F1-score is useful when you need a balance between precision and recall, especially in cases with imbalanced datasets.

Support

- **Definition**: Support refers to the number of actual occurrences of each class in the dataset. It provides context for the precision, recall, and F1-score metrics by showing how many instances of each class were present in the data.

Accuracy

- **Definition**: Accuracy measures the proportion of correctly classified instances (both true positives and true negatives) among all instances in the dataset. It indicates the overall effectiveness of the model but may not be informative in cases of imbalanced classes.

Accuracy= $\frac{TP+TN}{TP+TN+FP+FN}$ where TP is True Positive, TN is True Negative, FP is false positive and FN is False Negative.

Macro Average

- **Definition**: Macro average calculates the metric (such as precision, recall, or F1-score) independently for each class and then takes the average of these values. It treats all classes equally, regardless of their support, and provides a measure of the model's performance across all classes.

  Macro Average=$\frac{1}{c}\sum_{i=1}^{c} Metric$ where c is the number of classes.

Weighted Average

- **Definition**: Weighted average calculates the metric (such as precision, recall, or F1-score) for each class and then averages these values, weighted by the support (number of instances) of each class. It takes into account the imbalance in class distribution and provides an overall measure of performance, considering the prevalence of each class.

Weighted Average=$\frac{\sum_{i=1}^{c} (Support)iXMetric}{\sum_{i=i}^{c} (Support)}$

Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

1. **Logistic regression**

   Logistic regression uses sigmoid function to map values to different classes. Unlike linear regression, which gives a continuous outcome, logistic regression predicts classes of values. The equation for logistic regression is

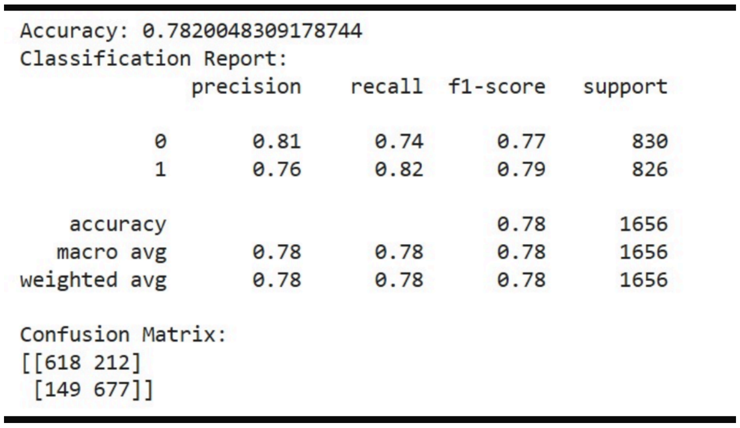
   logit(p) = ln($\frac{p}{1-p}$) $=$ $\beta_0 + \beta_1 x_1 + \ldots\ldots + \beta_n x_n$

   Where:

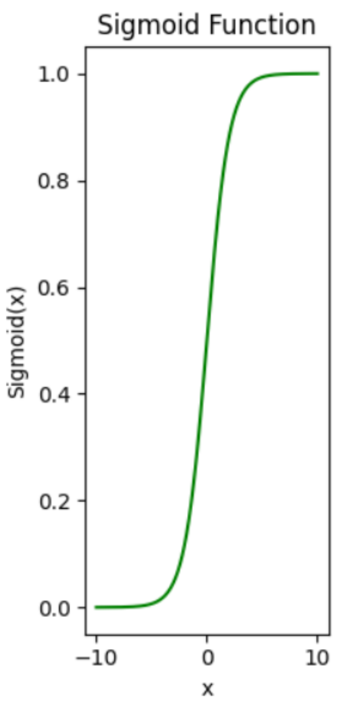
p is the probability of the outcome of interest(probability of churn), $x_1, x_2, \ldots x_n$ are the independent variables, β_0 is the intercept, $\beta_1, \ldots \beta_n$ are coefficients of the independent variables.

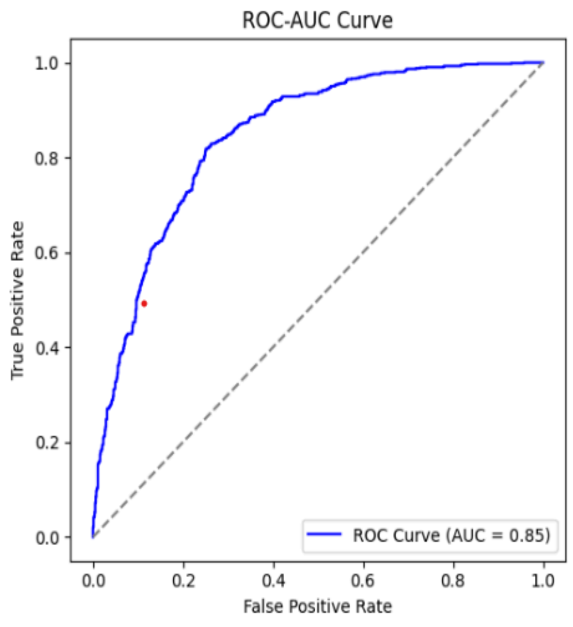
The logistic function (or sigmoid function) is used to transform the logit(linear combination of inputs) into a probability:

$$p = \frac{1}{1+e^{-(\beta_0 + \beta_1 x_1 + \ldots\ldots + \beta_n x_n)}}$$



```
Accuracy: 0.7820048309178744
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.74      0.77       830
           1       0.76      0.82      0.79       826

    accuracy                           0.78      1656
   macro avg       0.78      0.78      0.78      1656
weighted avg       0.78      0.78      0.78      1656

Confusion Matrix:
[[618 212]
 [149 677]]
```

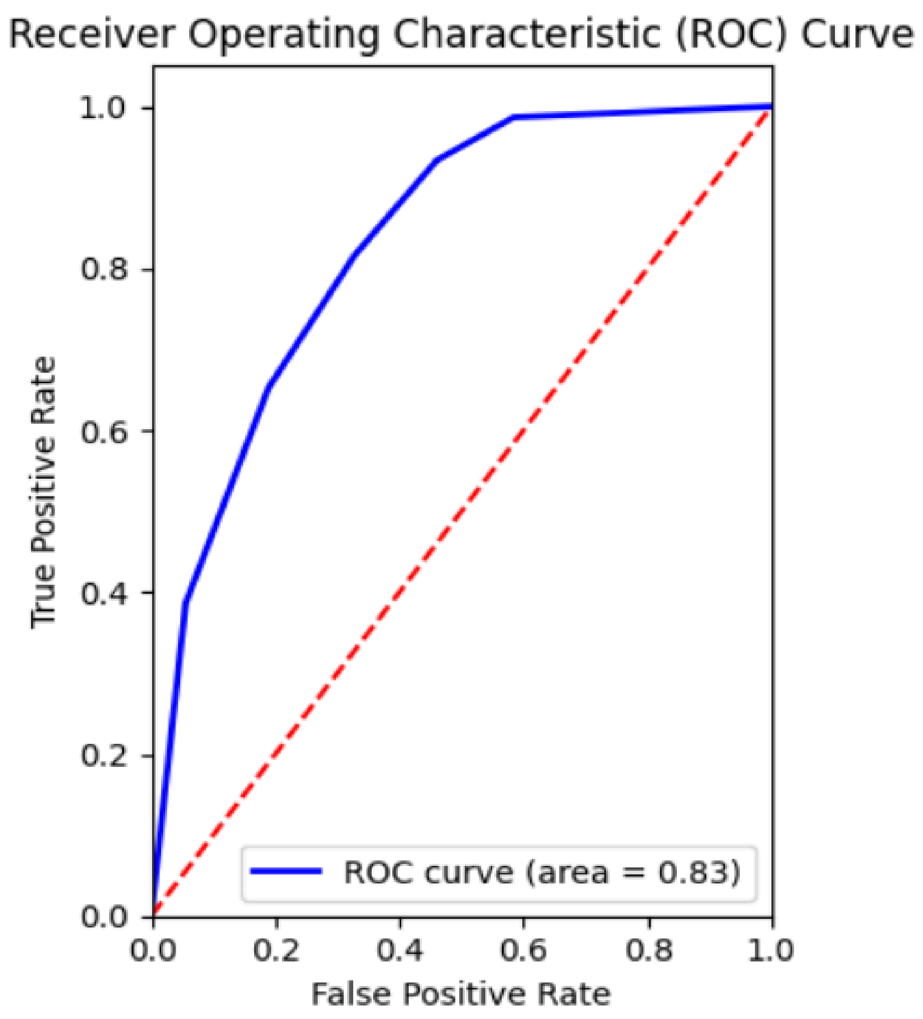




K nearest neighbors

The K-Nearest Neighbors (KNN) algorithm predicts the class of a given data point by comparing the dependent values of its K closest points.If K equals 3, then the algorithm will consider the 3 nearest neighbors of the value point and classify it in the majority class of the 3 neighbors. Selecting K as an odd number ensures that there's a definitive majority class.

```
 [132 674]]
```



