

```
# Import necessary libraries
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.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Load the dataset
df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")

# Data Preprocessing

# Drop customerID as it is not useful for prediction
df.drop("customerID", axis=1, inplace=True)

# Convert TotalCharges to numeric (some are spaces)
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors='coerce')

# Fill missing values with median
df["TotalCharges"].fillna(df["TotalCharges"].median(), inplace=True)

# Exploratory Data Analysis (EDA)

# Display churn distribution
sns.countplot(data=df, x="Churn")
plt.title("Churn Count")
plt.show()

# Check correlations
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

# Feature Engineering

# Encode categorical variables using LabelEncoder
le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    df[col] = le.fit_transform(df[col])

# Split features and target
X = df.drop("Churn", axis=1)
y = df["Churn"]

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Model Development

# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Train a Logistic Regression model
```

```
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)


# Evaluation

# Print evaluation metrics
print("Accuracy Score:", 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))

# Visualization

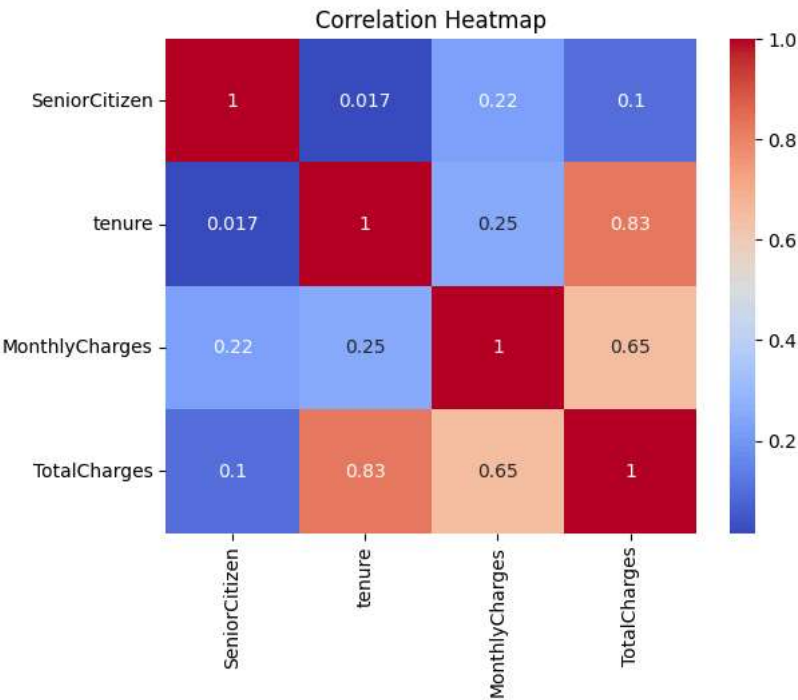
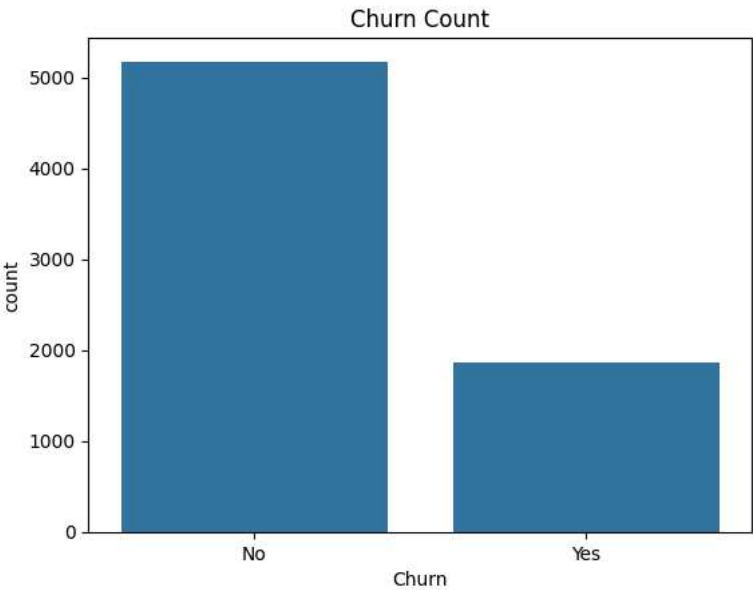
# Confusion matrix heatmap
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

df.head()
```

 <ipython-input-6-8a67a85996cb>:26: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as: The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[co:

```
df["TotalCharges"].fillna(df["TotalCharges"].median(), inplace=True)
```



