



18th July 2024



Telecom Churn Prediction

Infosys Springboard
Telecom Churn prediction modelling
Batch2

Presented By Garima kumari
Mentor: Bhaskar Naidu

Agenda

GitHub :



https://github.com/Garima1512/Batch2_Churn_Modeling_On_Telecom_Data

1. About telecom churn prediction
2. Our Mission and Vision
3. Our Goals
4. Our Milestones
5. Challenges
6. customers who have churned
6. Data Cleaning & Preprocessing
7. ML Model Building
8. Key Difference
9. Hyperparameter Tuning
10. Confusion Matrix & The ROC Curve
11. Conclusion



About Telecom Churn Prediction

▶ **“Telecom Churn Prediction”** aims to empower telecom companies with actionable insights and tools to reduce churn, retain customers, and improve overall business performance in a highly competitive **telecommunications market.**

Our Mission and Vision

Mission

Telecom churn prediction contributes to the overall success and competitiveness of telecom companies in a rapidly evolving market landscape.

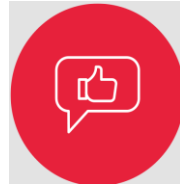
Vision

Telecom churn prediction can drive significant improvements in customer satisfaction, business performance, and competitiveness within the telecommunications sector.

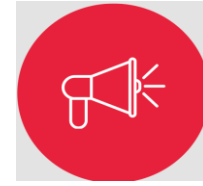
Our Goal



Minimize Churn Rate



Maximize Customer Retention



Optimize Business Performance

Our Milestones



1st

Data gathering &
understanding the dataset.

2nd

Data Cleaning and Pre-
processing.

3rd

Machine Learning model
building.

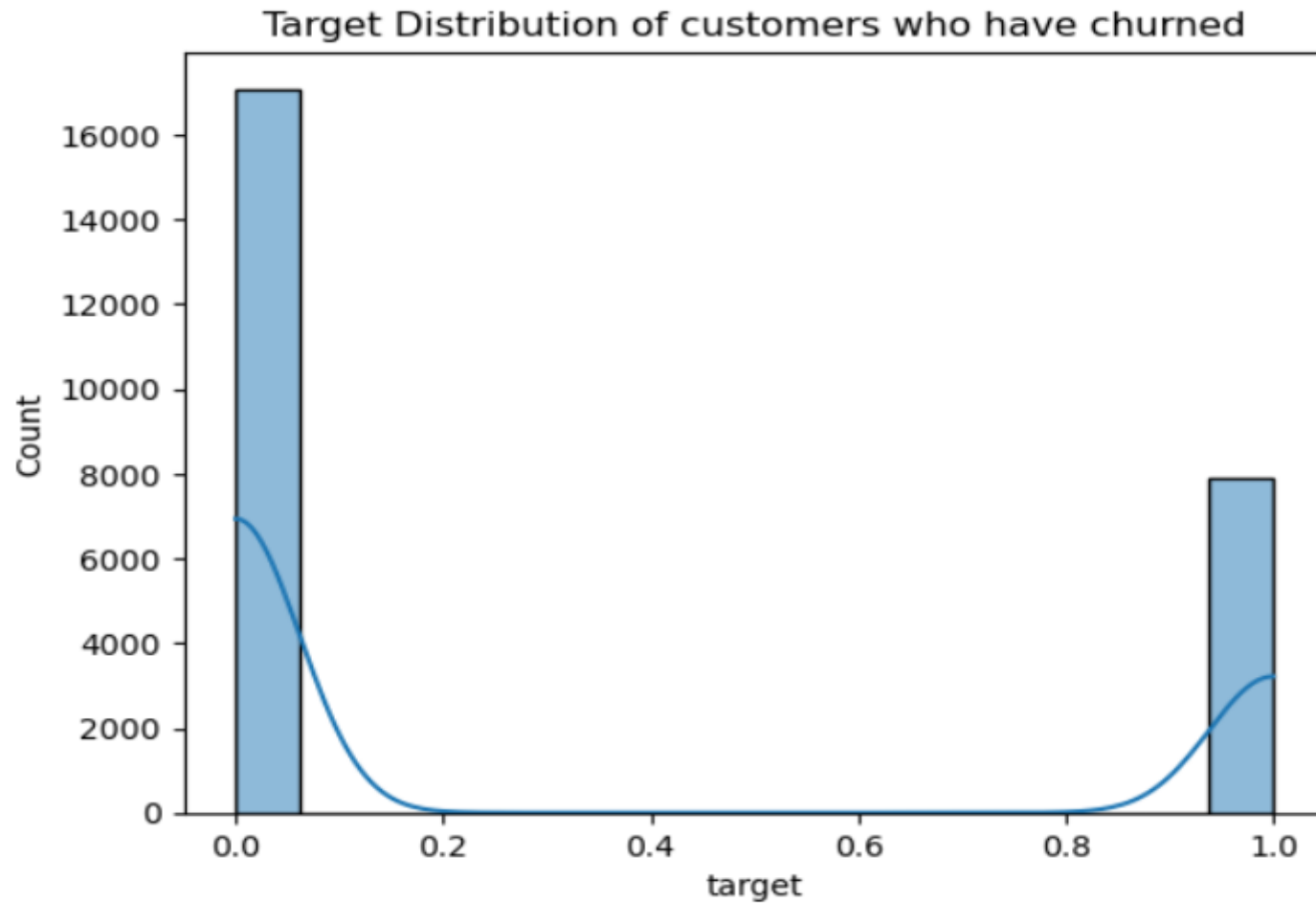
4th

Hyperparameter Tuning, ROC
Curve & Confusion Matrix

Challenges :