Distance Metrics

Euclidean Distance : The most common metric, calculated as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Manhattan Distance: Also knowns as L1 distance, calculated as:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Minkowski Distance : A generalized distance metric that includes both Euclidean and Manhattan distances as special cases:

$$d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$$

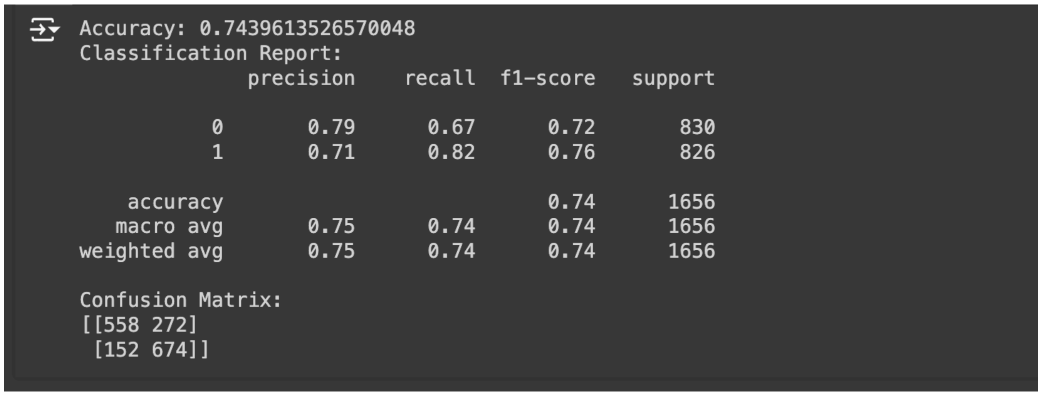
Hamming distance : Hamming distance is a metric used to measure the difference between two strings of equal length. It counts the number of positions at which the corresponding elements are different.

$$d(x, y) = \sum_{i=1}^{n} 1(x_i \neq y_i)$$

Chebyshev distance : Chebyshev distance is a metric used to measure the maximum absolute difference between the coordinates of two points.

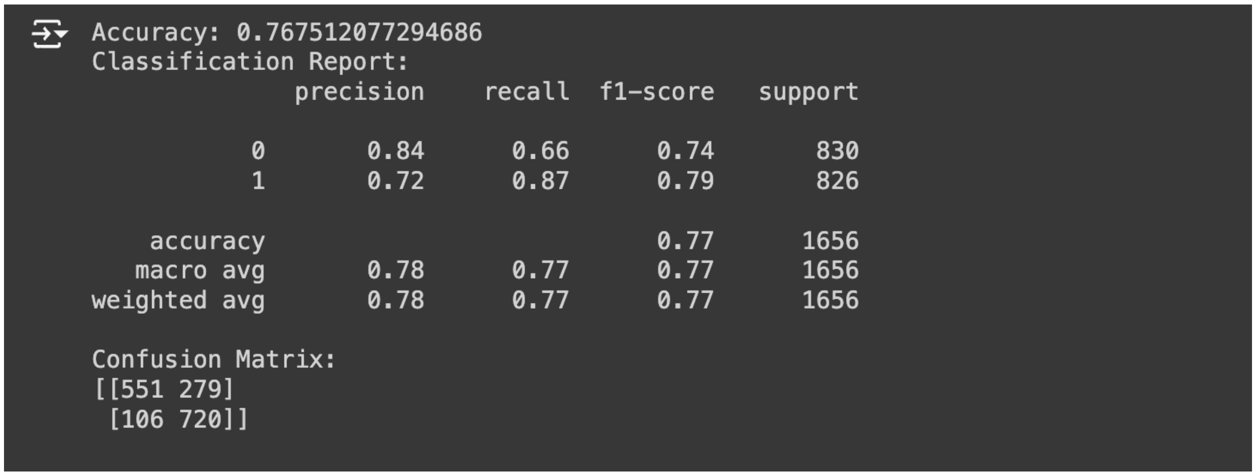
$$d(x, y) = max_{i=1}^{n} |x_i - y_i|$$

Euclidean distance:
k=5



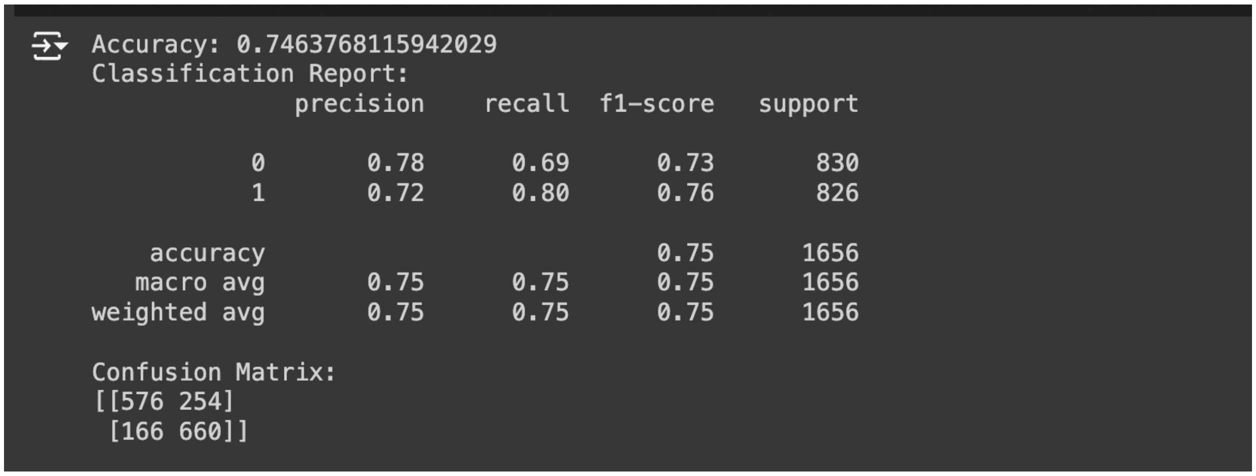
```
Accuracy: 0.7439613526570048
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.67      0.72       830
           1       0.71      0.82      0.76       826

    accuracy                           0.74      1656
   macro avg       0.75      0.74      0.74      1656
weighted avg       0.75      0.74      0.74      1656

Confusion Matrix:
[[558 272]
 [152 674]]
```

k=3



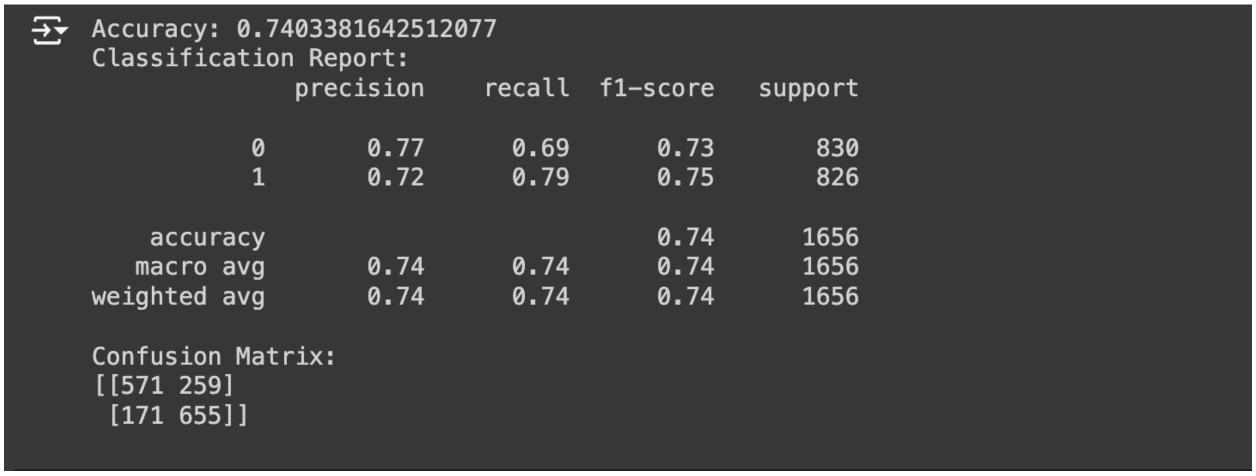
```
Accuracy: 0.767512077294686
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.66      0.74       830
           1       0.72      0.87      0.79       826

    accuracy                           0.77      1656
   macro avg       0.78      0.77      0.77      1656
weighted avg       0.78      0.77      0.77      1656

Confusion Matrix:
[[551 279]
 [106 720]]
```

