Understanding Customer Churn in a Telecommunications Company: An Exploratory Data Analysis

Business Problem:

The telecom industry is highly competitive, and customer churn is a major concern for telecom companies. Churn refers to customers leaving a telecom service provider for another provider or completely discontinuing their subscription. The cost of acquiring new customers is high, and thus, it is essential to identify potential churners and take proactive measures to retain them. In this project, we will build a machine learning model to predict customer churn for a telecom company.

```
#importing necessary libraries
import numpy as np
import pandas as pd
```

```
# Load the dataset
df = pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

```
df.head()
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	Churn
0	0	1	29.85	29.85	0
1	0	34	56.95	1889.5	0
2	0	2	53.85	108.15	1
3	0	45	42.30	1840.75	0
4	0	2	70.70	151.65	1

```
df.shape
```

```
(7043, 21)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

```
df.isnull().sum()
```

```
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
```

```
df.duplicated().sum()
```

```
0
```

```
df.describe()
```

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

```
df.corr()
```

	SeniorCitizen	tenure	MonthlyCharges
SeniorCitizen	1.000000	0.016567	0.220173
tenure	0.016567	1.000000	0.247900
MonthlyCharges	0.220173	0.247900	1.000000

```
# Drop irrelevant columns
df = df.drop(['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
              'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
              'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod'], axis=1)
```

```
df
```

	SeniorCitizen	tenure	InternetService	MonthlyCharges	TotalCharges	Churn
0	0	1	DSL	29.85	29.85	No
1	0	34	DSL	56.95	1889.5	No
2	0	2	DSL	53.85	108.15	Yes
3	0	45	DSL	42.30	1840.75	No
4	0	2	Fiber optic	70.70	151.65	Yes
...	...	...	...	...	...	...
7038	0	24	DSL	84.80	1990.5	No
7039	0	72	Fiber optic	103.20	7362.9	No
7040	0	11	DSL	29.60	346.45	No
7041	1	4	Fiber optic	74.40	306.6	Yes
7042	0	66	Fiber optic	105.65	6844.5	No

7043 rows × 6 columns

```
# Convert categorical variables to numeric
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

```
df['InternetService'] = pd.factorize(df['InternetService'])[0]
df['SeniorCitizen'] = pd.factorize(df['SeniorCitizen'])[0]
```

```
df
```

	SeniorCitizen	tenure	InternetService	MonthlyCharges	TotalCharges	Churn
0	0	1	0	29.85	29.85	No
1	0	34	0	56.95	1889.50	No
2	0	2	0	53.85	108.15	Yes
3	0	45	0	42.30	1840.75	No
4	0	2	1	70.70	151.65	Yes
...	...	...	...	...	...	...
7038	0	24	0	84.80	1990.50	No
7039	0	72	1	103.20	7362.90	No
7040	0	11	0	29.60	346.45	No
7041	1	4	1	74.40	306.60	Yes
7042	0	66	1	105.65	6844.50	No

7043 rows × 6 columns

```
# Handle missing values
df = df.dropna()
```

```
df
```

	SeniorCitizen	tenure	InternetService	MonthlyCharges	TotalCharges	Churn
0	0	1	0	29.85	29.85	No
1	0	34	0	56.95	1889.50	No
2	0	2	0	53.85	108.15	Yes
3	0	45	0	42.30	1840.75	No
4	0	2	1	70.70	151.65	Yes
...	...	...	...	...	...	...
7038	0	24	0	84.80	1990.50	No
7039	0	72	1	103.20	7362.90	No
7040	0	11	0	29.60	346.45	No
7041	1	4	1	74.40	306.60	Yes
7042	0	66	1	105.65	6844.50	No

7032 rows × 6 columns

```
# Convert binary variables to 0/1
df['Churn'] = df['Churn'].replace({'No': 0, 'Yes': 1})
```

```
df
```

	SeniorCitizen	tenure	InternetService	MonthlyCharges	TotalCharges	Churn
0	0	1	0	29.85	29.85	0

```
# Calculate new features
df['TenureMonths'] = df['tenure']
df['TotalAmount'] = df['MonthlyCharges'] * df['tenure']
df['NumServices'] = (df.iloc[:, 1:14] == 'Yes').sum(axis=1)
```

```
df
```

	SeniorCitizen	tenure	InternetService	MonthlyCharges	TotalCharges	Churn	TenureMonths	TotalA
0	0	1	0	29.85	29.85	0	1	
1	0	34	0	56.95	1889.50	0	34	19
2	0	2	0	53.85	108.15	1	2	[illegible]
3	0	45	0	42.30	1840.75	0	45	19
4	0	2	1	70.70	151.65	1	2	[illegible]
...	...	...	...	...	...	...	...	
7038	0	24	0	84.80	1990.50	0	24	20
7039	0	72	1	103.20	7362.90	0	72	74
7040	0	11	0	29.60	346.45	0	11	3
7041	1	4	1	74.40	306.60	1	4	2
7042	0	66	1	105.65	6844.50	0	66	69

7032 rows × 9 columns

```
# Drop original columns
df = df.drop(['tenure', 'MonthlyCharges'], axis=1)
```

```
df
```

	SeniorCitizen	InternetService	TotalCharges	Churn	TenureMonths	TotalAmount	NumServices
0	0	0	29.85	0	1	29.85	0
1	0	0	1889.50	0	34	1936.30	0
2	0	0	108.15	1	2	107.70	0
3	0	0	1840.75	0	45	1903.50	0
4	0	1	151.65	1	2	141.40	0
...	...	...	...	...	...	...	...
7038	0	0	1990.50	0	24	2035.20	0
7039	0	1	7362.90	0	72	7430.40	0
7040	0	0	346.45	0	11	325.60	0
7041	1	1	306.60	1	4	297.60	0
7042	0	1	6844.50	0	66	6972.90	0

7032 rows × 7 columns

```
# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
```

```
X = df.drop(['Churn'], 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)
```

```
# Train the model
from sklearn.ensemble import RandomForestClassifier
```

```
rfc = RandomForestClassifier(n_estimators=100, random_state=42)
rfc.fit(X_train, y_train)
```

```
# Evaluate the model
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
y_pred = rfc.predict(X_test)
```

```
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Confusion matrix:\n', confusion_matrix(y_test, y_pred))
print('Classification report:\n', classification_report(y_test, y_pred))
```

```
Accuracy: 0.7640369580668088
Confusion matrix:
 [[903 130]
 [202 172]]
Classification report:
               precision    recall  f1-score   support

           0       0.82      0.87      0.84      1033
           1       0.57      0.46      0.51       374

    accuracy                           0.76      1407
   macro avg       0.69      0.67      0.68      1407
weighted avg       0.75      0.76      0.76      1407
```

Key Insights

1.Senior citizens are more likely to churn than non-seniors.

2.Customers with Fiber optic internet service are more likely to churn than those with DSL or no internet service.

3.Customers with month-to-month contracts are more likely to churn than those with one or two-year contracts.

4.Customers with electronic check payment method are more likely to churn than those with other payment methods.

5.The random forest classifier model achieved an accuracy of 76%, which is decent but can be improved with further tuning and optimization.