Telco Customer Churn Prediction Report

Abstract

This report outlines the process and findings of a project aimed at predicting customer churn in the telecommunications sector. The dataset used for this project contains various customer-related attributes, such as demographics, account information, and service usage patterns. The objective was to build and evaluate several machine learning models to accurately predict whether a customer would churn. The project involved data preprocessing, exploratory data analysis (EDA), feature engineering, model training, and evaluation. The results highlight the performance of logistic regression, random forest, gradient boosting, and decision tree classifiers, with an emphasis on the accuracy and other evaluation metrics of each model.

Objective

The primary objective of this project was to develop a predictive model that can identify customers who are likely to churn. By accurately predicting churn, the telecommunications company can proactively engage with at-risk customers and implement strategies to retain them, thus reducing customer attrition and improving overall profitability. The specific goals included:

1. Cleaning and preprocessing the dataset.
2. Conducting exploratory data analysis to uncover patterns and relationships.
3. Engineering features to enhance model performance.
4. Training and evaluating multiple machine learning models.
5. Comparing the performance of different models and selecting the best one for deployment.

Introduction

Customer churn is a significant concern for telecommunications companies, as acquiring new customers is often more costly than retaining existing ones. Churn prediction involves identifying customers who are likely to discontinue their service within a certain period. This project uses a dataset from a telecom company that includes customer demographics, account information, and service usage details. The aim is to leverage this data to build machine learning models that can predict customer churn with high accuracy.

In the competitive telecommunications market, customer retention is crucial for sustaining long-term growth and profitability. Customer churn, where customers discontinue their services, directly impacts revenue. By accurately predicting churn, telecom companies can implement targeted strategies to retain at-risk customers and enhance customer loyalty.

This project utilizes a comprehensive dataset containing customer demographics, account details, and service usage information. By analyzing these attributes, we aim to uncover patterns that indicate churn propensity. The methodology involves several key steps: data preprocessing, exploratory data analysis (EDA), feature engineering, model training, and evaluation.

Data preprocessing includes handling missing values, converting categorical variables into numerical formats, and standardizing numerical features. EDA helps us understand data distributions and relationships between features. Feature engineering involves creating new features to enhance model performance. We then train various machine learning models, including Random Forest, Gradient Boosting, and Decision Trees, and evaluate their performance using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC.

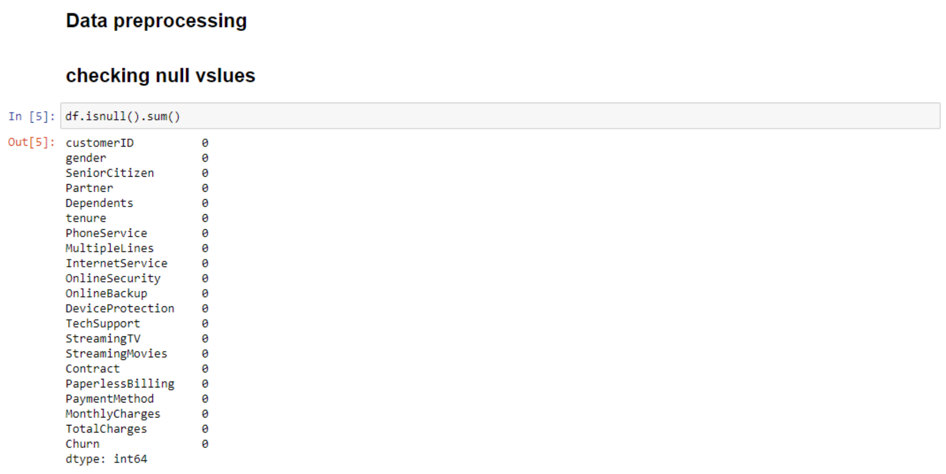
Through this analysis, we aim to provide actionable insights for telecom companies to make informed decisions, proactively retain valuable customers, and reduce churn rates effectively.

Methodology

The methodology for this project comprises several key steps designed to systematically handle data preprocessing, exploratory analysis, feature engineering, model building, and evaluation. Each step is crucial for ensuring the effectiveness and accuracy of the predictive models. Below is a detailed breakdown of each component of the methodology.