k=7



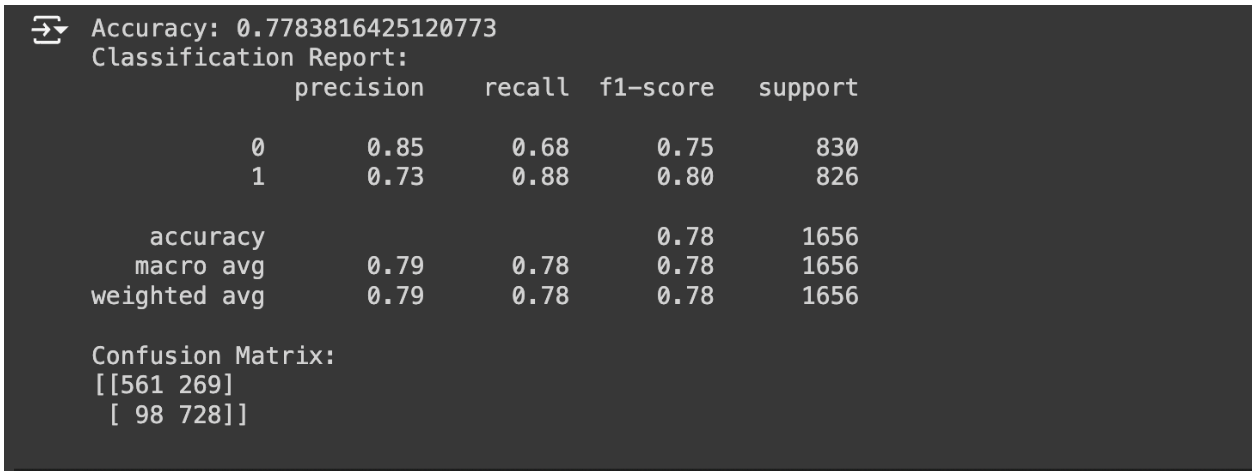
```
Accuracy: 0.7463768115942029
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.69      0.73       830
           1       0.72      0.80      0.76       826

    accuracy                           0.75      1656
   macro avg       0.75      0.75      0.75      1656
weighted avg       0.75      0.75      0.75      1656

Confusion Matrix:
[[576 254]
 [166 660]]
```

k=9



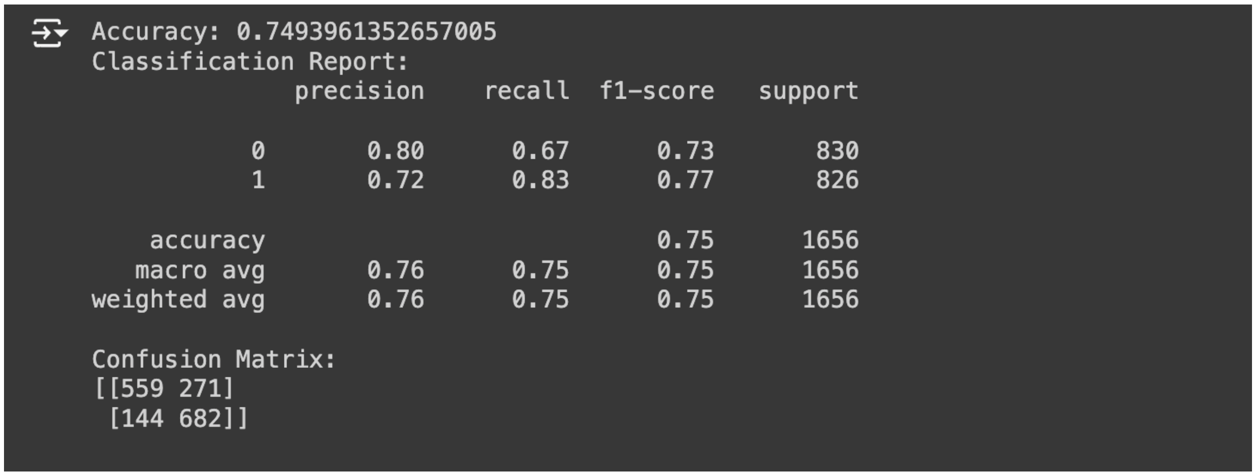
```
Accuracy: 0.7403381642512077
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       830
           1       0.72      0.79      0.75       826

    accuracy                           0.74      1656
   macro avg       0.74      0.74      0.74      1656
weighted avg       0.74      0.74      0.74      1656

Confusion Matrix:
[[571 259]
 [171 655]]
```

Manhattan
k=3



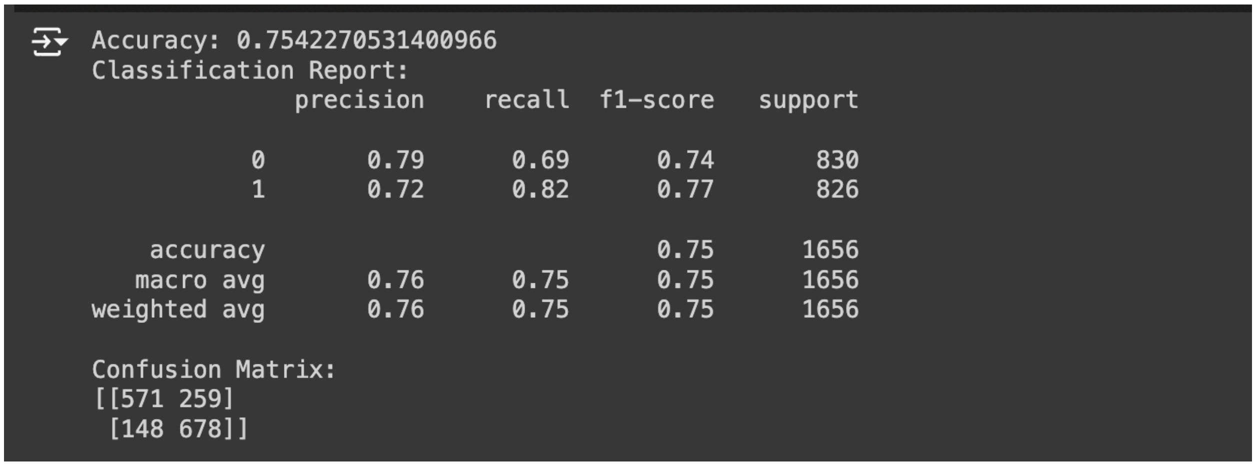
```
Accuracy: 0.7783816425120773
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.68      0.75       830
           1       0.73      0.88      0.80       826

    accuracy                           0.78      1656
   macro avg       0.79      0.78      0.78      1656
weighted avg       0.79      0.78      0.78      1656

Confusion Matrix:
[[561 269]
 [ 98 728]]
```

k=5



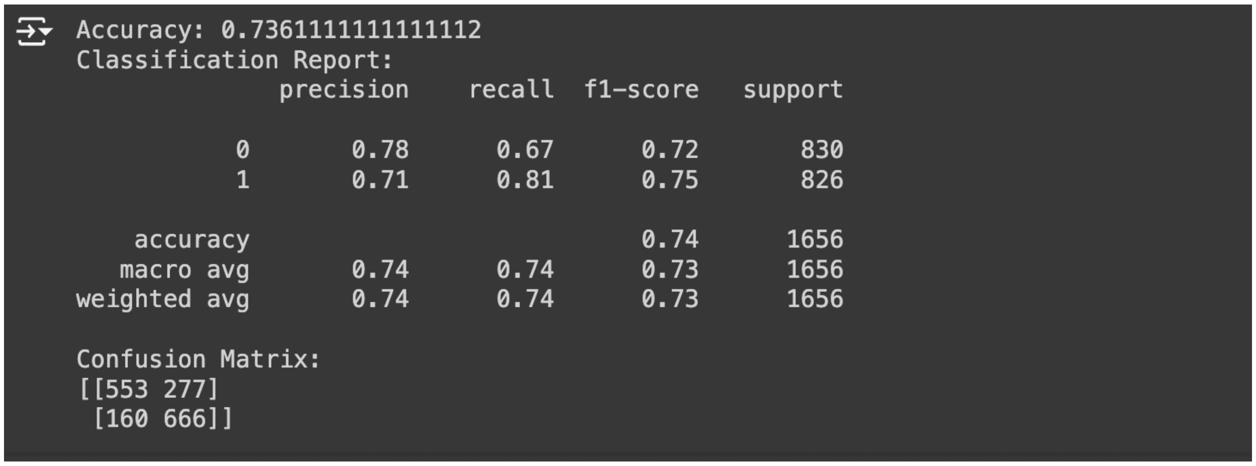
```
Accuracy: 0.7493961352657005
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.67      0.73       830
           1       0.72      0.83      0.77       826

    accuracy                           0.75      1656
   macro avg       0.76      0.75      0.75      1656
weighted avg       0.76      0.75      0.75      1656

Confusion Matrix:
[[559 271]
 [144 682]]
```

k=7



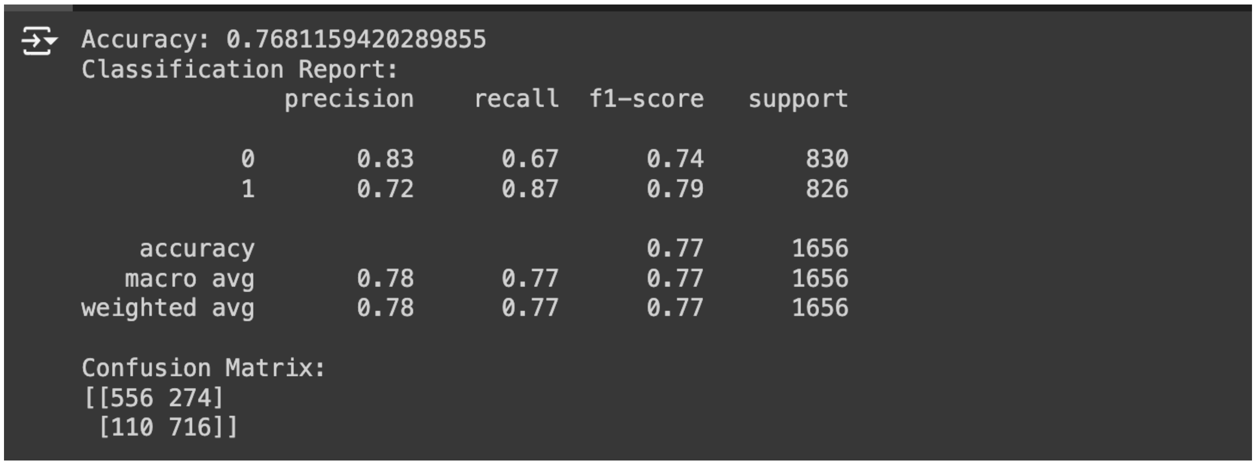
```
Accuracy: 0.7542270531400966
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.69      0.74       830
           1       0.72      0.82      0.77       826

    accuracy                           0.75      1656
   macro avg       0.76      0.75      0.75      1656
weighted avg       0.76      0.75      0.75      1656

Confusion Matrix:
[[571 259]
 [148 678]]
```

k=9



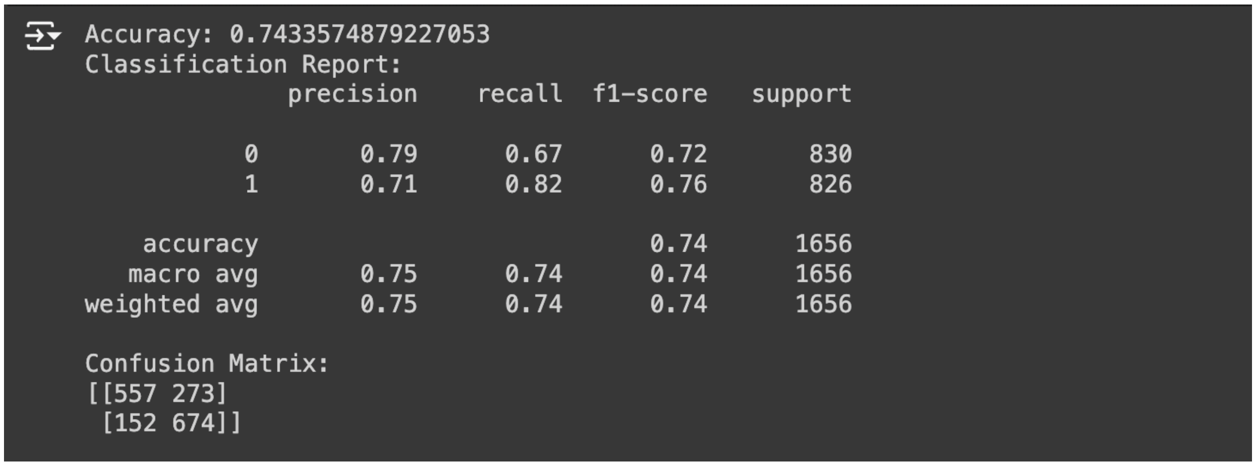
```
Accuracy: 0.7361111111111112
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.67      0.72       830
           1       0.71      0.81      0.75       826

    accuracy                           0.74      1656
   macro avg       0.74      0.74      0.73      1656
weighted avg       0.74      0.74      0.73      1656

Confusion Matrix:
[[553 277]
 [160 666]]
```

Minkowski
k=3 p=3



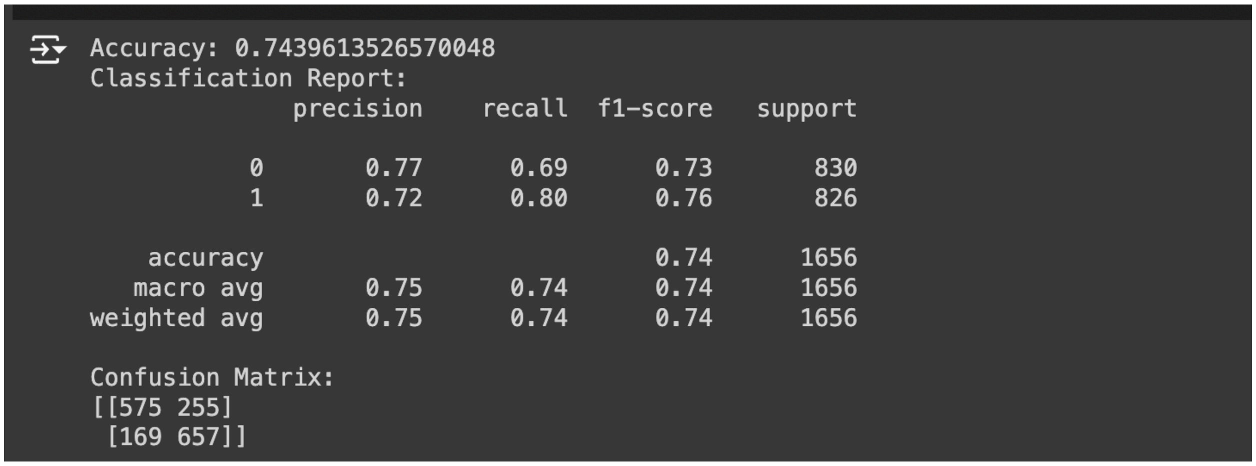
```
Accuracy: 0.7681159420289855
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.67      0.74       830
           1       0.72      0.87      0.79       826

    accuracy                           0.77      1656
   macro avg       0.78      0.77      0.77      1656
weighted avg       0.78      0.77      0.77      1656

Confusion Matrix:
[[556 274]
 [110 716]]
```

k=5 p=3



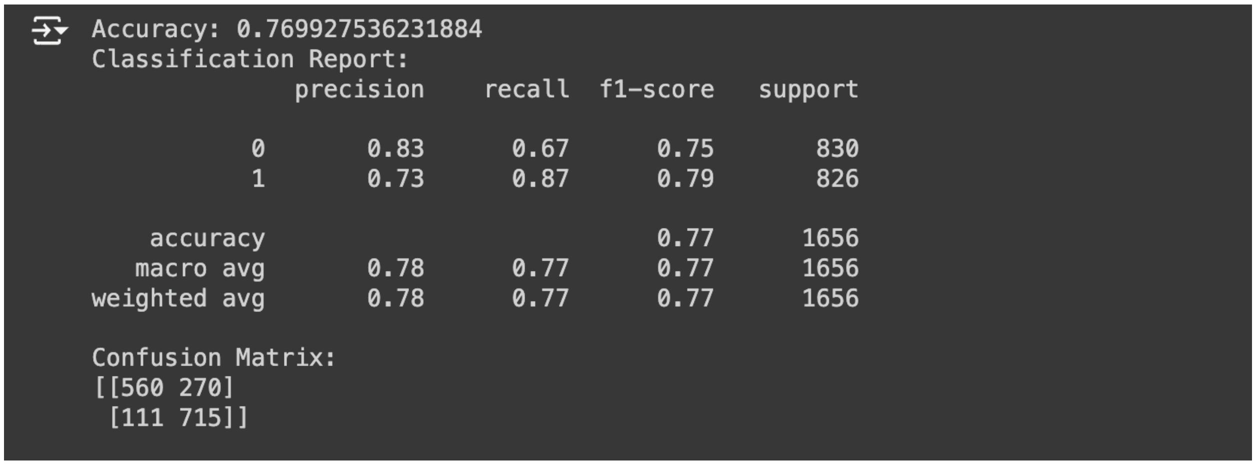
```
Accuracy: 0.7433574879227053
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.67      0.72       830
           1       0.71      0.82      0.76       826

    accuracy                           0.74      1656
   macro avg       0.75      0.74      0.74      1656
weighted avg       0.75      0.74      0.74      1656

Confusion Matrix:
[[557 273]
 [152 674]]
```

k=7 p=3



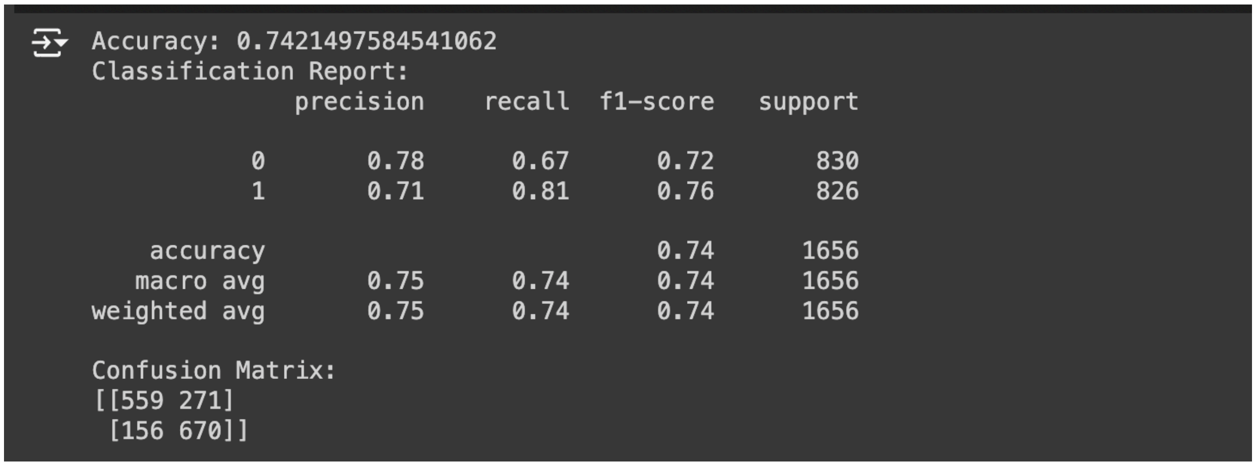
```
Accuracy: 0.7439613526570048
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       830
           1       0.72      0.80      0.76       826

    accuracy                           0.74      1656
   macro avg       0.75      0.74      0.74      1656
weighted avg       0.75      0.74      0.74      1656

Confusion Matrix:
[[575 255]
 [169 657]]
```

