Predicting Customer Churn in a Telecommunications Company

This document outlines a predictive model that can identify customers at risk of churning, enabling a telecommunications company to take proactive measures to retain them. The project involves data collection and processing, exploratory data analysis, feature engineering, model building, and model evaluation.

BY Mahrishi Rathore
Registration No:: 12115161

Steps involved in this project:

1. Data Collection and Processing
2. Exploratory Data Analysis (EDA)
3. Feature Engineering
4. Building the Churn Prediction Model
5. Model Evalution
6. Calculating Accuracy Precision recall and f1 score

Data Collection and Processing

The first step in the project is to load the customer data from a CSV file into a Pandas DataFrame. The data is then checked for any missing values, which are handled accordingly. Categorical variables are encoded into numerical values to prepare the data for modeling.

Load the Data

The customer data is loaded from a CSV file into a Pandas DataFrame, and the first and last 5 rows are printed to inspect the data.

1

Handle Missing Values

The dataset is checked for any missing values, and since there are no null values, no further action is required to handle missing data.

2

Encode Categorical Variables

Categorical columns are identified and encoded into numerical values, including converting 'Yes/No' columns, handling 'No internet service' values, and encoding other categorical features.

3

Handel Outliers

Here we handle the outlier which may have caused problem in our prediction algorithm. Use Z-Score and IQR method to handel them

And also handle duplicate value if there any in the dataset

Encode Categorical Variables

Next, we check for any missing values in the dataset. In this case, there are no null values, so we do not need to perform any operation to handle missing values.

We identify and encode categorical columns into numerical values to facilitate the modeling process.

1. **Convert Yes/No Columns**
Columns with 'Yes' and 'No' values are converted to 1 and 0, respectively.

2. **Convert Columns with 'No Internet Service‘**
Columns with values 'Yes', 'No', and 'No internet service' are encoded accordingly, with 'No internet service' assigned a unique value.

3. **Convert MultipleLines Column**
The 'MultipleLines' column, which includes 'Yes', 'No', and 'No phone service', is similarly encoded with unique values.

4. **Convert InternetService Column**
The 'InternetService' column, which includes 'DSL', 'No', and 'Fiber optic', is encoded with unique numerical values.

5. **Convert Contract Column**
The 'Contract' column, which includes 'Month-to-month', 'One year', and 'Two year', is encoded numerically.

6. **Convert PaymentMethod Column**
The 'PaymentMethod' column, which includes various payment methods, is also encoded numerically.

Exploratory Data Analysis (EDA)

The next step is to perform Exploratory Data Analysis (EDA) to understand the distribution of the data and identify relationships between different features.

Understand Data Distribution

The summary statistics and distribution of the target variable, 'Churn', are analyzed to gain insights into the data.