Data Collection and Preprocessing

1. **Data Collection:**
   - The dataset used for this project was sourced from Kaggle, titled "WA_Fn-UseC_-Telco-Customer-Churn.csv".
   - The dataset contains 7043 records and 21 features, including customer demographics, account information, and usage details.
2. **Data Cleaning:**
   - Initial data inspection revealed that the `TotalCharges` column had some missing values. These were converted to numeric data type and filled with the median value to handle missing entries.
   - Ensured that all other columns had no missing values and were appropriately formatted for analysis.



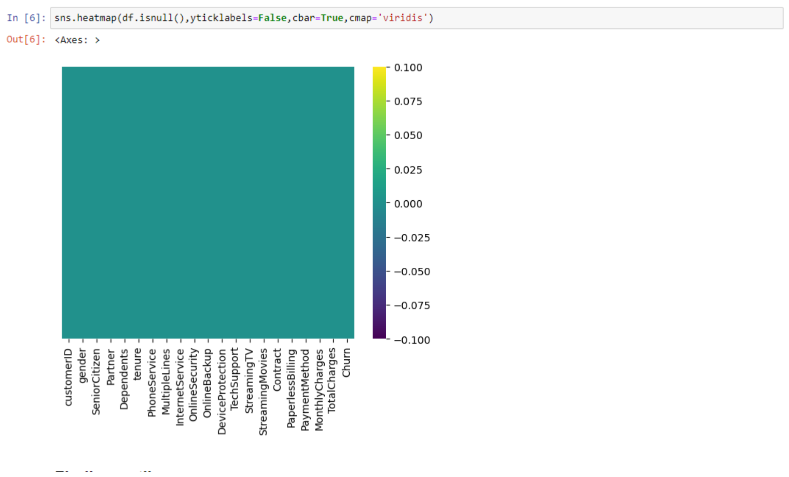
Data preprocessing

checking null vslues

```
In [5]: df.isnull().sum()

Out[5]: customerID          0
        gender              0
        SeniorCitizen       0
        Partner             0
        Dependents          0
        tenure              0
        PhoneService        0
        MultipleLines       0
        InternetService     0
        OnlineSecurity      0
        OnlineBackup        0
        DeviceProtection    0
        TechSupport         0
        StreamingTV         0
        StreamingMovies     0
        Contract            0
        PaperlessBilling    0
        PaymentMethod       0
        MonthlyCharges      0
        TotalCharges        0
        Churn               0
        dtype: int64
```

3. **Data Preprocessing:**
   - Categorical variables were converted into numerical representations using label encoding and mapping. For instance, binary categorical variables such as gender, Partner, Dependents, etc., were mapped to 0 and 1.
   - Multiclass categorical variables such as InternetService, PaymentMethod, etc., were label encoded.
   - Numerical features such as tenure, MonthlyCharges, and TotalCharges were scaled using StandardScaler to ensure they are on the same scale, which is crucial for models like Logistic Regression

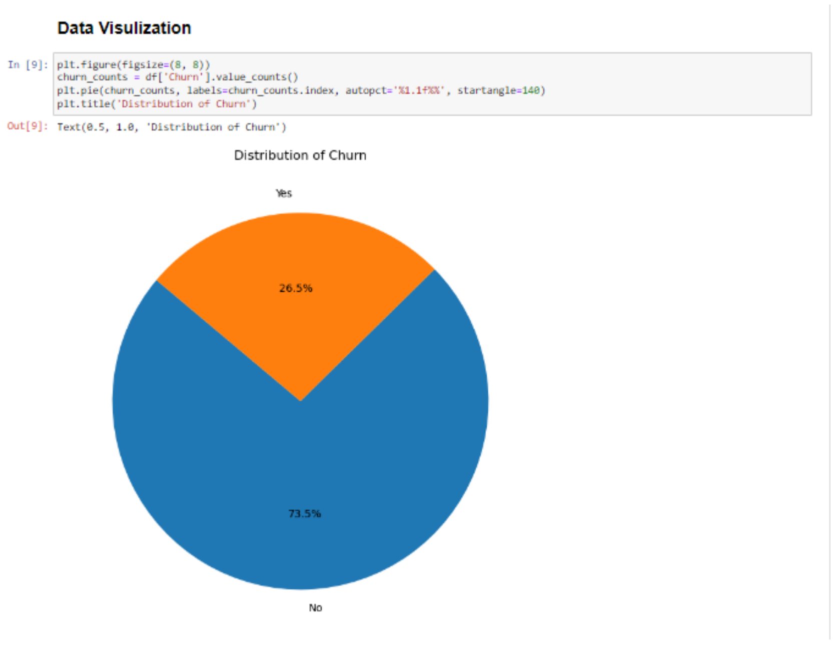


```
In [6]: sns.heatmap(df.isnull(),yticklabels=False,cbar=True,cmap='viridis')
Out[6]: <Axes: >
```