k=3 p=5



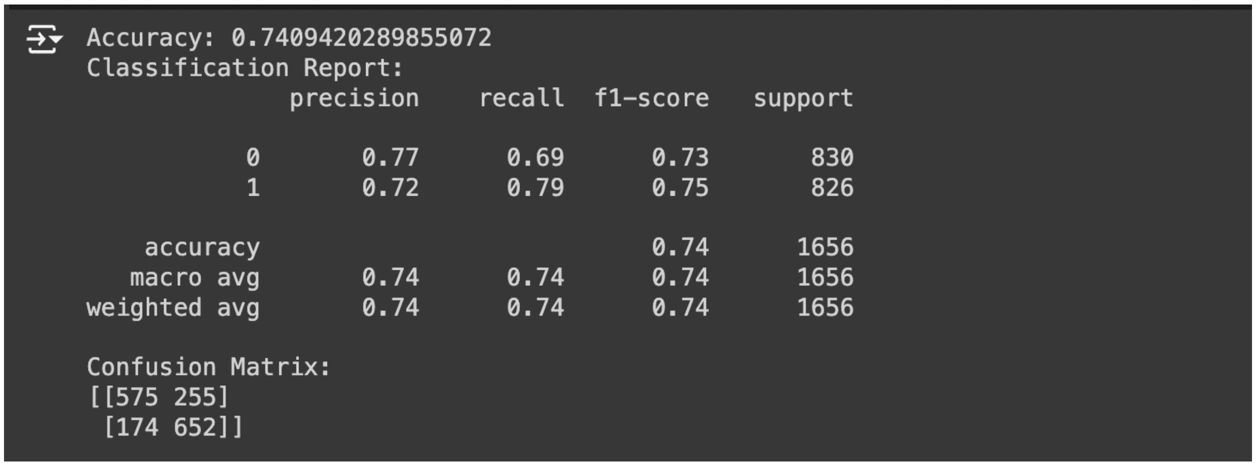
```
Accuracy: 0.769927536231884
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.67      0.75       830
           1       0.73      0.87      0.79       826

    accuracy                           0.77      1656
   macro avg       0.78      0.77      0.77      1656
weighted avg       0.78      0.77      0.77      1656

Confusion Matrix:
[[560 270]
 [111 715]]
```

k=5 p=5



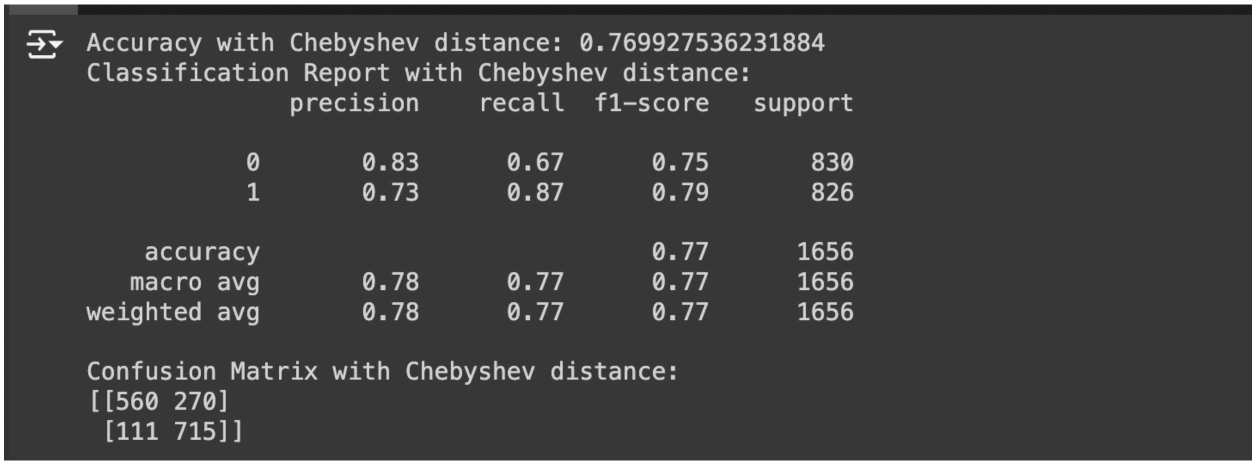
```
Accuracy: 0.7421497584541062
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.67      0.72       830
           1       0.71      0.81      0.76       826

    accuracy                           0.74      1656
   macro avg       0.75      0.74      0.74      1656
weighted avg       0.75      0.74      0.74      1656

Confusion Matrix:
[[559 271]
 [156 670]]
```

k=7 p=5



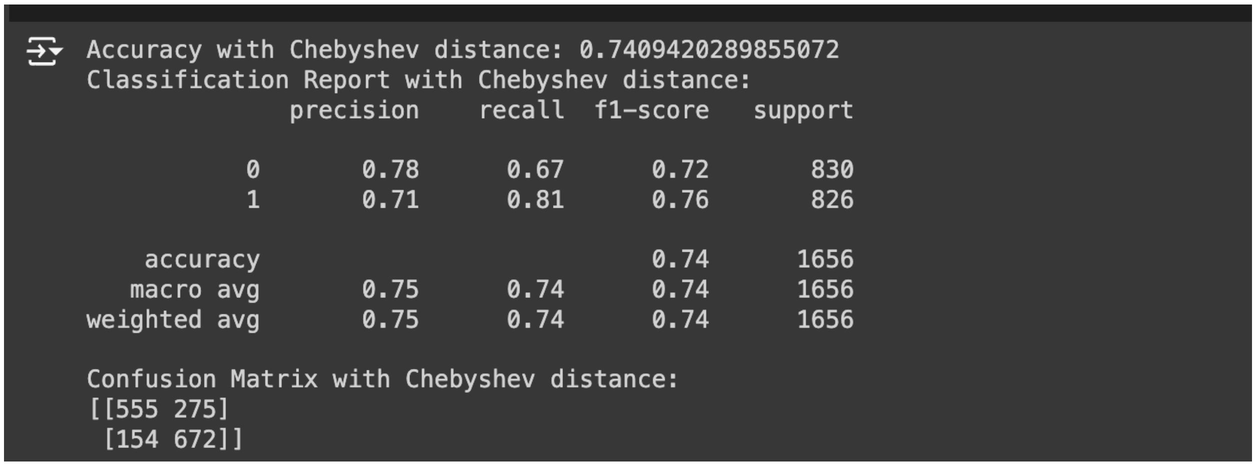
```
Accuracy: 0.7409420289855072
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       830
           1       0.72      0.79      0.75       826

    accuracy                           0.74      1656
   macro avg       0.74      0.74      0.74      1656
weighted avg       0.74      0.74      0.74      1656

Confusion Matrix:
[[575 255]
 [174 652]]
```

Chebyshev
k=3



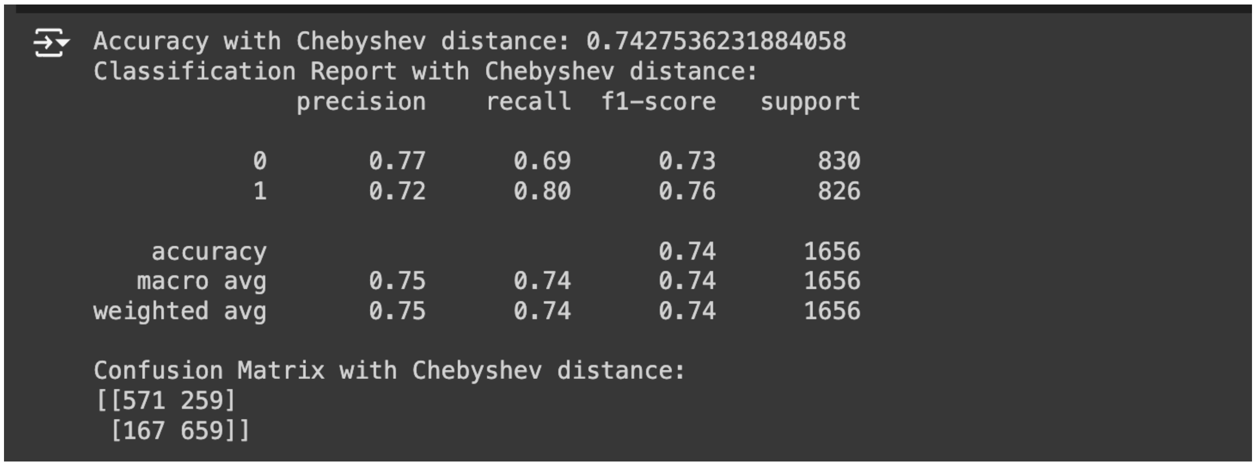
```
Accuracy with Chebyshev distance: 0.769927536231884
Classification Report with Chebyshev distance:
              precision    recall  f1-score   support

           0       0.83      0.67      0.75       830
           1       0.73      0.87      0.79       826

    accuracy                           0.77      1656
   macro avg       0.78      0.77      0.77      1656
weighted avg       0.78      0.77      0.77      1656

Confusion Matrix with Chebyshev distance:
[[560 270]
 [111 715]]
```