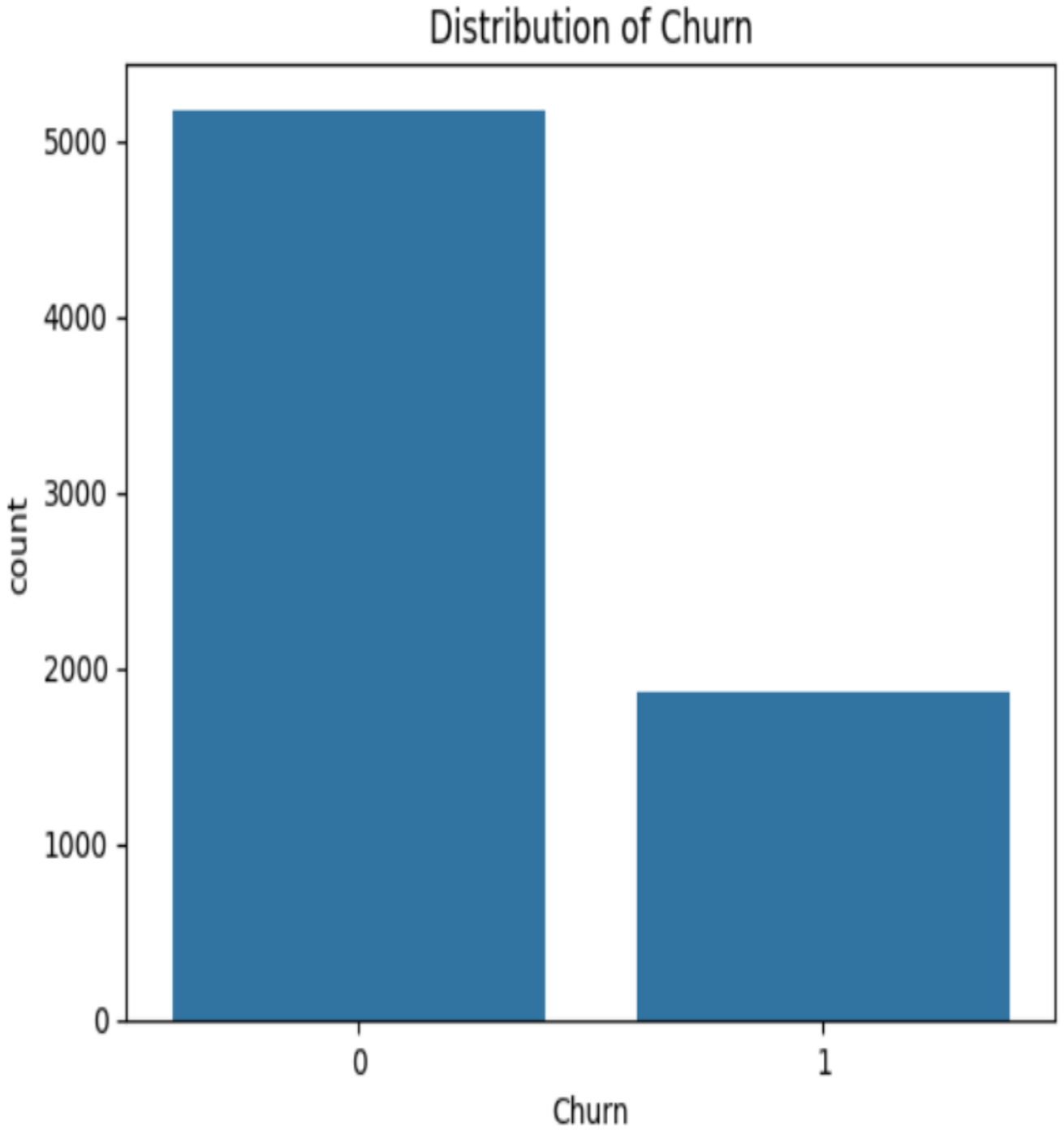
Identify Relationships

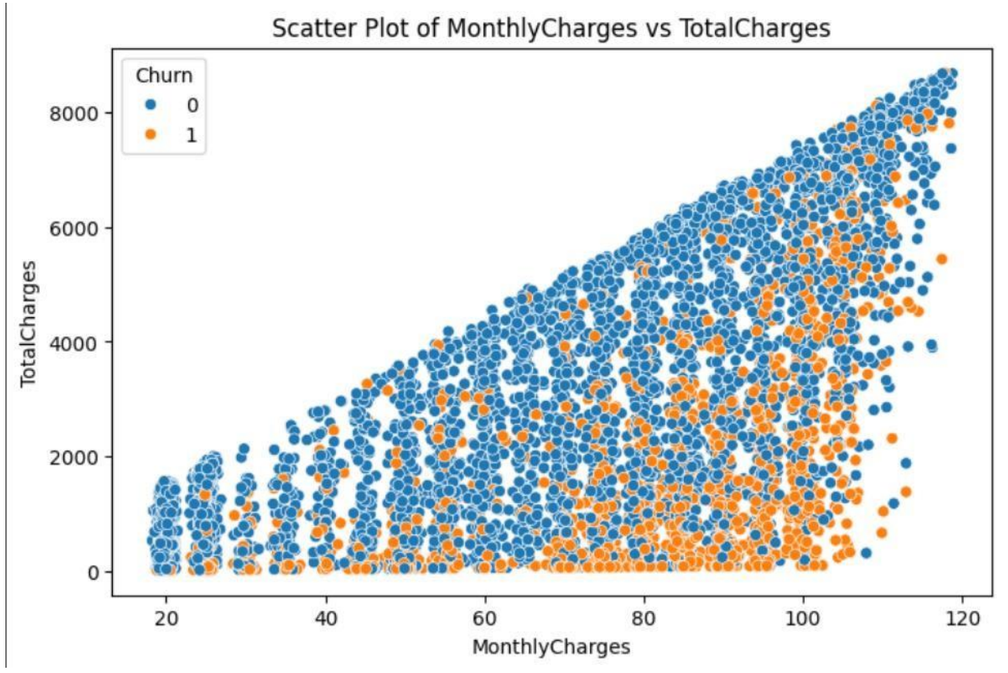
Relationships between features are explored using correlation matrices, count plots, histograms, box plots, and scatter plots.

