Infosys Springboard

Telecom Churn Prediction

By Gauri Bhardwaj (Batch 2)

Agendas

Code & Documentation Link

1) Introduction to Churn Prediction

2) Missions and Vision

3) Our Goals

4) Project Milestones

5) Challenges

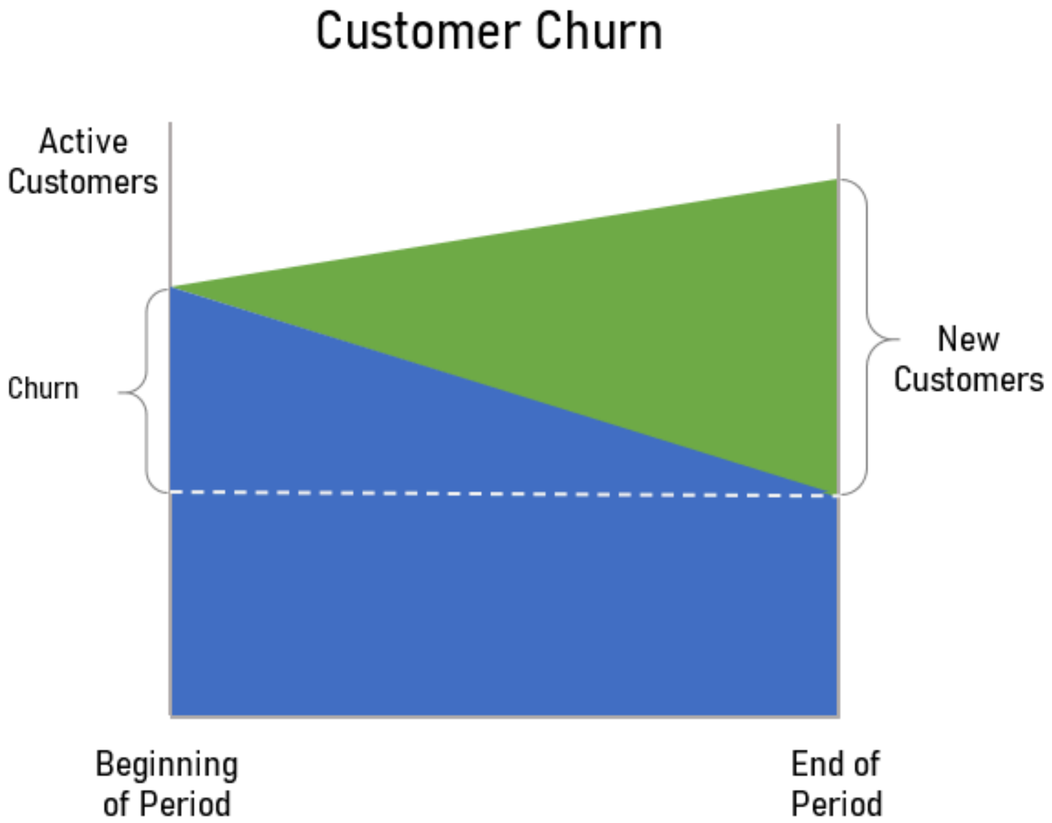
6) Customer Churn Model Workflow

7) The Machine Learning Process

8) ML Model Building

9) Hyperparameter Tuning

10) Model Evaluation & Conclusion



What is CHURN RATE?

Customer churn is the **percentage of customers that stopped using your company's product or service** during a certain time frame.

Churn rate is a key indicator of customer satisfaction.

- A **low** churn rate signifies satisfied customers
- A **high** churn rate indicates that customers are leaving

Churn serves as a valuable **measure of growth potential**. It tracks the customers you lose, while growth rates track the new customers you gain. Analyzing these metrics together reveals your business's overall growth. If your growth rate exceeds your churn rate, your business is expanding. Conversely, if churn surpasses growth, your business is contracting.

Goals

- Reduce Customer Churn
- Enhance customer satisfaction
- Optimize Marketing Efforts
- Increase Revenue
- Improve Competitive Edge

Vision

Customer satisfaction, business performance, and competitiveness in the telecom industry can all be significantly enhanced by telecom churn prediction.

Mission

In a market that is changing quickly, Telecom Churn Prediction helps telecom firms remain competitive and successful overall.

Project Milestones

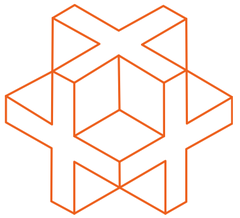
1	2	3	4
WEEK 1 AND 2	WEEK 3 AND 4	WEEK 5 AND 6	WEEK 7 AND 8
Data Collection and Data Wrangling	**Data Cleaning and Pre-Processing**	**Machine Learning and Model Implementation**	**Hyperparameter Tuning and Model Evaluation**

Challenges

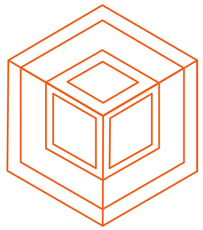
Data Quality & Availability

Ensuring the availability of high-quality, comprehensive, and up-to-date customer data for accurate predictions.

Imbalanced Data

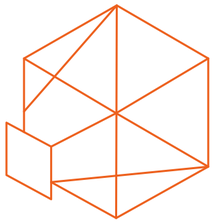


Dealing with the problem, where the no. of churned customers is smaller than the no. of retained customers.



