

# GRADIO APP FOR SALES PREDICTION FOR CUSTOMER CHURN



## INTRODUCTION

Churn prediction is a crucial task for businesses that depend on recurring revenue, like subscription-based services and membership organizations. It involves identifying customers who are likely to churn, allowing companies to take action to retain them and avoid revenue loss. In this article, I will explore how to develop and deploy a

churn prediction app using Gradio. This platform enables the easy creation and sharing of interactive web apps with machine learning models.

## **About Gradio**

Gradio is similar to Streamlit in that it's a free and open-source Python library that enables the development of customizable and user-friendly demos for machine learning models. It can be used with Python scripts or Jupyter Notebooks.

## **Process**

### **Workflow overview**

As indicated earlier, the workflow may be summarized as follows:

- Export ML items
- Set up environment
- Import ML items
- Build interface
- Write the function to process inputs and display outputs
- Deploy the app

## **Setting up an environment to build my app in**

To establish an environment for creating your app, it is advisable to employ a virtual environment to manage dependencies and guarantee that your project's packages do not clash with other globally installed packages on your system.

To generate and activate a virtual environment, follow these steps:

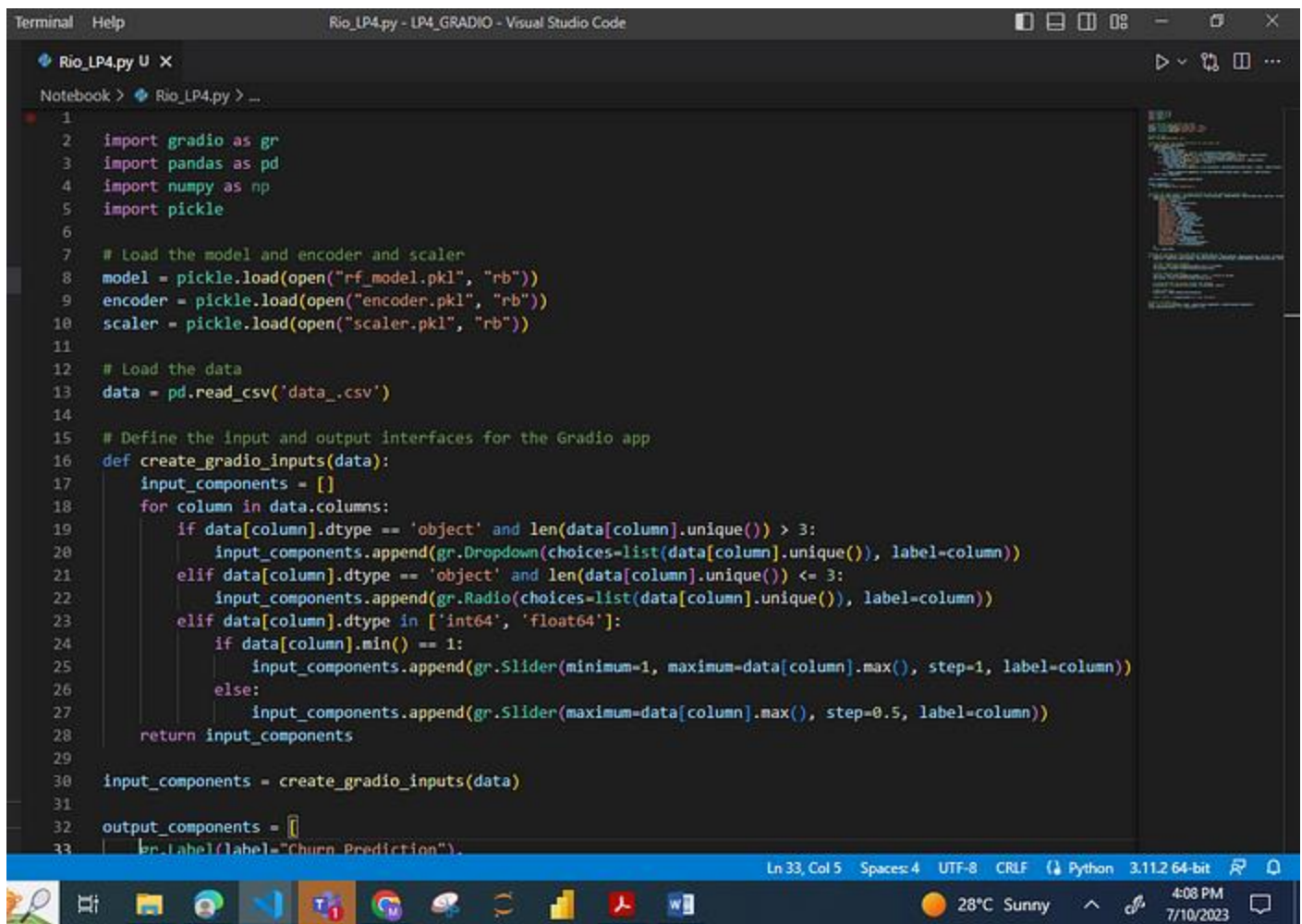
- *Navigate to your terminal.*
- *Move to your project directory.*
- *Execute the following command to create the virtual environment:* `python -m venv venv.`