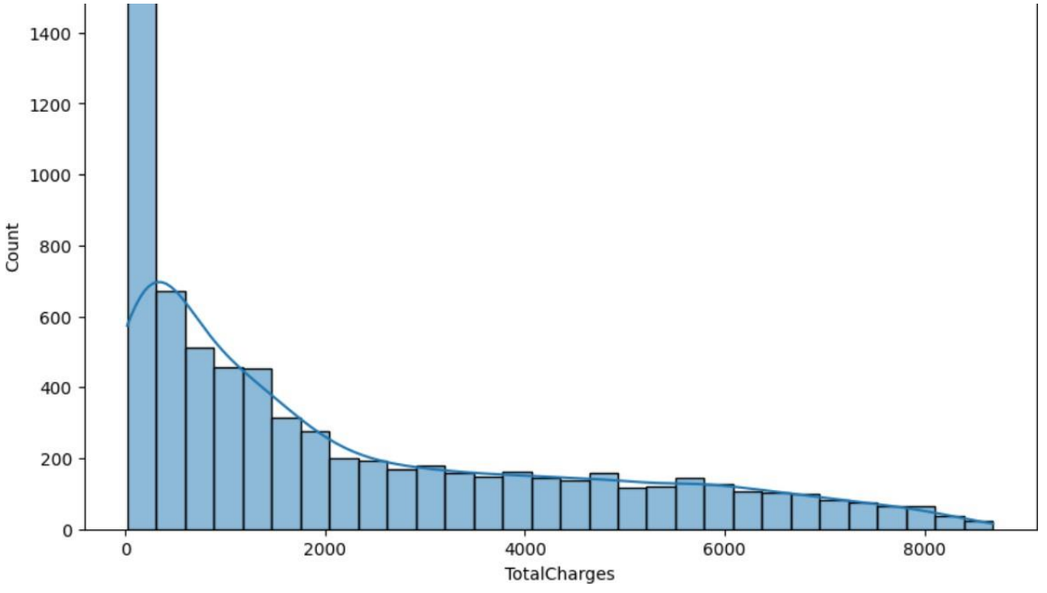
Feature Engineering

Created new features like 'ChargesRatio', 'HasMultipleServices', 'Senior_Partner', 'MonthlyChargesCategory', 'ServiceCount', and 'TenureGroup' to potentially improve model performance.

.


Distribution of Churn
count
Churn
0
1
0
1000
2000
3000
4000
5000


Scatter Plot of MonthlyCharges vs TotalCharges
Churn
0
1
TotalCharges
MonthlyCharges
0
2000
4000
6000
8000
20
40
60
80
100
120


Count
TotalCharges
0
200
400
600
800
1000
1200
1400
0
2000
4000
6000
8000

Building the Churn Prediction Model

The next step is to build the churn prediction model. The data is standardized, and then split into training and testing sets.

1

Logistic Regression

A Logistic Regression model is trained on the data.

2

Random Forest Classifier

A Random Forest Classifier model is trained on the data.

3

Gradient Boosting

A Gradient Boosting model is trained on the data.

Here Our Target variable is **Churn** .

4.1 Logistic Regresssion

+ Code + Text

```
[43] # Logistic Regression
     model_logistic=LogisticRegression()
```

```
[44] #training the LogisticRegression model with Training data
     model_logistic.fit(X_train,Y_train)
```

```
LogisticRegression()
```

```
[39] # standardization of data
     scaler=StandardScaler()
     scaler.fit(X)
     standardized_data=scaler.transform(X)
```

```
[40] X=standardized_data
     Y=df['Churn']
```

```
[41] X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

```
[42] print(X.shape,X_train.shape,X_test.shape)
```

```
(7043, 17) (5634, 17) (1409, 17)
```

4.2 RandomForest Classification

```
# RandomForest code
rf_params = {
    'n_estimators': 100,
    'max_depth': 10,
    'min_samples_split': 10,
    'min_samples_leaf': 4,
    'max_features': 'sqrt',
    'bootstrap': True,
    'random_state': 42
}
```

