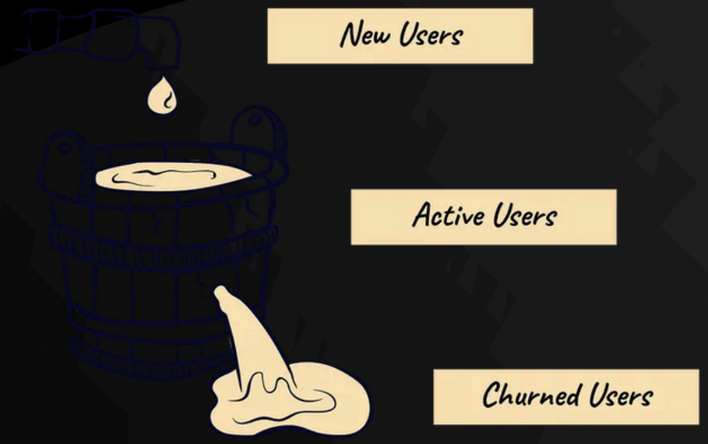
# TELECOM CHURN PREDICTION

*Presented by: Manas Dalai*
*Mentor : Mr. Bhaskar Naidu*

# CONTENT:

# About Telecom Churn Prediction

Churn refers to the process where customers discontinue their service with a telecom provider, either by switching to another provider or by stopping the use of telecom services altogether.




# Our Mission & Vission

## Mission

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

## Vission

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

# Our Goals


## Minimize Churn Rate

Minimizing churn prediction involves using machine learning to analyze customer data and identify those at risk of leaving. Companies can then implement targeted interventions, like personalized offers, to retain these customers. This strategy enhances customer satisfaction and reduces churn rates.

## Maximize Customer Retention

Maximizing customer retention involves strategies to keep existing customers engaged and satisfied with the service. This includes offering personalized experiences, proactive customer support, and loyalty programs. By focusing on retention, companies can reduce churn rates and build long-term customer loyalty.

## Optimize Business Performance

Optimizing business performance involves improving processes, resource allocation, and decision-making to achieve higher efficiency and effectiveness. This can lead to increased productivity, reduced costs, and enhanced overall business outcomes.

# Our Milestone

## 01 GATHERING & UNDERSTANDING

Gathering and understanding involves collecting information and analyzing it to gain insights and knowledge for informed decision-making.

## 02 DATA CLEAING & PREPROCESSING

Cleaning and preprocessing involve removing errors and inconsistencies from data and preparing it for analysis by organizing, transforming, and normalizing it.

## 03 MACHINE LEARNING

Machine learning and model building involve using algorithms to analyze data and create predictive models that can learn from patterns and make informed decisions or predictions.

## 04 HYPERPARAMETER TUNING & ROC

Hyperparameter tuning involves adjusting model parameters to optimize performance. ROC is a plot illustrating a binary classifier's diagnostic ability as its threshold varies.

# Challenges

### Data Quality & Integration

Incomplete or inaccurate data adversely affects model performance, compromising the reliability and effectiveness of analytical insights and predictions.

### Interpretability

Complex models like deep learning may provide accurate predictions but can be difficult to interpret, making it hard to understand the reasons behind churn predictions.

### Scalability

Telecom companies serve large customer bases, and churn prediction systems must be scalable to handle massive volumes of data and real-time predictions.

### Dynamic Behavior

Customer behavior can change over time, requiring continuous monitoring and updating of models.

# DATA CLEANING & PREPROCESSING

1. Convert datatypes of variables which are misclassified.
   - *Ensures accurate analysis and efficient processing.*
2. Removing duplicate records.
   - *Ensure accuracy, maintain integrity, and optimize efficiency in data management and analysis.*
3. Removing unique value variables.
   - *Necessary to avoid redundancy, improve efficiency, and ensure meaningful analysis.*
4. Removing Zero variance variables.
   - *Necessary to eliminate redundant information, improve model stability, and simplify interpretation.*

# DATA CLEANING & PREPROCESSING

5 Outlier Treatment.

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

6 Missing value Treatment.

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

7 Removing the highly correlated variables.

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

8 Multicollinearity (VIF>5)

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

# MACHINE LEARNING & MODEL BUILDING

| Logistic Regression | Decision Tree | Random Forest |
| --- | --- | --- |

# 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 outcome.*

*Logistic regression is crucial for predicting binary outcomes, offering interpretable results, probabilistic predictions, efficiency, and serving as a foundation for more complex models.*

# Code Snippet

```
[48]: x_train,x_test,y_train,y_test = train_test_split(ip,op,train_size=0.8, random_state=42)

[49]: #standard scaler
      sc = StandardScaler()
      x_train = sc.fit_transform(x_train)
      x_test = sc.fit_transform(x_test)
```

```
lr_model=LogisticRegression()
lr_model.fit(x_train,y_train)
```

▾ LogisticRegression

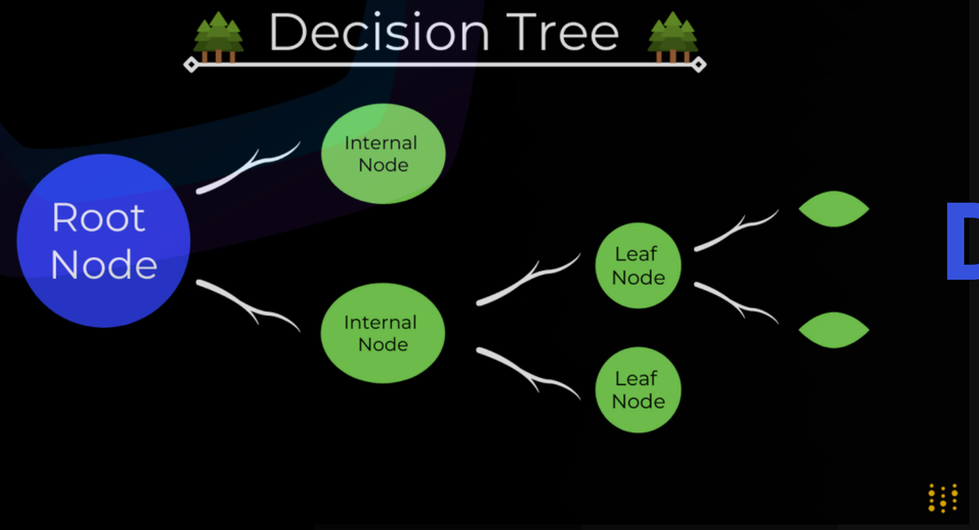
LogisticRegression()

```
lr_prediction = lr_model.predict(x_test)
lr_prediction
```

```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

```
# Calculate the accuracy of the Logistic Regression model on the test data
lr_accuracy = accuracy_score(y_test, lr_prediction)

# Print the accuracy score of the Logistic Regression model
print(f"Logistic Regression Accuracy: {lr_accuracy}")

# Print the classification report for the Logistic Regression model's predictions
print(classification_report(y_test, lr_prediction))
```

```
Logistic Regression Accuracy: 0.7868294067556O6
              precision    recall  f1-score   support

           0       0.81      0.91      0.86      2511
           1       0.69      0.47      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
```




# 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.*

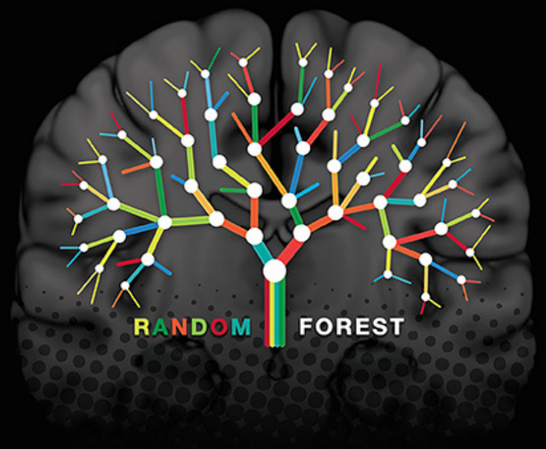
*Decision trees are crucial for their simplicity, ability to handle non-linear relationships, feature importance identification, and robustness to outliers, applicable to both classification and regression tasks.*

# Code Snippet

```
[48]: x_train,x_test,y_train,y_test = train_test_split(ip,op,train_size=0.8, random_state=42)

[49]: #standard scaler
      sc = StandardScaler()
      x_train = sc.fit_transform(x_train)
      x_test = sc.fit_transform(x_test)
```

```
[50]: dt_model = DecisionTreeClassifier()
      dt_model.fit(x_train,y_train)

[50]: ▾ DecisionTreeClassifier
      DecisionTreeClassifier()

[51]: dt_prediction = dt_model.predict(x_test)
      dt_prediction

[51]: array([1, 0, 0, ..., 1, 0, 0], dtype=int64)

[52]: # Calculate the accuracy of the Decision Tree model on the test data
      dt_accuracy = accuracy_score(y_test, dt_prediction)

      # Print the accuracy score of the Decision Tree model
      print(f"Decision Tree Accuracy: {dt_accuracy}")

      # Print the classification report for the Decision Tree model's predictions
      print(classification_report(y_test, dt_prediction))
```

```
Decision Tree Accuracy: 0.7022424070394551
              precision    recall  f1-score   support

           0       0.80      0.78      0.79      2511
           1       0.48      0.51      0.49      1012

    accuracy                           0.70      3523
   macro avg       0.64      0.64      0.64      3523
weighted avg       0.71      0.70      0.70      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.*

# Code Snippet

```
[48]: x_train,x_test,y_train,y_test = train_test_split(ip,op,train_size=0.8, random_state=42)

[49]: #standard scaler
      sc = StandardScaler()
      x_train = sc.fit_transform(x_train)
      x_test = sc.fit_transform(x_test)
```

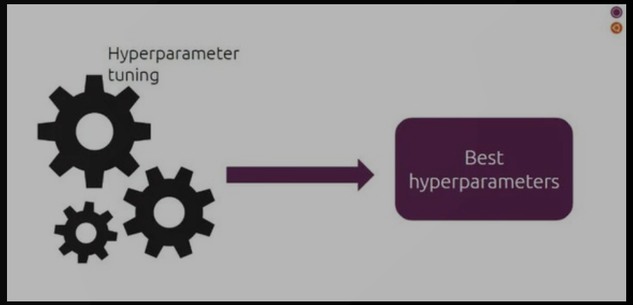
```
[64]: rf_model = RandomForestClassifier()
      rf_model.fit(x_train,y_train)

[64]: ▾ RandomForestClassifier
      RandomForestClassifier()

[65]: rf_prediction = rf_model.predict(x_test)
      rf_prediction

[65]: array([1, 0, 0, ..., 1, 0, 0], dtype=int64)

[66]: # Calculate the accuracy of the Random Forest Classifier model on the test data
      rf_accuracy = accuracy_score(y_test, rf_prediction)

      # Print the accuracy score of the Random Forest Classifier model
      print(f"Random Forest Classifier Accuracy: {rf_accuracy}")

      # Print the classification report for the Random Forest Classifier model's predictions
      print(classification_report(y_test, rf_prediction))
```

```
Random Forest Classifier Accuracy: 0.7848424638092535
              precision    recall  f1-score   support

           0       0.81      0.91      0.86      2511
           1       0.68      0.47      0.55      1012

    accuracy                           0.78      3523
   macro avg       0.75      0.69      0.71      3523
weighted avg       0.77      0.78      0.77      3523
```

# Key Difference

| Parameters | Logistic Regression | Decision Tree | Random Forest |
|---|---|---|---|
| **Training Accuracy** | **79.6%** | **78.9%** | 77.2% |
| Testing Accuracy | 78.6% | 77.8% | 78.2% |
| Precision | (0) 81% (1) 69% | (0) 81% (1) 65% | (0) 81% (1) 68% |
| Recall | (0) 91% (1) 47% | (0) 90% (1) 49% | (0) 92% (1) 45% |

# 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.

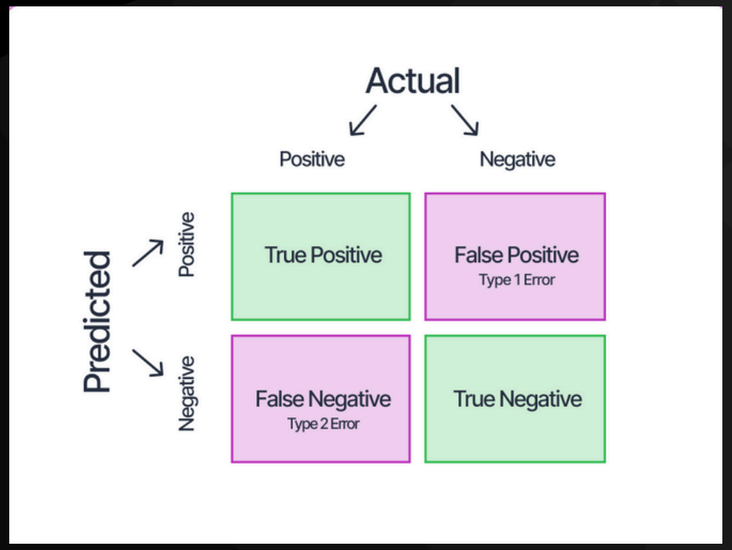
### Optimizes Computational Resources

Efficient tuning can lead to faster and more efficient model training and deployment.