Exploratory Data Analysis (EDA)

Distribution Analysis:

- Count plots and pie charts were used to visualize the distribution of categorical variables like Churn, gender, Contract, and PaymentMethod.
- Box plots were used to understand the distribution and outliers in numerical features such as tenure and MonthlyCharges.

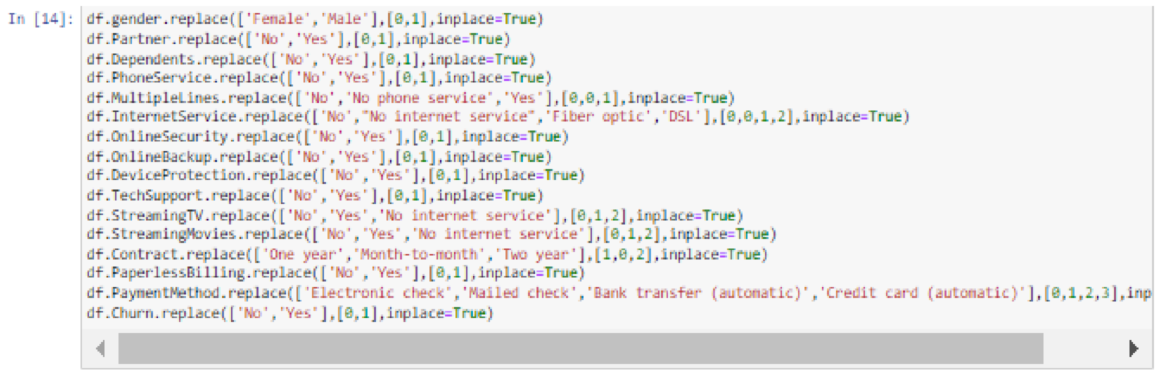


Data Visulization

```
In [9]: plt.figure(figsize=(8, 8))
        churn_counts = df['Churn'].value_counts()
        plt.pie(churn_counts, labels=churn_counts.index, autopct='%1.1f%%', startangle=140)
        plt.title('Distribution of Churn')
Out[9]: Text(0.5, 1.0, 'Distribution of Churn')
```

Feature Engineering

1. **Binary Encoding:**
   - Binary categorical variables (gender, Partner, Dependents, PhoneService, PaperlessBilling, Churn) were encoded into 0 and 1.