```
[821] #training the RandomForest model with Training data
      model_random_forest = RandomForestClassifier(**rf_params)
      model_random_forest.fit(X_train, Y_train)
```

```
RandomForestClassifier(max_depth=10, min_samples_leaf=4, min_samples_split=10,
                       random_state=42)
```

4.3 gradient boosting

```
[822] # Make predictions on the training set
      y_train_pred = gb_regressor.predict(X_train)

      # Make predictions on the test set
      y_test_pred = gb_regressor.predict(X_test)

      # Evaluate the model on the training set
      mse_train = mean_squared_error(Y_train, y_train_pred)
      mae_train = mean_absolute_error(Y_train, y_train_pred)
      r2_train = r2_score(Y_train, y_train_pred)
```

```
[823] # Evaluate the model on the test set
      mse_test = mean_squared_error(Y_test, y_test_pred)
```

Model Evaluation

The final step is to evaluate the performance of the trained models on the test data.

Logistic Regression

The Logistic Regression model is evaluated on the training and testing data, and the accuracy scores are reported.

Random Forest Classifier

The Random Forest Classifier model is evaluated on the training and testing data, and the accuracy scores are reported.

Gradient Boosting

The Gradient Boosting model is evaluated on the training and testing data, and the mean squared error, mean absolute error, and R-squared values are reported.

5.1 Logistic Regression

```
[824] #Accuracy Score
      #accuracy on training data
      X_train_prediction_model_logistic=model_logistic.predict(X_train)
      training_data_accuracy_model_logistic=accuracy_score(X_train_prediction_model_logistic,Y_train)
```

```
[825] print('Accuracy on Training Data: ',training_data_accuracy_model_logistic)
```

```
Accuracy on Training Data:  0.8020944266950657
```

```
[826] #Accuracy Score
      #accuracy on test data
      X_test_prediction_model_logistic=model_logistic.predict(X_test)
      testining_data_accuracy_model_logistic=accuracy_score(X_test_prediction_model_logistic,Y_test)
```

```
[827] print('Accuracy on Testing  Data: ',testining_data_accuracy_model_logistic)
```

```
Accuracy on Testing  Data:  0.8005677785663591
```

5.2 Random Forest Classifier

```
[828] #Accuracy Score
      #accuracy on training data
      rf_preds_train = model_random_forest.predict(X_train)
```

```
[829] print("Training Accuracy:", accuracy_score(Y_train, rf_preds_train))
```

```
Training Accuracy: 0.8379481718139865
```

```
[830] #Accuracy Score
      #accuracy on test data
      rf_preds_test = model_random_forest.predict(X_test)
```

```
[831] print("Testing Accuracy:", accuracy_score(Y_test, rf_preds_test))
```

```
Testing Accuracy: 0.808374733853797
```

- In this Logistic model give us the best outcome with training prediction as 80% and Testing prediction 80%.

  In Random Forest Classifier it lead to overfitting of data, since in training it showing 83.7% and in testing it showing 80.8%

  In this we MSE=.13

  MAE=.26

  R-Squared value=.36

```
Training Set Evaluation:
Mean Squared Error: 0.13
Mean Absolute Error: 0.26
R-squared: 0.35