## Toolkit Export

To export key components used during the modeling process from your notebook, you typically start by exporting the encoder, scaler, and model. These components can be combined into a dictionary and exported for easy access. In this scenario, the exports will be performed using Pickle, so the library must first be imported.

```
# Import Pickle
import pickle
```

## Load the Model and Encoder and Scaler



```
1
2 import gradio as gr
3 import pandas as pd
4 import numpy as np
5 import pickle
6
7 # Load the model and encoder and scaler
8 model = pickle.load(open("rf_model.pkl", "rb"))
9 encoder = pickle.load(open("encoder.pkl", "rb"))
10 scaler = pickle.load(open("scaler.pkl", "rb"))
11
12 # Load the data
13 data = pd.read_csv('data.csv')
14
15 # Define the input and output interfaces for the Gradio app
16 def create_gradio_inputs(data):
17     input_components = []
18     for column in data.columns:
19         if data[column].dtype == 'object' and len(data[column].unique()) > 3:
20             input_components.append(gr.Dropdown(choices=list(data[column].unique()), label=column))
21         elif data[column].dtype == 'object' and len(data[column].unique()) <= 3:
22             input_components.append(gr.Radio(choices=list(data[column].unique()), label=column))
23         elif data[column].dtype in ['int64', 'float64']:
24             if data[column].min() == 1:
25                 input_components.append(gr.Slider(minimum=1, maximum=data[column].max(), step=1, label=column))
26             else:
27                 input_components.append(gr.Slider(maximum=data[column].max(), step=0.5, label=column))
28     return input_components
29
30 input_components = create_gradio_inputs(data)
31
32 output_components = [
33     gr.Label(label="Churn Prediction")]
```

codes for packages used, model, encoder, and scalar

Our code utilizes Gradio, a web-based platform for developing and sharing custom machine learning interfaces, to create an app that predicts customer churn using a random forest model. The model is trained to forecast whether a customer is likely to churn based on various features. To build the app, we first load the machine learning model, encoder, and scaler from their respective pickle files. Pickling allows for the serialization of Python objects, enabling them to be saved to disk and loaded at a later time.

With the model, encoder, and scaler loaded, we create a Gradio interface that enables users to input data and obtain predictions from the model. The interface can be customized with various input and output types, labels, and descriptions. In this particular case, the input is a CSV file containing data on different features like customer age, tenure, and monthly charges, while the output is a CSV file containing the predicted values. Gradio handles the data preprocessing and post-processing automatically, eliminating any concerns for the user.

The app serves as a superb example of how machine learning can be made accessible to a broader audience. By providing a user-friendly interface for forecasting customer churn, businesses can gain a better understanding of their customer relationships, resulting in increased customer satisfaction and retention.

## **Input and output interfaces for the Gradio app**

In the preceding code snippet, we completed the loading of the machine-learning model, encoder, and scaler. We will now proceed to develop the input and output interfaces for the Gradio application.

```

14
15 # Define the input and output interfaces for the Gradio app
16 def create_gradio_inputs(data):
17     input_components = []
18     for column in data.columns:
19         if data[column].dtype == 'object' and len(data[column].unique()) > 3:
20             input_components.append(gr.Dropdown(choices=list(data[column].unique()), label=column))
21         elif data[column].dtype == 'object' and len(data[column].unique()) <= 3:
22             input_components.append(gr.Radio(choices=list(data[column].unique()), label=column))
23         elif data[column].dtype in ['int64', 'float64']:
24             if data[column].min() == 1:
25                 input_components.append(gr.Slider(minimum=1, maximum=data[column].max(), step=1, label=column))
26             else:
27                 input_components.append(gr.Slider(maximum=data[column].max(), step=0.5, label=column))
28     return input_components
29
30 input_components = create_gradio_inputs(data)
31
32 output_components = [
33     gr.Label(label="Churn Prediction"),
34 ]
35
36 # Convert the input values to a pandas DataFrame with the appropriate column names
37 def input_df_creator(gender, DeviceProtection, MonthlyCharges, PaymentMethod, PaperlessBilling, Contract, StreamingM
38     input_data = pd.DataFrame({
39         "gender": [gender],
40         "SeniorCitizen": [SeniorCitizen],
41         "Partner": [Partner],
42         "Dependents": [Dependents],
43         "tenure": [int(tenure)],
44         "PhoneService": [PhoneService],
45         "MultipleLines": [MultipleLines],
46         "InternetService": [InternetService],