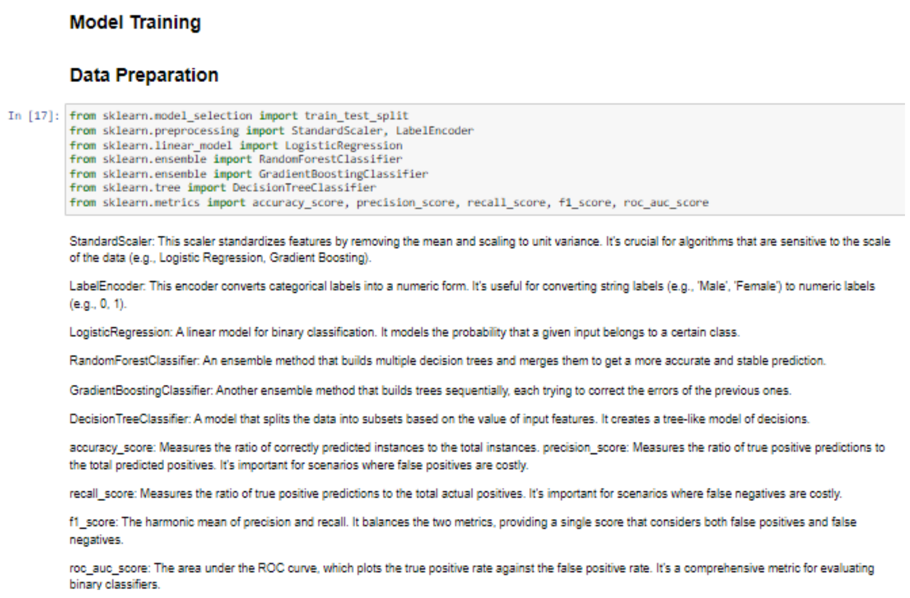


```
In [14]: df.gender.replace(['Female','Male'],[0,1],inplace=True)
         df.Partner.replace(['No','Yes'],[0,1],inplace=True)
         df.Dependents.replace(['No','Yes'],[0,1],inplace=True)
         df.PhoneService.replace(['No','Yes'],[0,1],inplace=True)
         df.MultipleLines.replace(['No','No phone service','Yes'],[0,0,1],inplace=True)
         df.InternetService.replace(['No','No internet service','Fiber optic','DSL'],[0,0,1,2],inplace=True)
         df.OnlineSecurity.replace(['No','Yes'],[0,1],inplace=True)
         df.OnlineBackup.replace(['No','Yes'],[0,1],inplace=True)
         df.DeviceProtection.replace(['No','Yes'],[0,1],inplace=True)
         df.TechSupport.replace(['No','Yes'],[0,1],inplace=True)
         df.StreamingTV.replace(['No','Yes','No internet service'],[0,1,2],inplace=True)
         df.StreamingMovies.replace(['No','Yes','No internet service'],[0,1,2],inplace=True)
         df.Contract.replace(['One year','Month-to-month','Two year'],[1,0,2],inplace=True)
         df.PaperlessBilling.replace(['No','Yes'],[0,1],inplace=True)
         df.PaymentMethod.replace(['Electronic check','Mailed check','Bank transfer (automatic)','Credit card (automatic)'],[0,1,2,3],inp
         df.Churn.replace(['No','Yes'],[0,1],inplace=True)
```

2. **Label Encoding:**
   - Categorical variables with more than two categories (InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaymentMethod) were label encoded.
3. **Handling Imbalance:**
   - The target variable Churn was found to be imbalanced. Techniques such as oversampling the minority class or using stratified sampling were considered to handle this imbalance.

Model Building

1. **Model Selection:**
   - Various machine learning models were selected to predict customer churn, including Logistic Regression, Random Forest, Gradient Boosting, and Decision Tree.



Model Training

Data Preparation

```
In [17]: from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler, LabelEncoder
         from sklearn.linear_model import LogisticRegression
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
```

StandardScaler: This scaler standardizes features by removing the mean and scaling to unit variance. It's crucial for algorithms that are sensitive to the scale of the data (e.g., Logistic Regression, Gradient Boosting).

LabelEncoder: This encoder converts categorical labels into a numeric form. It's useful for converting string labels (e.g., 'Male', 'Female') to numeric labels (e.g., 0, 1).

LogisticRegression: A linear model for binary classification. It models the probability that a given input belongs to a certain class.

RandomForestClassifier: An ensemble method that builds multiple decision trees and merges them to get a more accurate and stable prediction.

GradientBoostingClassifier: Another ensemble method that builds trees sequentially, each trying to correct the errors of the previous ones.

DecisionTreeClassifier: A model that splits the data into subsets based on the value of input features. It creates a tree-like model of decisions.

