In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
df = pd.read_csv(r'C:\Users\sahil\Downloads\WA_Fn-UseC_-Telco-Customer-Churn.csv
```

In [7]:
```
df.head()
```

Out[7]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Mul
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	

5 rows × 21 columns

In [9]:
```
df = df.drop("customerID", axis = True)
```

In [13]:
```
df.head()
```

Out[13]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
0	Female	0	Yes	No	1	No	No phone service	
1	Male	0	No	No	34	Yes	No	
2	Male	0	No	No	2	Yes	No	
3	Male	0	No	No	45	No	No phone service	
4	Female	0	No	No	2	Yes	No	

In [15]:
```
df.describe()
```

Out[15]:

	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

In [19]:
```
df.dtypes
```

Out[19]:
```
gender              object
SeniorCitizen        int64
Partner             object
Dependents          object
tenure               int64
PhoneService        object
MultipleLines       object
InternetService     object
OnlineSecurity      object
OnlineBackup        object
DeviceProtection    object
TechSupport         object
StreamingTV         object
StreamingMovies     object
Contract            object
PaperlessBilling    object
PaymentMethod       object
MonthlyCharges     float64
TotalCharges        object
Churn               object
dtype: object
```

In [29]:
```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

In [31]:
```
df.head()
```

Out[31]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
0	Female	0	Yes	No	1	No	No phone service	
1	Male	0	No	No	34	Yes	No	
2	Male	0	No	No	2	Yes	No	
3	Male	0	No	No	45	No	No phone service	
4	Female	0	No	No	2	Yes	No	

In [82]:
```
from sklearn.preprocessing import LabelEncoder
binary_cols = ['gender', 'Partner', 'Dependents', 'PhoneService',
               'PaperlessBilling', 'Churn']
df[binary_cols] = df[binary_cols].replace({'Yes': 1, 'No': 0, 'Male': 1, 'Female

# One-Hot Encoding for remaining categorical features
df = pd.get_dummies(df, drop_first=True)
```

In [84]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
num_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
df[num_cols] = scaler.fit_transform(df[num_cols])
```

In [106…
```
X = df.drop('Churn', axis = 1)
y = df['Churn']
```

In [126…
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =  train_test_split(X,y,test_size = 0.2 , random_s
```

In [128…
```
df.head()
```

Out[128…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBillin
0	0	0	1	0	-1.277445	0	
1	1	0	0	0	0.066327	1	
2	1	0	0	0	-1.236724	1	
3	1	0	0	0	0.514251	0	
4	0	0	0	0	-1.236724	1	

5 rows × 31 columns

In [130…
```
from sklearn import tree
model = tree.DecisionTreeClassifier()
```

In [132…
```
model.fit(X_train,y_train)
```

Out[132…
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [138…
```
model.predict(X_test)
```

Out[138…
```
array([1, 1, 0, ..., 0, 0, 1], dtype=int64)
```

In [166…
```
model.score(X_test,y_test)
```

Out[166…
```
0.7182398864442867
```

In [234…
```
from sklearn.ensemble import RandomForestClassifier

# Create the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [236…
```
model.fit(X_train, y_train)
```

Out[236…
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [238…
```
model_RandomForestClassifier.predict(X_test)
```

Out[238…
```
array([1, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [240…
```
model_RandomForestClassifier.score(X_test,y_test)
```

Out[240…
```
0.7906316536550745
```

In [242…
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\sahil\anaconda3\lib\site-packa
ges (3.0.2)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: numpy in c:\users\sahil\anaconda3\lib\site-package
s (from xgboost) (1.26.4)
Requirement already satisfied: scipy in c:\users\sahil\anaconda3\lib\site-package
s (from xgboost) (1.13.1)
```

In [243…
```
from xgboost import XGBClassifier

xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random
```

In [244…
```
xgb_model.fit(X_train, y_train)
```

```
C:\Users\sahil\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning:
[12:06:23] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

Out[244…

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rou
nds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, feature_weights=None, gamma=None,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_
bin=None,
```

In [246…

