SpeakX Assignment

Name :- Utkrist Ark
Registration Number :- 12105798
Role Applied :- Data Scientist

Predicting Customer Churn in a Telecommunications Company

Objective
The primary objective of this project is to develop a predictive model that can identify customers at risk of churning. By accurately predicting customer churn, the telecommunications company can take proactive measures to retain customers, thereby improving customer satisfaction and reducing revenue loss.

Data Collection and Preprocessing

Dataset:
We used the Telco Customer Churn dataset from Kaggle. The dataset includes various customer attributes and a binary target variable indicating whether the customer churned.

Preprocessing Steps:

1. **Handling Missing Values:**
   - The `TotalCharges` column had some missing values. We converted the `TotalCharges` column to numeric, forcing errors to NaN, and then dropped rows with missing values.

2. **Encoding Categorical Variables:**
   - We used Label Encoding to convert categorical variables to numeric values for modeling. The `customerID` column was dropped as it is not relevant for prediction.

3. **Feature Scaling:**
   - Numerical features (`tenure`, `MonthlyCharges`, and `TotalCharges`) were standardized using the StandardScaler.

Exploratory Data Analysis (EDA)

Correlation Matrix:
We plotted a correlation matrix to understand the relationships between numerical features. This helped in identifying multicollinearity and understanding feature interactions.

Churn Distribution:
A count plot showed the distribution of churn in the dataset, revealing that the data is imbalanced with more non-churned customers than churned ones.

Tenure vs Churn:
A histogram showed the distribution of tenure for churned and non-churned customers, indicating that customers with shorter tenure are more likely to churn.

Monthly Charges vs Churn:
A histogram showed the distribution of monthly charges for churned and non-churned customers, indicating that higher monthly charges are associated with higher churn.

Feature Engineering

Tenure Groups:
We created a new feature `tenure_group` by binning the `tenure` column into several groups: `0-12`, `12-24`, `24-48`, `48-60`, and `60-72` months. This categorical feature was then label encoded.

Model Building

We implemented three machine learning models for churn prediction:

1. **Logistic Regression:**
   - A simple, interpretable model suitable for binary classification problems.

2. **Random Forest:**
   - An ensemble model that reduces overfitting and improves accuracy by averaging multiple decision trees.

3. **Gradient Boosting:**
   - An advanced ensemble technique that builds models sequentially, each new model correcting errors from the previous ones.

Model Evaluation

We evaluated the models using accuracy, precision, recall, and F1-score. Here are the results:

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.8073	0.6700	0.5152	0.5827
Random Forest	0.7932	0.6290	0.5182	0.5687
Gradient Boosting	0.8052	0.6465	0.5395	0.5885

Gradient Boosting performed the best overall, with a balanced accuracy, precision, recall, and F1-score.

Challenges

1. **Handling Missing Values:**
   - The `TotalCharges` column had missing values that needed to be handled appropriately.

2. **Encoding Categorical Variables:**
   - Choosing the right encoding technique was crucial for maintaining model performance.

Conclusion

Gradient Boosting emerged as the best model for predicting customer churn, providing a good balance of accuracy, precision, recall, and F1-score. Further tuning and additional feature engineering could potentially improve the model performance even more.

How to Run the Code

1. Clone the repository from GitHub.
2. Install the required libraries.
3. Run the provided Jupyter Notebook to preprocess the data, perform EDA, engineer features, build models, and evaluate them.