Complex Customer Behavior

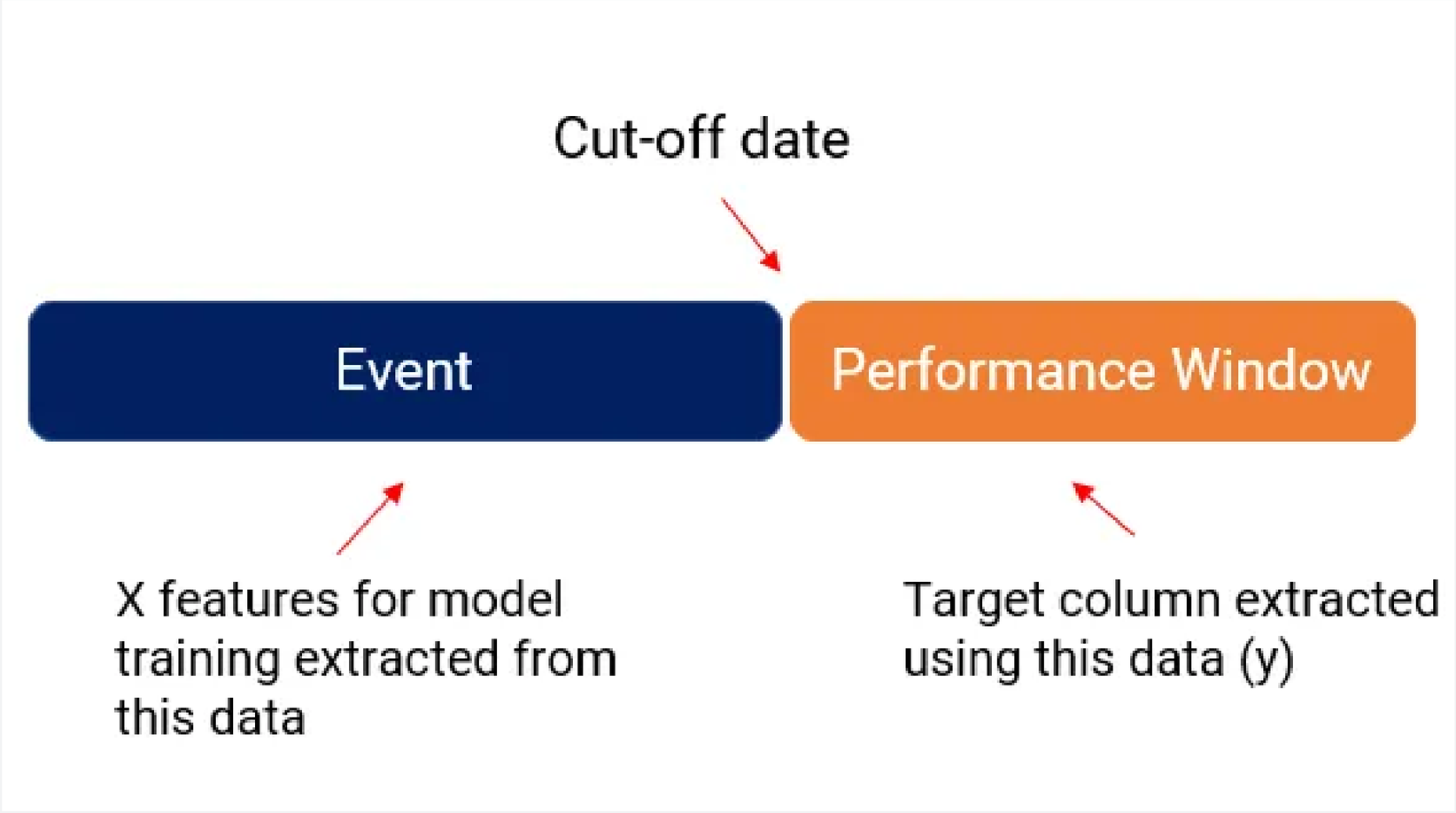
Understanding and modeling the complex and often unpredictable behavior of customers that leads to churn.



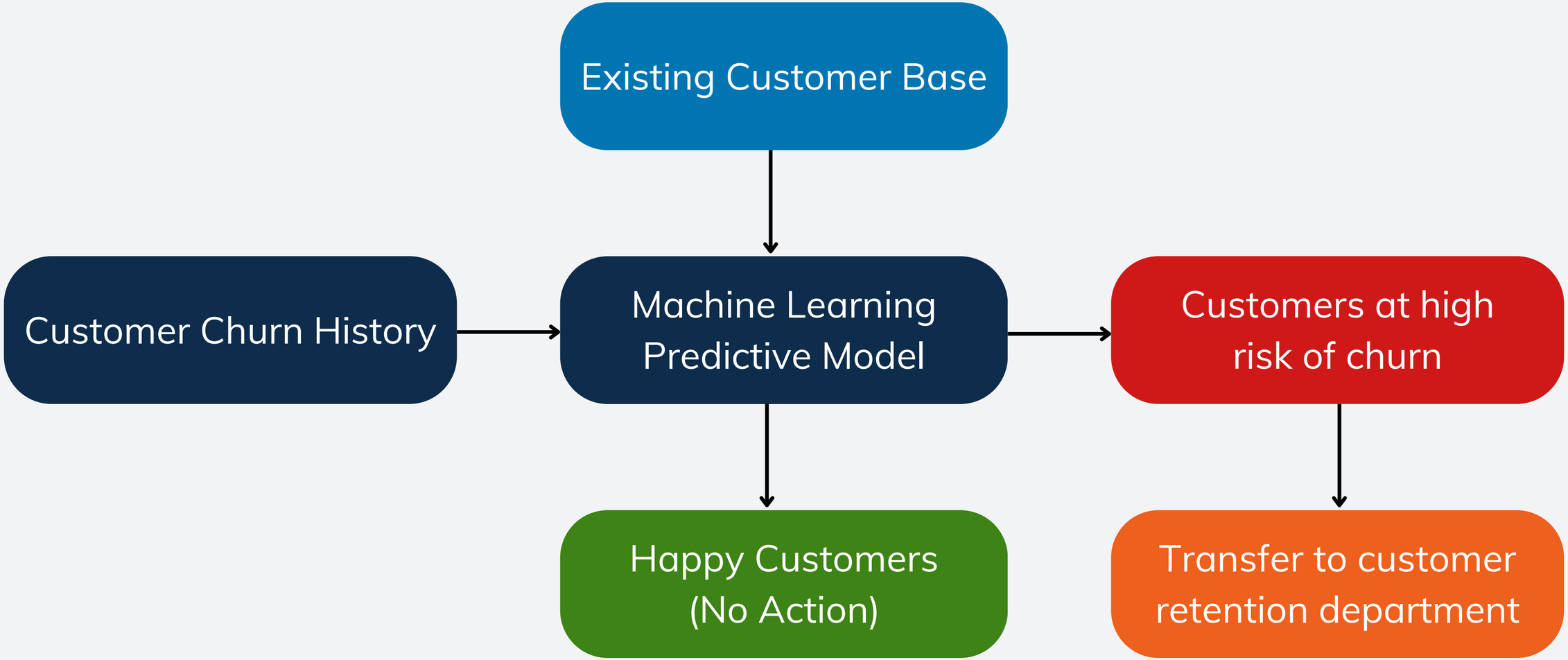
Dynamic Market Conditions

Adapting models to reflect the rapidly changing market conditions and evolving customer preferences.