# Code Snippet

```
[55]: # Define the parameter grid for hyperparameter tuning
param_grid = {'max_depth': [3, 5, 7, 10], 'min_samples_split': [2, 5, 10]}

# Initialize GridSearchCV with the Decision Tree model, parameter grid, 5-fold cross-validation, and accuracy scoring
grid_search_dt = GridSearchCV(dt_model, param_grid, cv=5, scoring='accuracy')

# Fit the GridSearchCV to the training data
grid_search_dt.fit(x_train, y_train)

# Get the best estimator (model) from the grid search
best_dt_model = grid_search_dt.best_estimator_

# Predict the test data using the best model
dt_predictions_best = best_dt_model.predict(x_test)

# Calculate the accuracy of the best model on the test data
best_dt_accuracy = accuracy_score(y_test, dt_predictions_best)

# Print the best accuracy score
print(f"Best Decision Tree Accuracy: {best_dt_accuracy}")

# Print the classification report for the best model's predictions
print(classification_report(y_test, dt_predictions_best))
```

```
Best Decision Tree Accuracy: 0.7785977859778598
              precision    recall  f1-score   support

           0       0.81      0.90      0.85      2511
           1       0.65      0.49      0.56      1012

    accuracy                           0.78      3523
   macro avg       0.73      0.69      0.71      3523
weighted avg       0.77      0.78      0.77      3523
```




# 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.*

# COMPONETS

## True Positive (TP)

The number of instances correctly predicted as positive.

## True Negative (TN)

The number of instances correctly predicted as negative.

## False Positive (FP)

The number of instances incorrectly predicted as positive (Type I error).

## False Negative (NP)

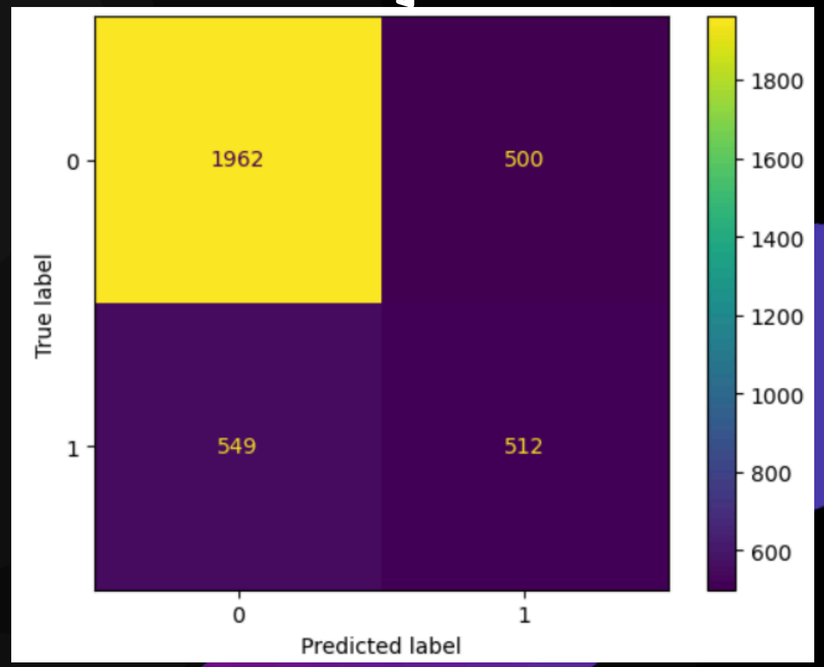
The number of instances incorrectly predicted as negative (Type II error).

# Code Snippet

```
[53]: cm = confusion_matrix(dt_prediction,y_test)
      cm
```

```
[53]: array([[1962,  500],
             [ 549,  512]], dtype=int64)
```

```
[54]: cmd = ConfusionMatrixDisplay(cm)
      cmd.plot()
      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 lifetime value leads to steady revenue streams and sustainable business growth.

# THANK YOU