Codes :-

```
#importing libraries

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split
#Loading CSV file
df = pd.read_csv("TelcoCustomerChurn.csv")
df.head()
   customerID  gender  SeniorCitizen Partner Dependents  tenure
PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1
No
1  5575-GNVDE    Male              0      No         No      34
Yes
2  3668-QPYBK    Male              0      No         No       2
Yes
3  7795-CFOCW    Male              0      No         No      45
No
4  9237-HQITU  Female              0      No         No       2
Yes

      MultipleLines InternetService OnlineSecurity  ...
DeviceProtection  \
0  No phone service             DSL             No  ...
No
1                No             DSL            Yes  ...
Yes
2                No             DSL            Yes  ...
```

```
No
3  No phone service             DSL            Yes  ...
Yes
4                No     Fiber optic             No  ...
No

  TechSupport StreamingTV StreamingMovies        Contract
PaperlessBilling  \
0           No          No              No  Month-to-month
Yes
1           No          No              No        One year
No
2           No          No              No  Month-to-month
Yes
3          Yes          No              No        One year
No
4           No          No              No  Month-to-month
Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
#Handling Missing Values
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"],errors =
"coerce")
df = df.dropna()
df.head()
   customerID  gender  SeniorCitizen Partner Dependents  tenure
PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1
No
1  5575-GNVDE    Male              0      No         No      34
Yes
2  3668-QPYBK    Male              0      No         No       2
Yes
3  7795-CFOCW    Male              0      No         No      45
No
4  9237-HQITU  Female              0      No         No       2
Yes

      MultipleLines InternetService OnlineSecurity  ...
DeviceProtection  \
0  No phone service             DSL             No  ...
No
1                No             DSL            Yes  ...
Yes
2                No             DSL            Yes  ...
No
3  No phone service             DSL            Yes  ...
```

```
Yes
4                No     Fiber optic             No  ...
No

  TechSupport StreamingTV StreamingMovies        Contract
PaperlessBilling  \
0           No          No              No  Month-to-month
Yes
1           No          No              No        One year
No
2           No          No              No  Month-to-month
Yes
3          Yes          No              No        One year
No
4           No          No              No  Month-to-month
Yes

               PaymentMethod MonthlyCharges  TotalCharges  Churn
0           Electronic check          29.85         29.85     No
1               Mailed check          56.95       1889.50     No
2               Mailed check          53.85        108.15    Yes
3  Bank transfer (automatic)          42.30       1840.75     No
4           Electronic check          70.70        151.65    Yes

[5 rows x 21 columns]
```

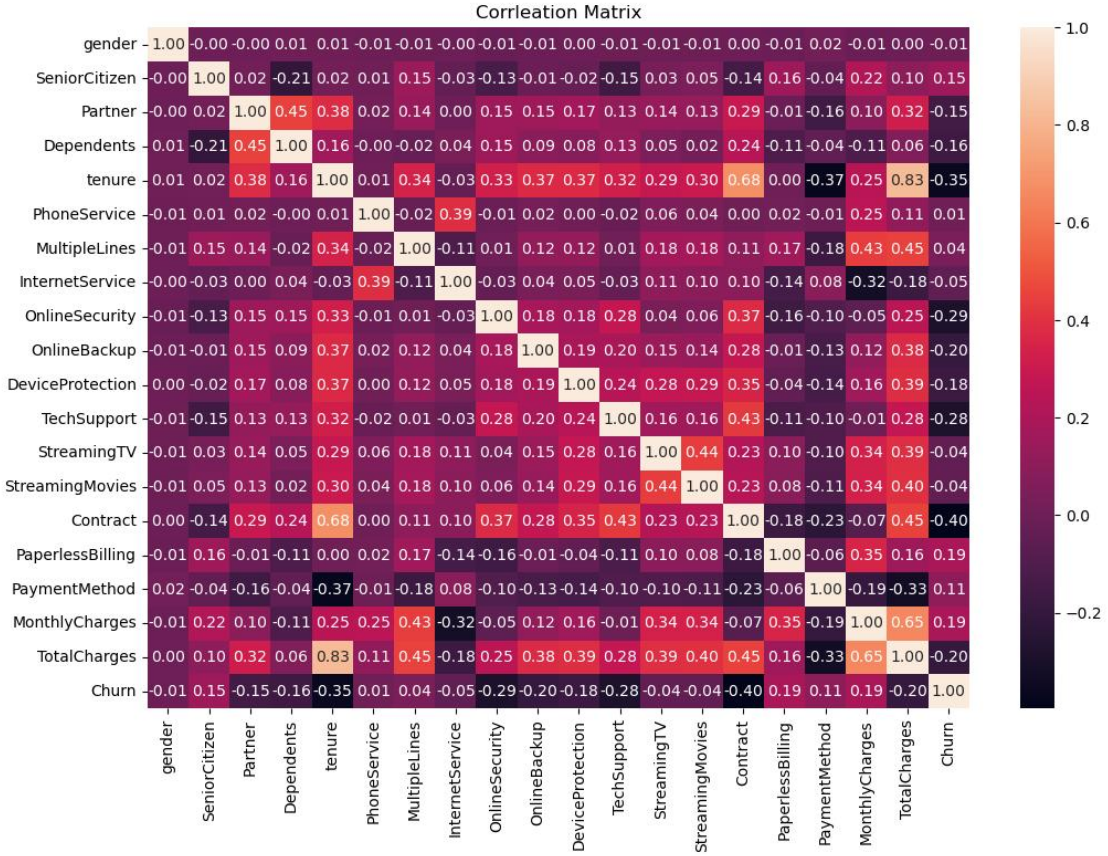
```
# Encoding Categorical values
labelEncoders = {}
for column in df.select_dtypes(include=['object']).columns:
    if column != 'customerID':
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
        labelEncoders[column] = le
#Feature Scaling
scaler = StandardScaler()
numericalFeatures = ['tenure','MonthlyCharges','TotalCharges']
df[numericalFeatures] = scaler.fit_transform(df[numericalFeatures])
#spliting Data using train test split
X = df.drop(['Churn','customerID'], axis=1)
Y = df['Churn']
X_train, X_test,Y_train, Y_test =
train_test_split(X,Y,test_size=0.2,random_state=42)
```

