TELECOM CHURN PREDICTION

GitHub

https://shorturl.at/Ur03Nt

Presented by-Nyasha Singh
Mentor-Mr. Bhaskar Naidu

INFOSYS

AGENDA

1-Introduction

2-Our Mission & Vision

3-Our Goal

4-Our Milestones

5-Challenges

6-Data Cleaning & Preprocessing

7-ML Model Implementation

8-Hyperparameter Tuning

9-Confusion Matrix & ROC curve

10-Conclusion

INTRODUCTION

Telecom churn prediction involves using statistical and machine learning techniques to identify customers who are likely to discontinue their service with a telecom provider. The goal is to predict churn by analyzing customer behavior and usage patterns.

CUSTOMER ATTRITION

IMPACT ON BUSINESS

REASONS FOR CHURN

OUR MISSION & VISION

MISSION

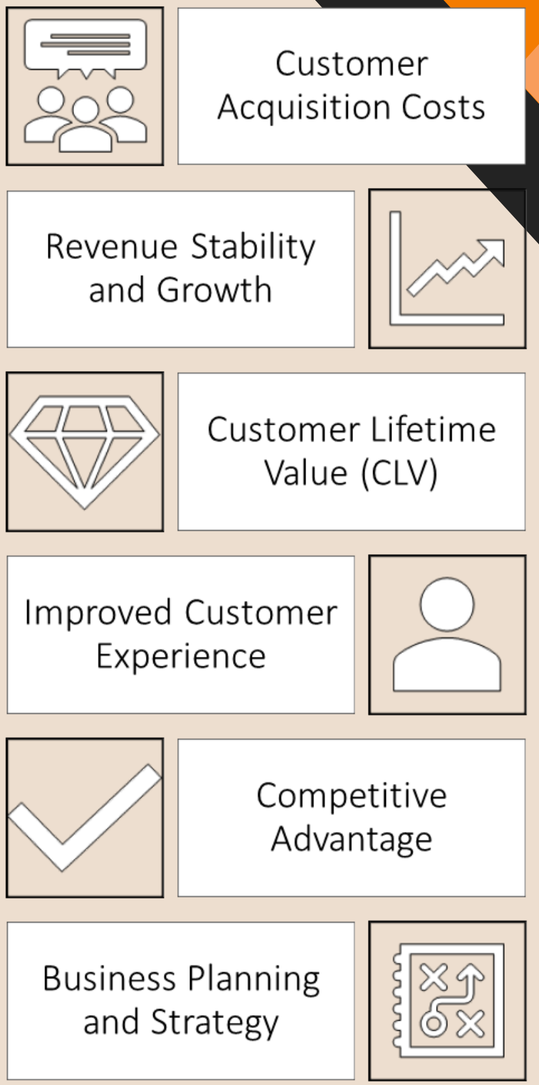
To leverage advanced data analytics and machine learning techniques to accurately predict customer churn, enabling proactive retention strategies and enhancing customer satisfaction for telecom providers.

VISION

To drive significant improvements in customer satisfaction, business performance, and competitiveness within the telecommunications sector through innovative and effective churn prediction solutions.

OUR GOALS

-Customer Retention
-Cost Efficiency
-Improved Customer Experience
-Revenue Growth



OUR MILESTONES

1st-Data gathering & understanding the dataset

2nd-Data Cleaning and Pre-processing

3rd-Machine Learning and model Implementation

4th-Hyperparameter Tuning , ROC curve & Confusion Matrix

Data Cleaning & Pre-Processing

1-Convert data types of variables which are misclassified.
2-Removing Duplicate Records.
3-Remove Unique Values Variables.
4-Removing Zero Variance Variables.
5-Outlier Treatment.
6-Missing Value Treatment.
7-Removing the highly correlated variables.
8-Muliticollinearity(VIF>5)

OUTPUT

```
In [10]: df.dtypes  #to get the types of data present

Out[10]: s6.new.rev.p2.m2              float64
         s1.new.rev.m1                 float64
         s3.og.rev.4db.p5              float64
         s3.new.rev.4db.p5             float64
         s4.usg.ins.p2                   int64
                                        ...
         s3.og.rev.all.m2              float64
         s3.new.rev.m2                 float64
         prop.og.mou.any.p6            float64
         prop.loc.i2i.mou.og.mou.p3    float64
         s3.rev.p1                     float64
         Length: 111, dtype: object
```

```
Missing values in each column:
s6.new.rev.p2.m2              0
s1.new.rev.m1                 0
s3.og.rev.4db.p5              0
s3.new.rev.4db.p5             0
s4.usg.ins.p2                 0
                             ..
s3.og.rev.all.m2              0
s3.new.rev.m2                 0
prop.og.mou.any.p6            0
prop.loc.i2i.mou.og.mou.p3    0
s3.rev.p1                     0
Length: 111, dtype: int64
Summary of missing values in each column:
Empty DataFrame
Columns: [Missing Values, Percentage]
Index: []
```

```
# Display the columns removed and the new shape of the dataset
print("Removed zero variance columns:", zero_variance_cols)
print("New shape of the dataset:", data_cleaned.shape)
```

```
Removed zero variance columns: []
New shape of the dataset: (25000, 111)
```

```
In [8]: df.shape #gives actual shape of rows & columns

Out[8]: (25000, 111)
```

