1. Introduction

This project predicts customer churn using ML to identify patterns in customer behavior.

2. Import libraries

```
# 1. Introduction
# This project predicts customer churn using ML to identify patterns in customer behavior.

# 2. Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import (
    classification_report, confusion_matrix, accuracy_score,
    precision_score, recall_score, f1_score
)
import warnings
warnings.filterwarnings('ignore')
```

3. Read dataset

```
# 3. Read dataset
df = pd.read_csv("/content/WA_Fn-UseC_-Telco-Customer-Churn.csv")  # Replace with your path
```

4. Exploratory data analysis (EDA)

```
# 4. Exploratory data analysis (EDA)

# 4.1 Shape of dataset
print("Shape:", df.shape)

# 4.2 Preview dataset
print(df.head())

# 4.3 Summary of dataset
print(df.info())

# 4.4 Statistical properties
print(df.describe())
```

```
Shape: (7043, 21)
   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]
<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
```

5. Feature scaling

```
# 5. Feature selection
df.drop(['customerID'], axis=1, inplace=True)

# Convert TotalCharges to numeric
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df['TotalCharges'].fillna(df['TotalCharges'].mean(), inplace=True)
```

6. Convert categorical columns to numeric columns

```
# 6. Convert categorical columns to numeric columns
# 6.1 Explore Gender
print(df['gender'].value_counts())

# 6.2 Explore Geography (replacing with TenureGroups as a categorical example)
print(df['Contract'].value_counts())

# Label encode binary columns
le = LabelEncoder()
binary_cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn']
for col in binary_cols:
    df[col] = le.fit_transform(df[col])

# One-hot encode remaining categoricals
df = pd.get_dummies(df, drop_first=True)
```

```
gender
Male      3555
Female    3488
Name: count, dtype: int64
Contract
Month-to-month    3875
Two year          1695
One year          1473
Name: count, dtype: int64
```

7. Feature Scaling

```
# 7. Feature Scaling
X = df.drop('Churn', axis=1)
y = df['Churn']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

8. Model Training

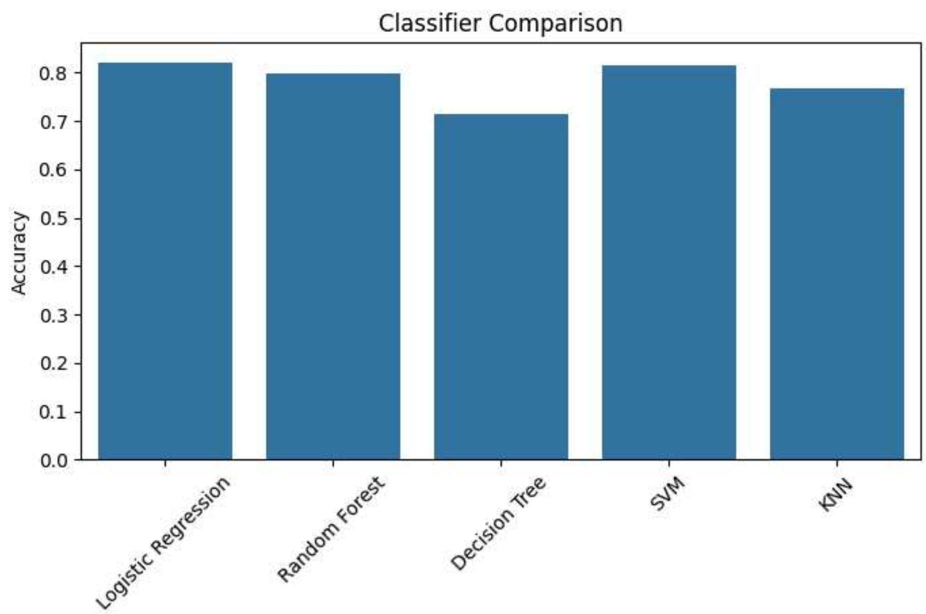
```
# 8. Model Training
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# 8.1 Predict accuracy with different algorithms
models = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "SVM": SVC(),
    "KNN": KNeighborsClassifier()
}

accuracy_scores = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracy_scores[name] = acc
    print(f"{name} Accuracy: {acc:.2f}")
```

```
Logistic Regression Accuracy: 0.82
Random Forest Accuracy: 0.80
Decision Tree Accuracy: 0.71
SVM Accuracy: 0.81
KNN Accuracy: 0.77
```

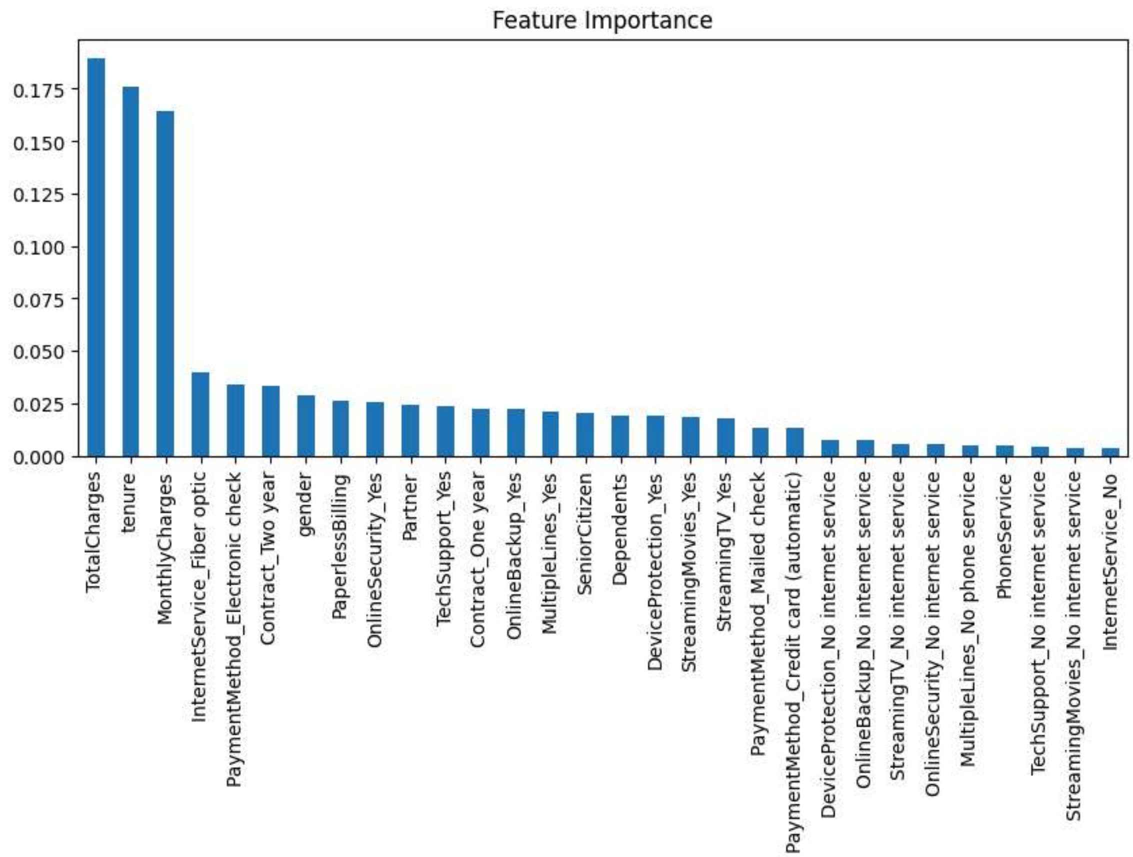
```
# 8.2 Plot classifier accuracy scores
plt.figure(figsize=(8, 4))
sns.barplot(x=list(accuracy_scores.keys()), y=list(accuracy_scores.values()))
plt.ylabel("Accuracy")
plt.title("Classifier Comparison")
plt.xticks(rotation=45)
plt.show()
```



9. Feature Importance

```
# 9. Feature Importance
# 9.1 Using Random Forest
rf = RandomForestClassifier()
rf.fit(X_train, y_train)

importances = rf.feature_importances_
feat_df = pd.Series(importances, index=X.columns).sort_values(ascending=False)
feat_df.plot(kind='bar', figsize=(10, 4), title="Feature Importance")
plt.show()
```

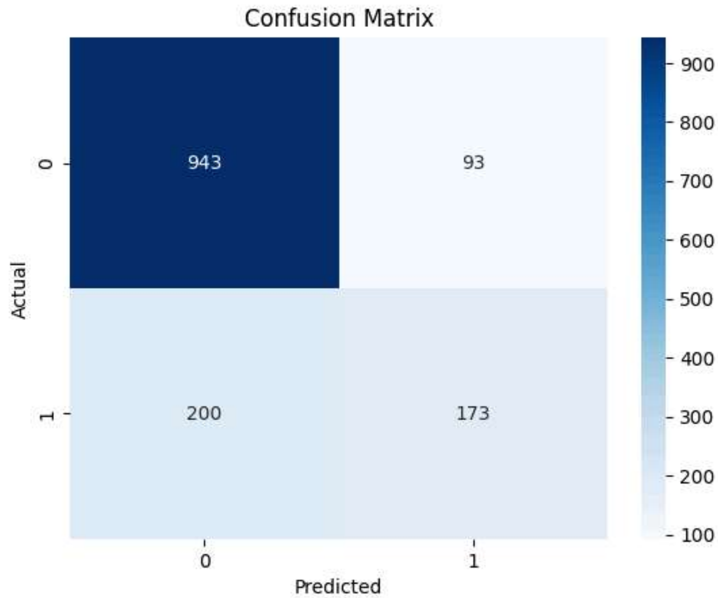



```
# 9.2 Drop least important feature
least_important = feat_df.idxmin()
print("Dropping:", least_important)
X_reduced = df.drop(columns=['Churn', least_important])
X_reduced_scaled = scaler.fit_transform(X_reduced)
```

```
Dropping: InternetService_No
```

10. Confusion Matrix

```
# 10. Confusion Matrix
y_pred = rf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



11. Classification Metrics

```
# 11. Classification Metrics
print("11.1 Classification Report")
print(classification_report(y_test, y_pred))

# 11.2 Accuracy
acc = accuracy_score(y_test, y_pred)
print("11.2 Accuracy:", acc)

# 11.3 Error
print("11.3 Error:", 1 - acc)

# 11.4 Precision
print("11.4 Precision:", precision_score(y_test, y_pred))

# 11.5 Recall
print("11.5 Recall:", recall_score(y_test, y_pred))

# 11.6 True Positive Rate (Recall)
print("11.6 TPR:", recall_score(y_test, y_pred))

# 11.7 False Positive Rate
fpr = cm[0][1] / (cm[0][0] + cm[0][1])
print("11.7 FPR:", fpr)

# 11.8 Specificity (True Negative Rate)
tnr = cm[0][0] / (cm[0][0] + cm[0][1])
print("11.8 Specificity:", tnr)

# 11.9 F1 Score
print("11.9 F1 Score:", f1_score(y_test, y_pred))

# 11.10 Support
print("11.10 Support:", classification_report(y_test, y_pred, output_dict=True)['1']['support'])
```

```
11.1 Classification Report
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.65      0.46      0.54       373

    accuracy                           0.79      1409
   macro avg       0.74      0.69      0.70      1409
weighted avg       0.78      0.79      0.78      1409

11.2 Accuracy: 0.7920511000709723
11.3 Error: 0.20794889992902765
11.4 Precision: 0.650375939849624
```

```
11.5 Recall: 0.46380697050938335
11.6 TPR: 0.46380697050938335
11.7 FPR: 0.08976833976833977
11.8 Specificity: 0.9102316602316602
11.9 F1 Score: 0.5414710485133021
11.10 Support: 373.0
```

12. cross validation

```
# 12. Cross-Validation
cv_scores = cross_val_score(RandomForestClassifier(), X_scaled, y, cv=5)
print("CV Accuracy:", cv_scores)
print("Mean CV Accuracy:", np.mean(cv_scores))
```

```
CV Accuracy: [0.79063165 0.79701916 0.77217885 0.79829545 0.79616477]
Mean CV Accuracy: 0.7908579787405638
```

13.Result

```
# 13. Results and Conclusion
best_model = max(accuracy_scores, key=accuracy_scores.get)
print(f"\nBest model: {best_model} with accuracy of {accuracy_scores[best_model]:.2f}")
print("Random Forest's top features:")
print(feat_df.head(5))
```

```
Best model: Logistic Regression with accuracy of 0.82
Random Forest's top features:
TotalCharges                      0.189387
tenure                            0.175794
MonthlyCharges                    0.164157
InternetService_Fiber optic       0.039872
PaymentMethod_Electronic check    0.034138
dtype: float64
```