- **Data Quality and Integration:** Telecom companies often have vast amounts of data stored across multiple systems, including customer demographics, call logs, usage patterns, and billing information. Integrating and cleansing this data for analysis can be challenging, and poor data quality can lead to inaccurate predictions.
- **Imbalanced Data:** Telecom datasets are often imbalanced, with a small percentage of customers churning compared to those who remain. Imbalanced data can bias predictive models and lead to inaccurate churn predictions.
- **Dynamic Customer Behavior:** Customer behavior and preferences evolve over time, making it challenging to build accurate predictive models that adapt to changing patterns and trends.
- **Scalability:** Telecom companies serve large customer bases, and churn prediction systems must be scalable to handle massive volumes of data and real-time predictions.
- **Real-Time Prediction:** Implementing real-time churn prediction systems that can provide timely interventions to prevent churn is challenging due to the need for fast data processing and decision-making.

Target Distribution of customers who have churned



Data Cleaning & Pre-Processing

Convert datatypes of variables which are misclassified.

- ensures accurate analysis and efficient processing.

Removing duplicate records.

- ensure accuracy, maintain integrity, and optimize efficiency in data management and analysis.

Removing unique value variables.

- necessary to avoid redundancy, improve efficiency, and ensure meaningful analysis.

Removing Zero variance variables.

- necessary to eliminate redundant information, improve model stability, and simplify interpretation.



Data Cleaning & Pre-Processing

Outlier Treatment.

- necessary to maintain data integrity, improve model performance, and ensure robust and interpretable analyses.

Missing value Treatment.

- essential for maintaining data integrity, improving model performance, and ensuring accurate and statistically valid analyses.

Removing the highly correlated variables.

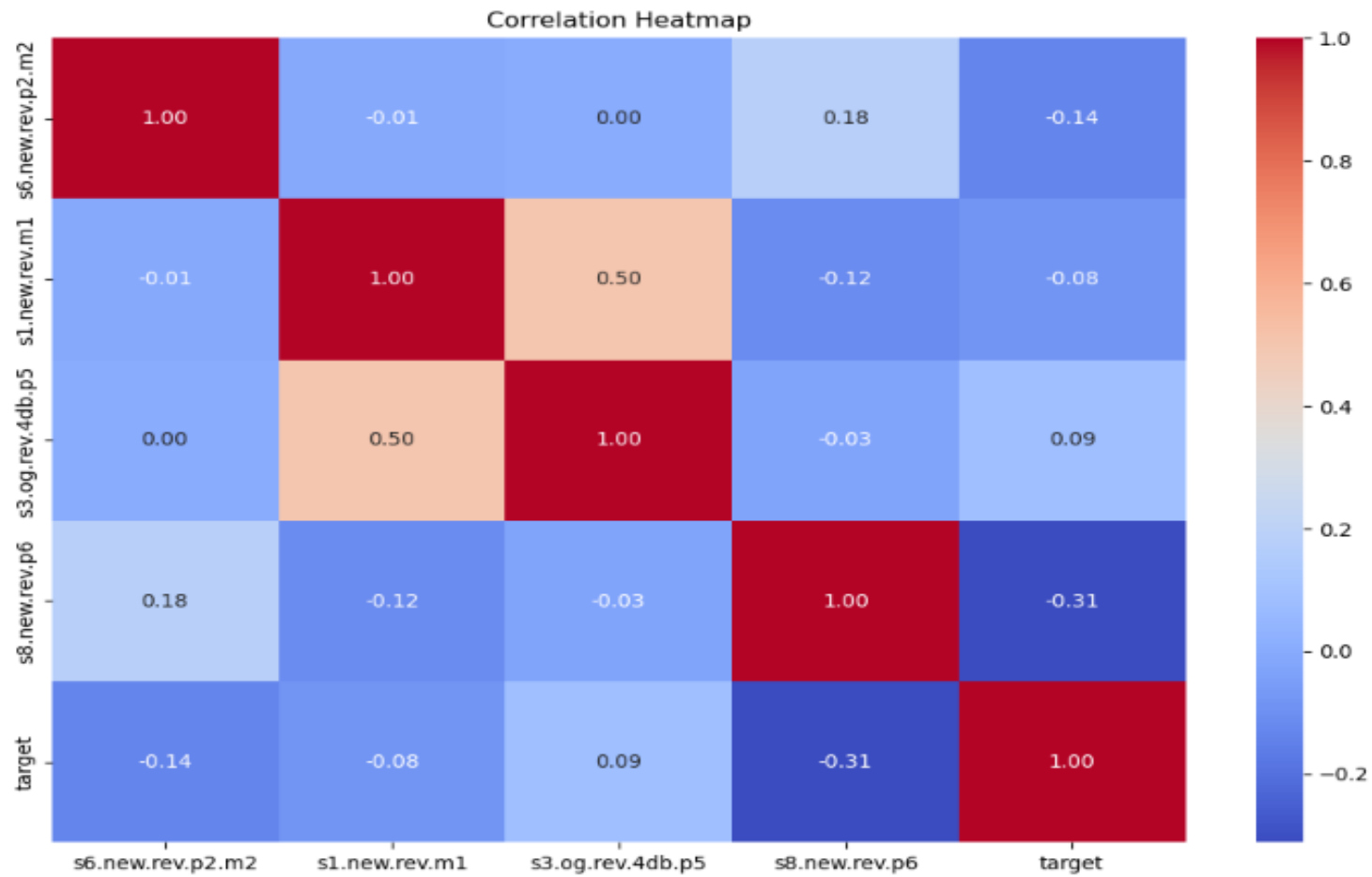
- necessary to reduce redundancy, improve model stability, and enhance efficiency in model training and interpretation.