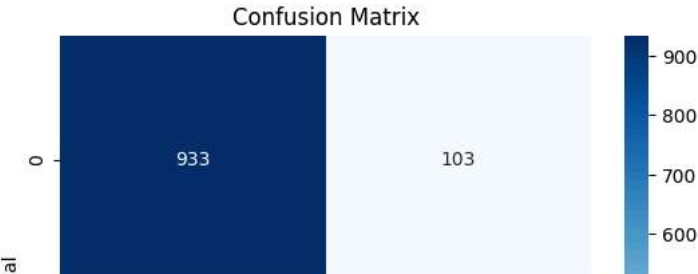
Accuracy Score: 0.815471965933286

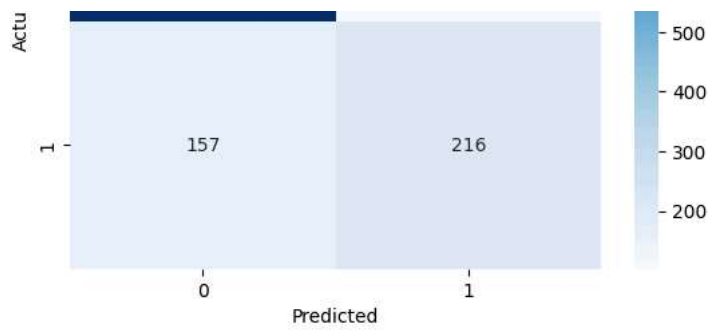
Confusion Matrix:

```
[[933 103]
 [157 216]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.86	0.90	0.88	1036
1	0.68	0.58	0.62	373
accuracy			0.82	1409
macro avg	0.77	0.74	0.75	1409
weighted avg	0.81	0.82	0.81	1409





	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	D
0	0	0	1	0	1	0	1	0	0	2	
1	1	0	0	0	34	1	0	0	2	0	
2	1	0	0	0	2	1	0	0	2	2	
3	1	0	0	0	45	0	1	0	2	0	
4	0	0	0	0	2	1	0	1	0	0	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
!pip install gradio==3.35.2
```

```
Collecting gradio==3.35.2
  Downloading gradio-3.35.2-py3-none-any.whl.metadata (15 kB)
Collecting aiofiles (from gradio==3.35.2)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (3.11.15)
Requirement already satisfied: altair>=4.2.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (5.5.0)
Collecting fastapi (from gradio==3.35.2)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpeg (from gradio==3.35.2)
  Downloading ffmpeg-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client>=0.2.7 (from gradio==3.35.2)
  Downloading gradio_client-1.10.0-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: httpx in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.14.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (0.31.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (3.1.6)
Requirement already satisfied: markdown-it-py>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py[linkify]>=2.0.0) (3.0.0)
Requirement already satisfied: markupsafe in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (3.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (3.10.0)
Collecting mdit-py-plugins<=0.3.3 (from gradio==3.35.2)
  Downloading mdit_py_plugins-0.3.3-py3-none-any.whl.metadata (2.8 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (2.0.2)
Requirement already satisfied: orjson in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (3.10.18)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (2.2.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (11.2.1)
Requirement already satisfied: pydantic in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (2.11.4)
Collecting pydub (from gradio==3.35.2)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: pygments>=2.12.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (2.19.1)
Collecting python-multipart (from gradio==3.35.2)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (2.32.3)
Collecting semantic-version (from gradio==3.35.2)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting uvicorn>=0.14.0 (from gradio==3.35.2)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: websockets>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio==3.35.2) (15.0.1)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/dist-packages (from altair>=4.2.0->gradio==3.35.2) (4.21.0)
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/dist-packages (from altair>=4.2.0->gradio==3.35.2) (1.42.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from altair>=4.2.0->gradio==3.35.2) (24.2)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from altair>=4.2.0->gradio==3.35.2) (4.12.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client>=0.2.7->gradio==3.35.2) (2025.4.26)
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (from httpx->gradio==3.35.2) (4.9.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx->gradio==3.35.2) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx->gradio==3.35.2) (1.0.9)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from httpx->gradio==3.35.2) (3.10)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx->gradio==3.35.2) (0.14.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.14.0->gradio==3.35.2) (3.16.1)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.14.0->gradio==3.35.2) (4.67.1)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.14.0->gradio==3.35.2) (1.20.4)
Requirement already satisfied: mdurl<=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.0.0->markdown-it-py[linkify]) (0.1.2)
Requirement already satisfied: linkify-it-py<3,>=1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py[linkify]>=2.0.0) (2.0.3)
INFO: pip is looking at multiple versions of mdit-py-plugins to determine which version is compatible with other requirements. This operation may have a few seconds of extra time.
Collecting mdit-py-plugins<=0.3.3 (from gradio==3.35.2)
  Downloading mdit_py_plugins-0.3.2-py3-none-any.whl.metadata (2.8 kB)
  Downloading mdit_py_plugins-0.3.1-py3-none-any.whl.metadata (2.8 kB)
```