Dataset Composition



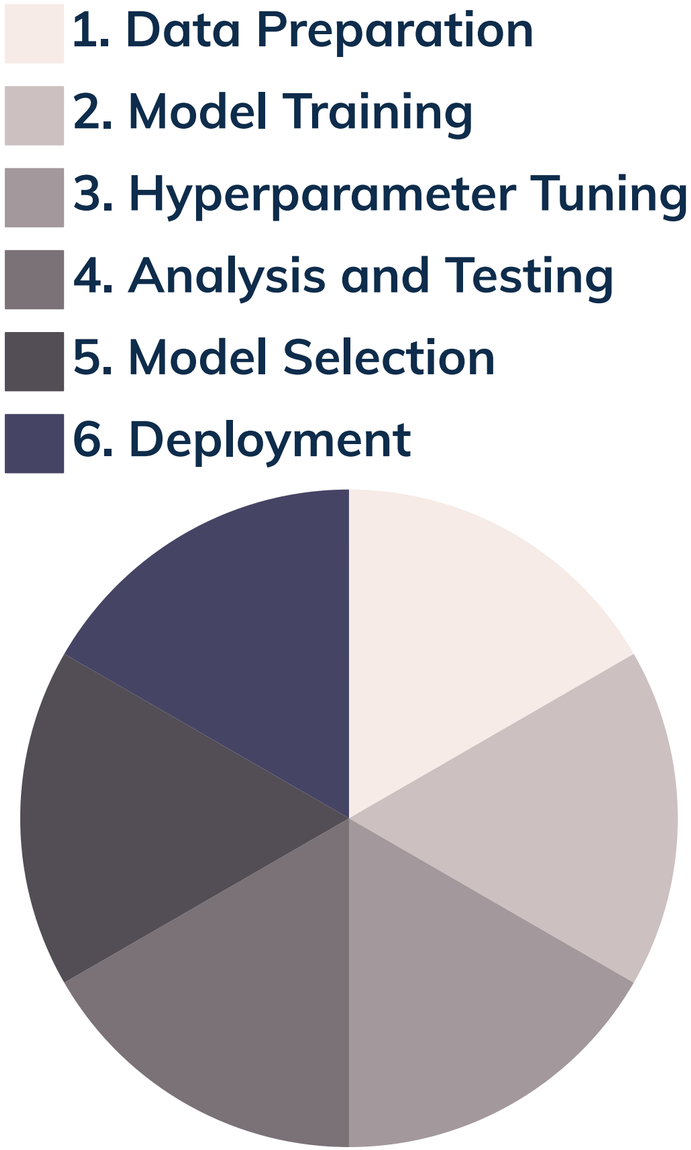
Customer Churn Model Workflow



The Machine Learning Process

Points for discussion

More and more companies realize that specific job titles and profiles can be accomplished while working from home.



Data Preparation

1) Data Gathering

- Identifying various data sources
- Collect Data
- Integrate data obtained from different sources

2) Data Exploration

- Understand quality, characteristics and nature of your data
- Find correlations, general trends and outliers

3) Data Wrangling

- Cleaning the data and converting raw data into usable format
- Deal with **missing values, invalid data, outliers, duplicates**
- Filtering Techniques of feature selection

4) Train- Test Split: CROSS-VALIDATION

- Technique for evaluating the performance of a machine learning algorithm.

- The procedure involves taking a dataset and dividing it into two subsets. The **training subset** is used to fit the model and the **testing subset** is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values.

Nevertheless, common split percentages include:

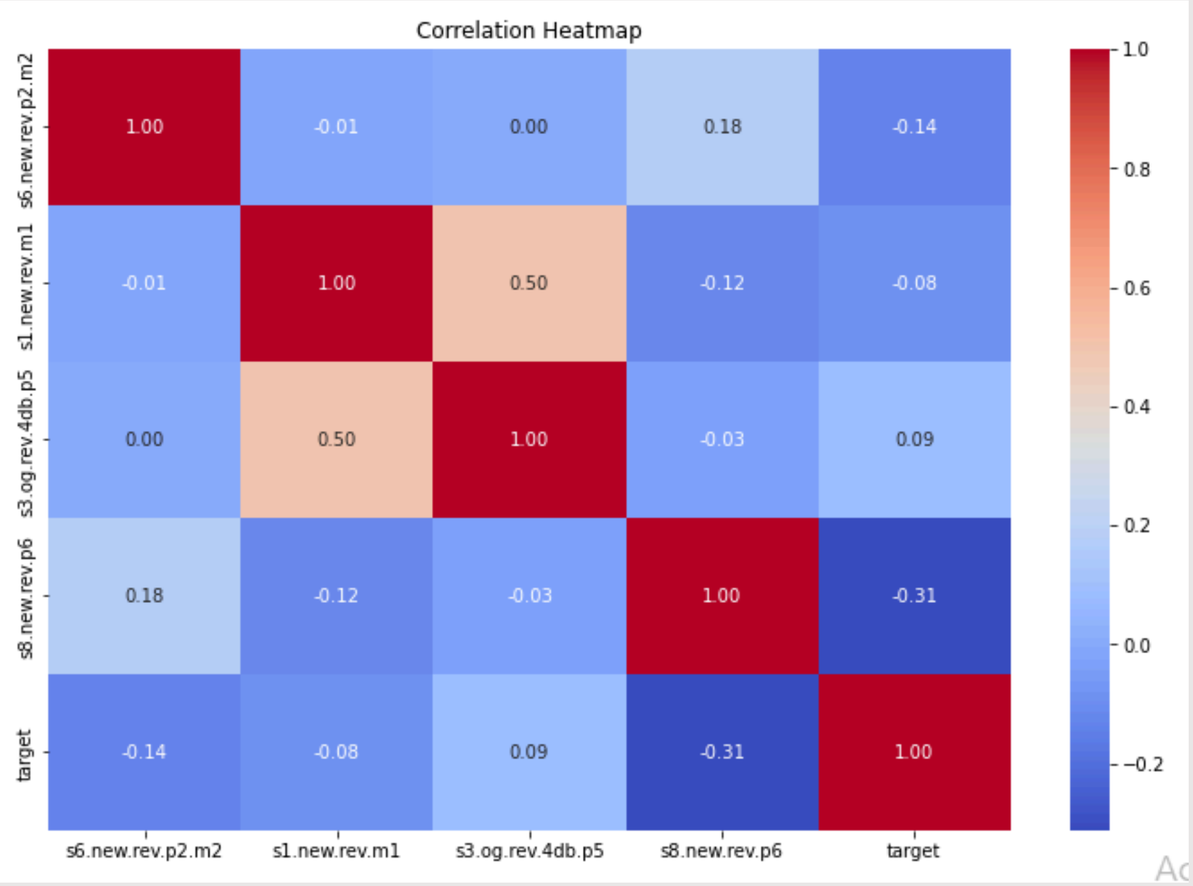
- Train: 80%, Test: 20%
- Train: 67%, Test: 33%