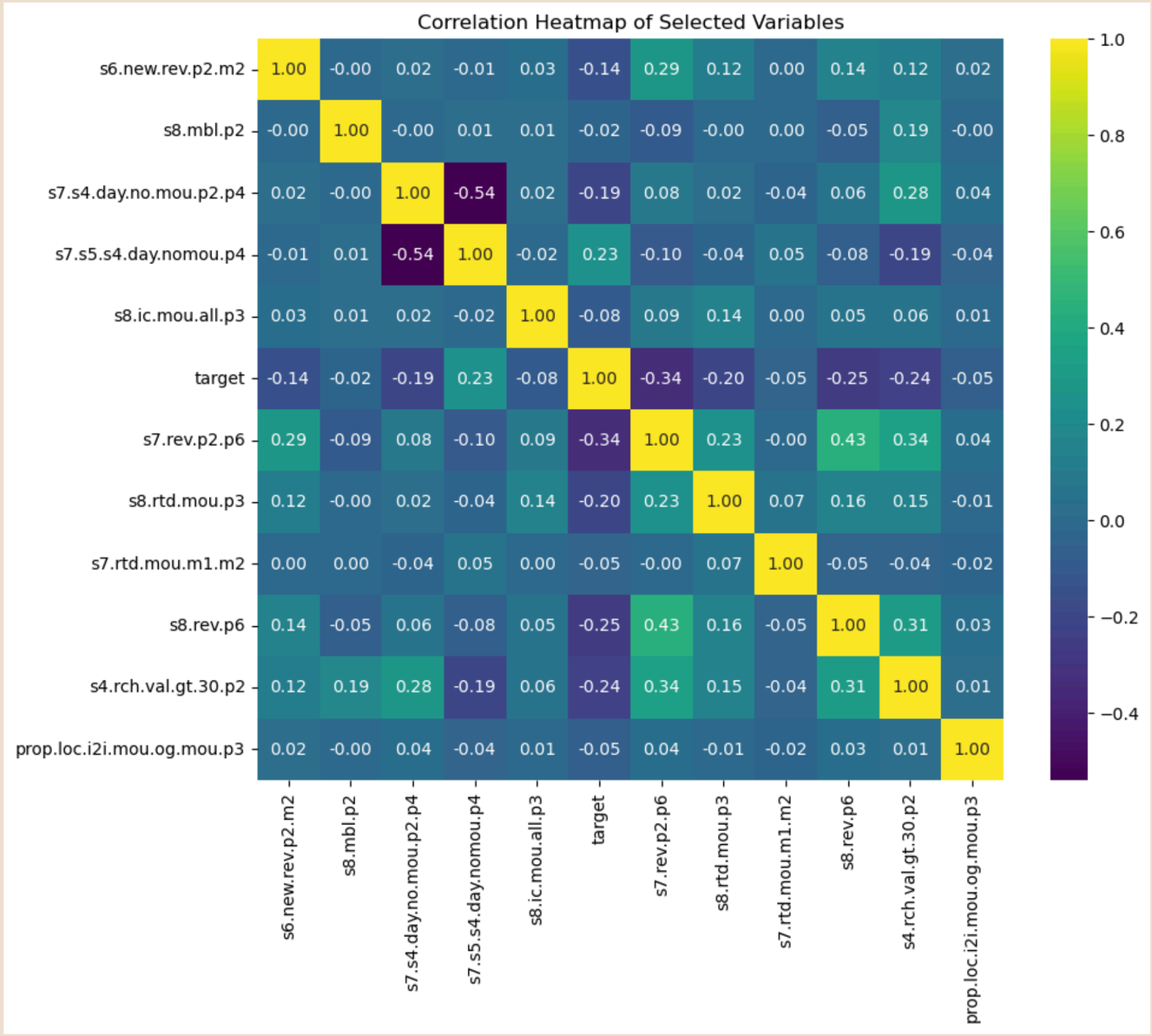
```
Selected numeric columns with VIF <= 5:
['s6.new.rev.p2.m2', 's8.mbl.p2', 's7.s4.day.no.mou.p2.p4', 's7.s5.s4.day.nomou.p4', 's8.ic.mou.all.p
3', 'target', 's7.rev.p2.p6', 's8.rtd.mou.p3', 's7.rtd.mou.m1.m2', 's8.rev.p6', 's4.rch.val.gt.30.p
2', 'prop.loc.i2i.mou.og.mou.p3']
```

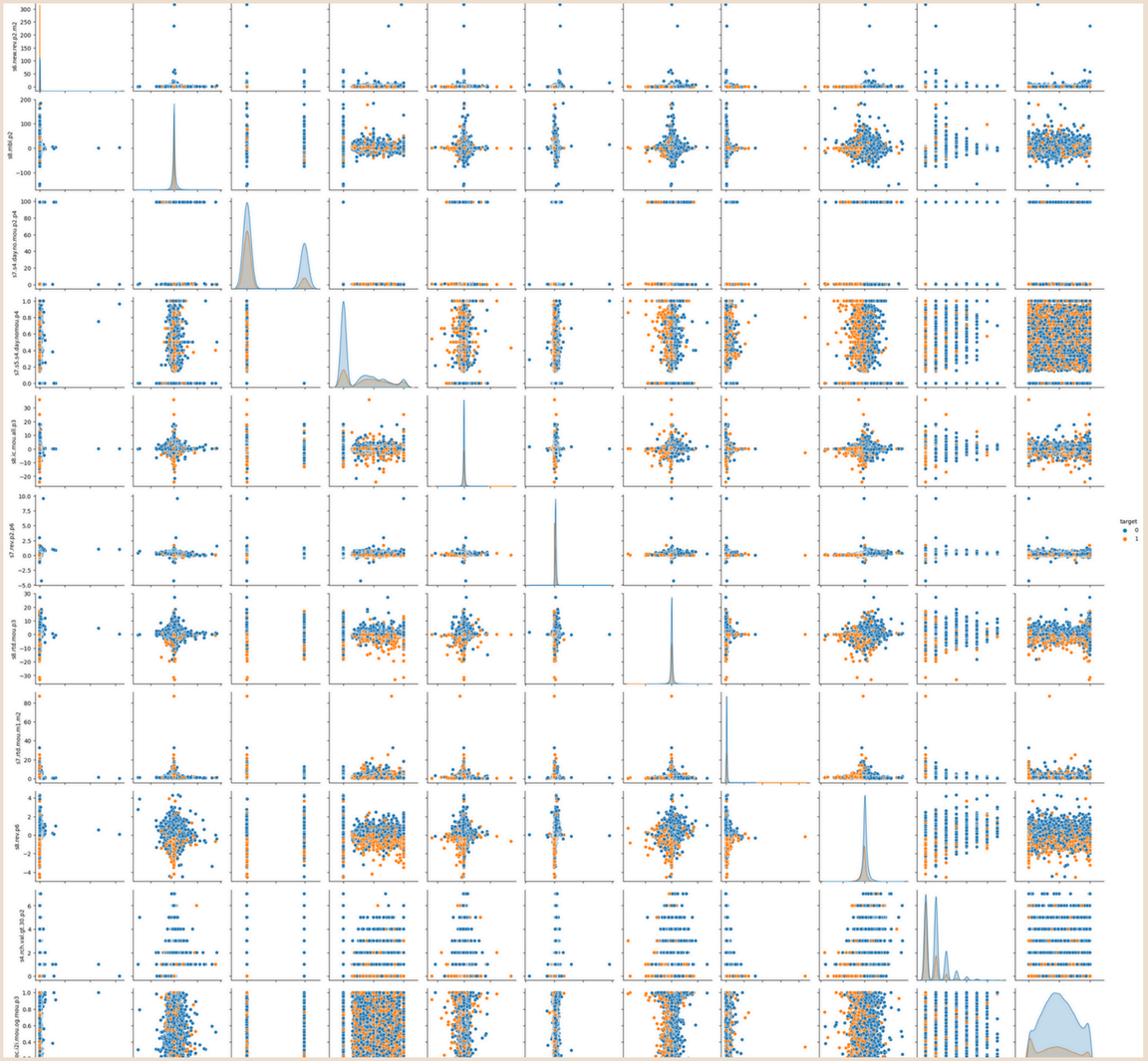
```
In [13]: #to check the duplicate values are present in dataset or not

duplicate_rows = df[df.duplicated()]
duplicate_rows
```

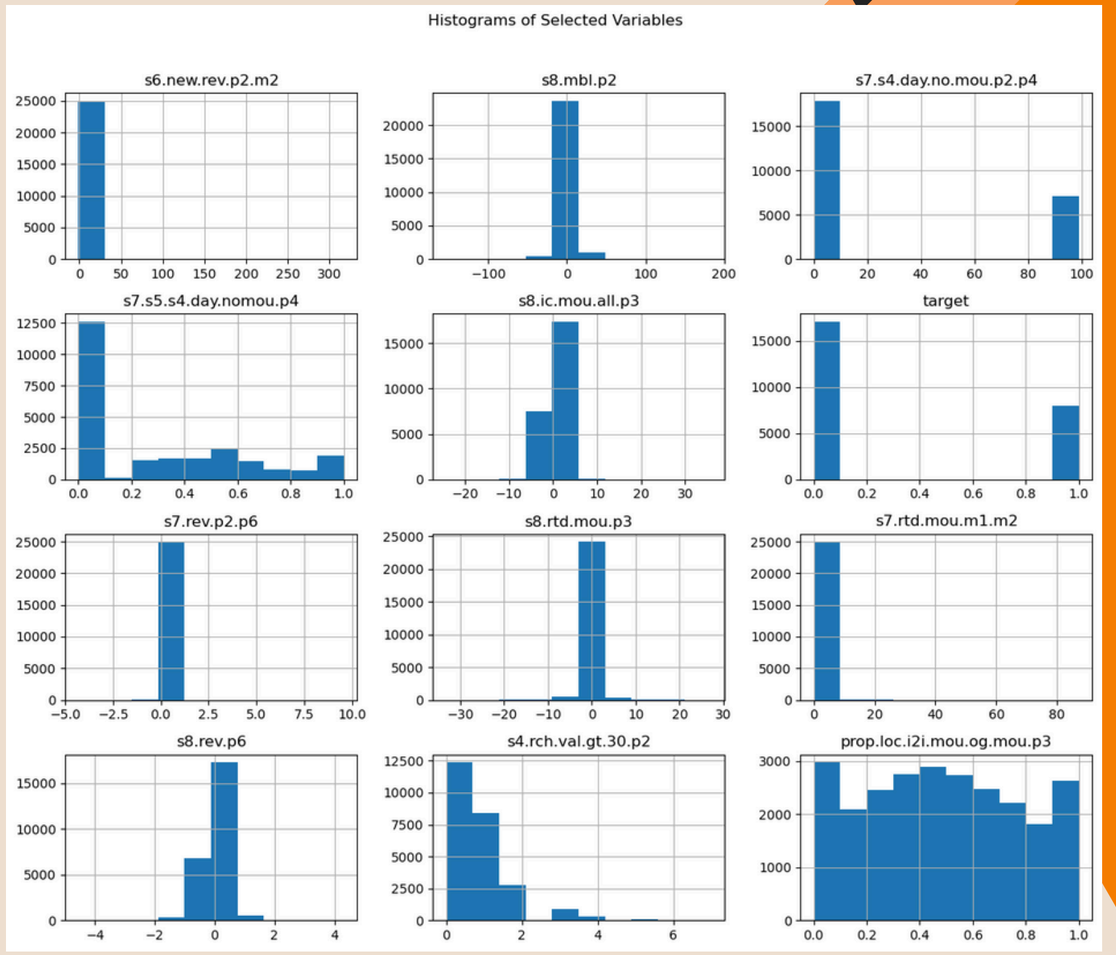
Out[13]:

s6.new.rev.p2.m2	s1.new.rev.m1	s3.og.rev.4db.p5	s3.new.rev.4db.p5	s4.usg.ins.p2	s4.og.unq.any.p2	s2.rch.val.p6	s1.og.rev.all