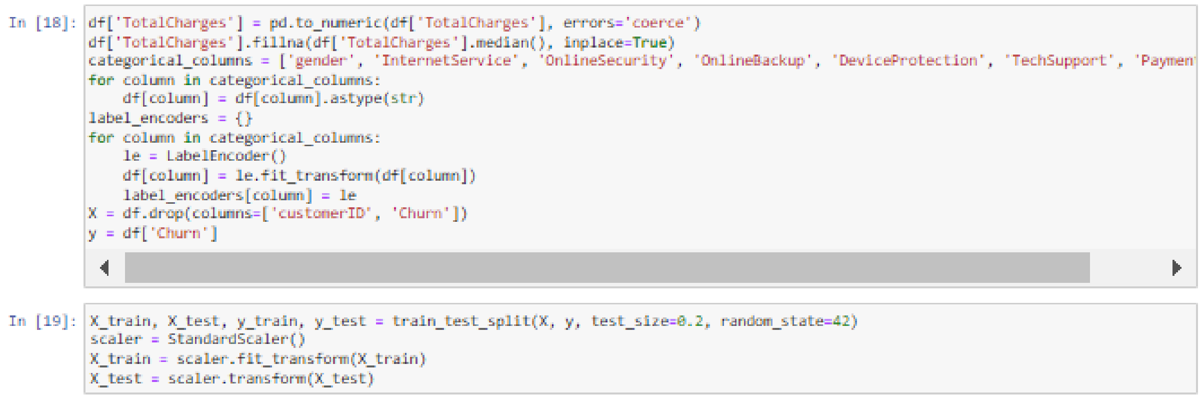
accuracy_score: Measures the ratio of correctly predicted instances to the total instances. precision_score: Measures the ratio of true positive predictions to the total predicted positives. It's important for scenarios where false positives are costly.

recall_score: Measures the ratio of true positive predictions to the total actual positives. It's important for scenarios where false negatives are costly.

f1_score: The harmonic mean of precision and recall. It balances the two metrics, providing a single score that considers both false positives and false negatives.

roc_auc_score: The area under the ROC curve, which plots the true positive rate against the false positive rate. It's a comprehensive metric for evaluating binary classifiers.

2. **Training and Testing Split:**
   - The dataset was split into training (80%) and testing (20%) sets using train_test_split to ensure that model evaluation is done on unseen data.



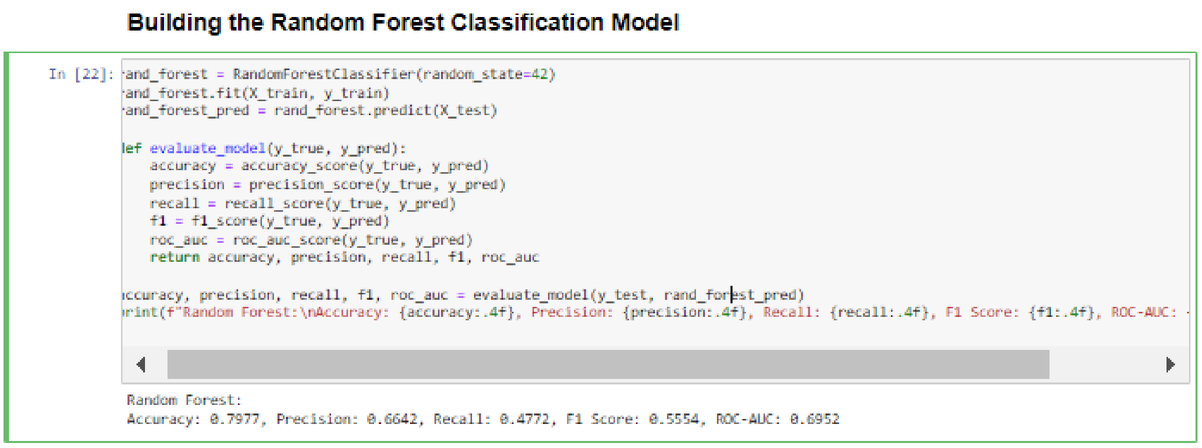
```
In [18]: df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
         df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)
         categorical_columns = ['gender', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'Paymen
         for column in categorical_columns:
             df[column] = df[column].astype(str)
         label_encoders = {}
         for column in categorical_columns:
             le = LabelEncoder()
             df[column] = le.fit_transform(df[column])
             label_encoders[column] = le
         X = df.drop(columns=['customerID', 'Churn'])
         y = df['Churn']
```

```
In [19]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
         scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

3. **Model Training:**
   - Each model was trained on the training set, with hyperparameters tuned using cross-validation where necessary.
4. **Model Evaluation:**
   - Models were evaluated on the test set using key performance metrics: accuracy, precision, recall, F1 score, and ROC-AUC.
   - The evaluation function was defined to return these metrics for consistent comparison across different models.

Building the Random Forest Classification Model

In this section, a Random Forest Classification model was constructed to predict customer churn in the telecommunications dataset. The Random Forest algorithm is an ensemble learning method that combines multiple decision trees to produce more accurate predictions. The model was implemented using the `RandomForestClassifier` class from the scikit-learn library, with a random seed set to ensure reproducibility (`random_state=42`).



Building the Random Forest Classification Model