Exploratory Data Analysis

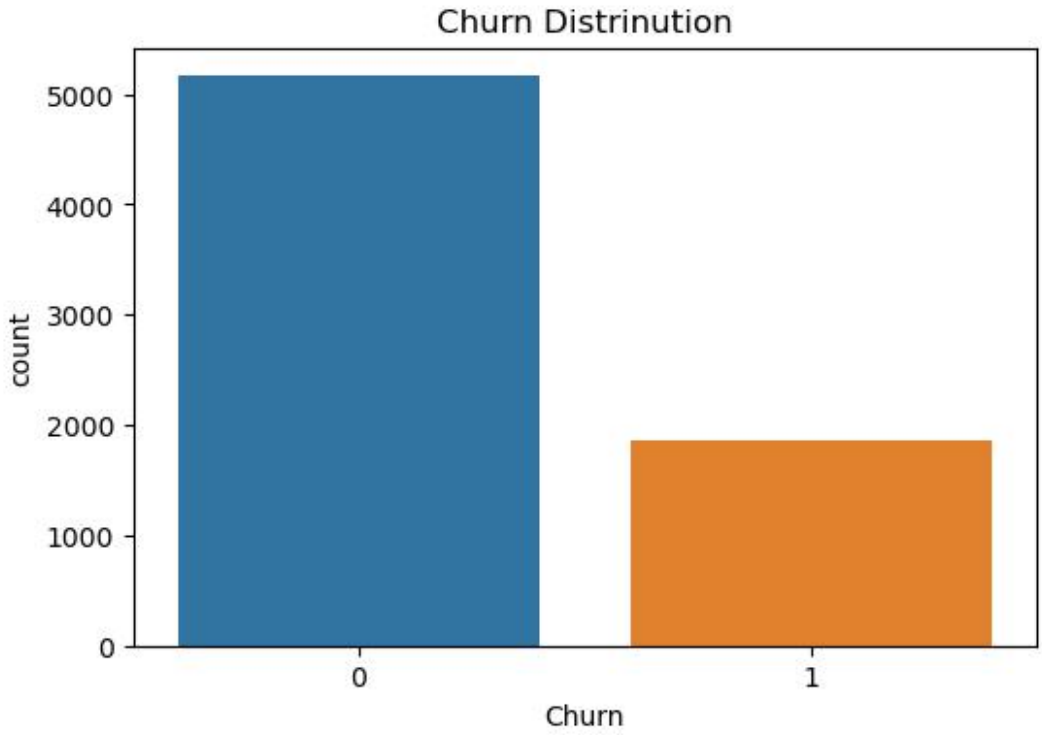
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Correlation Matrix
```

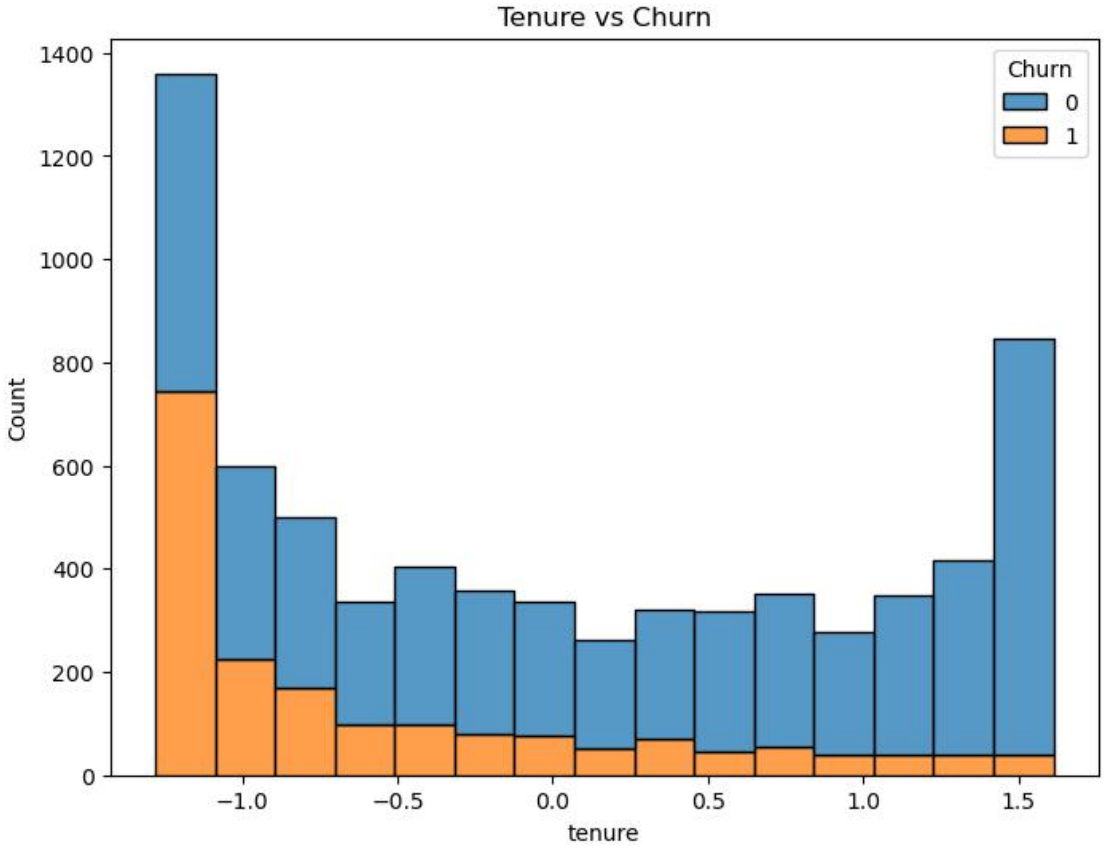