0 rows × 111 columns

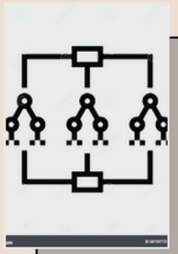

Correlation Heatmap of Selected Variables
s6.new.rev.p2.m2 1.00 -0.00 0.02 -0.01 0.03 -0.14 0.29 0.12 0.00 0.14 0.12 0.02
s8.mbl.p2 -0.00 1.00 -0.00 0.01 0.01 -0.02 -0.09 -0.00 0.00 -0.05 0.19 -0.00
s7.s4.day.no.mou.p2.p4 0.02 -0.00 1.00 -0.54 0.02 -0.19 0.08 0.02 -0.04 0.06 0.28 0.04
s7.s5.s4.day.nomou.p4 -0.01 0.01 -0.54 1.00 -0.02 0.23 -0.10 -0.04 0.05 -0.08 -0.19 -0.04
s8.ic.mou.all.p3 0.03 0.01 0.02 -0.02 1.00 -0.08 0.09 0.14 0.00 0.05 0.06 0.01
target -0.14 -0.02 -0.19 0.23 -0.08 1.00 -0.34 -0.20 -0.05 -0.25 -0.24 -0.05
s7.rev.p2.p6 0.29 -0.09 0.08 -0.10 0.09 -0.34 1.00 0.23 -0.00 0.43 0.34 0.04
s8.rtd.mou.p3 0.12 -0.00 0.02 -0.04 0.14 -0.20 0.23 1.00 0.07 0.16 0.15 -0.01
s7.rtd.mou.m1.m2 0.00 0.00 -0.04 0.05 0.00 -0.05 -0.00 0.07 1.00 -0.05 -0.04 -0.02
s8.rev.p6 0.14 -0.05 0.06 -0.08 0.05 -0.25 0.43 0.16 -0.05 1.00 0.31 0.03
s4.rch.val.gt.30.p2 0.12 0.19 0.28 -0.19 0.06 -0.24 0.34 0.15 -0.04 0.31 1.00 0.01
prop.loc.i2i.mou.og.mou.p3 0.02 -0.00 0.04 -0.04 0.01 -0.05 0.04 -0.01 -0.02 0.03 0.01 1.00
s6.new.rev.p2.m2
s8.mbl.p2
s7.s4.day.no.mou.p2.p4
s7.s5.s4.day.nomou.p4
s8.ic.mou.all.p3
target
s7.rev.p2.p6
s8.rtd.mou.p3
s7.rtd.mou.m1.m2
s8.rev.p6
s4.rch.val.gt.30.p2
prop.loc.i2i.mou.og.mou.p3
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


target
0
1


Histograms of Selected Variables
s6.new.rev.p2.m2
s8.mbl.p2
s7.s4.day.no.mou.p2.p4
s7.s5.s4.day.nomou.p4
s8.ic.mou.all.p3
target
s7.rev.p2.p6
s8.rtd.mou.p3
s7.rtd.mou.m1.m2
s8.rev.p6
s4.rch.val.gt.30.p2
prop.loc.i2i.mou.og.mou.p3

MACHINE LEARNING MODEL BUILDING

DECISION TREE

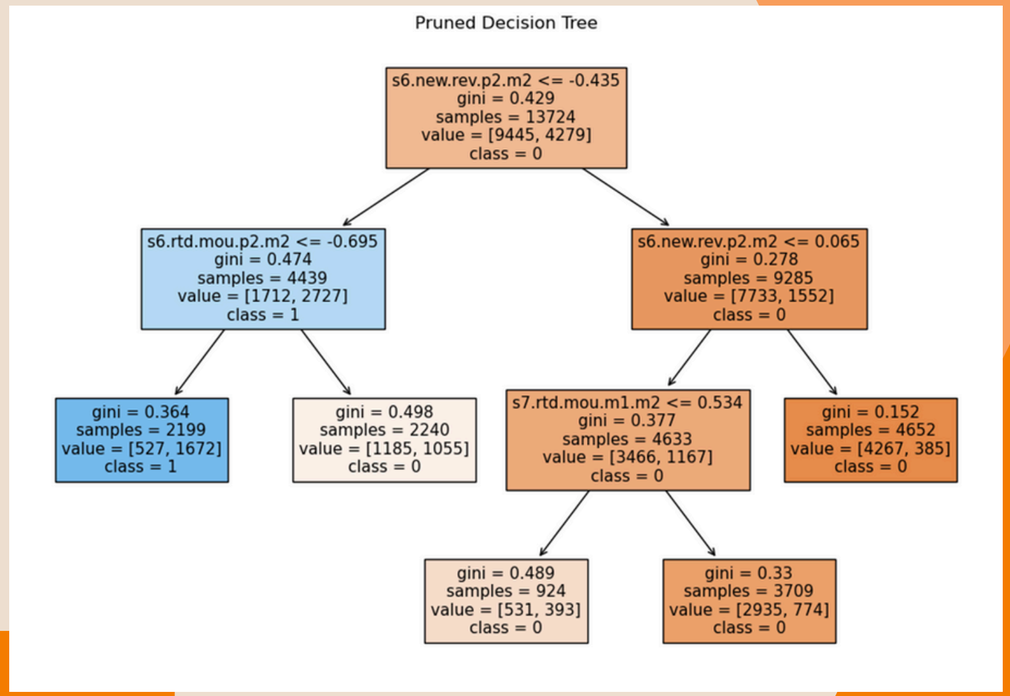


RANDOM FOREST

LOGISTIC REGRESSION

DECISION TREE

A decision tree is a machine-learning model used for classification and regression tasks. It splits the data into subsets based on the value of input features, forming a tree-like structure of decision nodes and leaf nodes, where each path from the root to a leaf represents a decision rule.



HYPERPARAMETER TUNING

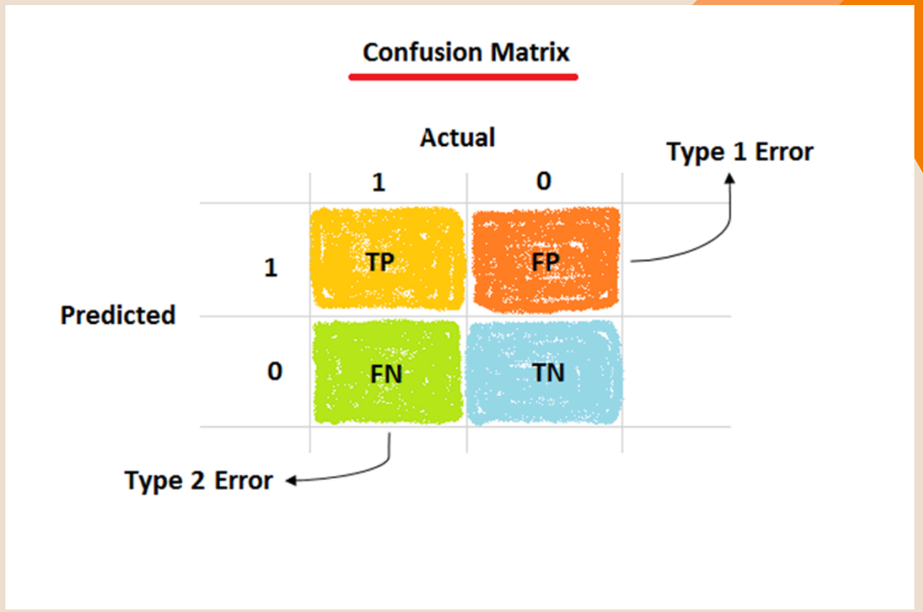
Hyperparameters: Parameters set before the learning process begins, unlike model parameters learned from data.

Tuning: The process of finding the optimal set of hyperparameters to improve model performance.

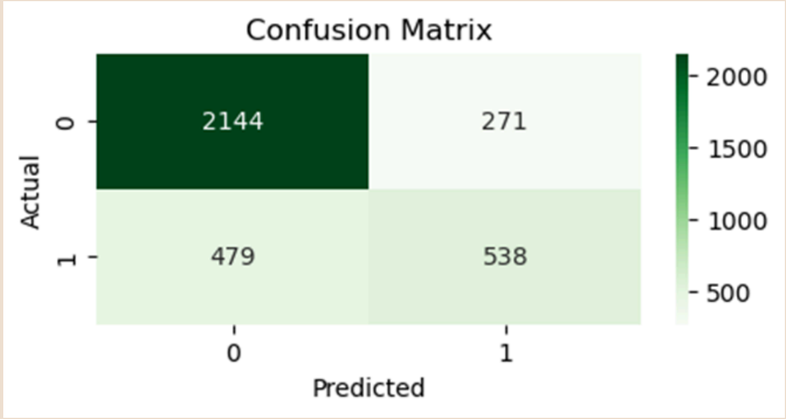
```
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_lea
f': 10, 'min_samples_split': 2}
Accuracy: 0.78
              precision    recall  f1-score   support

           0       0.82      0.89      0.85      2415
           1       0.67      0.53      0.59      1017

    accuracy                           0.78      3432
   macro avg       0.74      0.71      0.72      3432
weighted avg       0.77      0.78      0.77      3432
```

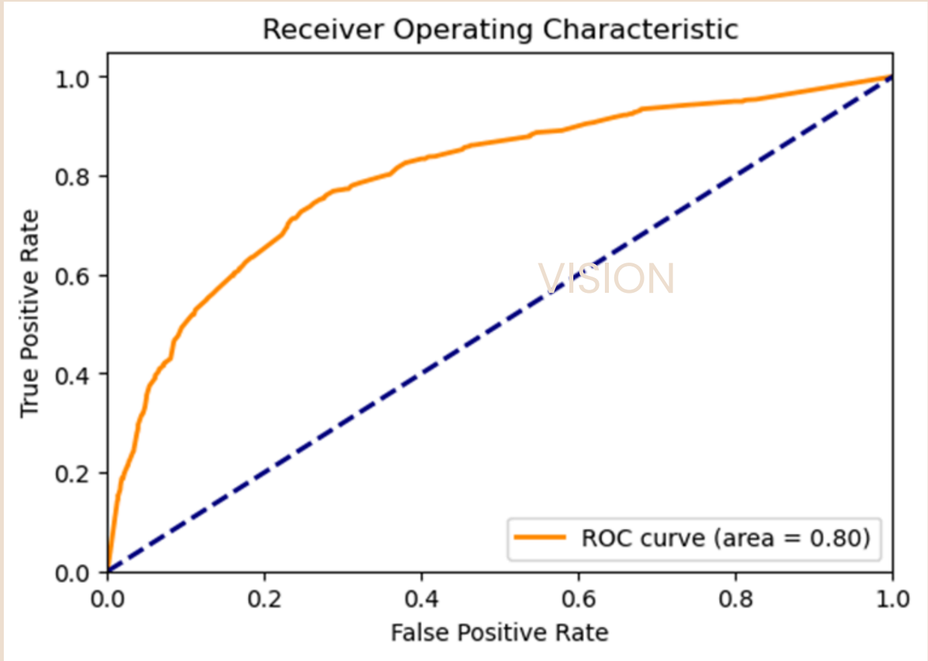
CONFUSION MATRIX

A confusion matrix evaluates a classification model's performance by comparing predictions with actual labels. It includes True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). This matrix helps calculate metrics like accuracy, precision, recall, and F1-score, offering a detailed performance overview.