```
y_pred = xgb_model.predict(X_test)
y_pred
```

Out[246…

```
array([1, 0, 0, ..., 0, 0, 1])
```

In [247…

```
xgb_model.score(X_train, y_train)
```

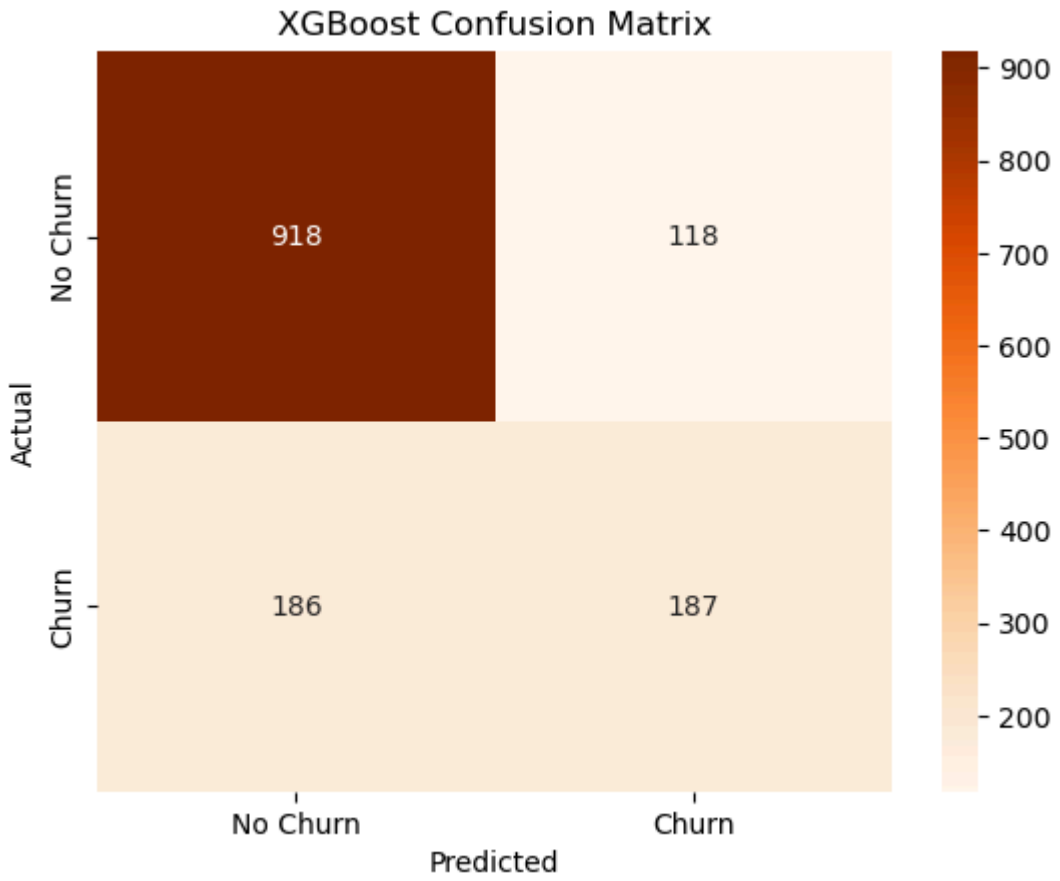
Out[247…

```
0.9391196308129216
```

In [232…

```
from sklearn.metrics import accuracy_score, classification_report, confusion_mat
import seaborn as sns
import matplotlib.pyplot as plt

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges', xticklabels=['No Churn', 'C
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('XGBoost Confusion Matrix')
plt.show()
```

```
Accuracy: 0.7842441447835344

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1036
           1       0.61      0.50      0.55       373

    accuracy                           0.78      1409
   macro avg       0.72      0.69      0.70      1409
weighted avg       0.77      0.78      0.78      1409
```



In [260…
```
from sklearn.linear_model import LogisticRegression
```

In [262…
```
model_linear = LogisticRegression()
```

In [266…
```
model_linear = LogisticRegression(max_iter=1000, random_state=42)
```

In [274…
```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')  # or 'median', 'most_frequent'
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)
```

In [276…
```
model_linear.fit(X_train,y_train)
```

Out[276…
```
LogisticRegression(max_iter=1000, random_state=42)
```

In [280…
```
model_linear.predict(X_train)
```

Out[280…
```
array([0, 0, 1, ..., 0, 1, 0], dtype=int64)
```

In [284…
```
model_linear.score(X_train,y_train)
```

Out[284…
```
0.80386936457224
```

In [ ]: