1.Upload the Dataset

```
from google.colab import files
import pandas as pd

# Upload the dataset
uploaded = files.upload()

# Load the uploaded CSV file into a DataFrame
import io
df = pd.read_csv(io.BytesIO(uploaded[list(uploaded.keys())[0]]))

# Show first few rows
print("Dataset Preview:")
df.head()
```

```
Choose files student-dataset.csv
• student-dataset.csv(text/csv) - 56993 bytes, last modified: 08/05/2025 - 100% done
Saving student-dataset.csv to student-dataset.csv
Dataset Preview:
```

	school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;activ
0	
1	
2	
3	
4	

Next steps: Generate code with df | View recommended plots | New interactive sheet

2. Load the Dataset

```
import pandas as pd
import io
from google.colab import files

# Upload the dataset
uploaded = files.upload()

# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a DataFrame using the temporary path
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

# Display basic information about the dataset
print("Dataset Loaded Successfully!")
print("\nFirst 5 rows:")
print(df.head())

print("\nDataset Info:")
print(df.info())

print("\nSummary Statistics:")
print(df.describe())
```

```
sv
v) - 56993 bytes, last modified: 08/05/2025 - 100% done
 to student-dataset (1).csv
y!


amsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;activit
';4;4;"at_home";"teacher...
';1;1;"at_home";"other";...
';1;1;"at_home";"other";...
';4;2;"health";"services...
';3;3;"other";"other";"h...


 DataFrame'>
 to 394
mns):


s;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;acti




ess;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;ac
                                  395
                                  395
";"T";1;1;"other";"at_home";...
                                    1
```

3.Data Exploration

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files

# Upload the dataset
uploaded = files.upload()

# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a DataFrame using the temporary path
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

# Display basic information
print("First 5 rows of the dataset:")
print(df.head())

print("\nDataset Info:")
df.info()

print("\nSummary statistics:")
print(df.describe(include='all'))

# Check for missing values
print("\nMissing values per column:")
print(df.isnull().sum())

# Check class balance for churn
# ----> Check if 'churn' column exists (case-insensitive)
if 'churn' in df.columns.str.lower():
    churn_column = df.columns[df.columns.str.lower() == 'churn'][0]  # Get actual column name
    print(f"\n{churn_column} distribution:")
    print(df[churn_column].value_counts(normalize=True))
else:
    print("\n'Churn' column not found in the dataset. Please check the column name.")

# Visualize class distribution
# ----> If churn column exists, proceed with visualization
if 'churn' in df.columns.str.lower():
    sns.countplot(data=df, x=churn_column)
    plt.title(f'{churn_column} Distribution')
    plt.show()
```

```
else:
    print("\nSkipping visualization due to missing 'Churn' column.")

# Plot correlation heatmap for numerical features
# ----> Check if there are any numerical features before plotting the heatmap
numerical_df = df.select_dtypes(include='number')
if numerical_df.empty:
    print("\nNo numerical features found in the dataset. Skipping correlation heatmap.")
else:
    plt.figure(figsize=(12, 8))
    sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm')
    plt.title('Correlation Heatmap')
    plt.show()


# Categorical feature analysis vs churn
# ----> If churn column exists, proceed with categorical feature analysis
if 'churn' in df.columns.str.lower():
    categorical_features = df.select_dtypes(include='object').columns.drop(churn_column)

    for col in categorical_features:
        plt.figure(figsize=(8, 4))
        sns.countplot(data=df, x=col, hue=churn_column)
        plt.title(f'{col} vs {churn_column}')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()
else:
    print("\nSkipping categorical feature analysis due to missing 'Churn' column.")
```

```
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.
Saving student-dataset.csv to student-dataset (4).csv
First 5 rows of the dataset:
  school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;activit
0  GP;"F";18;"U";"GT3";"A";4;4;"at_home";"teacher...
1  GP;"F";17;"U";"GT3";"T";1;1;"at_home";"other";...
2  GP;"F";15;"U";"LE3";"T";1;1;"at_home";"other";...
3  GP;"F";15;"U";"GT3";"T";4;2;"health";"services...
4  GP;"F";16;"U";"GT3";"T";3;3;"other";"other";"h...