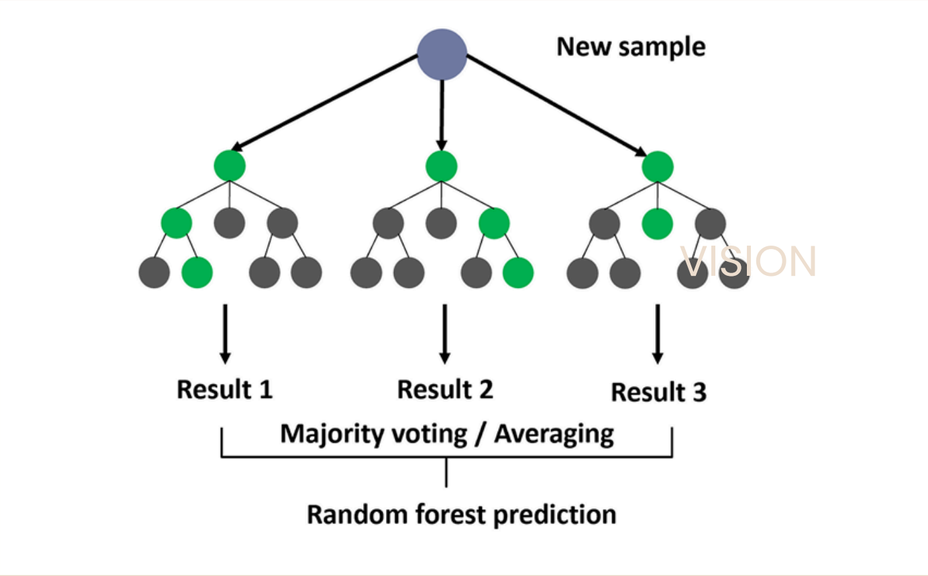
CONFUSION MATRIX & ROC CURVE





RANDOM FOREST

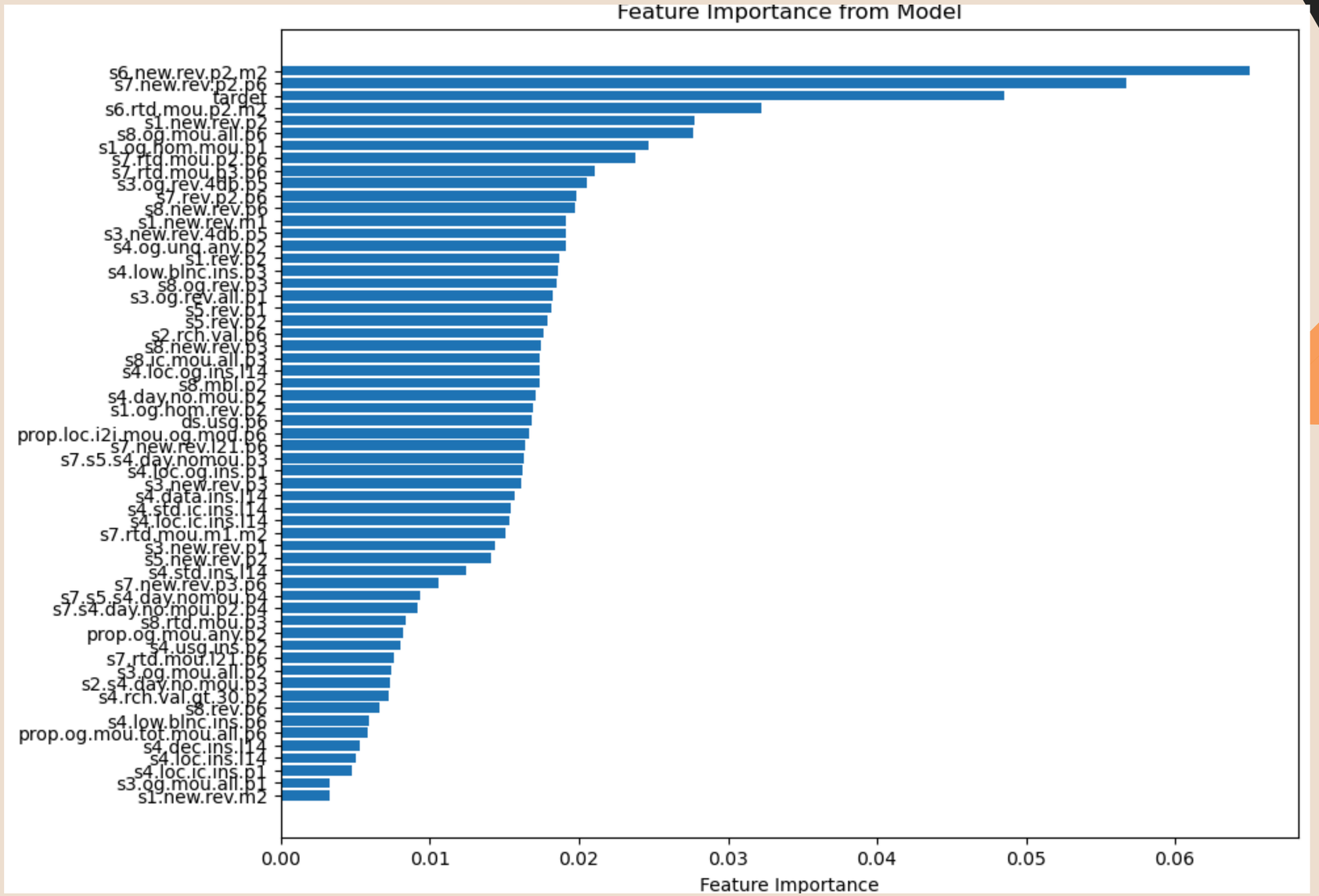
Random Forest is an ensemble learning method primarily used for classification and regression tasks. It operates by constructing multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.