Downloading mdit_py_plugins-0.3.0-py3-none-any.whl.metadata (2.8 kB)

```
import gradio as gr
```

```

/usr/local/lib/python3.11/dist-packages/gradio_client/documentation.py:106: UserWarning: Could not get documentation group for <class 'gradio_client.documentation.DocumentationGroup'>: {exc}
warnings.warn(f"Could not get documentation group for {cls}: {exc}")
/usr/local/lib/python3.11/dist-packages/gradio_client/documentation.py:106: UserWarning: Could not get documentation group for <class 'gradio_client.documentation.DocumentationGroup'>: {exc}
warnings.warn(f"Could not get documentation group for {cls}: {exc}")

```

```
def predict_churn(gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService, MultipleLines,
                  OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV,
                  StreamingMovies, Contract, PaperlessBilling, PaymentMethod, MonthlyCharges,
                  TotalCharges):
```

```
    # Create input array for the model
```

```
    input_data = [[gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService,
                    MultipleLines, OnlineSecurity, OnlineBackup, DeviceProtection,
                    TechSupport, StreamingTV, StreamingMovies, Contract,
                    PaperlessBilling, PaymentMethod, MonthlyCharges, TotalCharges]]
```

```
    # Scale input data using the existing scaler
```

```
    scaled_input = scaler.transform(input_data)
```

```
    # Make prediction using the trained model
```

```
    prediction = model.predict(scaled_input)[0]
```

```
    # Return prediction as a string ("Churn" or "No Churn")
```

```
    return "Churn" if prediction == 1 else "No Churn"
```

```
import gradio as gr
```

```
# Define input components for the interface
```

```
input_components = [
    gr.inputs Dropdown(choices=["Female", "Male"], default="Female", label="Gender"), # Pass the list of st
    gr.inputs Dropdown(choices=["No", "Yes"], default="No", label="Senior Citizen"),
    gr.inputs Dropdown(choices=["No", "Yes"], default="No", label="Partner"),
    gr.inputs Dropdown(choices=["No", "Yes"], default="No", label="Dependents"),
    gr.inputs Slider(minimum=1, maximum=72, step=1, default=1, label="Tenure"),
    gr.inputs Dropdown(choices=["No", "Yes"], default="No", label="Phone Service"),
    gr.inputs Dropdown(choices=["No phone service", "No", "Yes"], default="No phone service", label="Multi
    gr.inputs Dropdown(choices=["No internet service", "No", "Yes"], default="No internet service", label='
    gr.inputs Dropdown(choices=["No internet service", "No", "Yes"], default="No internet service", label='
    gr.inputs Dropdown(choices=["No internet service", "No", "Yes"], default="No internet service", label='
    gr.inputs Dropdown(choices=["No internet service", "No", "Yes"], default="No internet service", label='
    gr.inputs Dropdown(choices=["No internet service", "No", "Yes"], default="No internet service", label='
    gr.inputs Dropdown(choices=["No internet service", "No", "Yes"], default="No internet service", label='
    gr.inputs Dropdown(choices=["Month-to-month", "One year", "Two year"], default="Month-to-month", label=
    gr.inputs Dropdown(choices=["No", "Yes"], default="No", label="Paperless Billing"),
    gr.inputs Dropdown(choices=["Electronic check", "Mailed check", "Bank transfer (automatic)", "Credit ca
    gr.inputs Slider(minimum=18.25, maximum=118.75, step=0.01, default=18.25, label="Monthly Charges"),
    gr.inputs Slider(minimum=18.8, maximum=8684.8, step=0.01, default=18.8, label="Total Charges")
]
```

```
# Create the Gradio interface
```

```
iface = gr.Interface(
    fn=predict_churn,
    inputs=input_components,
    outputs="text",
    title="Customer Churn Prediction",
    description="Predict whether a customer will churn or not."
)
```

```
# Launch the interface
```

```
iface.launch()
```