k=5



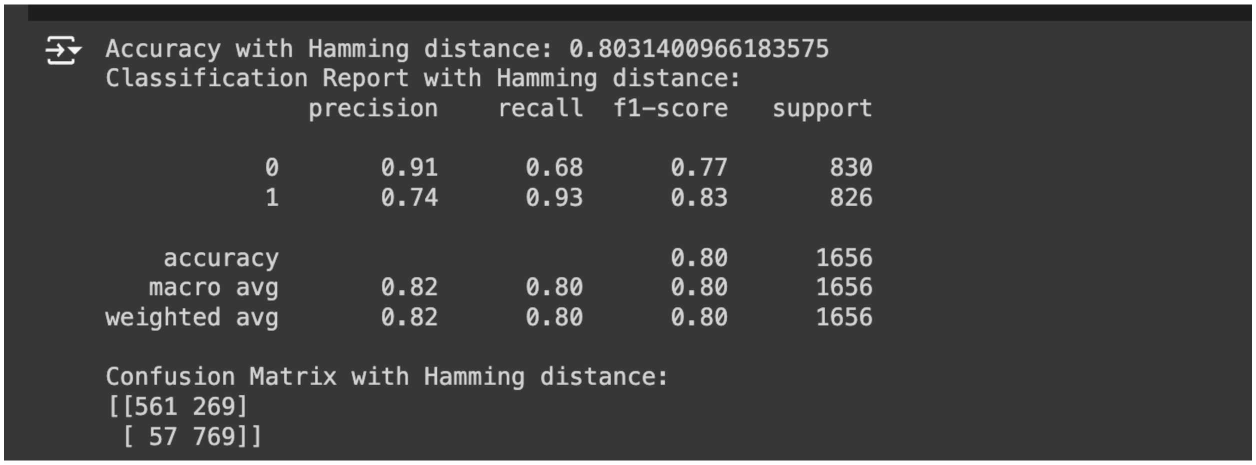
```
Accuracy with Chebyshev distance: 0.7409420289855072
Classification Report with Chebyshev distance:
              precision    recall  f1-score   support

           0       0.78      0.67      0.72       830
           1       0.71      0.81      0.76       826

    accuracy                           0.74      1656
   macro avg       0.75      0.74      0.74      1656
weighted avg       0.75      0.74      0.74      1656

Confusion Matrix with Chebyshev distance:
[[555 275]
 [154 672]]
```

k=7



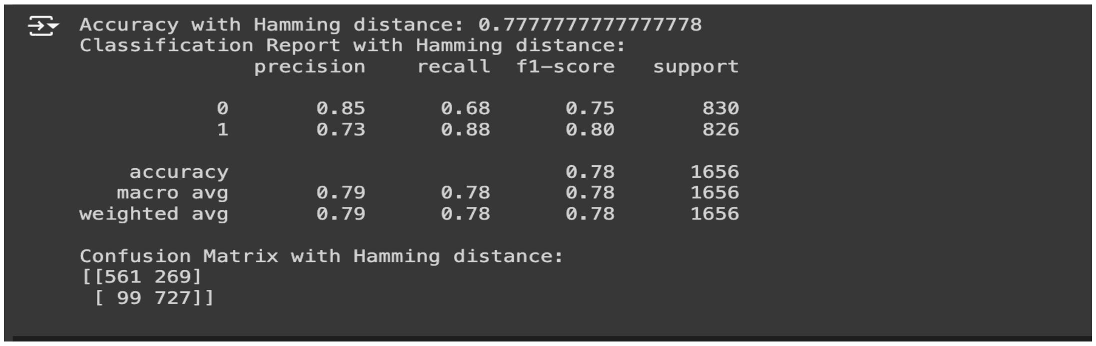
```
Accuracy with Chebyshev distance: 0.7427536231884058
Classification Report with Chebyshev distance:
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       830
           1       0.72      0.80      0.76       826

    accuracy                           0.74      1656
   macro avg       0.75      0.74      0.74      1656
weighted avg       0.75      0.74      0.74      1656

Confusion Matrix with Chebyshev distance:
[[571 259]
 [167 659]]
```

Hamming
k=3



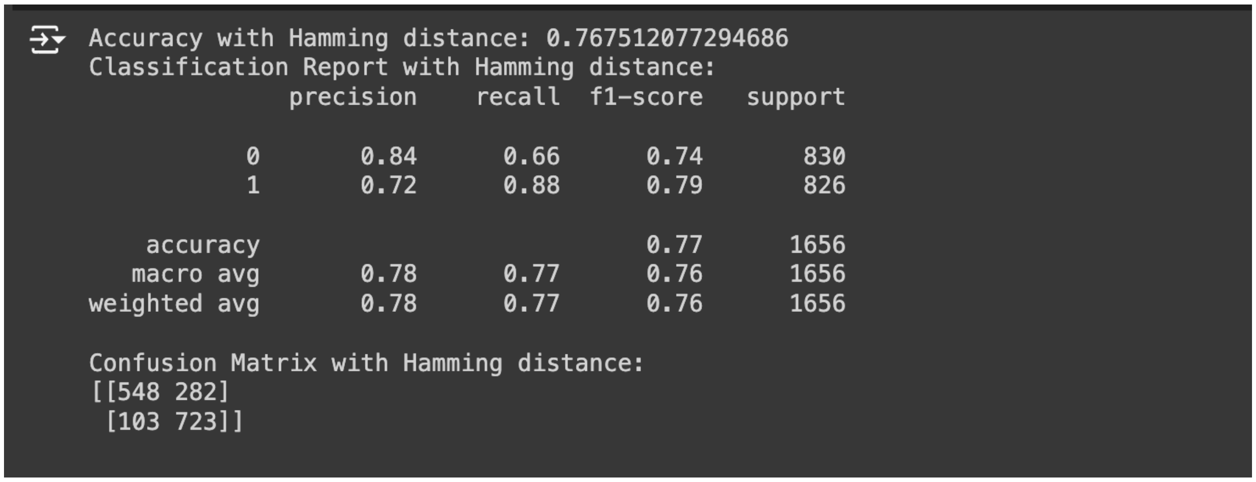
```
Accuracy with Hamming distance: 0.8031400966183575
Classification Report with Hamming distance:
              precision    recall  f1-score   support

           0       0.91      0.68      0.77       830
           1       0.74      0.93      0.83       826

    accuracy                           0.80      1656
   macro avg       0.82      0.80      0.80      1656
weighted avg       0.82      0.80      0.80      1656

Confusion Matrix with Hamming distance:
[[561 269]
 [ 57 769]]
```

k=5



```
Accuracy with Hamming distance: 0.7777777777777778
Classification Report with Hamming distance:
              precision    recall  f1-score   support

           0       0.85      0.68      0.75       830
           1       0.73      0.88      0.80       826

    accuracy                           0.78      1656
   macro avg       0.79      0.78      0.78      1656
weighted avg       0.79      0.78      0.78      1656

Confusion Matrix with Hamming distance:
[[561 269]
 [ 99 727]]
```

k=7



```
Accuracy with Hamming distance: 0.767512077294686
Classification Report with Hamming distance:
              precision    recall  f1-score   support

           0       0.84      0.66      0.74       830
           1       0.72      0.88      0.79       826

    accuracy                           0.77      1656
   macro avg       0.78      0.77      0.76      1656
weighted avg       0.78      0.77      0.76      1656

Confusion Matrix with Hamming distance:
[[548 282]
 [103 723]]
```

Sno	K	Metric	p	Accuracy
1	3	Euclidean	NA	76.75%
2	5	Euclidean	NA	74.39%

3	7	Euclidean	NA	74.635%
4	9	Euclidean	NA	74.033%
5	3	Manhattan	NA	77.83%
6	5	Manhattan	NA	74.93%
7	7	Manhattan	NA	75.42%
8	9	Manhattan	NA	73.61%
9	3	Minkowski	3	76.81
10	5	Minkowski	3	74.33
11	7	Minkowski	3	74.39
12	3	Minkowski	5	76.99%
13	5	Minkowski	5	74.21%
14	7	Minkowski	5	74.09
15	3	Hamming	NA	80.31
16	5	Hamming	NA	77.77
17	7	Hamming	NA	76.75
18	3	Chebyshev	NA	76.99
19	5	Chebyshev	NA	74.09
20	7	Chebyshev	NA	74.27

Decision Tree

A decision tree is a versatile machine learning algorithm that can handle both categorical and numerical data. It is a tree-like model used for decision-making and predictive modeling.