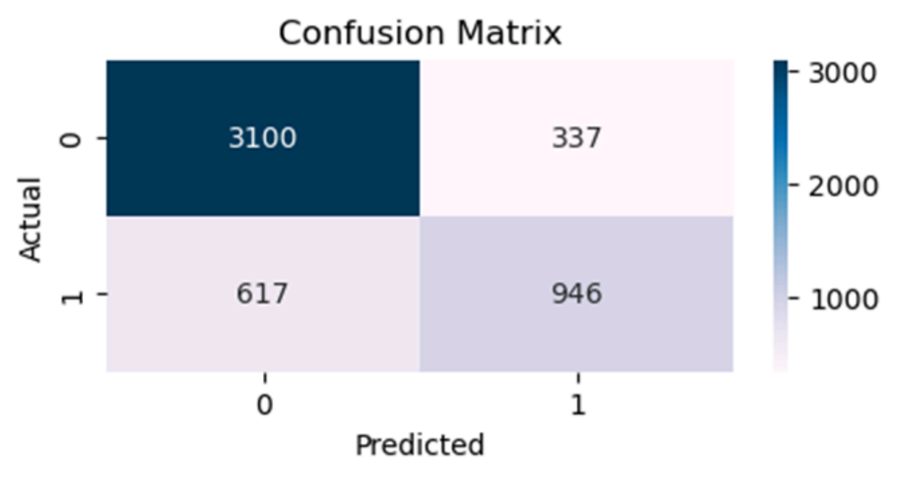
CODE SNIPPET & RESULT

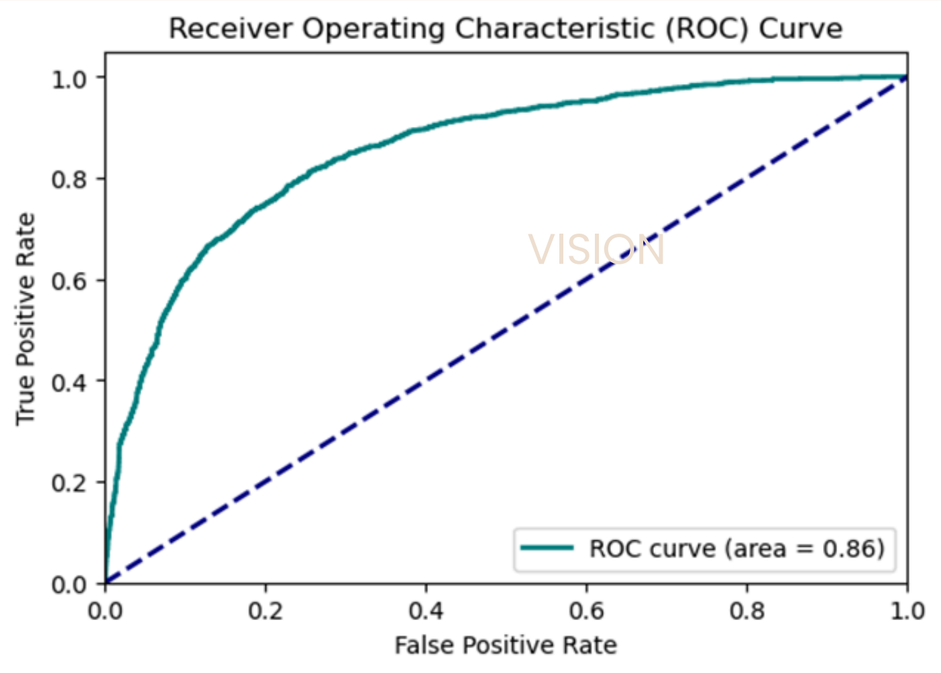
```python
# Step 4: Splitting into training and testing sets
X = df_sampled.drop(columns=['target'])  # Replace 'target_column' with your actual target variable
y = df_sampled['target']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 5: Training a Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)  # Example Random Forest classifier w
model.fit(X_train, y_train)

# Step 6: Predictions and evaluation
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.80
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      2415
           1       0.71      0.57      0.63      1017

    accuracy                           0.80      3432
   macro avg       0.77      0.73      0.75      3432
weighted avg       0.79      0.80      0.80      3432
```


Feature Importance from Model
s6.new.rev.p2.m2
s7.new.rev.p2.p6
target
s6.rtd.mou.p2.m2
s1.new.rev.p2
s8.og.mou.all.p6
s1.og.hom.mou.p1
s7.rtd.mou.p2.p6
s7.rtd.mou.p3.p6
s3.og.rev.4db.p5
s7.rev.p2.p6
s8.new.rev.p6
s1.new.rev.m1
s3.new.rev.4db.p5
s4.og.unq.any.p2
s1.rev.p2
s4.low.blnc.ins.p3
s8.og.rev.p3
s3.og.rev.all.p1
s5.rev.p1
s5.rev.p2
s2.rch.val.p6
s8.new.rev.p3
s8.ic.mou.all.p3
s4.loc.og.ins.l14
s8.mbl.p2
s4.day.no.mou.p2
s1.og.hom.rev.p2
ds.usg.p6
prop.loc.i2i.mou.og.mod.p6
s7.new.rev.l21.p6
s7.s5.s4.day.nomou.p3
s4.loc.og.ins.p1
s3.new.rev.p3
s4.data.ins.l14
s4.std.ic.ins.l14
s4.loc.ic.ins.l14
s7.rtd.mou.m1.m2
s3.new.rev.p1
s5.new.rev.p2
s4.std.ins.l14
s7.new.rev.p3.p6
s7.s5.s4.day.nomou.p4
s7.s4.day.no.mou.p2.p4
s8.rtd.mou.p3
prop.og.mou.any.p2
s4.usg.ins.p2
s7.rtd.mou.l21.p6
s3.og.mou.all.p2
s2.s4.day.no.mou.p3
s4.rch.val.gt.30.p2
s8.rev.p6
s4.low.blnc.ins.p6
prop.og.mou.tot.mou.all.p6
s4.dec.ins.l14
s4.loc.ins.l14
s4.loc.ic.ins.p1
s3.og.mou.all.p1
s1.new.rev.m2
0.00
0.01
0.02
0.03
0.04
0.05
0.06
Feature Importance

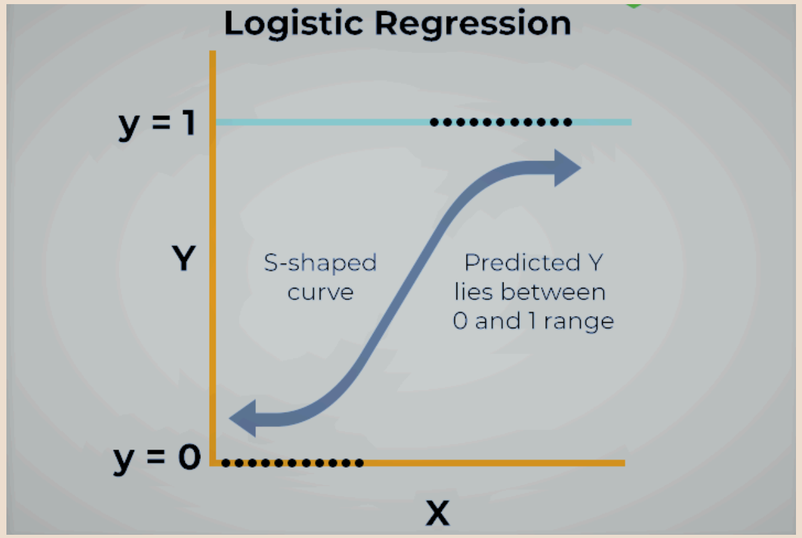
CONFUSION MATRIX & ROC CURVE





LOGISTIC REGRESSION

Logistic Regression is commonly used in churn prediction due to its simplicity, interpretability, and effectiveness in handling binary classification tasks.
Logistic Regression provides a probabilistic output, which means it predicts the likelihood of a customer churning. This is valuable for businesses as they can prioritize interventions for customers with higher probabilities of churning.