Multicollinearity ($VIF > 5$).

- crucial to ensure accurate, stable, and interpretable models.



Heatmap of the correlation matrix



Machine Learning Model Building



Logistic Regression



Random Forest



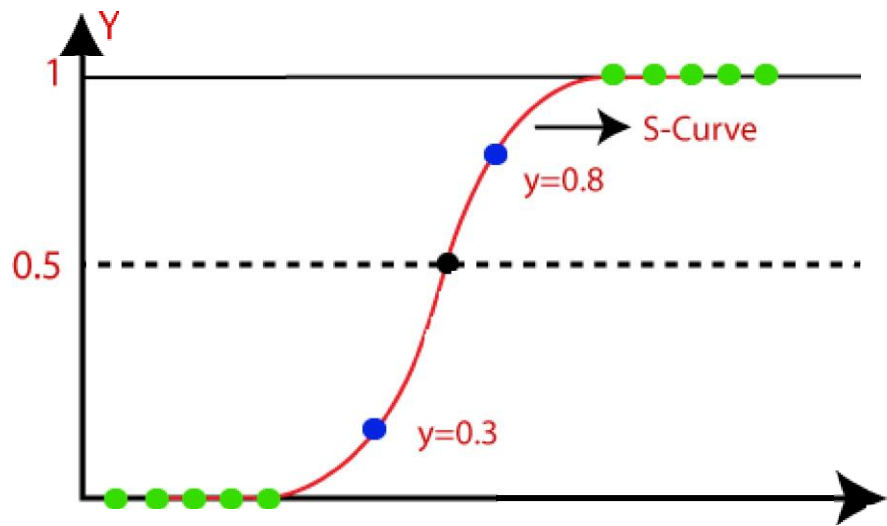
Decision Tree



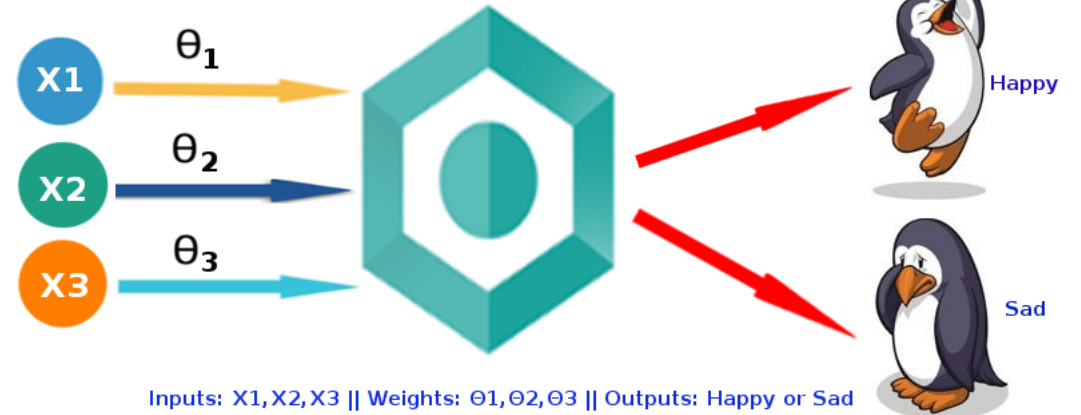
Logistic Regression

- Logistic regression is a type of supervised machine learning algorithm used for binary classification problems, where the target variable is a binary outcome (0 or 1). It's an extension of linear regression, but instead of predicting a continuous value, it predicts the probability of the binary out-come.
- Logistic regression is crucial for predicting binary outcome, offering interpretable results, probabilistic predictions, efficiency, and serving as a foundation for more complex models.

Pictures:



Logistic Regression Model



Code

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

logreg = LogisticRegression(max_iter=5000, solver="liblinear")

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

acc_prec = accuracy_score(y_test, y_pred)*100
print("\nAccuracy Percentage:", round(acc_prec, 3), "%")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

