```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import joblib
```

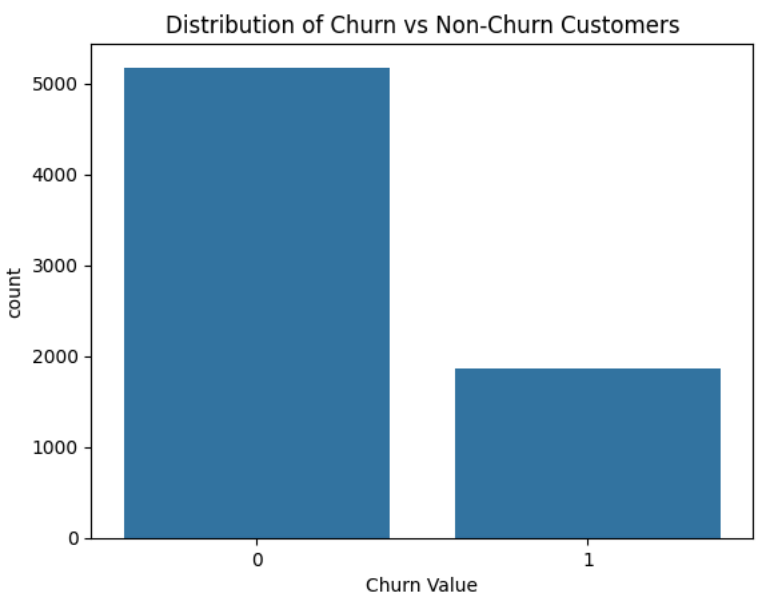
```
df = pd.read_excel('Telco_customer_churn.xlsx')
print(df.head())
print(df.info())
```

```
4  34.039224, -118.266293  34.039224 -118.266293    Male  ...  Month-to-month

  Paperless Billing             Payment Method  Monthly Charges Total Charges  \
0               Yes               Mailed check            53.85        108.15
1               Yes           Electronic check            70.70        151.65
2               Yes           Electronic check            99.65         820.5
3               Yes           Electronic check           104.80       3046.05
4               Yes  Bank transfer (automatic)           103.70        5036.3

  Churn Label Churn Value Churn Score  CLTV                   Churn Reason
0         Yes           1          86  3239   Competitor made better offer
1         Yes           1          67  2701                          Moved
2         Yes           1          86  5372                          Moved
3         Yes           1          84  5003                          Moved
4         Yes           1          89  5340  Competitor had better devices

[5 rows x 33 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 33 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   CustomerID         7043 non-null   object
 1   Count              7043 non-null   int64
 2   Country            7043 non-null   object
 3   State              7043 non-null   object
 4   City               7043 non-null   object
 5   Zip Code           7043 non-null   int64
 6   Lat Long           7043 non-null   object
 7   Latitude           7043 non-null   float64
 8   Longitude          7043 non-null   float64
 9   Gender             7043 non-null   object
 10  Senior Citizen     7043 non-null   object
 11  Partner            7043 non-null   object
 12  Dependents         7043 non-null   object
 13  Tenure Months      7043 non-null   int64
 14  Phone Service      7043 non-null   object
 15  Multiple Lines     7043 non-null   object
 16  Internet Service   7043 non-null   object
 17  Online Security    7043 non-null   object
 18  Online Backup      7043 non-null   object
 19  Device Protection  7043 non-null   object
 20  Tech Support       7043 non-null   object
 21  Streaming TV       7043 non-null   object
 22  Streaming Movies   7043 non-null   object
 23  Contract           7043 non-null   object
 24  Paperless Billing  7043 non-null   object
 25  Payment Method     7043 non-null   object
 26  Monthly Charges    7043 non-null   float64
 27  Total Charges      7043 non-null   object
 28  Churn Label        7043 non-null   object
 29  Churn Value        7043 non-null   int64
 30  Churn Score        7043 non-null   int64
 31  CLTV               7043 non-null   int64
 32  Churn Reason       1869 non-null   object
dtypes: float64(3), int64(6), object(24)
memory usage: 1.8+ MB
None
```

