Customer Churn Prediction
Part B

1 Customer Churn Prediction Project Part (B)

1.0.1 1. Load the Data

[11]:
```python
import pandas as pd

# Load the dataset
df = pd.read_csv("Customer_data - customer_data.csv")
df.head()
```

[11]:
```
   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]
```

1.0.2 2. Data Cleaning and Preprocessing

[12]:
```
# Convert TotalCharges to numeric and drop missing
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df = df.dropna(subset=['TotalCharges'])

# Replace SeniorCitizen binary to string
df['SeniorCitizen'] = df['SeniorCitizen'].replace({1: 'Yes', 0: 'No'})

# Drop customerID
df.drop('customerID', axis=1, inplace=True)

# Encode target variable
df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})

# One-hot encoding
df_encoded = pd.get_dummies(df, drop_first=True)
df_encoded.head()
```

[12]:
```
   tenure  MonthlyCharges  TotalCharges  Churn  gender_Male  \
0       1           29.85         29.85      0        False
1      34           56.95       1889.50      0         True
2       2           53.85        108.15      1         True
3      45           42.30       1840.75      0         True
4       2           70.70        151.65      1        False

   SeniorCitizen_Yes  Partner_Yes  Dependents_Yes  PhoneService_Yes  \
0              False         True           False             False
1              False        False           False              True
2              False        False           False              True
3              False        False           False             False
4              False        False           False              True

   MultipleLines_No phone service  …  StreamingTV_No internet service  \
0                            True  …                            False
1                           False  …                            False
2                           False  …                            False
3                            True  …                            False
4                           False  …                            False

   StreamingTV_Yes  StreamingMovies_No internet service  StreamingMovies_Yes  \
0            False                                False                False
1            False                                False                False
2            False                                False                False
3            False                                False                False
```

```
4           False                                    False                False

   Contract_One year  Contract_Two year  PaperlessBilling_Yes  \
0              False              False                  True
1               True              False                 False
2              False              False                  True
3               True              False                 False
4              False              False                  True

   PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
0                                  False                            True
1                                  False                           False
2                                  False                           False
3                                  False                           False
4                                  False                            True

   PaymentMethod_Mailed check
0                       False
1                        True
2                        True
3                       False
4                       False

[5 rows x 31 columns]
```

1.0.3 3. Train-Test Split and Feature Scaling

[13]:
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = df_encoded.drop('Churn', axis=1)
y = df_encoded['Churn']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Scale
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

1.0.4 4. Train Logistic Regression and Random Forest Models

[14]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# Use scaled data for logistic regression
log_model = LogisticRegression(max_iter=2000, solver='lbfgs')
log_model.fit(X_train_scaled, y_train)

# Use original data for random forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

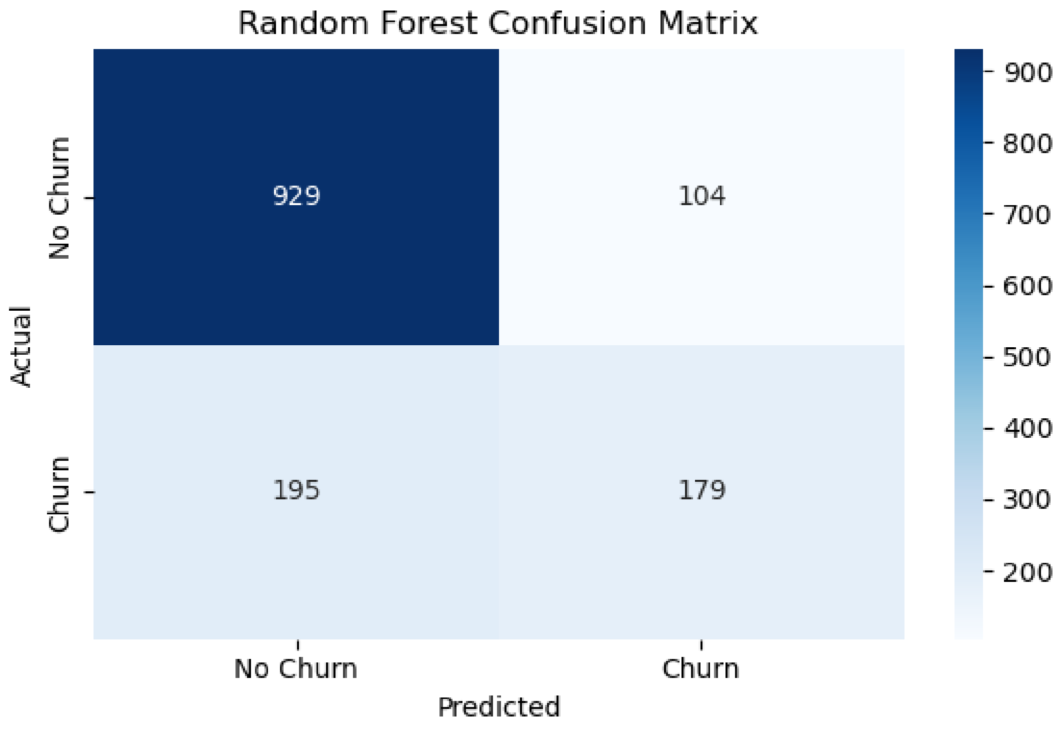
[14]:
```
RandomForestClassifier(random_state=42)
```

1.0.5 5. Evaluate Model Performance

[15]:
```
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Predict
log_pred = log_model.predict(X_test_scaled)
rf_pred = rf_model.predict(X_test)

# Reports
print("Logistic Regression Report:\n", classification_report(y_test, log_pred))
print("Random Forest Report:\n", classification_report(y_test, rf_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, rf_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Churn',␣
↪'Churn'], yticklabels=['No Churn', 'Churn'])
plt.title("Random Forest Confusion Matrix")
plt.ylabel("Actual")
plt.xlabel("Predicted")
plt.tight_layout()
plt.show()
```

```
Logistic Regression Report:
               precision    recall  f1-score   support

           0       0.83      0.89      0.86      1033
           1       0.62      0.52      0.56       374

    accuracy                           0.79      1407
   macro avg       0.73      0.70      0.71      1407
```

```
weighted avg       0.78      0.79      0.78      1407

Random Forest Report:
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1033
           1       0.63      0.48      0.54       374

    accuracy                           0.79      1407
   macro avg       0.73      0.69      0.70      1407
weighted avg       0.77      0.79      0.78      1407
```



1.1 6. Save Final Models and Processed Data

[16]:
```python
import joblib

joblib.dump(log_model, "logistic_regression_model_scaled.pkl")
joblib.dump(rf_model, "random_forest_model.pkl")
df_encoded.to_csv("processed_customer_data.csv", index=False)
```

Notebook completed with scaling for Logistic Regression