Accuracy Percentage: 78.882 %

Confusion Matrix:

```
[[2285  226]
 [ 518  494]]
```

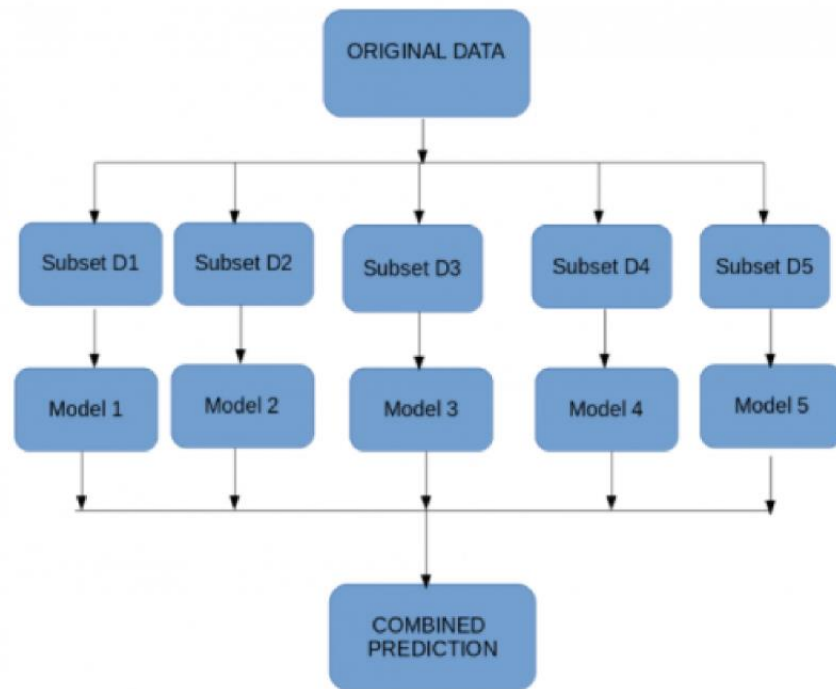
Classification Report:

	precision	recall	f1-score	support
0	0.82	0.91	0.86	2511
1	0.69	0.49	0.57	1012
accuracy			0.79	3523
macro avg	0.75	0.70	0.72	3523
weighted avg	0.78	0.79	0.78	3523

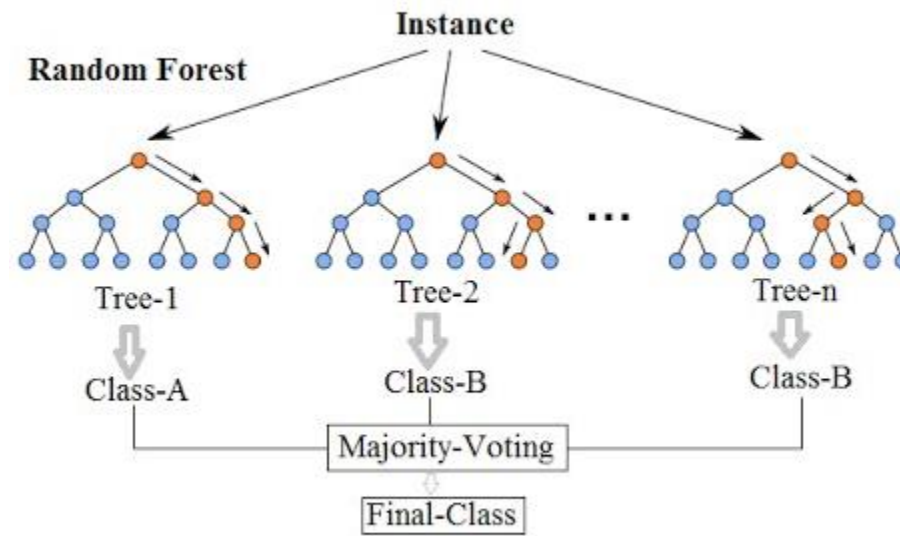
Random Forest

- Random Forest is a supervised learning algorithm that combines multiple decision trees to improve the accuracy and robustness of predictions. It is a popular ensemble learning method that is widely used in classification and regression tasks.
- Random Forests enhance accuracy and robustness through ensemble learning, resist overfitting, highlight feature importance, handle missing values, scale well with large datasets, and are versatile for both classification and regression.

Pictures:



Random Forest Simplified



Code

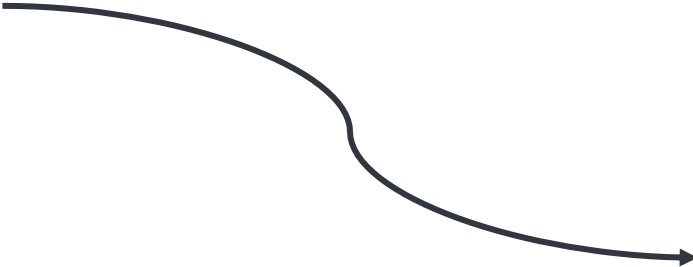
```
rf_model = RandomForestClassifier(n_estimators=100,random_state=42)
rf_model.fit(X_train,y_train)
y_pred = rf_model.predict(X_test)

initial_y_pred=rf_model.predict(X_test)

print("Initial Model Confusion Matrix:")
print(confusion_matrix(y_test,initial_y_pred))

print ("\nInitial Model Classification Report:")
print (classification_report(y_test, initial_y_pred))

print("\nInitial Model Accuracy Score:")
print(accuracy_score(y_test,initial_y_pred))
acc_prec = accuracy_score(y_test, initial_y_pred)* 100
print ("\n Accuracy Percentage:", round(acc_prec, 3), "%")
```



Initial Model Confusion Matrix:

```
[[2286  225]
 [ 528  484]]
```

Initial Model Classification Report:

	precision	recall	f1-score	support
0	0.81	0.91	0.86	2511
1	0.68	0.48	0.56	1012
accuracy			0.79	3523
macro avg	0.75	0.69	0.71	3523
weighted avg	0.78	0.79	0.77	3523

Initial Model Accuracy Score:

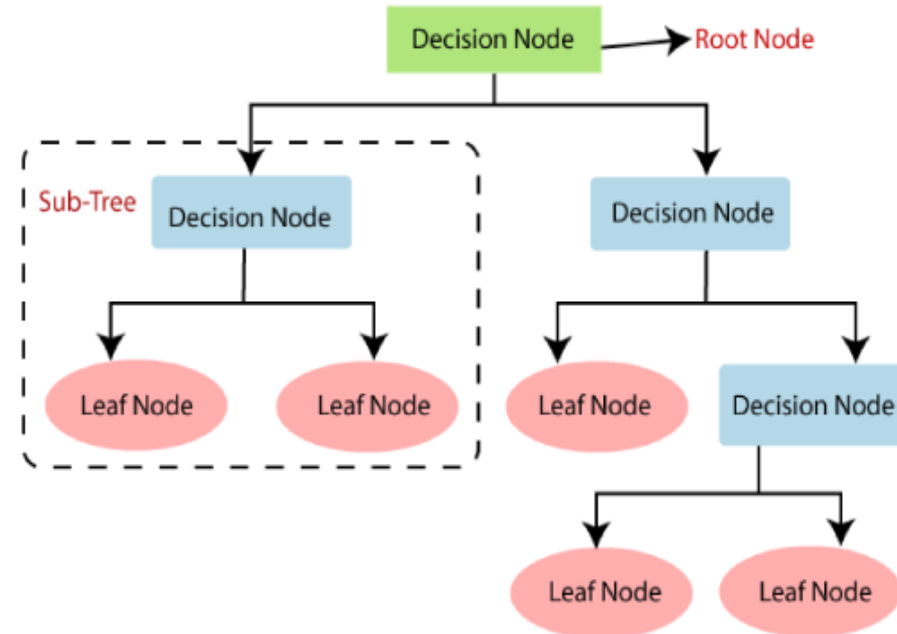
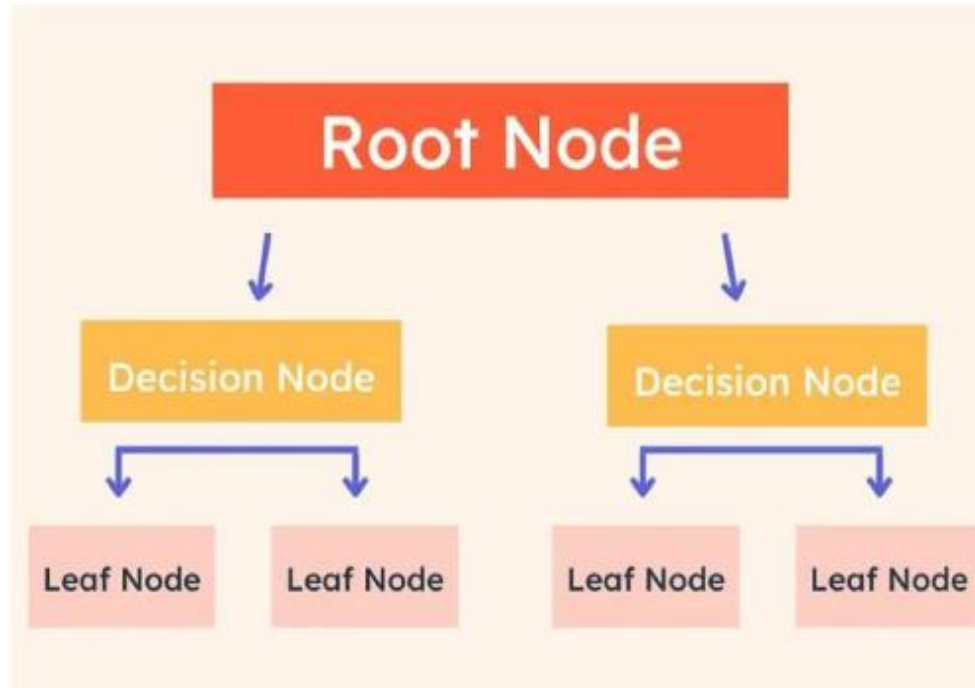
0.7862617087709338

Accuracy Percentage: 78.626 %

Decision Tree

- Decision trees in machine learning are a supervised learning algorithm that enables developers to analyze the possible consequences of a decision and predict outcomes for future data. A decision tree is a tree-like model that starts at the root and branches out to demonstrate various outcomes.
- Decision trees are crucial for their simplicity, ability to handle non-linear relationships, feature importance identification, and robustness to outliers, applicable to both classification and regression tasks.

Pictures:



Code

```
X = Churn_Data_Clean.drop(columns = ['target'],axis=1)
y = Churn_Data_Clean['target']

X_train,X_test,y_train,y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=42)

initial_model = DecisionTreeClassifier(random_state = 42)
initial_model.fit(X_train,y_train)

initial_y_pred = initial_model.predict(X_test)

print ("initial Model Confusion Matrix:")
print (confusion_matrix(y_test,initial_y_pred ))

print ("\ninitial Model Classification Report:")
print (classification_report(y_test, initial_y_pred))

print ("\ninitial Model Accuracy Score:")
print (accuracy_score(y_test, initial_y_pred))
acc_prec = accuracy_score(y_test, initial_y_pred)* 100
print ("\nAccuracy Percentage:", round(acc_prec, 3), "%")
```

initial Model Confusion Matrix:

```
[[1967  544]
 [ 511  501]]
```

initial Model Classification Report:

	precision	recall	f1-score	support
0	0.79	0.78	0.79	2511
1	0.48	0.50	0.49	1012
accuracy			0.70	3523
macro avg	0.64	0.64	0.64	3523
weighted avg	0.70	0.70	0.70	3523

initial Model Accuracy Score:
0.7005393130854386

Accuracy Percentage: 70.054 %

Key Difference:

Parameters	Logistic Regression	Decision Tree	Random Forest
Accuracy	78.882 %	70.054 %	78.626 %
Precision	(0) 82 % (1) 69 %	(0) 79 % (1) 48 %	(0) 81 % (1) 68 %
Recall	(0) 91 % (1) 49 %	(0) 64 % (1) 70 %	(0) 91 % (1) 48 %

Why Logistic Regression?

Logistic Regression offers a several advantages: They deliver a higher accuracy resist overfitting highlight feature importance and effective handle noise and missing values. Logistic regression is a statistical model used for classification and predictive analytics. It estimates the probability of an event occurring based on a given dataset of independent variables. After evaluating all three models, we selected logistic regression due to its superior performance.

Hyperparameter Tuning

What is Hyperparameter Tuning ?

Hyperparameter Tuning is the process of finding the best settings for a machine learning model to improve its performance. This involves adjusting parameters that control the learning process, like the number of trees in a Random Forest or the learning rate in a neural network. By testing different combinations of these parameters, we identify the ones that yield the best results for our specific data.



Importance:

- **Improves Model Performance:** Proper tuning can significantly enhance the predictive accuracy and generalization of a model.
- **Reduces Overfitting/Underfitting:** Helps in finding the balance between bias and variance, ensuring the model performs well on unseen data.
- **Optimizes Computational Resources:** Efficient tuning can lead to faster and more efficient model training and deployment.

Code:

```
param_grid = {
    'solver': ['liblinear', 'saga'],
    'penalty': ['l1', 'l2'],
    'C': [0.01, 0.1, 1, 10, 100],
    'max_iter': [1000, 2000, 5000]
}

logreg = LogisticRegression()

grid_search = GridSearchCV(logreg, param_grid, cv=5, verbose=1, n_jobs=-1)

grid_search.fit(X_train_scaled, y_train)

best_params = grid_search.best_params_

print(f'Best parameters: {best_params}')

best_estimator = grid_search.best_estimator_
y_pred_best = best_estimator.predict(X_test_scaled)

accuracy_best = accuracy_score(y_test, y_pred_best)
conf_matrix_best = confusion_matrix(y_test, y_pred_best)
class_report_best = classification_report(y_test, y_pred_best)
```

Fitting 5 folds for each of 60 candidates, totalling 300 fits
Best parameters: {'C': 1, 'max_iter': 1000, 'penalty': 'l1', 'solver': 'saga'}
Accuracy of Best Model: 0.7885325007096224

Confusion Matrix of Best Model:

```
[[2280  231]
 [ 514  498]]
```

Classification Report of Best Model:

	precision	recall	f1-score	support
0	0.82	0.91	0.86	2511
1	0.68	0.49	0.57	1012
accuracy			0.79	3523
macro avg	0.75	0.70	0.72	3523
weighted avg	0.78	0.79	0.78	3523

Accuracy Percentage of Best Model: 78.853 %

Confusion Matrix:

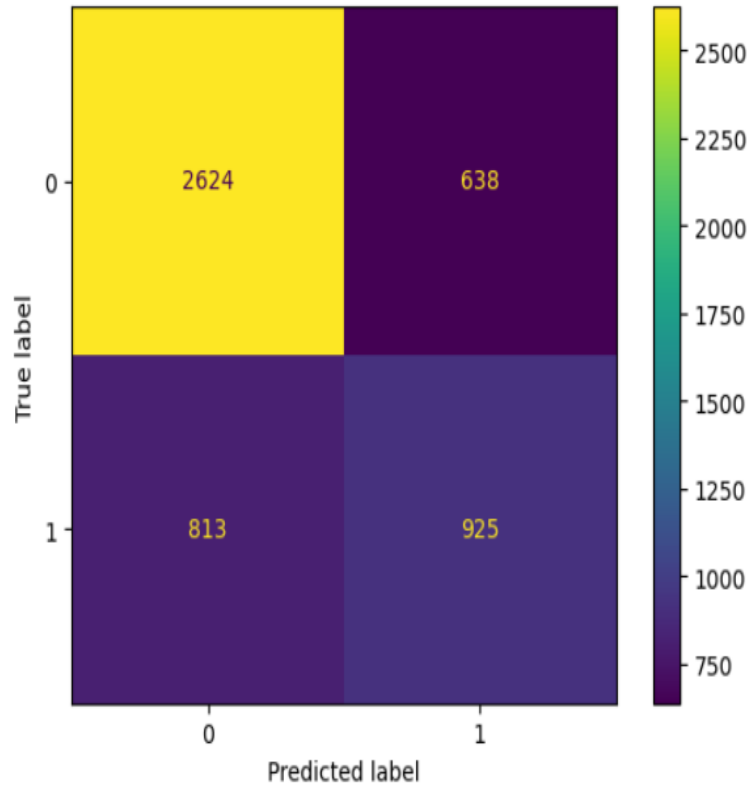
- What is Confusion Matrix ?
- A confusion matrix is a table that evaluates a classification model's performance by showing the counts of true positives, true negatives, false positives, and false negatives. It helps derive metrics like accuracy, precision, recall, and F1 score for a more detailed performance assessment.