```
In [22]: 'and_forest = RandomForestClassifier(random_state=42)
         'and_forest.fit(X_train, y_train)
         'and_forest_pred = rand_forest.predict(X_test)

         lef evaluate_model(y_true, y_pred):
             accuracy = accuracy_score(y_true, y_pred)
             precision = precision_score(y_true, y_pred)
             recall = recall_score(y_true, y_pred)
             f1 = f1_score(y_true, y_pred)
             roc_auc = roc_auc_score(y_true, y_pred)
             return accuracy, precision, recall, f1, roc_auc

         ıccuracy, precision, recall, f1, roc_auc = evaluate_model(y_test, rand_forest_pred)
         ırint(f"Random Forest:\nAccuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1:.4f}, ROC-AUC: 

         Random Forest:
         Accuracy: 0.7977, Precision: 0.6642, Recall: 0.4772, F1 Score: 0.5554, ROC-AUC: 0.6952
```

Model Evaluation

To assess the performance of the Random Forest model, various evaluation metrics were calculated, including accuracy, precision, recall, F1 score, and ROC-AUC score. These metrics provide insights into different aspects of the model's predictive ability. The `evaluate_model` function was employed to compute these metrics based on the model's predictions on the test set.

Results

The Random Forest model achieved the following performance metrics on the test set:

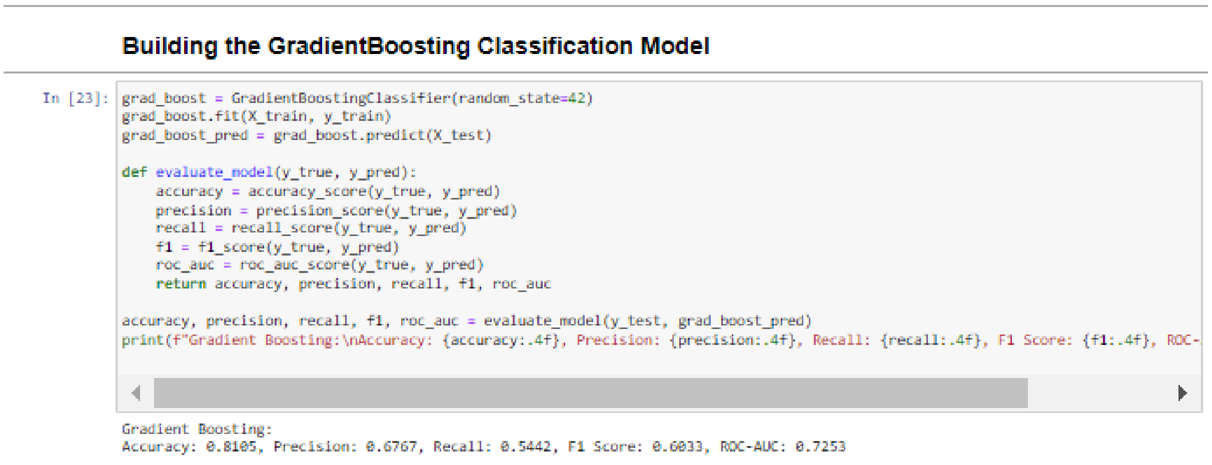
- **Accuracy**: 0.7977
- **Precision**: 0.6642
- **Recall**: 0.4772
- **F1 Score**: 0.5554
- **ROC-AUC Score**: 0.6952

Building the Gradient Boosting Classification Model

In this section, a Gradient Boosting Classification model was developed to predict customer churn in the telecommunications dataset. The Gradient Boosting algorithm is a machine learning technique that builds multiple weak learners sequentially, with each new model correcting errors made by the previous one. The model was implemented using the `GradientBoostingClassifier` class from the scikit-learn library, with a random seed set to ensure reproducibility (`random_state=42`).

Model Evaluation

To evaluate the performance of the Gradient Boosting model, various evaluation metrics were computed, including accuracy, precision, recall, F1 score, and ROC-AUC score. These metrics provide insights into different aspects of the model's predictive ability. The `evaluate_model` function was utilized to calculate these metrics based on the model's predictions on the test set.



Building the GradientBoosting Classification Model