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 1 columns):
 #   Column
---  ------
 0   school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;acti
dtypes: object(1)
memory usage: 3.2+ KB

Summary statistics:
       school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;ac
count                                                 395
unique                                                395
top     MS;"M";19;"U";"LE3";"T";1;1;"other";"at_home";...
freq                                                    1

Missing values per column:
school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;activitie
dtype: int64

'Churn' column not found in the dataset. Please check the column name.

Skipping visualization due to missing 'Churn' column.

No numerical features found in the dataset. Skipping correlation heatmap.

Skipping categorical feature analysis due to missing 'Churn' column
```

4.Check for Missing Values and Duplicates

```
import pandas as pd
import io
from google.colab import files

# Upload the dataset
uploaded = files.upload()
```

```
# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a DataFrame using the temporary path
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

# Check for missing values
print("Missing values per column:\n")
missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])

# Percentage of missing values (optional)
print("\nPercentage of missing values per column:\n")
missing_percentage = (df.isnull().sum() / len(df)) * 100
print(missing_percentage[missing_percentage > 0])

# Check for duplicate rows
print("\nNumber of duplicate rows:")
duplicate_rows = df.duplicated().sum()
print(duplicate_rows)

# Show duplicate rows (optional)
if duplicate_rows > 0:
    print("\nExample duplicate rows:")
    print(df[df.duplicated()].head())
```

```
Choose files students_dataset.csv
```

- **students_dataset.csv**(text/csv) - 313 bytes, last modified: 24/04/2025 - 100% done

```
Saving students_dataset.csv to students_dataset.csv
Missing values per column:

Age           4
Marks         4
Attendance    4
dtype: int64

Percentage of missing values per column:

Age           25.0
Marks         25.0
Attendance    25.0
dtype: float64

Number of duplicate rows:
0
```

5.Visualize a Few Features

```
import pandas as pd
import io
from google.colab import files

# Upload the dataset
uploaded = files.upload()

# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a DataFrame using the temporary path
df = pd.read_csv(io.BytesIO(uploaded[file_name]))
import pandas as pd
import io
from google.colab import files

# Upload the dataset
uploaded = files.upload()

# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a DataFrame using the temporary path
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

# Check for missing values
print("Missing values per column:\n")
print(df.isnull().sum())
```

```
Choose files students_dataset.csv
• students_dataset.csv(text/csv) - 313 bytes, last modified: 24/04/2025 - 100% done
Saving students_dataset.csv to students_dataset (1).csv
Choose files students_dataset.csv
• students_dataset.csv(text/csv) - 313 bytes, last modified: 24/04/2025 - 100% done
Saving students_dataset.csv to students_dataset (2).csv
Missing values per column:

Name          0
Age           4
Marks         4
Attendance    4
Passed        0
dtype: int64
```

6. Identify Target and Features

```
import pandas as pd
import io
from google.colab import files

# Upload the dataset
uploaded = files.upload()

# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a DataFrame using the temporary path
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

# View column names to identify the target
print("Column names:\n", df.columns)

# ... (rest of your code)
```

```
Choose files students_dataset.csv
• students_dataset.csv(text/csv) - 313 bytes, last modified: 24/04/2025 - 100% done
Saving students_dataset.csv to students_dataset (3).csv
Column names:
 Index(['Name', 'Age', 'Marks', 'Attendance', 'Passed'], dtype='object')
```

7. Convert Categorical Columns to Numerical

```
import pandas as pd
import io
from google.colab import files
from sklearn.preprocessing import LabelEncoder

# Upload the dataset
uploaded = files.upload()

# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a
```

```
Choose files students_dataset.csv
• students_dataset.csv(text/csv) - 313 bytes, last modified: 24/04/2025 - 100% done
Saving students_dataset.csv to students_dataset (5).csv
```

8. One-Hot Encoding

```
import pandas as pd
import io
from google.colab import files

# Upload the dataset
uploaded = files.upload()

# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a DataFrame using the temporary path
```

```
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

# Drop customerID if present
if 'customerID' in df.columns:
    df
```

```
Choose files students_dataset.csv
```

- **students_dataset.csv**(text/csv) - 313 bytes, last modified: 24/04/2025 - 100% done