Data Preprocessing Steps

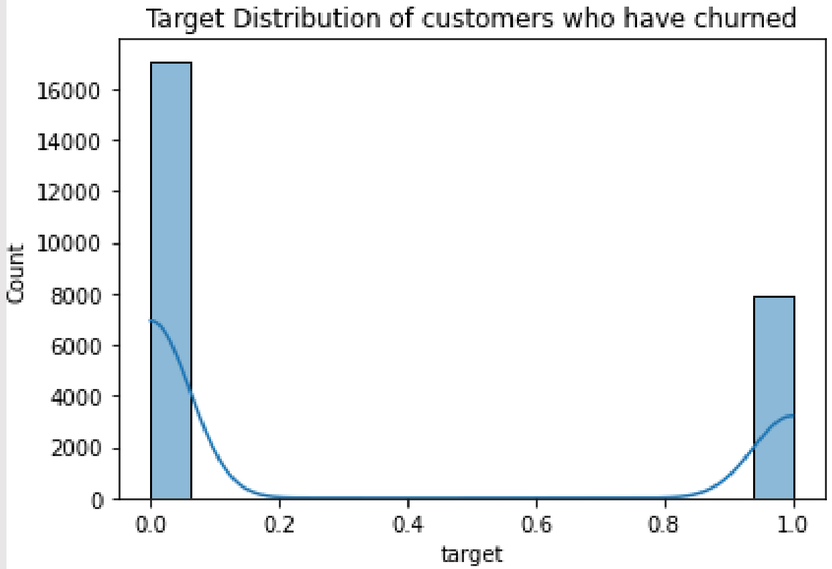
Data Pre-processing

1) Convert data type of variables which are misclassified.
2) Removing Duplicate records
3) Removing Unique value variables
4) Removing Zero variance variables
5) Outlier Treatment
    - Using Boxplot: Q3+(1.5*IQR) & Q1-(1.5*IQR)
    - Standardization: +/- 3 Sigma approach
    - Capping & Flooring
6) Missing Value Treatment
    - Remove records if NA's are less than 5%
    - Remove if NA's are 50% in any variable
    - Impute with Mean/Median, if variable is numeric and with Mode if variable is categorical
7) Removing the highly correlated variables
8) Multicollinearity (VIF > 5)

Understanding the Dataset




```python
# Distribution Plot for 'target' column
sns.histplot(data=churn_data, x="target", kde=True, multiple="stack")

# Display the plot
plt.title("Target Distribution of customers who have churned")
plt.show()
```



MODEL TRAINING

Model training is a crucial process in the field of Machine Learning, which involves **teaching a model to identify patterns, relationships, or trends within a given dataset** in order to generate meaningful insights. During the model training process, a dataset is carefully curated as the primary source of knowledge.

WRT SUPERVISED LEARNING

In this method, the training dataset is labeled, by **comparing the model's predictions with these labeled data points**, the algorithm identifies any discrepancies and adjusts the model accordingly. This iterative process continues until the model achieves a desired level of accuracy.

WRT UNSUPERVISED LEARNING

By **exploring the relationships between data points**, the unsupervised learning algorithm identifies clusters or groups that share similar characteristics.

CLASSIFICATION MODELS

1. LOGISTIC REGRESSION

It is a predictive analysis algorithm and based on the concept of **probability**.

Some of the examples of classification problems:

- Classifying Emails: **spam or not spam**,
- Classifying Online transactions: Fraud or not Fraud

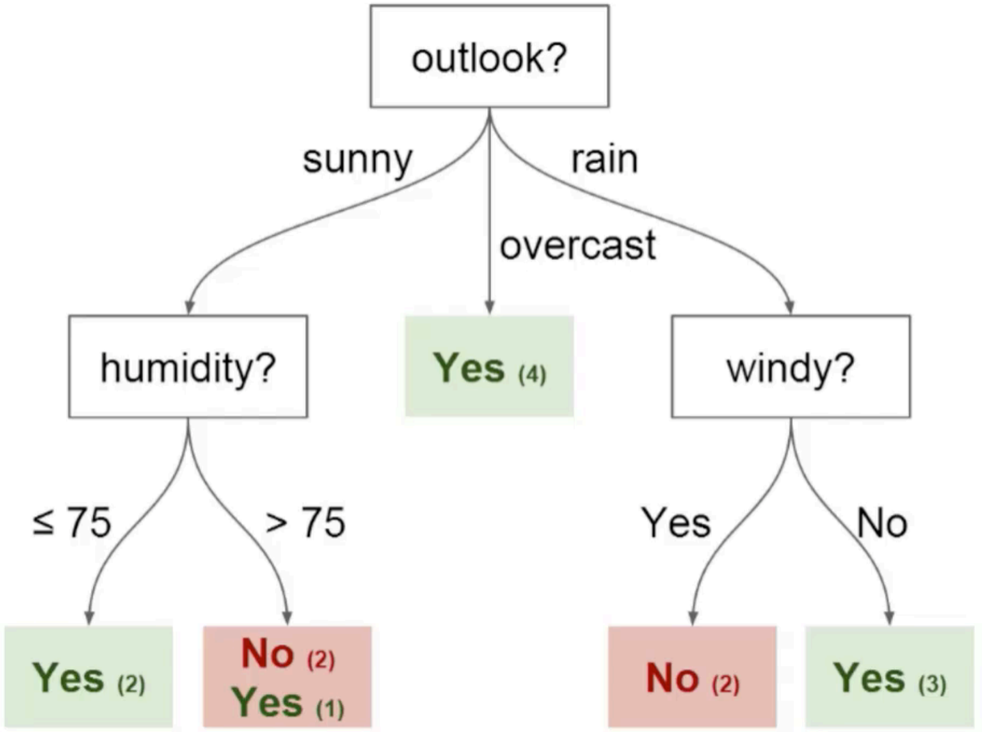
Logistic regression transforms its output using the logistic **sigmoid function** to return a probability value.

3. Logistic Regression

```python
from sklearn.preprocessing import StandardScaler

# Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize the Logistic Regression model with increased max_iter
logreg = LogisticRegression(max_iter=5000, solver="liblinear")

# Fit the model to the training data
logreg.fit(X_train, y_train)

# Predict on the test data
y_pred = logreg.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
Best Parameters: {'C': 0.01, 'max_iter': 1000, 'penalty': 'l1'
Accuracy of Best Model: 0.8028
Confusion Matrix of Best Model:
[[3092  345]
 [ 641  922]]
Classification Report of Best Model:
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      3437
           1       0.73      0.59      0.65      1563

    accuracy                           0.80      5000
   macro avg       0.78      0.74      0.76      5000
weighted avg       0.80      0.80      0.80      5000


Accuracy Percentage of Best Model: 80.28 %
```

2. DECISION TREES



A decision tree is a **supervised** machine learning algorithm used for both classification and regression tasks.

It models decisions and their possible consequences in a tree-like structure, consisting of **nodes** representing tests on attributes, **branches** representing the outcomes of these tests, and **leaf nodes** representing class labels or target values.

The main objective of a decision tree is to split the dataset into subsets based on the most significant attribute, creating a structure that mirrors human decision-making processes.

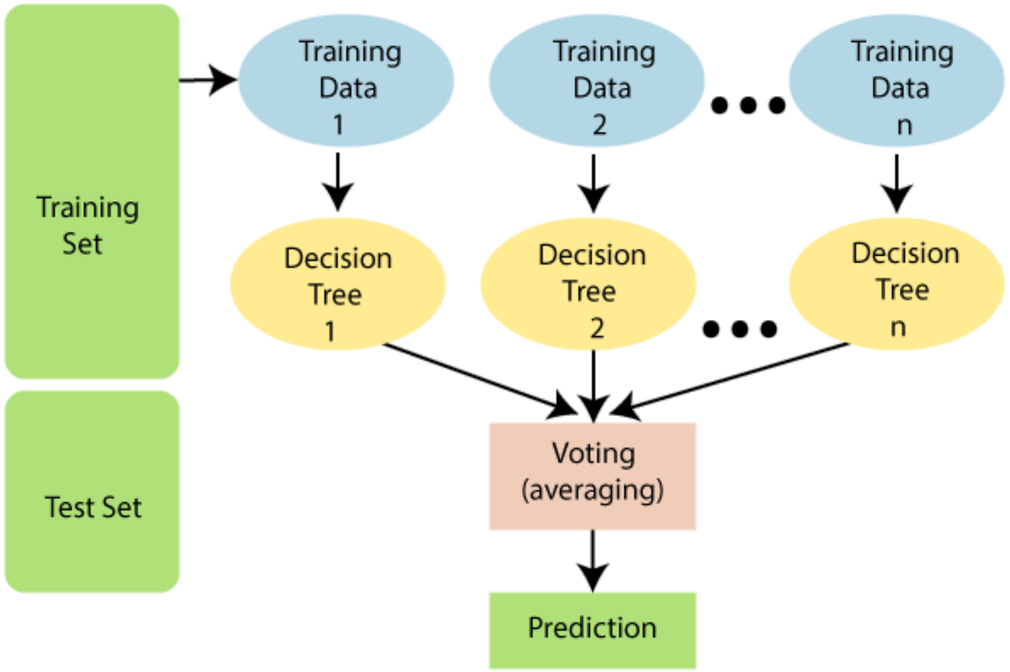
1. Decision Trees

In [18]:

```python
# Build and train the initial decision tree model
initial_model = DecisionTreeClassifier(random_state=42)
initial_model.fit(X_train, y_train)

# Make predictions with the initial model
initial_y_pred = initial_model.predict(X_test)

# Evaluate the initial model
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), "%")
```

```
Best Parameters found by GridSearchCV:
{'max_depth': 10, 'max_features': None}

Tuned Model Confusion Matrix:
[[2924  513]
 [ 615  948]]

Tuned Model Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.85      0.84      3437
           1       0.65      0.61      0.63      1563

    accuracy                           0.77      5000
   macro avg       0.74      0.73      0.73      5000
weighted avg       0.77      0.77      0.77      5000


Tuned Model Accuracy: 77.44 %
```

3. RANDOM FOREST



Random Forest is a **supervised** machine learning algorithm used for both classification and regression tasks. It operates by constructing a **multitude of decision trees** during training and outputting the mode of the classes or mean prediction of the individual trees.

The main objective is to **improve predictive accuracy and control overfitting** by combining the predictions of multiple decision trees. The algorithm's strength lies in its ability to reduce the variance observed in individual decision trees, leading to improved generalization and robustness against overfitting.

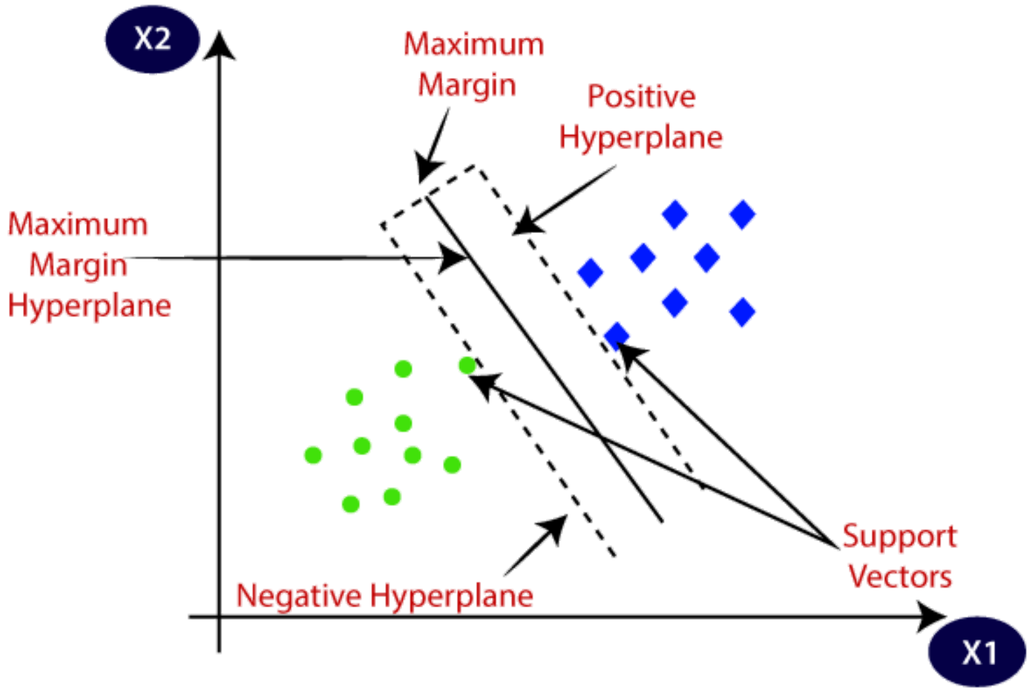
2. Random Forest

```python
# Build and train the initial random forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)

# Make predictions with the initial model
initial_y_pred = rf_model.predict(X_test)

# Evaluate the initial model
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), "%")
```

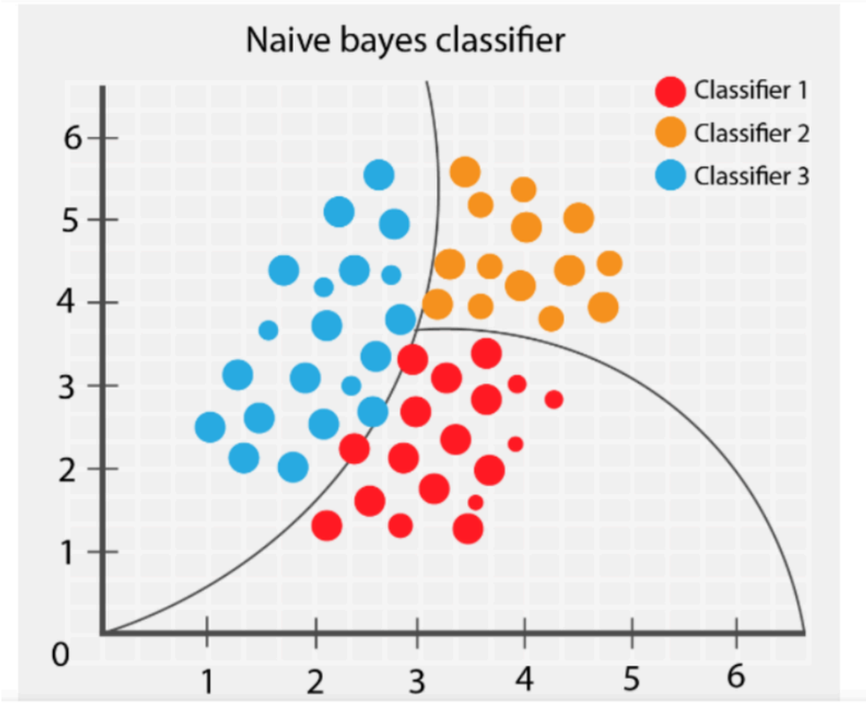
```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best Hyperparameters: {'max_depth': 10, 'max_features': 'sqrt',

Tuned Model Confusion Matrix:
[[3103  334]
 [ 649  914]]

Tuned Model Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      3437
           1       0.73      0.58      0.65      1563

    accuracy                           0.80      5000
   macro avg       0.78      0.74      0.76      5000
weighted avg       0.80      0.80      0.80      5000


Tuned Model Accuracy: 80.34 %
```

Some other Classification Models

4. SUPPORT VECTOR MACHINE



5. NAIVE BAYES CLASSIFIER



Hyperparameter Tuning

Process of selecting the optimal values for a machine learning model's hyperparameters. Hyperparameters are **settings that control the learning process** of the model, such as the **learning rate**, the number of neurons in a neural network, or the kernel size in a support vector machine. The goal of hyperparameter tuning is to find the values that lead to the best performance on a given task.