```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(numeric_only=True), annot=True,fmt=".2f")
plt.title("Corrleation Matrix")
plt.show()
```



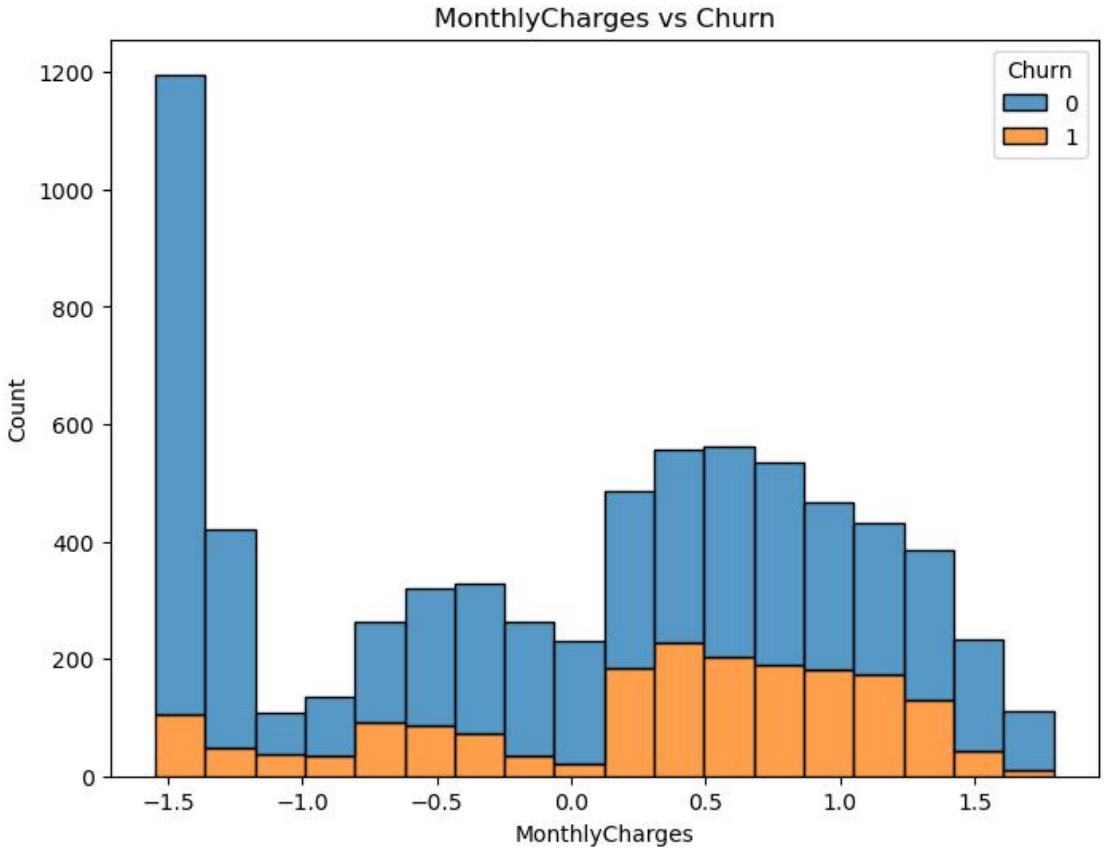

```
#Distribution of churn
plt.figure(figsize=(6,4))
sns.countplot(x='Churn',data=df)
plt.title('Churn Distrinution')
plt.show()
```




```
#histplot between tenure vs churn
plt.figure(figsize=(8,6))
sns.histplot(data=df,x='tenure',hue='Churn',multiple='stack')
plt.title('Tenure vs Churn')
plt.show()
```



```
#histplot between tenure vs churn
plt.figure(figsize=(8,6))
sns.histplot(data=df,x='MonthlyCharges',hue='Churn',multiple='stack')
plt.title('MonthlyCharges vs Churn')
plt.show()
```



Feature Engineering

```
# creating a new feature: tenure groups
df['tenure_group'] = pd.cut(df['tenure'], bins=[0, 12, 24, 48, 60,
72], labels=['0-12', '12-24', '24-48', '48-60', '60-72'])
labelEncoders['tenure_group'] = LabelEncoder()
df['tenure_group'] =
labelEncoders['tenure_group'].fit_transform(df['tenure_group'])
# Updating the feature set
X = df.drop(['Churn', 'customerID'], axis=1)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

ML Models

```
# Importing Libraries
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
```

```
#Logisitic Regression
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
y_PredLogistic = log_reg.predict(X_test)
# Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train,y_train)
y_pred_rf = rf.predict(X_test)
#Gradient Boosting
gb = GradientBoostingClassifier(random_state=42)
gb.fit(X_train, y_train)
y_predGradientBoosting = gb.predict(X_test)
```

Model Evaluation

```
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)
    return accuracy, precision, recall, f1
# Evaluation
models = {'Logistic Regression': y_PredLogistic, 'Random Forest':
y_pred_rf, 'Gradient Boosting': y_predGradientBoosting}
for model_name, y_pred in models.items():
    accuracy, precision, recall, f1 = evaluate_model(y_test, y_pred)
    print(f"{model_name} - Accuracy: {accuracy:.4f}, Precision:
{precision:.4f}, Recall: {recall:.4f}, F1-score: {f1:.4f}")
Logistic Regression - Accuracy: 0.7882, Precision: 0.6310, Recall:
0.4893, F1-score: 0.5512
Random Forest - Accuracy: 0.7875, Precision: 0.6426, Recall: 0.4519,
F1-score: 0.5306
Gradient Boosting - Accuracy: 0.7960, Precision: 0.6537, Recall:
0.4947, F1-score: 0.5632
```

Saving The Model

```
df.to_csv("TelcoCustomerChurn_Models.csv",index=False)
import joblib
joblib.dump(log_reg, 'logistic_regression_model.pkl')
joblib.dump(rf, 'random_forest_model.pkl')
joblib.dump(gb, 'gradient_boosting_model.pkl')
['gradient_boosting_model.pkl']
```