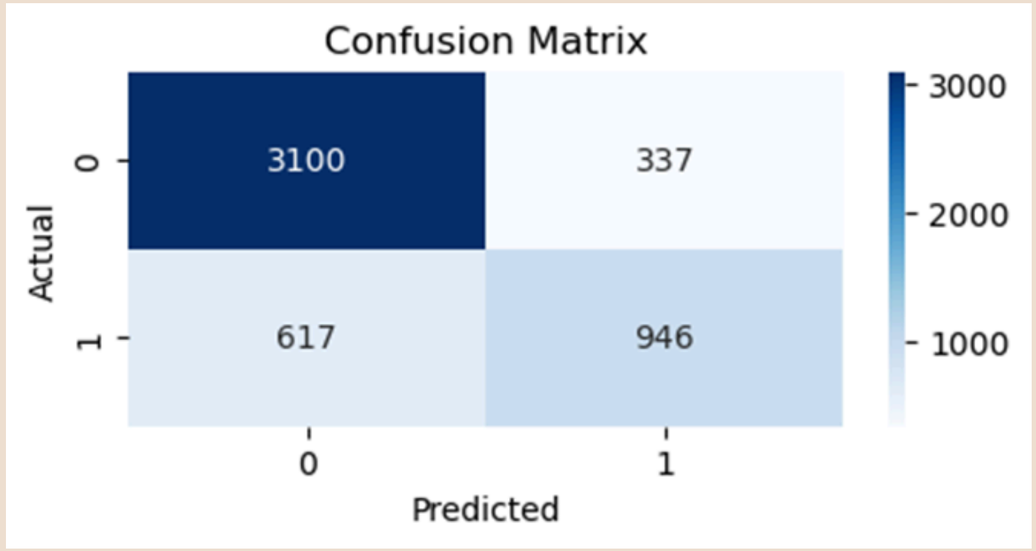


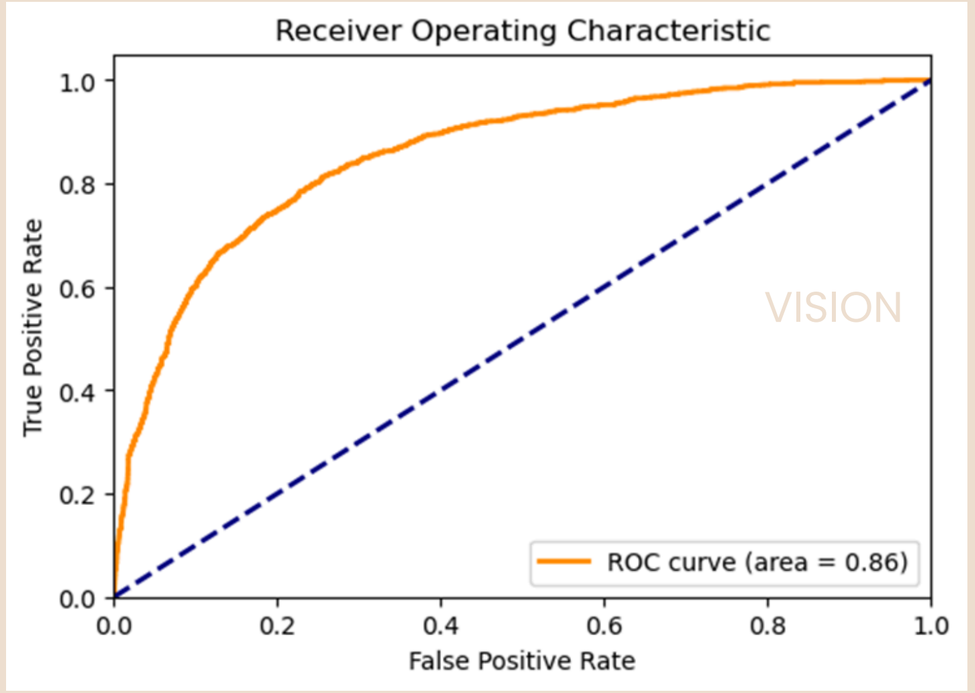
CODE SNIPPET & RESULT

```python
# Step 1: Prepare the data
target_column = 'target'
X = df.drop(columns=[target_column])
y = df[target_column]

# Step 2: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Train the logistic regression model
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train, y_train)

# Step 4: Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.81
              precision    recall  f1-score   support

           0       0.83      0.90      0.87      3437
           1       0.74      0.60      0.66      1563

    accuracy                           0.81      5000
   macro avg       0.78      0.75      0.76      5000
weighted avg       0.80      0.81      0.80      5000
```

CONFUSION MATRIX & ROC CURVE





KEY DIFFERENCE

PARAMETERS	LOGISTIC REGRESSION	RANDOM FOREST	DECISION TREE
Training Accuracy	90%	82%	80%
Testing Accuracy	81%	80%	78%
Precision	(0)83% (1)74%	(0) 83% (1)71%	(0)82% (1)67%
Recall	(0)90% (1)60%	(0) 90% (1)57%	(0)89% (1)53%

Final Model Selection: Logistic Regression

After extensive evaluation and hyperparameter tuning, we compared the performance of three models: Decision Tree, Random Forest, and Logistic Regression. The Decision Tree model achieved an accuracy of 78%, while the Random Forest model improved upon this with an accuracy of 80%. However, Logistic Regression emerged as the top performer with an accuracy of 81%.

CONCLUSION

In our project, we evaluated Decision Tree, Random Forest, and Logistic Regression models. Despite tuning, Decision Tree achieved 78% accuracy and Random Forest 80%. Logistic Regression excelled with 81% accuracy. Its high accuracy, efficiency with large datasets, simplicity, and robustness against overfitting make it the best choice for our project. Additionally, Logistic Regression's ease of implementation and clear interpretability allow for better understanding and communication of results. Consequently, Logistic Regression is our final model, offering a balanced solution in performance, interpretability, and computational efficiency. This ensures our project meets its objectives with reliable and actionable insights.

THANK YOU !