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

import pandas as pd
df=pd.read_csv("Telco-Customer-Churn.csv")
df.head()

#Data Exploration
df.info()
df.describe()
 df.columns
df.shape

#checking missing value and duplicates
print(df.isnull().sum())
print(f"Duplicted Rows :{df.duplicated().sum()}")

#visualize features
 import seaborn as sns
 import matplotlib.pyplot as plt
sns.countplot(x='Churn', data=df)
 plt.title('Distribution of Churn')
```

```
 plt.show()
sns.histplot(df['tenure'],kde=True)
plt.title('Distribution of Tenure')
 plt.show()
sns.boxplot(x='Churn', y='MonthlyCharges', data=df)
plt.title('Monthly Charges vs Churn')
plt.show()


#identifing target and features
target='Churn'
 features=df.drop(columns=[target]).columns.tolist()
#convert catgo to numeric
df['Totalcharges']=pd.to_numeric(df['TotalCharges'],errors='c
oerce') df.dropna(inplace=True)
#one-hot encode
df_encoded=pd.get_dummies(df,drop_first=True)
#featue scaling
 from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaled_features=scaler.fit_transform(df_encoded.drop(colu
mns=['Churn_Yes']))
X=pd.DataFrame(scaled_features,columns=df_encoded.drop(
columns=['Churn_Yes']).columns) y=df_encoded['Churn_Yes']
#train-test split
```

```
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)

df['Churn'].value_counts()

# Apply SMOTE to balance the dataset

from imblearn.over_sampling import SMOTE smote =
SMOTE(random_state=42) X_resampled, y_resampled =
smote.fit_resample(X_train, y_train)

#model building

from sklearn.ensemble import RandomForestClassifier from
sklearn.linear_model import LogisticRegression from
sklearn.metrics import accuracy_score

# Random Forest (balanced)

 rf_model = RandomForestClassifier(class_weight='balanced')
rf_model.fit(X_resampled, y_resampled)

# Logistic Regression (balanced)

 lr_model = LogisticRegression(class_weight='balanced',
max_iter=1000) lr_model.fit(X_resampled, y_resampled)


#evalution

 from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score


# Random Forest Evaluation
```

```
print("Random Forest Results:")
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)
 print(confusion_matrix(y_test, rf_preds))
 print(classification_report(y_test, rf_preds))
 print("RF Accuracy:", accuracy_score(y_test, rf_preds))
# Logistic Regression Evaluation
print("\nLogistic Regression Results:")
lr_model.fit(X_train, y_train)
 lr_preds = lr_model.predict(X_test)
 print(confusion_matrix(y_test, lr_preds))
print(classification_report(y_test, lr_preds))
 print("LR Accuracy:", accuracy_score(y_test,lr_preds))


print(f"After SMOTE - Churn Distribution:
{y_resampled.value_counts()}")


# Compare accuracies and choose the better model
 if accuracy_score(y_test, rf_preds) > accuracy_score(y_test,
lr_preds):
  model = rf_model
 print("Selected RF model")
 else:
```

```
  model = lr_model
 print("Selected LR model")


import joblib
joblib.dump(model, 'churn_prediction_model.pkl')
joblib.dump(X.columns.tolist(), 'columns.pkl')


!pip install gradio pandas joblib --quiet
import gradio as gr
 import pandas as pd import joblib
model = joblib.load('churn_prediction_model.pkl') columns =
joblib.load('columns.pkl')
def predict_churn(gender, senior_citizen, partner,
dependents, tenure, monthly, total, phone_service,
multiple_lines, internet_service):    input_data = {
  'gender': gender,
  'SeniorCitizen': senior_citizen,
 'partner': partner,
 'dependents': dependents,
   'tenure': tenure,
  'MonthlyCharges': monthly,
 'TotalCharges': total,
    'PhoneService': phone_service,
```

```
 'MultipleLines': multiple_lines,
     'InternetService': internet_service
 }
   df_input = pd.DataFrame([input_data])
  df_encoded =
pd.get_dummies(df_input).reindex(columns=columns,
fill_value=0)    prediction = model.predict(df_encoded)
   return "Churn" if prediction[0] == 1 else "No Churn"
# Create Gradio interface
 iface = gr.Interface  (
  fn=predict_churn,
   inputs=[
   gr.Dropdown(['Female', 'Male'], label="Gender"),
gr.Dropdown(['No', 'Yes'], label="Senior Citizen"),
gr.Dropdown(['No', 'Yes'], label="Partner"),
    gr.Dropdown(['No', 'Yes'], label="Dependents"),
  gr.Slider(0, 72, label="Tenure (months)"),
gr.Number(minimum=0.0, label="Monthly Charges"),
gr.Number(minimum=0.0, label="Total Charges"),
gr.Dropdown(['No', 'Yes'], label="Phone Service"),
gr.Dropdown(['No phone service', 'No', 'Yes'], label="Multiple
Lines"),       gr.Dropdown(['DSL', 'Fiber optic', 'No'],
label="Internet Service")    ],    outputs="text",
title="Customer Churn Prediction",
 description="Enter customer details to predict churn." )
```

```
# Launch the interface and share it publicly
iface.launch(share=True)
```