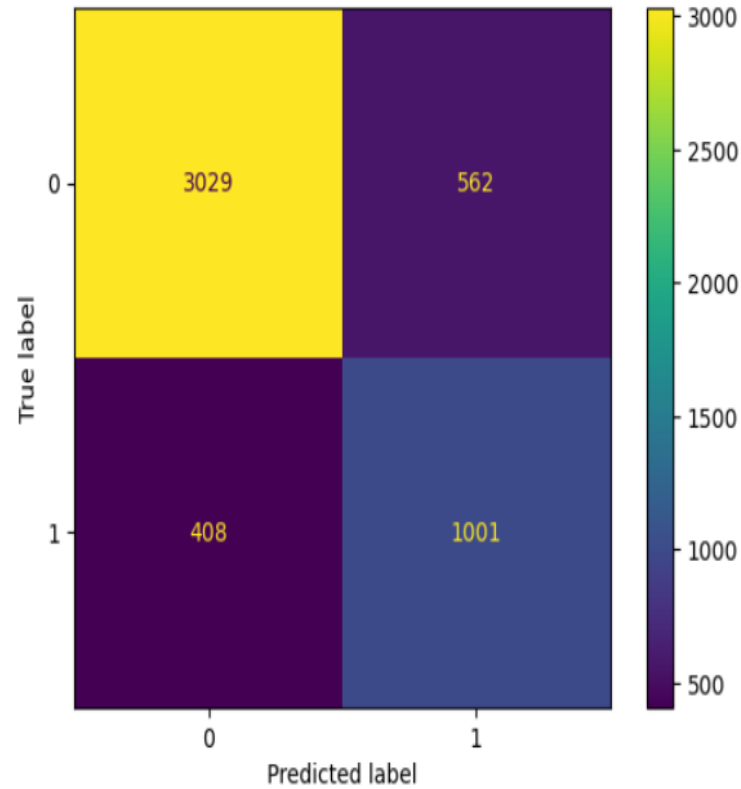
Components :

- **True Positives (TP):** The number of instances correctly predicted as positive.
- **True Negatives (TN):** The number of instances correctly predicted as negative.
- **False Positives (FP):** The number of instances incorrectly predicted as positive (Type I error).
- **False Negatives (FN):** The number of instances incorrectly predicted as negative (Type II error).

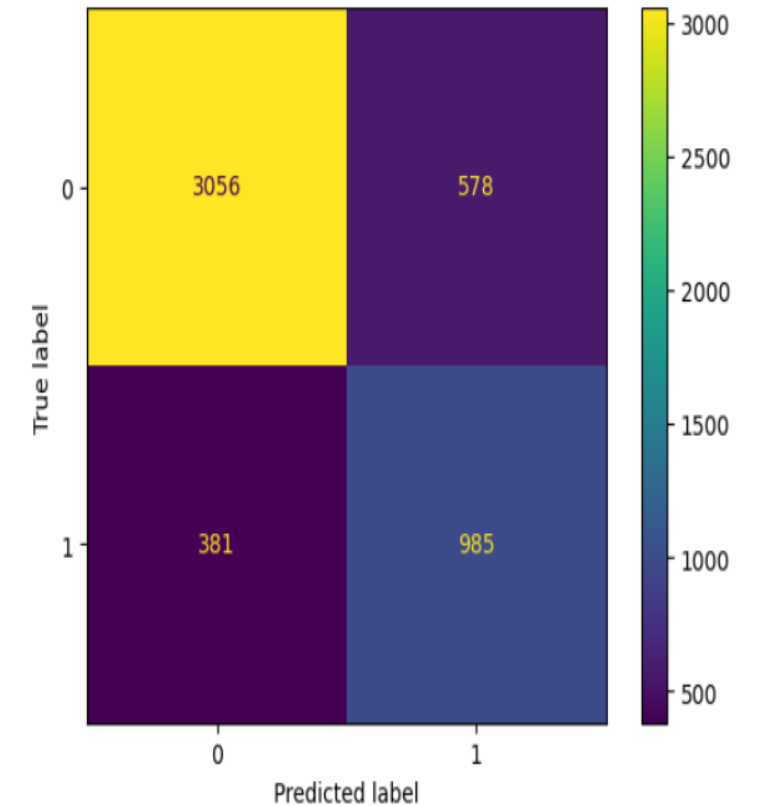
Pictures:



Decision Tree



Logistic Regression



Random Forest

The ROC Curve

What is ROC Curve ?