```
Saving students_dataset.csv to students_dataset (6).csv
```

9. Feature Scaling

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Print the column names to identify the correct target column
print(df.columns)

# Separate features and target
# Update the target variable name if necessary based on the printed columns
# The error is likely because 'churn' is not the correct column name.
# Check the output of df.columns and replace 'Churn' with the actual column name.
# target = 'Churn'  # Replace 'Churn' with the actual column name if different
# Based on the column names printed, the correct target column name is likely 'Attendence'
# Make sure the target column name matches exactly the name in your dataframe
# It is case sensitive, for example if the column is 'attendence' it should be target = 'attendence'
# The target column name might have a typo, let's assume it's 'Attendance' instead of 'Attendence'
target = 'Attendance'  # Replace with the actual target column name if different

# Check if the target column exists, if not, raise an error with suggestions
if target not in df.columns:
    possible_targets = [col for col in df.columns if target.lower() in col.lower()]
    suggestions = f", did you mean {possible_targets[0]}?" if possible_targets else ""
    raise KeyError(f"Target column '{target}' not found in the DataFrame. Please check the column name{suggestions}")

X = df.drop(columns=target)
y = df[target]
```

```
Index(['Name', 'Age', 'Marks', 'Attendance', 'Passed'], dtype='object')
```

10.Train-Test Split

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Instead of loading from 'churn_encoded.csv', use the existing df
# df = pd.read_csv('churn_encoded.csv')  # Remove this line

# Define the target and features
# Ensure the target variable
```

11.Model Building

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import LabelEncoder  # Import LabelEncoder

# ... (rest of your code)

# Before splitting into train and test sets:
# Convert categorical columns in X to numerical using Label Encoding
for col in X.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])

# ... (rest of your code - train_test_split, model fitting, etc.)
```

12. Evaluation

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import LabelEncoder  # Import LabelEncoder
from sklearn.impute import SimpleImputer # Import SimpleImputer

# ... (rest of your code)

# Before splitting into train and test sets:
# Convert categorical columns in X to numerical using Label Encoding
for col in X.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])

# Impute missing values with the mean (or other strategy)
imputer = SimpleImputer(strategy='mean') # You can choose other strategies like 'median', 'most_frequent'
X = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Assuming you want a 80/20 split

# Initialize and train the model
model = LogisticRegression(max_iter=1000)  # Increase max_iter if needed
model.fit(X_train, y_train)

# ... (rest of your code - model fitting, etc.)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

13.Make Predictions from New Input

```
# Initialize StandardScaler
scaler = StandardScaler()

# Store the column names before scaling (X_train is likely a DataFrame here)
# Convert X_train to a DataFrame if it's a NumPy array
if isinstance(X_train, np.ndarray):
    X_train = pd.DataFrame(X_train, columns=X.columns)  # Assuming X was the original DataFrame

# Now you can access the columns
X_train_cols = X_train.columns

# Fit the scaler and transform X_train
X_train_scaled = scaler.fit_transform(X_train)

# ... (rest of your code)
```

14. Convert to DataFrame and Encode

```
import pandas as pd

# Example new input data (replace these with actual input values)
new_input = {
    'gender': ['Female'],
    'SeniorCitizen': [0],
    'Partner': ['Yes'],
    'Dependents': ['No'],
    'tenure': [12],
    'PhoneService': ['Yes'],
    'MultipleLines': ['No'],
    'InternetService': ['Fiber optic'],
    'OnlineSecurity': ['No'],
    'OnlineBackup': ['Yes'],
    'DeviceProtection': ['No'],
    'TechSupport': ['No'],
    'StreamingTV': ['Yes'],
    'StreamingMovies': ['No'],
    'Contract': ['Month-to-month'],
    'PaperlessBilling': ['Yes'],
    'PaymentMethod': ['Electronic check'],
    'MonthlyCharges': [75.9],
```

```
    'TotalCharges': [890.5]
}

# Convert to DataFrame
new_df = pd.DataFrame(new_input)

# One-hot encode (must match training-time columns)
new_df_encoded = pd.get_dummies(new_df)

# Reindex to match training feature columns (X_train must be available)
new_df_encoded = new_df_encoded.reindex(columns=X_train.columns, fill_value=0)

print("Encoded new customer data:")
print(new_df_encoded.head())
```

```
Encoded new customer data:
   Name  Age  Marks  Passed
0     0    0      0       0
```

15.Predict the Final Grade

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from google.colab import files
import io
from sklearn.impute import SimpleImputer # Import SimpleImputer


# Upload the dataset
uploaded = files.upload()

# Get the filename from the uploaded dictionary
file_name = list(uploaded.keys())[0]

# Load the uploaded CSV file into a DataFrame
```

```
Choose files students_dataset.csv
• students_dataset.csv(text/csv) - 313 bytes, last modified: 24/04/2025 - 100% done
Saving students_dataset.csv to students_dataset (12).csv
```

16.Deployment-Building an Interactive App

```
pip install streamlit pandas scikit-learn joblib
```

```
Collecting streamlit
  Downloading streamlit-1.45.0-py3-none-any.whl.metadata (8.9 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (1.4.2)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: blinker<2,>=1.5.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (5.5.2)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (8.1.8)
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.11/dist-packages (from streamlit) (2.0.2)
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.11/dist-packages (from streamlit) (24.2)
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (11.2.1)
Requirement already satisfied: protobuf<7,>=3.20 in /usr/local/lib/python3.11/dist-packages (from streamlit) (5.29.4)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (18.1.0)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.11/dist-packages (from streamlit) (2.32.3)
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (9.1.2)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.11/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/lib/python3.11/dist-packages (from streamlit) (4.13.2)
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl.metadata (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 1.6 MB/s eta 0:00:00
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.11/dist-packages (from streamlit) (3.1.44)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.11/dist-packages (from streamlit) (6.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.2)
```

```
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from altair<6,>=4.0->streamlit) (3.1.6)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/dist-packages (from altair<6,>=4.0->streamlit) (4.23.0)
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/dist-packages (from altair<6,>=4.0->streamlit) (1.37.1)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11/dist-packages (from gitpython!=3.1.19,<4,>=3.0.7->streamlit)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.27->streamlit) (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.27->streamlit) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.27->streamlit) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.27->streamlit) (2025.4
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/dist-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->altair<6,>=4.0->streamlit) (3.0.
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->al
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->str
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair<6,>=4.0->streamli
Downloading streamlit-1.45.0-py3-none-any.whl (9.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.9/9.9 MB 42.6 MB/s eta 0:00:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 37.4 MB/s eta 0:00:00
Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl (79 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.1/79.1 kB 4.6 MB/s eta 0:00:00
Installing collected packages: watchdog, pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.45.0 watchdog-6.0.0
```

```
import joblib
joblib.dump(model, "churn_model.pkl")
joblib.dump(scaler, "scaler.pkl")
```

```
['scaler.pkl']
```

```
import streamlit as st
import pandas as pd
import numpy as np
import joblib  # for loading saved model and scaler

# Load the model and scaler
model = joblib.load("churn_model.pkl")
scaler = joblib.load("scaler.pkl")

# ... (rest of your code)

# Encode input to match training features
input_encoded = pd.get_dummies(input_data)
model_features = model.feature_names_in_ if hasattr(model, 'feature_names_in_') else scaler.feature_names_in_

# Get missing columns in input_encoded compared to model_features
missing_cols = set(model_features) - set(input_encoded.columns)

# Add missing columns with 0 values to input_encoded
for col in missing_cols:
    input_encoded[col] = 0

# Ensure all columns are present and in the correct order
input_encoded = input_encoded.reindex(columns=model_features, fill_value=0)

#
```

17.Create a Prediction Function

```
import pandas as pd

def predict_churn(input_data, model, scaler, feature_columns):
    """
    Predicts churn from input customer data.

    Parameters:
    - input_data (dict): Raw customer input as a dictionary.
    - model (sklearn classifier): Trained model.
    - scaler (sklearn scaler): Fitted scaler object.
    - feature_columns (list): Columns used during model training.

    Returns:
    - prediction (int): 1 = churn, 0 = no churn
```

```
    - probability (float): Probability of churn
    """

    # Convert input to DataFrame
    input_df = pd.DataFrame([input_data])

    # One-hot encode categorical features
    input_encoded = pd.get_dummies(input_df)

    # Align encoded input with training feature columns
    input_encoded = input_encoded.reindex(columns=feature_columns, fill_value=0)

    # Scale the input
    input_scaled = scaler.transform(input_encoded)

    # Make prediction
    prediction = model.predict(input_scaled)[0]
    probability = model.predict_proba(input_scaled)[0][1]

    return prediction, probability
```

17.Create the Gradio Interface

```
!pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.29.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.10.0 (from gradio)
  Downloading gradio_client-1.10.0-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.30.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.18)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.2.1)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.4)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.9-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.3)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradio)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0.16
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.67.1
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2.9
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
```

```
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (
```

```
import gradio as gr
import pandas as pd
import joblib

# Load model, scaler, and feature columns
model = joblib.load("churn_model.pkl")
scaler = joblib.load("scaler.pkl")
feature_columns = model.feature_names_in_

# Prediction function
def predict_churn_gradio(
    gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService,
    MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection,
    TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling,
    PaymentMethod, MonthlyCharges, TotalCharges
):
    input_dict = {
        'gender': gender,
        'SeniorCitizen': SeniorCitizen,
        'Partner': Partner,
        'Dependents': Dependents,
        'tenure': tenure,
        'PhoneService': PhoneService,
        'MultipleLines': MultipleLines,
        'InternetService': InternetService,
        'OnlineSecurity': OnlineSecurity,
        'OnlineBackup': OnlineBackup,
        'DeviceProtection': DeviceProtection,
        'TechSupport': TechSupport,
        'StreamingTV': StreamingTV,
        'StreamingMovies': StreamingMovies,
        'Contract': Contract,
        'PaperlessBilling': PaperlessBilling,
        'PaymentMethod': PaymentMethod,
        'MonthlyCharges': MonthlyCharges,
        'TotalCharges': TotalCharges
    }

    df = pd.DataFrame([input_dict])
    df_encoded = pd.get_dummies(df)
    df_encoded = df_encoded.reindex(columns=feature_columns, fill_value=0)
    df_scaled = scaler.transform(df_encoded)
    prediction = model.predict(df_scaled)[0]
    probability = model.predict_proba(df_scaled)[0][1]

    label = "Churn" if prediction == 1 else "No Churn"
    return f"Prediction: {label} | Probability: {probability:.2f}"

# Define Gradio interface
gr_interface = gr.Interface(
    fn=predict_churn_gradio,
    inputs=[
        gr.Dropdown(["Male", "Female"], label="Gender"),
        gr.Radio([0, 1], label="Senior Citizen"),
        gr.Radio(["Yes", "No"], label="Partner"),
        gr.Radio(["Yes", "No"], label="Dependents"),
        gr.Slider(0, 72, step=1, label="Tenure (months)"),
        gr.Radio(["Yes", "No"], label="Phone Service"),
        gr.Radio(["Yes", "No", "No phone service"], label="Multiple Lines"),
        gr.Dropdown(["DSL", "Fiber optic", "No"], label="Internet Service"),
        gr.Radio(["Yes", "No", "No internet service"], label="Online Security"),
        gr.Radio(["Yes", "No", "No internet service"], label="Online Backup"),
        gr.Radio(["Yes", "No", "No internet service"], label="Device Protection"),
        gr.Radio(["Yes", "No", "No internet service"], label="Tech Support"),
        gr.Radio(["Yes", "No", "No internet service"], label="Streaming TV"),
        gr.Radio(["Yes", "No", "No internet service"], label="Streaming Movies"),
        gr.Dropdown(["Month-to-month", "One year", "Two year"], label="Contract"),
        gr.Radio(["Yes", "No"], label="Paperless Billing"),
        gr.Dropdown([
            "Electronic check", "Mailed check", "Bank transfer (automatic)", "Credit card (automatic)"
        ], label="Payment Method"),
        gr.Number(label="Monthly Charges"),
        gr.Number(label="Total Charges")
```

```
        ],
        outputs="text",
        title="Customer Churn Prediction",
        description="Enter customer details to predict if they are likely to churn."
    )

    # Launch the app
    gr_interface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatically

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://600a6186de2146fcf8.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working dir
```