Components of a Decision Tree:

1. **Head Node:**

   The topmost node representing the entire dataset. It acts as the root from which the tree grows.

2. **Node:**

    Represents decisions or test points. Each node corresponds to a feature in the dataset and tests a specific attribute.

3. **Edges:**

    Represent the outcome of the test conducted at each node. They lead to subsequent nodes and define the tree's branches.

4. **Leaf Nodes:**

    These are the final decisions or predictions made by the decision tree. Leaf nodes do not split further and represent the output or class label.

Gini Impurity:

Gini impurity is a metric used to evaluate the quality of a split in a decision tree. It measures how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the set. The formula for Gini impurity is:

Gini impurity=$1-(p^2+q^2)$

- p is the probability of an element being classified as the first class.
- q is the probability of an element being classified as the second class.

Since $p+q=1p + q = 1p+q=1$ for binary classification, the formula can also be expressed as:

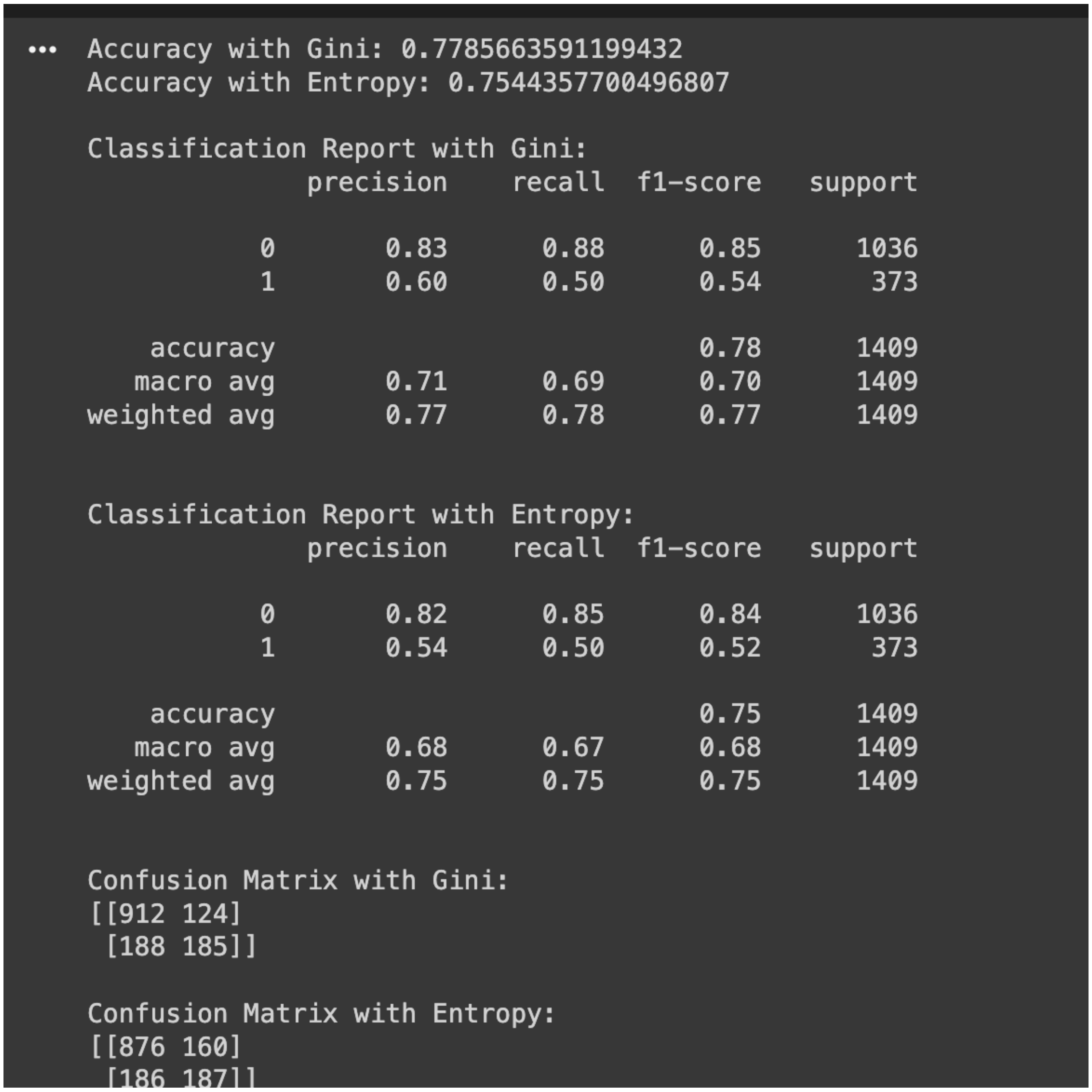
Gini impurity=2pq

. Gini impurity reaches its minimum (zero) when all cases in the node fall into a single target category, indicating a pure node. A lower Gini impurity indicates a better split.

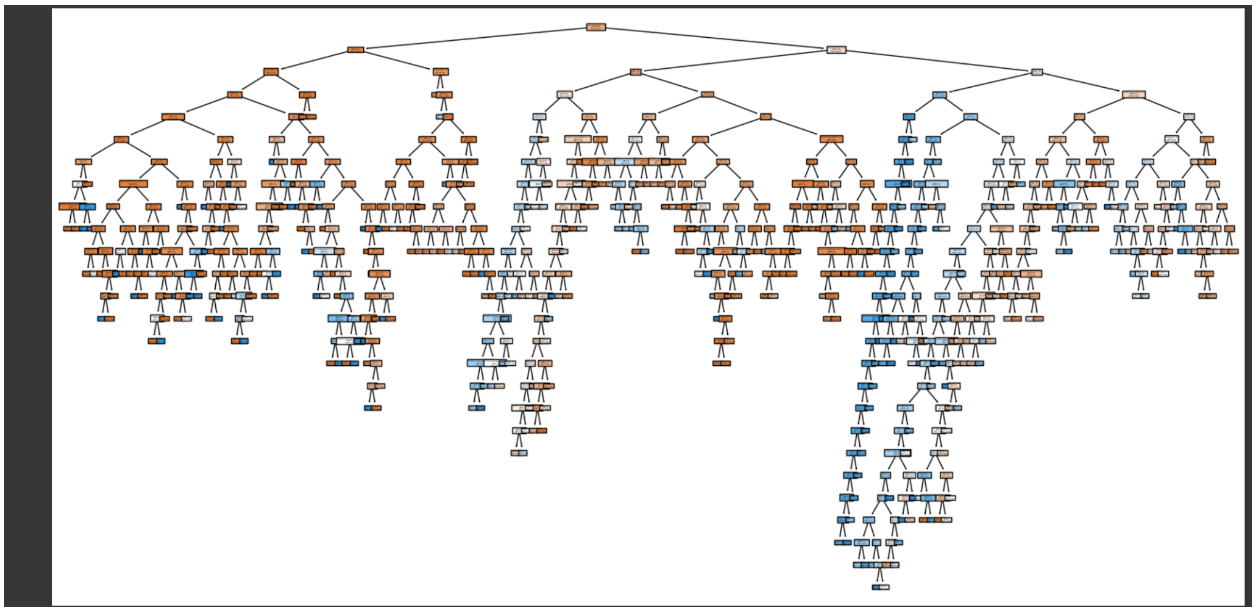
Decision trees are intuitive and easy to interpret, making them a popular choice for many classification and regression tasks in machine learning.

Entropy:
Entropy is a measure of impurity or disorder used to determine the best split in decision trees. It quantifies the uncertainty or unpredictability in the data. In the context of binary classification, entropy measures how mixed the classes are in a given node.

$H(D) = - p_1 log_2(p_1) - p_2 log_2(p_2)$
Where, p_1 is the proportion of the first class in the dataset
And p_2 is the proportion of the second class in the dataset.