```

Input, output data, and columns used for pandas data frame

## Convert the input values to a pandas DataFrame with the appropriate column names

The code below defines a function that is responsible for converting the input values from the Gradio interface into a Pandas DataFrame with appropriate column names. This process is crucial for preparing input data for the machine learning model to make predictions.

```

import pandas as pd

def preprocess_input(input_dict):
    # Convert the input values to a Pandas DataFrame
    input_df = pd.DataFrame({
        'gender': [input_dict['gender']],
        'senior_citizen': [input_dict['senior_citizen']],
        'partner': [input_dict['partner']],

```

```

    'dependents': [input_dict['dependents']],
    'tenure': [input_dict['tenure']],
    'phone_service': [input_dict['phone_service']],
    'multiple_lines': [input_dict['multiple_lines']],
    'internet_service': [input_dict['internet_service']],
    'online_security': [input_dict['online_security']],
    'online_backup': [input_dict['online_backup']],
    'device_protection': [input_dict['device_protection']],
    'tech_support': [input_dict['tech_support']],
    'streaming_tv': [input_dict['streaming_tv']],
    'streaming_movies': [input_dict['streaming_movies']],
    'contract': [input_dict['contract']],
    'paperless_billing': [input_dict['paperless_billing']],
    'payment_method': [input_dict['payment_method']],
    'monthly_charges': [input_dict['monthly_charges']],
    'total_charges': [input_dict['total_charges']]
    })

    return input_df

```

The aim of the function mentioned earlier is to validate input data and make sure that the column names match the required input of the machine learning model. It does so by creating a dictionary with keys as column names and values as input data and then converting it into a Pandas DataFrame. This step is essential for the model to generate accurate predictions.

In the next code segment, all the elements that we have built so far will be merged to develop a full-fledged Gradio application for predicting customer churn. The users can input customer information, get churn probability predictions, and visualize the results in an easily understandable format.

## Launch the Gradio app



We have successfully developed a customer churn prediction application by utilizing the Gradio framework and machine learning in a concise code snippet. This app enables businesses to promptly and conveniently input customer information and receive churn probability predictions. By utilizing this app, businesses can take preemptive measures to prevent churn and retain customers, which can result in revenue growth and expansion.

```
C:\Users\KOFI\Downloads> tc3.py > ...
52     "StreamingMovies": [StreamingMovies],
53     "Contract": [Contract],
54     "PaperlessBilling": [PaperlessBilling],
55     "PaymentMethod": [PaymentMethod],
56     "MonthlyCharges": [float(MonthlyCharges)],
57     "TotalCharges": [float(TotalCharges)],
58     })
59     return input_data
60
61 # Define the function to be called when the Gradio app is run
62 def predict_churn(gender, DeviceProtection, MonthlyCharges, PaymentMethod, PaperlessBilling, Contract, StreamingMovi
63     input_df = input_df_creator(gender, DeviceProtection, MonthlyCharges, PaymentMethod, PaperlessBilling, Contract,
64
65     # Encode categorical variables
66     cat_cols = data.select_dtypes(include=['object']).columns
67     cat_encoded = encoder.transform(input_df[cat_cols])
68
69     # Scale numerical variables
70     num_cols = data.select_dtypes(include=['int64', 'float64']).columns
71     num_scaled = scaler.transform(input_df[num_cols])
72
73     # joining encoded and scaled columns back together
74     processed_df = pd.concat([num_scaled, cat_encoded], axis=1)
75
76     # Make prediction
77     prediction = model.predict(processed_df)
78
79     return "Churn" if prediction[0] == 1 else "No Churn"
80
81 # Launch the Gradio app
82 iface = gr.Interface(predict_churn, inputs=input_components, outputs=output_components)
83 iface.launch(inbrowser=True, show_error=True)
```

codes for pre-process and predicting sales

In addition to its functionality, this app can be easily customized to suit the individual needs of any business. By adjusting the input components or fine-tuning the machine learning model, businesses



can create a churn prediction application that is customized to their particular industry and customer base.

## Gradio Customer Churn Prediction Web App

This is App is a deployment of Customer Churn prediction Machine Learning model built with random forest

Gender

Female

Senior Citizen

Yes

No

Partner

Yes

No

Dependents

Yes

No

Tenure (months)

1

Phone Service

Yes

No

Multiple Lines

No

Internet Service

Fiber optic

Online Security

No

Online Backup

No

Device Protection

No

Tech Support

No

TV Streaming

No

Movie Streaming

No

Contract

Month-to-month

Paperless Billing

Yes

No

Payment Method

Electronic check

Monthly Charges

0

Total Charges

0

output

Submit forms to view prediction...

Flag

This project highlights the potential of blending machine learning with user-friendly app-building platforms like Gradio. Through this fusion, businesses can leverage the latest advancements in AI to make informed decisions based on data analysis and remain ahead of their competitors.