Test Set Evaluation:
Mean Squared Error: 0.13
Mean Absolute Error: 0.27
R-squared: 0.32
```

FINAL RESULT

- AS we have seen that ,
- In this Logistic model give us the best outcome with training prediction as 80%
- and
- Testing prediction 80%.

  In Random Forest Classifier in training it showing 83.7% and

  in testing it showing 80%

  In this we MSE=.13

  MAE=.26

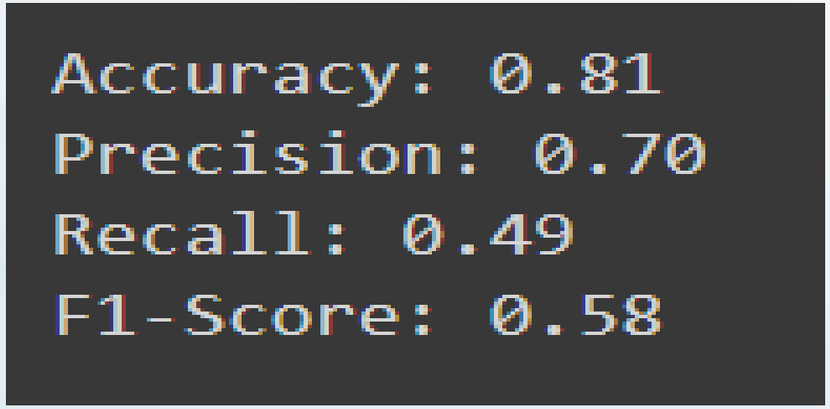
  R-Squared value=.36

  Accuracy: 0.81

  Precision: 0.70

  Recall: 0.49

  Fl-score: 0.58



Here the Best predictive output is given by Logistic Model and random forest as they give 80% accuracy with test data.

we have perform other various method to improve accuracy like combining various feature to create a new feature. We can also perform cross validation and other method to prevent the overfitting of data in Random Forest Classifier. With accuracy .81 we can say it's a decent model for predicting churn

Problem and Challenges

- Firstly, data cleaning was main problem we have given mainly all columns as categorical, converting them to numerical columns for prediction process was challenging.
- Then the we find the outliers and duplicate and handel them using appropriate method like for outlier Z- score method and IQR method.
- Then we have performed feature engineering on various column so that we can optimize the feature for prediction purpose
- when I firstly started doing this project I have faced the problem of overfitting in Random Forest Classification which lead to wrong prediction. So solve this problem I have included the various classification parameter which prevent it from overfitting like (n_estimators': 100, 'max_depth': 10, 'min_samples_split': 10, 'min_samples_leaf': 4, 'max_features': 'sqrt', 'bootstrap': True, 'random_state‘)

Earlier before including the parameter to this

5.2 Random Forest Classifier

```
[116] #Accuracy Score
      #accuracy on training data
      X_train_prediction_model_random_forest=model_random_forest.predict(X_train)
      training_data_accuracy_model_random_forest=accuracy_score(X_train_prediction_model_random_forest,Y_train)
```

```
[117] print('Accuracy on Training Data: ',training_data_accuracy_model_random_forest)
```

```
Accuracy on Training Data:  0.9980475683351083
```

```
[118] #Accuracy Score
      #accuracy on test data
      X_test_prediction_model_random_forest=model_random_forest.predict(X_test)
      testining_data_accuracy_model_random_forest=accuracy_score(X_test_prediction_model_random_forest,Y_test)
```

```
[119] print('Accuracy on Testing  Data: ',testining_data_accuracy_model_random_forest)
```

```
Accuracy on Testing  Data:  0.7934705464868701
```

After including the parameter to Random Forest Classification

5.2 Random Forest Classifier

```
[828] #Accuracy Score
      #accuracy on training data
      rf_preds_train = model_random_forest.predict(X_train)
```

```
[829] print("Training Accuracy:", accuracy_score(Y_train, rf_preds_train))
```

```
Training Accuracy: 0.8379481718139865
```

```
[830] #Accuracy Score
      #accuracy on test data
      rf_preds_test = model_random_forest.predict(X_test)
```

```
[831] print("Testing Accuracy:", accuracy_score(Y_test, rf_preds_test))
```

```
Testing Accuracy: 0.808374733853797
```

As we can see earlier the random forest classification was overfitting but when we added the parameter it , not showing the overfitting and predicting the values with accuracy of 80 %

Key Takeaways

- The project aims to develop a predictive model to identify customers at risk of churning, enabling the telecommunications company to take proactive measures to retain them.
- The process involves data collection and processing, exploratory data analysis, feature engineering, model building, and model evaluation.
- Three different models are trained and evaluated: Logistic Regression, Random Forest Classifier, and Gradient Boosting.
- The performance of the models is assessed on the training and testing data, with the Gradient Boosting model showing the best overall performance.

Conclusion

In conclusion, this project I developed a predictive model to identify customers at risk of churning, which can enable the telecommunications company to take proactive measures to retain them. The detailed process of data collection, exploratory analysis, feature engineering, model building, and model evaluation has been documented, providing a comprehensive guide for the company to implement and build upon this solution.

1

Data Collection

Gather customer data from various sources.

2

Data Processing

Clean, preprocess, and encode the data for modeling.

3

Model Building

Train and evaluate multiple machine learning models.

4

Deployment

Implement the best-performing model in production.