PREDICTING CUSTOMER CHURN USING MACHINE LEARINGTO UNCOVER HIDDEN PATTERN

program

```
# 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

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve

from imblearn.over_sampling import SMOTE

import gradio as gr

# 2. Load Dataset

df = pd.read_csv('customer_churn.csv')  # Replace with your actual file

# 3. Data Preprocessing

df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

df = df.dropna()

# Encode categorical features

df = pd.get_dummies(df, drop_first=True)

# 4. Split Features & Target

X = df.drop('Churn_Yes', axis=1)

y = df['Churn_Yes']

# 5. Train-Test Split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# 6. Handle Imbalance with SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
# 7. Feature Scaling
scaler = StandardScaler()
X_train_resampled = scaler.fit_transform(X_train_resampled)
X_test = scaler.transform(X_test)
# 8. Model Training (Random Forest)
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_resampled, y_train_resampled)
# 9. Evaluation
y_pred = rf_model.predict(X_test)
y_proba = rf_model.predict_proba(X_test)[:, 1]
print("Classification Report:\n", classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, y_proba))
# 10. Gradio Interface Function
def predict_churn(tenure, monthly_charges, total_charges, support_calls,
contract_type, internet_service):
  # Prepare input as dataframe
  input_data = {
    'tenure': [tenure],
    'MonthlyCharges': [monthly_charges],
    'TotalCharges': [total_charges],
    'SupportCalls': [support_calls],
    'Contract_One year': [1 if contract_type == 'One year' else 0],
    'Contract_Two year': [1 if contract_type == 'Two year' else 0],
    'InternetService_Fiber optic': [1 if internet_service == 'Fiber optic' else 0],
```

```
        'InternetService_No': [1 if internet_service == 'No' else 0]
    }
    input_df = pd.DataFrame(input_data)
    input_scaled = scaler.transform(input_df)
    proba = rf_model.predict_proba(input_scaled)[0][1]
    result = "Likely to Churn" if proba > 0.5 else "Not Likely to Churn"
        return f"Churn Probability: {proba * 100:.2f}%\nPrediction: {result}"
# 11. Deploy with Gradio
iface = gr.Interface(
    fn=predict_churn,
    inputs=[
        gr.Number(label="Tenure (months)"),
        gr.Number(label="Monthly Charges"),
        gr.Number(label="Total Charges"),
        gr.Number(label="Number of Support Calls"),
        gr.Radio(["Month-to-month", "One year", "Two year"], label="Contract Type"),
        gr.Radio(["DSL", "Fiber optic", "No"], label="Internet Service")
    ],
    outputs="text",
    title="Customer Churn Predictor",
    description="Predicts the probability of customer churn based on input details."
)

iface.launch()
```

Output

📊 Classification Report:

	precision	recall	f1-score	support
0	0.85	0.95	0.90	1033
1	0.78	0.52	0.62	374
accuracy			0.84	1407
macro avg	0.82	0.74	0.76	1407
weighted avg	0.83	0.84	0.83	1407