The ROC curve is a graphical representation of the True Positive Rate against the False Positive Rate.

It is a powerful tool for evaluating the performance of different classification models, to choose the best threshold for a given model, and to assess the overall quality of a model.



ROC Curve Code:

```
# Predict probabilities with the best estimator
y_pred_proba = best_model.predict_proba(X_test)[: , 1]

# Compute ROC curve
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_pred_proba)
auc_score_lr = roc_auc_score(y_test, y_pred_proba)

plt.figure(figsize=(10, 6))

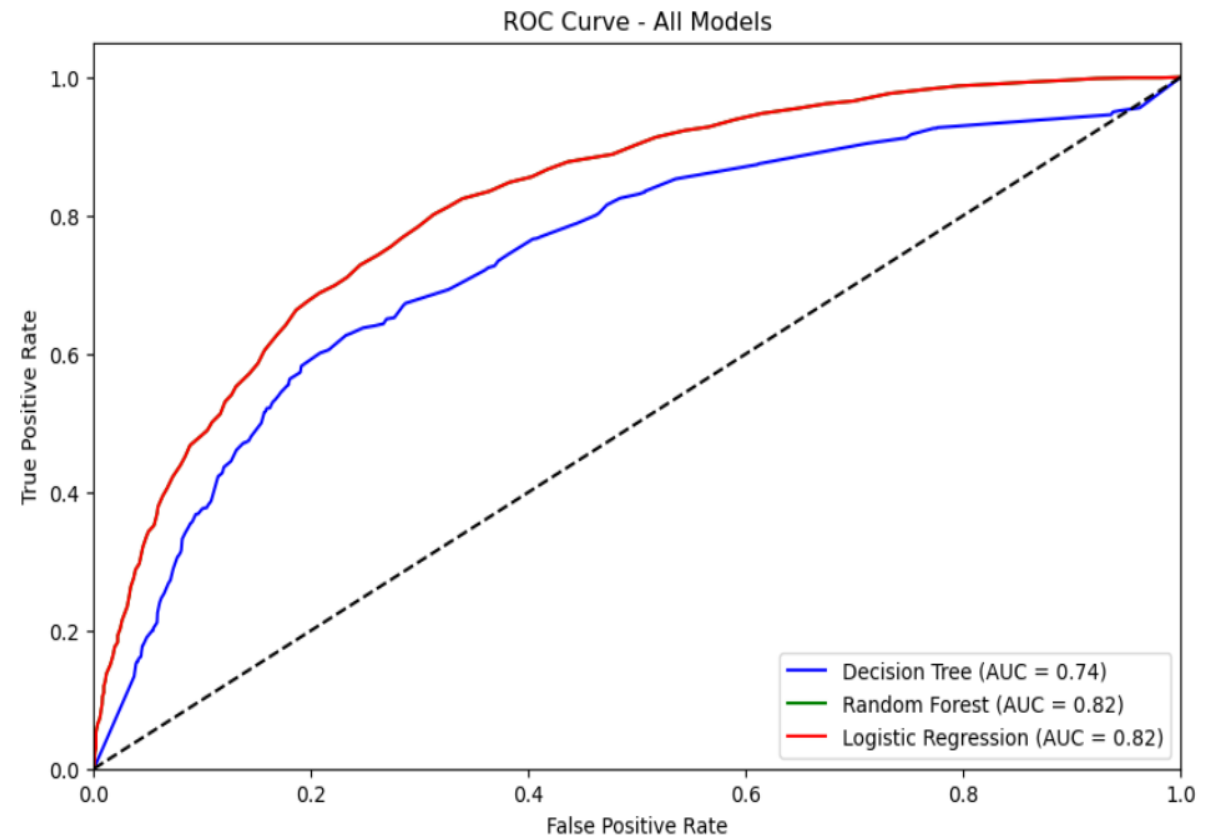
# Plot Decision Tree ROC curve
plt.plot(fpr_dt, tpr_dt, label=f'Decision Tree (AUC = {auc_score_dt:.2f})', color='blue')

# Plot Random Forest ROC curve
plt.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {auc_score_rf:.2f})', color='green')

# Plot Logistic Regression ROC curve
plt.plot(fpr_lr, tpr_lr, label=f'Logistic Regression (AUC = {auc_score_lr:.2f})', color='red')

# Plot diagonal line
plt.plot([0, 1], [0, 1], 'k--')

# Set plot attributes
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - All Models')
plt.legend(loc='lower right')
plt.show()
```



Conclusion:

- **Improved Customer Retention:** By accurately predicting which customers are likely to churn, telecom companies can implement targeted strategies to retain valuable customers, ensuring their satisfaction and loyalty.
- **Cost Reduction:** Preventing churn is more cost-effective than acquiring new customers. Churn prediction allows for efficient resource allocation and reduces costs associated with customer attrition.
- **Enhanced Customer Experience:** Understanding customer behavior and preferences enables telecom providers to offer personalized experiences, address concerns proactively, and significantly improve overall satisfaction.
- **Increased Revenue:** Retaining existing customers and maximizing their life time value leads to steady revenue streams and sustainable business growth.
- **Competitive Advantage:** Effective churn prediction and management provide telecom companies with a competitive edge through superior service, tailored offers, and proactive retention initiatives.