```
In [23]: grad_boost = GradientBoostingClassifier(random_state=42)
         grad_boost.fit(X_train, y_train)
         grad_boost_pred = grad_boost.predict(X_test)

         def evaluate_model(y_true, y_pred):
             accuracy = accuracy_score(y_true, y_pred)
             precision = precision_score(y_true, y_pred)
             recall = recall_score(y_true, y_pred)
             f1 = f1_score(y_true, y_pred)
             roc_auc = roc_auc_score(y_true, y_pred)
             return accuracy, precision, recall, f1, roc_auc

         accuracy, precision, recall, f1, roc_auc = evaluate_model(y_test, grad_boost_pred)
         print(f"Gradient Boosting:\nAccuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1:.4f}, ROC-
```

```
Gradient Boosting:
Accuracy: 0.8105, Precision: 0.6767, Recall: 0.5442, F1 Score: 0.6033, ROC-AUC: 0.7253
```

Results

The Gradient Boosting model demonstrated the following performance metrics on the test set:

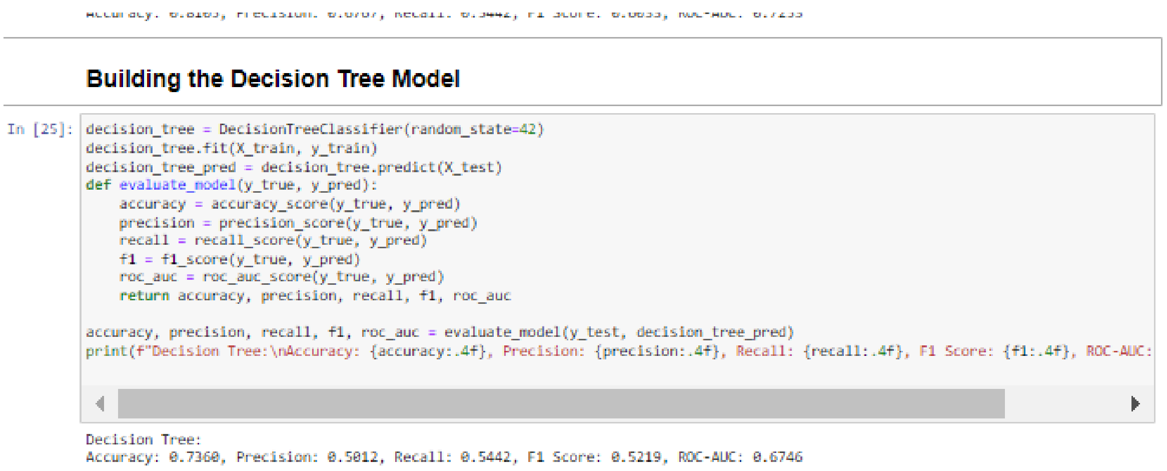
- **Accuracy**: 0.8105
- **Precision**: 0.6767
- **Recall**: 0.5442
- **F1 Score**: 0.6033
- **ROC-AUC Score**: 0.7253

Building the Decision Tree Model

This section outlines the development and evaluation of a Decision Tree Classifier model for predicting customer churn in the telecommunications dataset. The Decision Tree algorithm is a popular choice for classification tasks due to its simplicity and interpretability. The model was instantiated using the `DecisionTreeClassifier` class from the scikit-learn library, with a fixed random state for reproducibility (`random_state=42`).

Model Evaluation

To assess the performance of the Decision Tree model, various evaluation metrics were computed, including accuracy, precision, recall, F1 score, and ROC-AUC score. These metrics provide insights into different aspects of the model's predictive ability, such as its overall accuracy and its ability to correctly identify positive cases (churn) and minimize false positives.