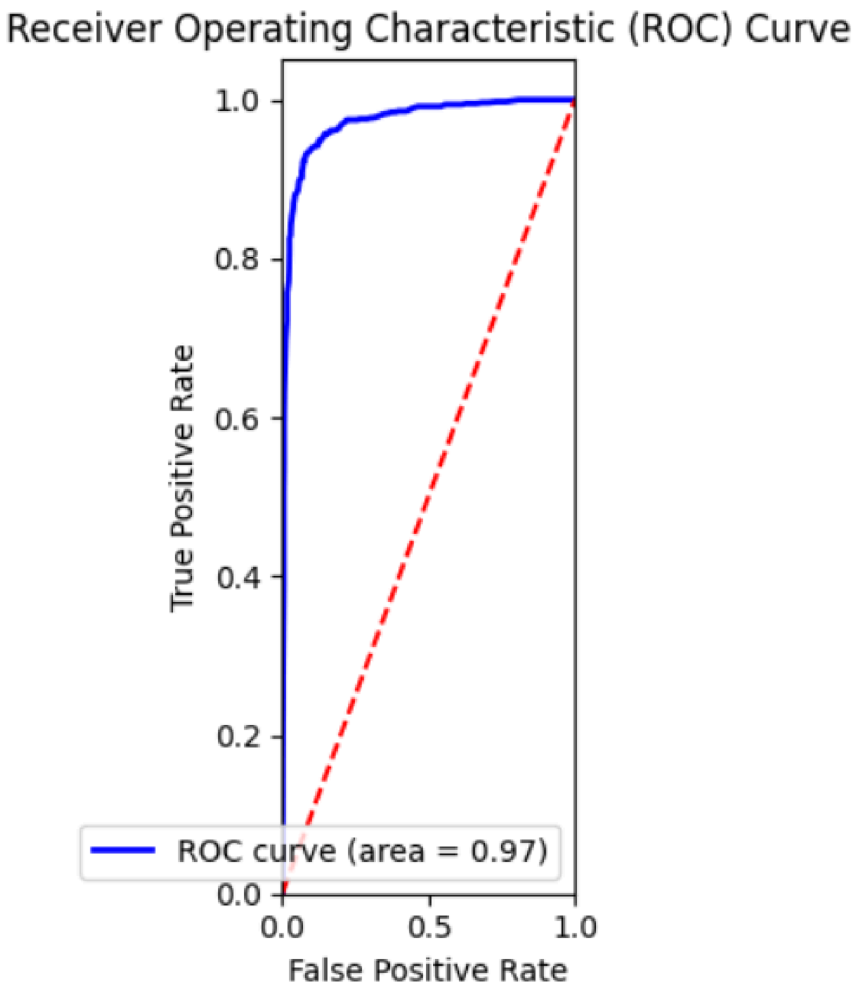
```
•••  Accuracy with Gini: 0.7785663591199432
     Accuracy with Entropy: 0.7544357700496807

     Classification Report with Gini:
                   precision    recall  f1-score   support

                0       0.83      0.88      0.85      1036
                1       0.60      0.50      0.54       373

         accuracy                           0.78      1409
        macro avg       0.71      0.69      0.70      1409
     weighted avg       0.77      0.78      0.77      1409


     Classification Report with Entropy:
                   precision    recall  f1-score   support

                0       0.82      0.85      0.84      1036
                1       0.54      0.50      0.52       373

         accuracy                           0.75      1409
        macro avg       0.68      0.67      0.68      1409
     weighted avg       0.75      0.75      0.75      1409


     Confusion Matrix with Gini:
     [[912 124]
      [188 185]]

     Confusion Matrix with Entropy:
     [[876 160]
      [186 187]]
```



Random Forest

It builds multiple decision trees and combines their outputs to produce a more accurate and stable prediction.
Combines the predictions of multiple models to improve overall performance and reduce the risk of overfitting.
Bagging: A technique where multiple models are trained on different subsets of the training data to improve performance and reduce variance.
The greater number of trees in the forest leads to higher accuracy[link]. Each decision tree in the forest is built using a random subset of features, which helps in reducing the correlation between individual trees and improving generalization.

```
[  55  791]]
```

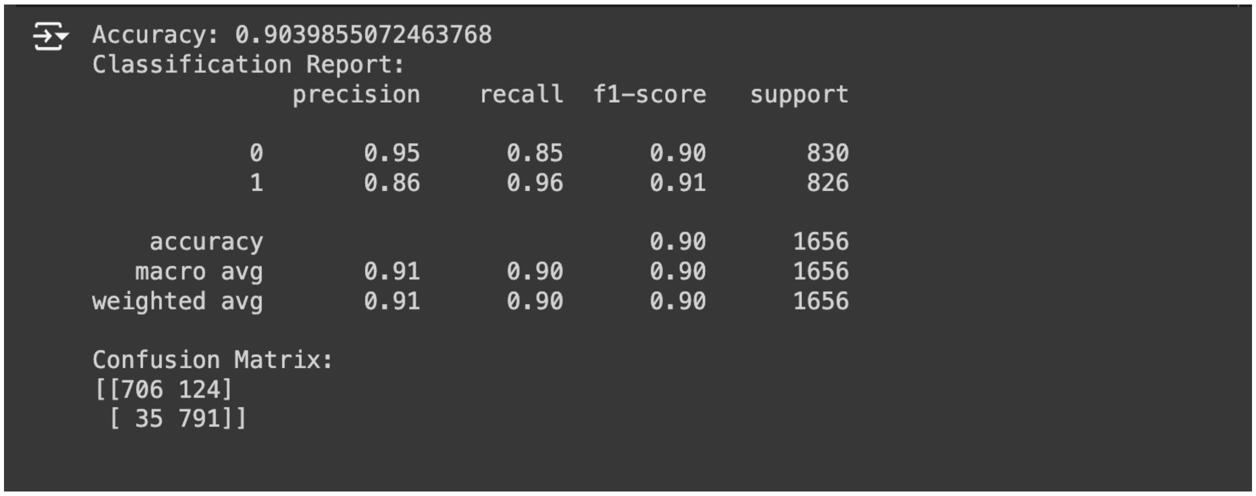


1. Bootstrap Sampling : Create B Bootstrap samples.where each sample S_b is crated by randomly sampling with replacement from the training dataset.
2. Decision Tree Training : For each bootstrap sample S_b:
   Train a decision tree T_b , At each node,randomly select a subset of features F_r and choose the best split from these features.
3. Aggregation : For classification:
   Predict class $\widehat{y}$ by majority vote:

$$\widehat{y} = \text{mode}(T_b(x) \text{ for all trees } T_b)$$

Steps in Building a Random Forest:

1. Generate Bootstrap Samples: Create multiple subsets of the training data by sampling with replacement (bootstrapping). Each subset will be used to train a separate decision tree.
2. Train Decision Trees: For each bootstrap sample:
    - Build a decision tree using the subset of data.
    - At each node in the tree, randomly select a subset of features and choose the best feature to split on based on criteria like Gini impurity or entropy.
3. Aggregate Predictions:
    - For Classification: Combine the predictions from all decision trees using majority voting (i.e., the class that receives the most votes is the predicted class).



```
Accuracy: 0.9039855072463768
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.85      0.90       830
           1       0.86      0.96      0.91       826

    accuracy                           0.90      1656
   macro avg       0.91      0.90      0.90      1656
weighted avg       0.91      0.90      0.90      1656

Confusion Matrix:
[[706 124]
 [ 35 791]]
```

Performance Comparison of Binary Classification Algorithms

We compared Linear Regression, Logistic regression, KNN, Decision tree, and random forest models to conclude the best models amongst these for binary classification. The random forest model outperformed all the other models with an accuracy of 90.39%, whereas all the other models had an accuracy of around 74%.

For the K-Nearest Neighbors (KNN) model, the accuracy of 80.31% was observed when the number of nearest neighbors considered was 3, and the distance metric was Hamming. All the other varieties of the KNN model showed an accuracy of about 75% on average.

Other models, including Logistic Regression and Decision Tree, demonstrated relatively lower performance, with accuracy scores ranging from 74% to 79%. These findings suggest that while these models have their merits, they were not as effective for our specific dataset as the Random Forest model.

Additionally, we evaluated the Linear Regression model. The Linear Regression model produced a Mean Squared Error (MSE) of 0.1549 and an R-squared value of 0.1645, indicating limited explanatory power and accuracy for this classification task.

Detailed Results:

- Random Forest: Achieved the highest accuracy at 90.39%, indicating superior performance in binary classification tasks.
- K-Nearest Neighbors (KNN):

Best accuracy: 80.31% (with 3 nearest neighbors and Hamming distance).

Other configurations: Approximately 74% accuracy.

- Logistic Regression and Decision Tree: Both models had accuracies between 74% and 79%, reflecting moderate effectiveness.
- Linear Regression:

  Mean Squared Error (MSE): 0.1549

  R-squared: 0.1645

  These metrics indicate that Linear Regression is less suitable for binary classification in this context.

In conclusion, the Random Forest model is recommended for its excellent performance and accuracy, making it the most suitable choice for our binary classification needs.

Value Table for Linear Regression

Algorithm	R squared Value	Mean Squared Value
Linear Regression	0.16448742824354945	0.1549427939207768

Precision-Recall Table:

Algorithm	Precision		Recall		F1 Score		Accuracy
Logistic Regression	0	1	0	1	0	1	
	0.81	0.76	0.74	0.82	0.77	0.79	78%
K-Nearest Neighbour	0	1	0	1	0	1	
Best Case k=3 (Hamming)	0.91	0.74	0.68	0.93	0.77	0.83	80%
Worst Case k=9 (Manhattan)	0.78	0.71	0.67	0.81	0.72	0.75	73%

Random Forest	0	1	0	1	0	1	
	0.95	0.86	0.85	0.96	0.90	0.91	90%

Conclusion
We compared multiple algorithms for binary classification, and the random forest model was the most effective with an overall accuracy of 90.39%. For the KNN model, the best accuracy of 80.31% was observed,when number of nearest neighbors was 3 and distance metric was hamming, while all the other cases had a similar accuracy of around 74%. All the other models(Logistic regression and Decision tree) were also not that effective, with the accuracy ranging from 74%-79%.