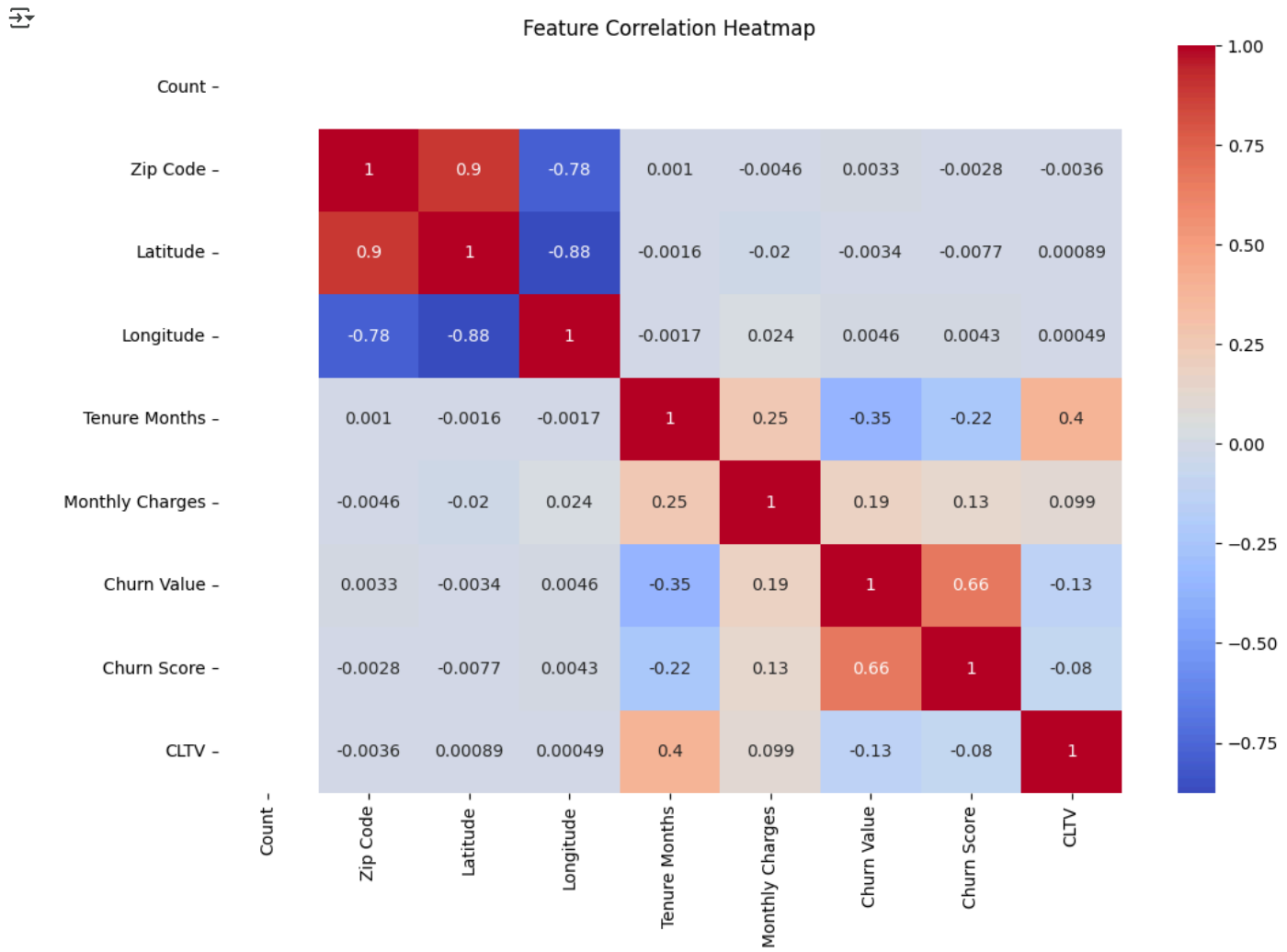
```
print(df.columns)
```

```
Index(['CustomerID', 'Count', 'Country', 'State', 'City', 'Zip Code',
       'Lat Long', 'Latitude', 'Longitude', 'Gender', 'Senior Citizen',
       'Partner', 'Dependents', 'Tenure Months', 'Phone Service',
       'Multiple Lines', 'Internet Service', 'Online Security',
       'Online Backup', 'Device Protection', 'Tech Support', 'Streaming TV',
       'Streaming Movies', 'Contract', 'Paperless Billing', 'Payment Method',
       'Monthly Charges', 'Total Charges', 'Churn Label', 'Churn Value',
       'Churn Score', 'CLTV', 'Churn Reason'],
      dtype='object')
```

```
sns.countplot(data=df, x='Churn Value')
plt.title('Distribution of Churn vs Non-Churn Customers')
plt.show()
```




```
plt.figure(figsize=(12,8))

numerical_df = df.select_dtypes(include=['int64', 'float64'])

sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.show()
```




```
label_encoder = LabelEncoder()
df['Churn Value'] = label_encoder.fit_transform(df['Churn Value'])
```

```
df = df.drop(columns=['CustomerID'])
```

```
print(df.columns.tolist())
```

```
['Count', 'Country', 'State', 'City', 'Zip Code', 'Lat Long', 'Latitude', 'Longitude', 'Gender', 'Senior Citizen', 'Partner', 'Deper
```

```
columns_to_drop = [
    'Count', 'Country', 'State', 'City', 'Zip Code',
    'Lat Long', 'Latitude', 'Longitude', 'Churn Label',
    'Churn Score', 'CLTV', 'Churn Reason'
]
df = df.drop(columns=columns_to_drop)
```

```
df = pd.get_dummies(df, columns=['Contract', 'Payment Method', 'Internet Service'])
```

```
df['Total Charges'] = pd.to_numeric(df['Total Charges'], errors='coerce')
df = df.dropna()
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
numerical_cols = ['Tenure Months', 'Monthly Charges', 'Total Charges']
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
```

```
X = df.drop(columns=['Churn Value'])
y = df['Churn Value']

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_state=42, stratify=y
)
```

```
binary_columns = ['Gender', 'Senior Citizen', 'Partner', 'Dependents',
                  'Phone Service', 'Multiple Lines', 'Online Security',
                  'Online Backup', 'Device Protection', 'Tech Support',
                  'Streaming TV', 'Streaming Movies', 'Paperless Billing']

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for col in binary_columns:
    df[col] = le.fit_transform(df[col])
```

```
from sklearn.preprocessing import LabelEncoder

binary_columns = ['Gender', 'Senior Citizen', 'Partner', 'Dependents',
                  'Phone Service', 'Multiple Lines', 'Online Security',
                  'Online Backup', 'Device Protection', 'Tech Support',
                  'Streaming TV', 'Streaming Movies', 'Paperless Billing']

le = LabelEncoder()
for col in binary_columns:
    df[col] = le.fit_transform(df[col])
```

```
from sklearn.preprocessing import StandardScaler

num_cols = ['Tenure Months', 'Monthly Charges', 'Total Charges']
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])
```

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

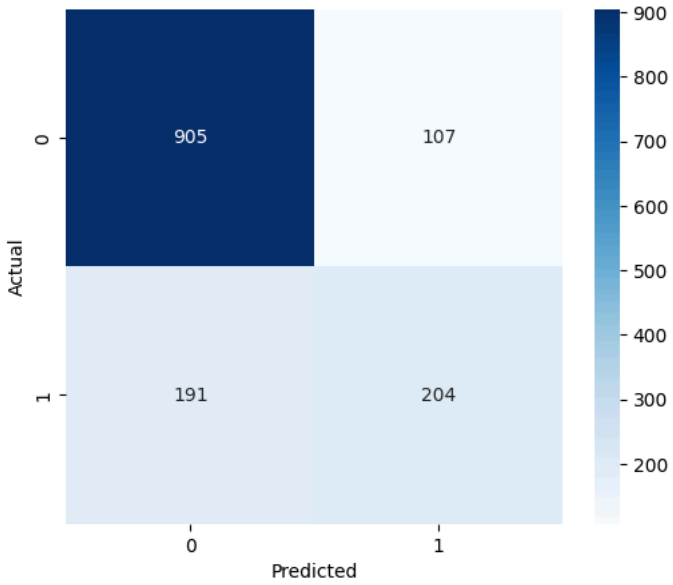
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.ensemble import RandomForestClassifier

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

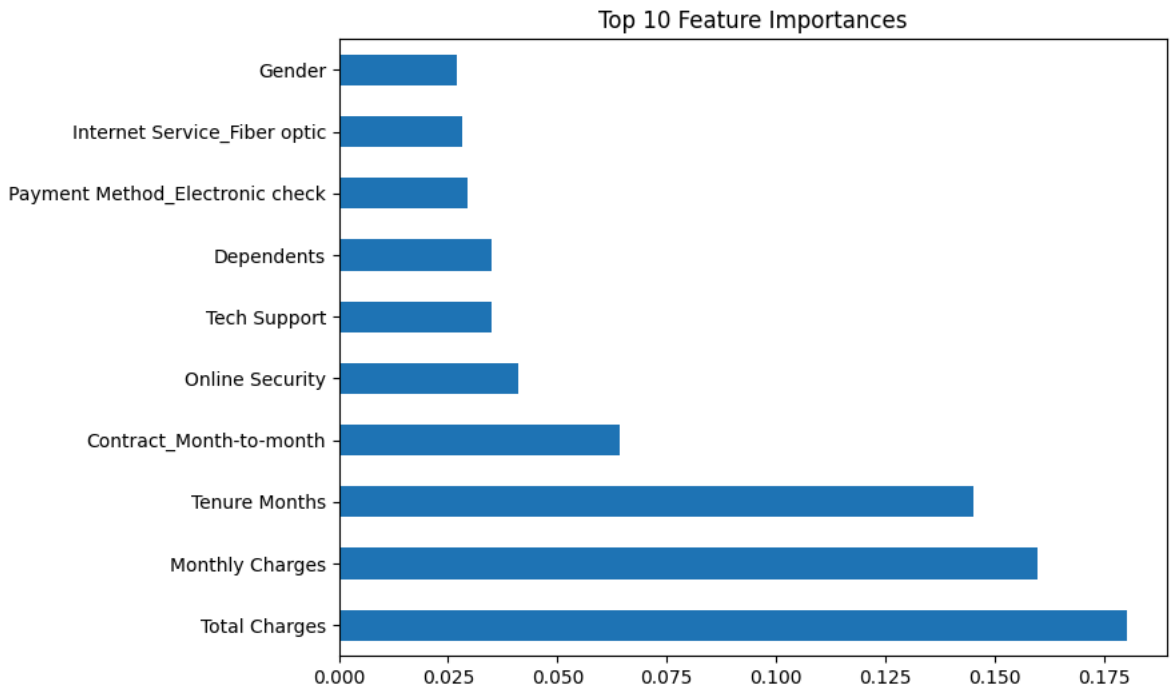
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