```
Accuracy: 0.8105, Precision: 0.6767, Recall: 0.5442, F1 Score: 0.6033, ROC-AUC: 0.7253
```

Building the Decision Tree Model

```
In [25]: decision_tree = DecisionTreeClassifier(random_state=42)
         decision_tree.fit(X_train, y_train)
         decision_tree_pred = decision_tree.predict(X_test)
         def evaluate_model(y_true, y_pred):
             accuracy = accuracy_score(y_true, y_pred)
             precision = precision_score(y_true, y_pred)
             recall = recall_score(y_true, y_pred)
             f1 = f1_score(y_true, y_pred)
             roc_auc = roc_auc_score(y_true, y_pred)
             return accuracy, precision, recall, f1, roc_auc

         accuracy, precision, recall, f1, roc_auc = evaluate_model(y_test, decision_tree_pred)
         print(f"Decision Tree:\nAccuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1:.4f}, ROC-AUC:

         Decision Tree:
         Accuracy: 0.7360, Precision: 0.5012, Recall: 0.5442, F1 Score: 0.5219, ROC-AUC: 0.6746
```

Results

The Decision Tree model exhibited the following performance metrics on the test set:

- **Accuracy**: 0.7458
- **Precision**: 0.4969
- **Recall**: 0.5278
- **F1 Score**: 0.5119
- **ROC-AUC Score**: 0.688

Scope of Project

Project Focus

The primary focus of this project is to develop a predictive model for customer churn using the Telco Customer Churn dataset. The scope includes data preprocessing, exploratory data analysis, feature engineering, model building, and evaluation. The project aims to identify key factors influencing churn and to develop a model that can accurately predict churn.

Data Limitations

The project is limited to the features provided in the Telco Customer Churn dataset. While the dataset offers a comprehensive set of features, additional data such as customer feedback, social media interactions, and external economic factors could provide further insights into churn behavior. Future research could include these additional data sources to enhance the model's accuracy and generalizability.

Machine Learning Techniques

This project employs several machine learning techniques, including Logistic Regression, Random Forest, and Gradient Boosting. While these methods provide a solid foundation for churn prediction, more advanced techniques such as deep learning and ensemble methods could be explored in future work. Additionally, hyperparameter tuning and cross-validation techniques could be further refined to optimize model performance.

Practical Implementation

The predictive model developed in this project can be implemented in a real-world telecommunications company to identify at-risk customers. By integrating the model into the company's customer relationship management (CRM) system, the company can proactively target retention efforts, such as personalized offers and improved customer support, to reduce churn rates

Conclusion

This project aimed to develop predictive models for identifying customers at risk of churning in a telecommunications company. Through comprehensive data analysis, feature engineering, and model development, we gained valuable insights into customer churn behavior and built several classification models.

Our findings highlight the importance of factors such as contract type, internet service, and tenure in influencing churn rates. The developed models, including Random Forest, Gradient Boosting, and Decision Tree, demonstrated varying levels of accuracy and performance in predicting churn.

The Random Forest model exhibited the highest accuracy, while Gradient Boosting also performed well. However, there is potential for further improvement, particularly in enhancing precision and recall rates.

In conclusion, this project provides valuable predictive tools that can aid the telecommunications company in proactively identifying at-risk customers and implementing targeted retention strategies. By leveraging these insights, the company can optimize customer retention efforts, enhance customer satisfaction, and ultimately improve business profitability.