1. DECISION TREE

- Accuracy Before 72.4%
- Accuracy After 77.4%

2. RANDOM FOREST

- Accuracy Before 80.34%
- Accuracy After 80.52%

3. LOGISTIC REGRESSION

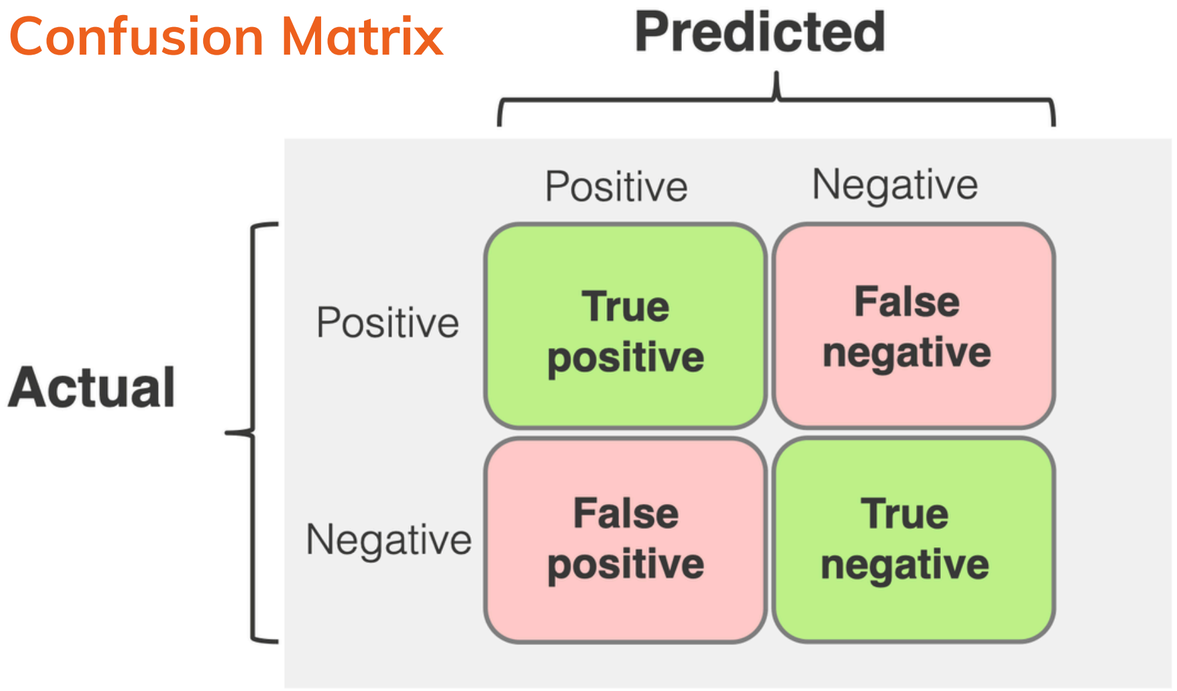
- Accuracy Before 80.28%
- Accuracy After 80.48%

Model Training → Hyperparameter Tuning → Find Accuracy and evaluate performance

Popularly used performance metrics for classification problems are:

- Accuracy
- Precision
- Recall
- F-score
- ROC

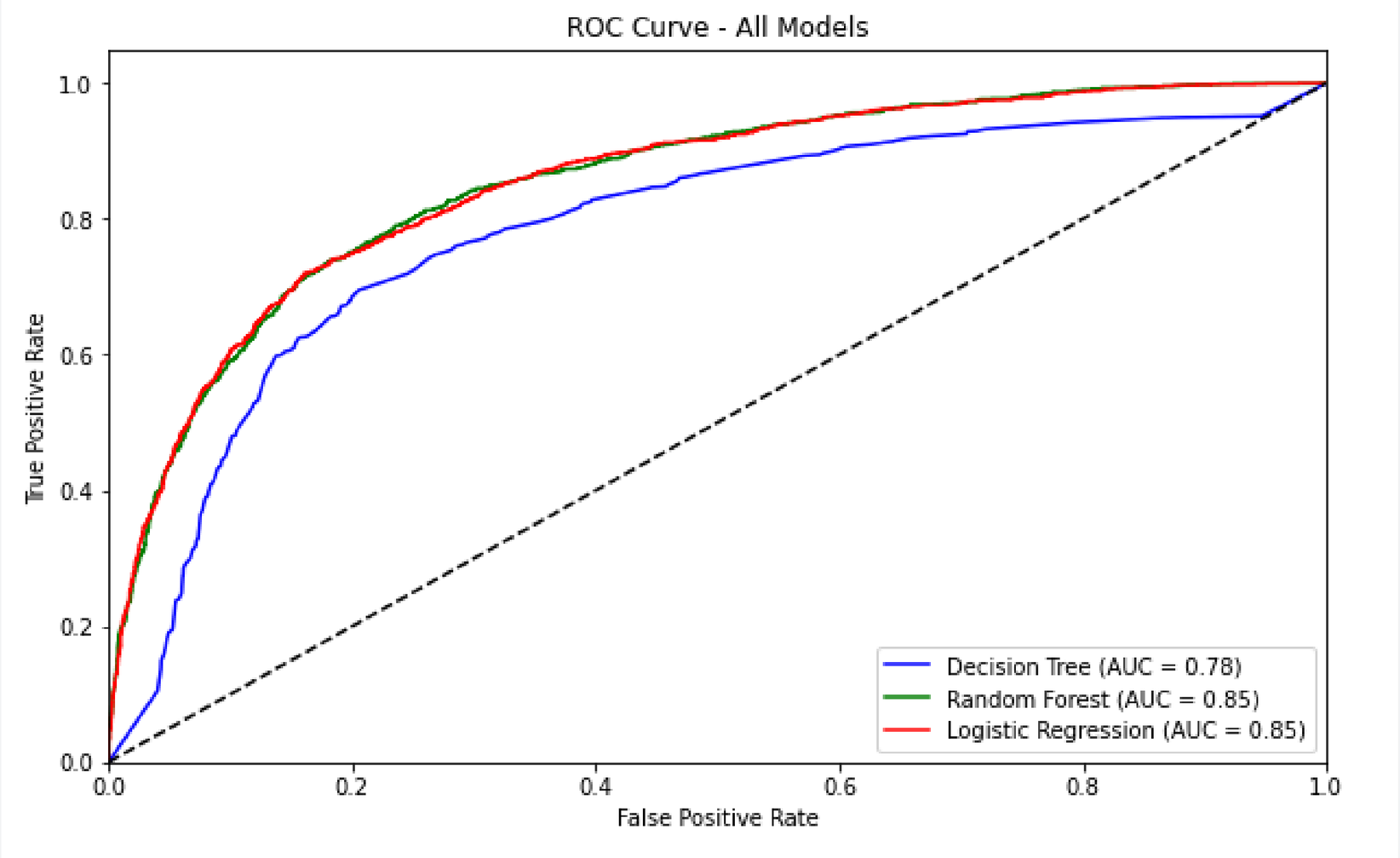
Confusion Matrix



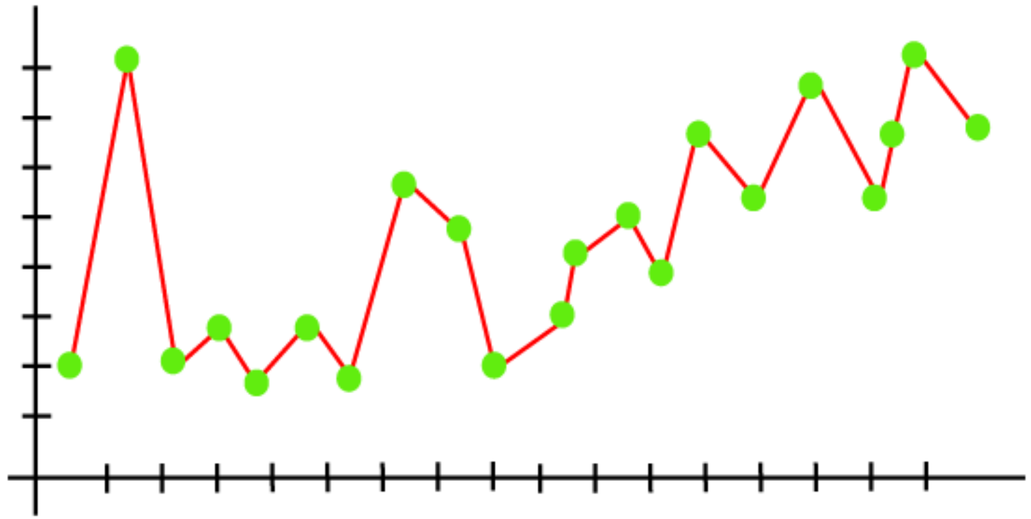
Key Differences

Parameters	Decision Tree	Random Forest	Logistic Regression
Training Accuracy	77.34	80.52	80.48
Testing Accuracy	72.4	80.34	80.28
Precision	(0) 83% (1) 65%	(0) 83% (1) 73%	(0) 83% (1) 73%
Recall	(0) 85% (1) 61%	(0) 90% (1) 58%	(0) 90% (1) 59%

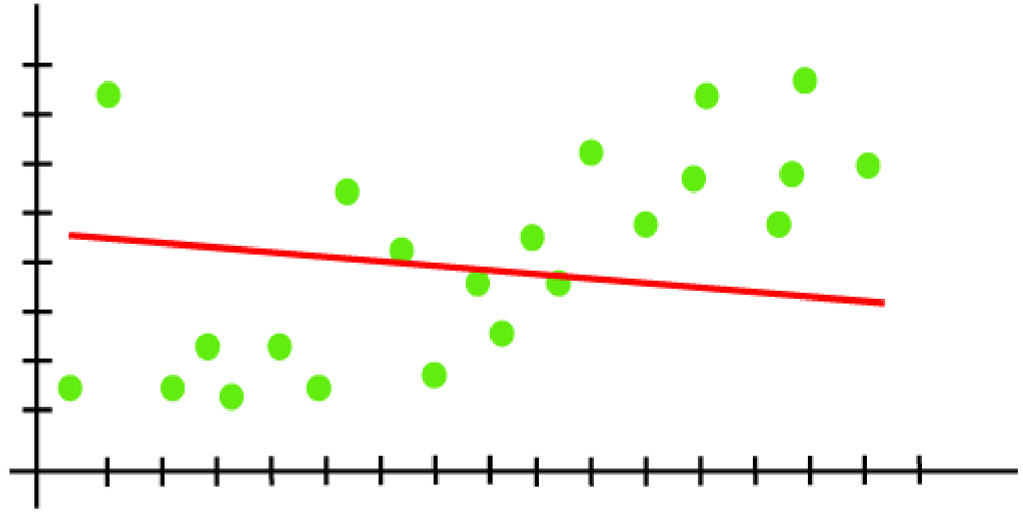
Cumulative ROC Curve



Model Optimization: Reduce Overfitting and Underfitting



Overfitting occurs when our machine learning model **tries to cover all the data points** or more than the required data points present in the given dataset. Even if model works well for training data, it fails for new samples.



In the case of underfitting, the model is not able to learn enough from the training data, and hence it reduces the accuracy and produces unreliable predictions. High bias and Low variance.

Why Random Forest?

Introduction

When tasked with predicting telecom churn, it is crucial to select a model that not only provides high accuracy but also delivers robust and reliable predictions. After analyzing and evaluating various models, Random Forest emerged as the superior choice. This conclusion was based on multiple factors including accuracy, precision, recall, and overall performance metrics.

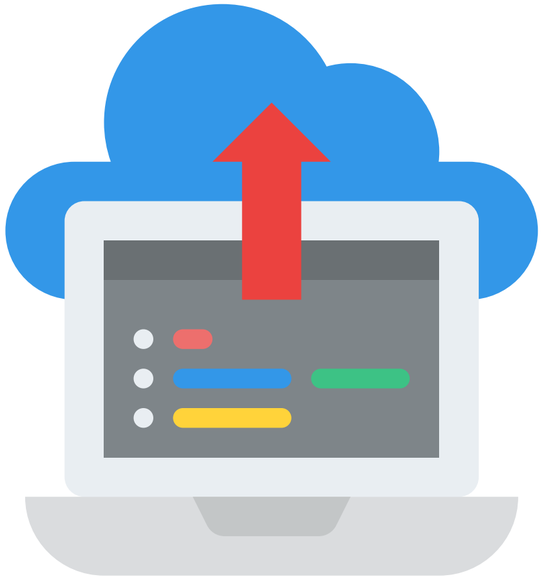
- HIGHEST ACCURACY (80.52)
- BALANCED PERFORMANCE METRICS
- ROBUSTNESS TO OVERFITTING
- SCALABILITY - LARGE DATASETS

Final Testing and Deployment

LAST STEP OF MACHINE LEARNING CYCLE:

If the model we developed is producing an accurate result as per the customer's requirements with an acceptable speed then we deploy our model into the real-world system.

Before deploying ensure whether the model is increasing its performance using available data or not.



Thank You!