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

y_pred = rf_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Confusion Matrix Graph
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
Accuracy: 0.7882018479033405
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1012
           1       0.66      0.52      0.58       395

    accuracy                           0.79      1407
   macro avg       0.74      0.71      0.72      1407
weighted avg       0.78      0.79      0.78      1407
```



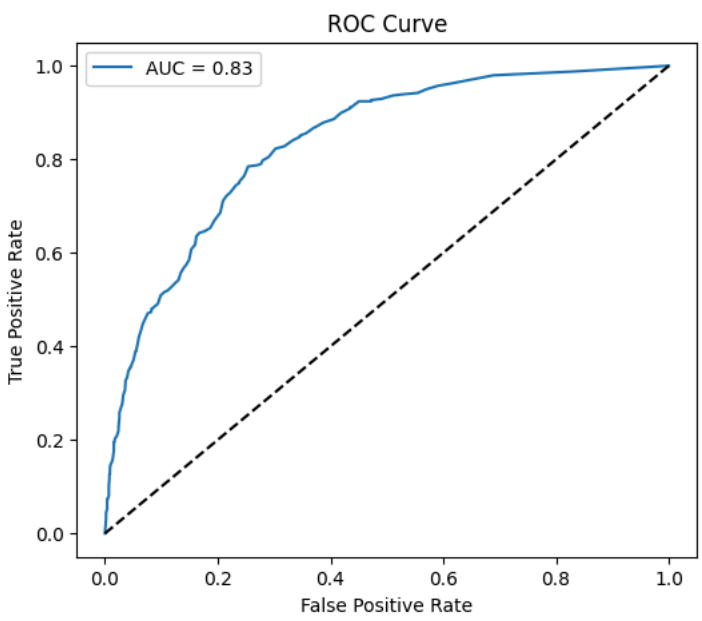

```
import pandas as pd

feat_importances = pd.Series(rf_model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh', figsize=(8,6))
plt.title("Top 10 Feature Importances")
plt.show()
```




```
from sklearn.model_selection import cross_val_score

cv_scores = cross_val_score(rf_model, X, y, cv=5)
print("5-Fold CV Accuracy Scores:", cv_scores)
print("Mean CV Accuracy:", cv_scores.mean())
```

```
5-Fold CV Accuracy Scores: [0.77967306 0.80170576 0.77311522 0.79302987 0.79445235]
Mean CV Accuracy: 0.7883952519459196
```

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

y_prob = rf_model.predict_proba(X_test)[:,1]  # probability of positive class
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc_score = roc_auc_score(y_test, y_prob)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'AUC = {auc_score:.2f}')
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```




```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=3, n_jobs=-1, verbose=1)
grid_search.fit(X_train, y_train)
print("Best Params:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
Best Params: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Best CV Score: 0.8049777777777778
```

```
import joblib
joblib.dump(shap_values, 'shap_values.pkl')
```

```
['shap_values.pkl']
```

```
import joblib
joblib.dump(rf_model, 'churn_model.pkl')
print("Model saved and workflow ended successfully!")
```

```
Model saved and workflow ended successfully!
```

```
import matplotlib.pyplot as plt

# Confusion matrix example
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
```

```
plt.savefig('confusion_matrix.png')  # save plot
plt.close()

print("All plots saved. Workflow finished.")
```

```
All plots saved. Workflow finished.
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.savefig("confusion_matrix.png")
plt.close()

print("Confusion Matrix saved as 'confusion_matrix.png'")
```

```
Confusion Matrix saved as 'confusion_matrix.png'
```

```
import pandas as pd

feat_importances.nlargest(10).plot(kind='barh', figsize=(8,6))
plt.title("Top 10 Feature Importances")
plt.savefig("feature_importance.png")
plt.close()

print("Feature Importance saved as 'feature_importance.png'")
```

```
Feature Importance saved as 'feature_importance.png'
```

```
from sklearn.metrics import roc_curve, roc_auc_score

fpr, tpr, _ = roc_curve(y_test, rf_model.predict_proba(X_test)[:,1])
auc_score = roc_auc_score(y_test, rf_model.predict_proba(X_test)[:,1])

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f"AUC = {auc_score:.2f}")
plt.plot([0,1],[0,1],'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.savefig("roc_curve.png")
plt.close()

print("ROC Curve saved as 'roc_curve.png'")
```

```
ROC Curve saved as 'roc_curve.png'
```