**Marrakech Telecom Customer Churn Prediction App with Python and Gradio**

****

Customer churn is a major problem for many businesses, especially in the telecom industry, where customers can easily switch to other providers. Customer churn can lead to loss of revenue, market share, and customer loyalty. Therefore, it is important for telecom companies to identify the customers who are likely to churn and take proactive actions to retain them.

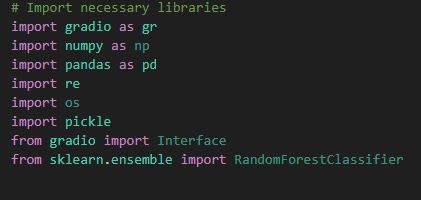
One way to do this is to use machine learning to build a predictive model that can analyze the customer data and output the probability of churn for each customer. This can help the telecom company to segment the customers based on their churn risk and design targeted marketing campaigns or offer incentives to reduce the churn rate.

In this article, we shall explain how to build a telecom customer churn prediction app with Python and Gradio. Gradio is a library that allows you to create interactive web interfaces for your machine learning models with just a few lines of code. You can use Gradio to easily test and debug your models, or share them with others through a link or an embeddable widget.

We will use a random forest classifier to train our model and then use Gradio to create a user-friendly app that can take user inputs and output the predicted churn status of the customer.

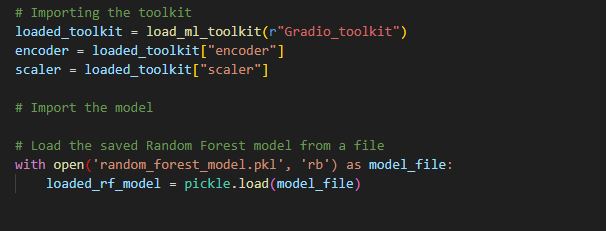
**Step 1: Import necessary libraries**

First, we imported some libraries that we will use for our project. We will use Numpy and Pandas for data manipulation, re and os for regular expressions and file operations, pickle for saving and loading objects, sklearn for machine learning, and gradio for creating the app interface

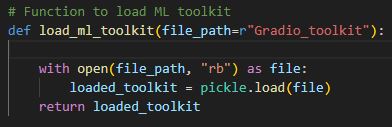


**Step 2: Load the machine learning toolkit**

Next, we loaded the machine learning toolkit that contains some preprocessed items that we will use for our model.



* An encoder that transforms the categorical features into numeric values using one-hot encoding.
* A scaler that scales the numeric features into a range between 0 and 1 using min-max scaling.
* A trained random forest model that predicts the churn probability based on the input features.
* We have saved these items in a pickle file called “Gradio\_toolkit” that we will load into our script.



We then call the function and assign the loaded items to their respective variables and also loaded the random forest model from another pickle file called “random\_forest\_model.pkl”

**Step 3: Define a function to process and predict**

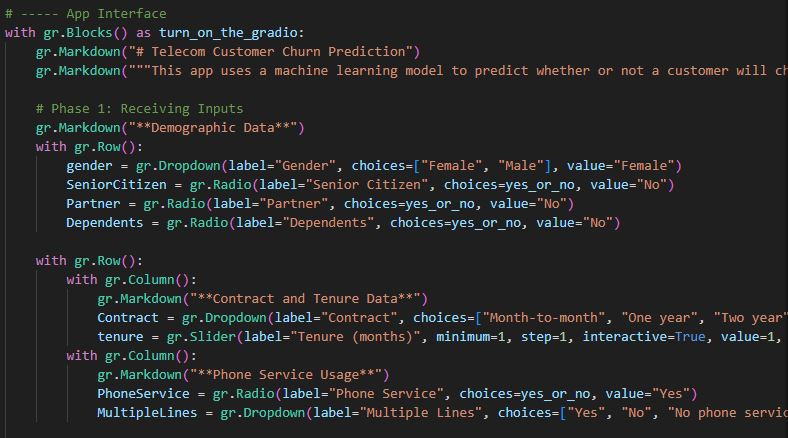
Now after loaded our machine learning toolkit, we defined a function that can takes user inputs and return the predicted churn status of the customer. We followed this steps:-

* The inputs is Converted into a pandas DataFrame with the expected column names.
* Encode the categorical columns using the encoder that we loaded earlier.
* Scale the numeric columns using the scaler that we loaded earlier.
* Rename the column names to remove any non-alphanumeric characters that might cause errors.
* Use the random forest model that we loaded earlier to make a prediction based on the processed input features.
* Return a dictionary that contains the prediction labels and probabilities.

**Step 4: Create the app interface**

Finally, we used Gradio to create the app interface that will allow users to interact with our model. We need to do the following:

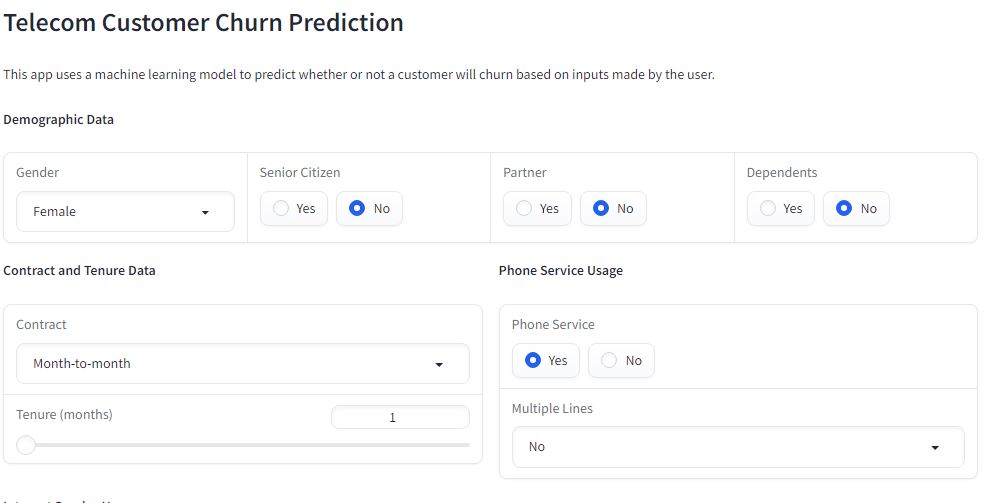
* Define some variable limits and lists of options that we will use for our input widgets.
* Create a gradio.Blocks object that will contain our app layout.
* Use gradio.Markdown to add some headings and descriptions for our app.
* Use gradio.Row and gradio.Column to arrange our input widgets in a grid-like structure.
* Use gradio.Dropdown, gradio.Radio, gradio.Slider, and gradio.Button to create different types of input widgets for our features.
* Use gradio.Label to create an output widget that will display the prediction result.
* Use gradio.Interface to create an interface object that will link our input and output widgets with our process\_and\_predict function.
* Use interface.launch() to launch our app in a browser window or an inline notebook.

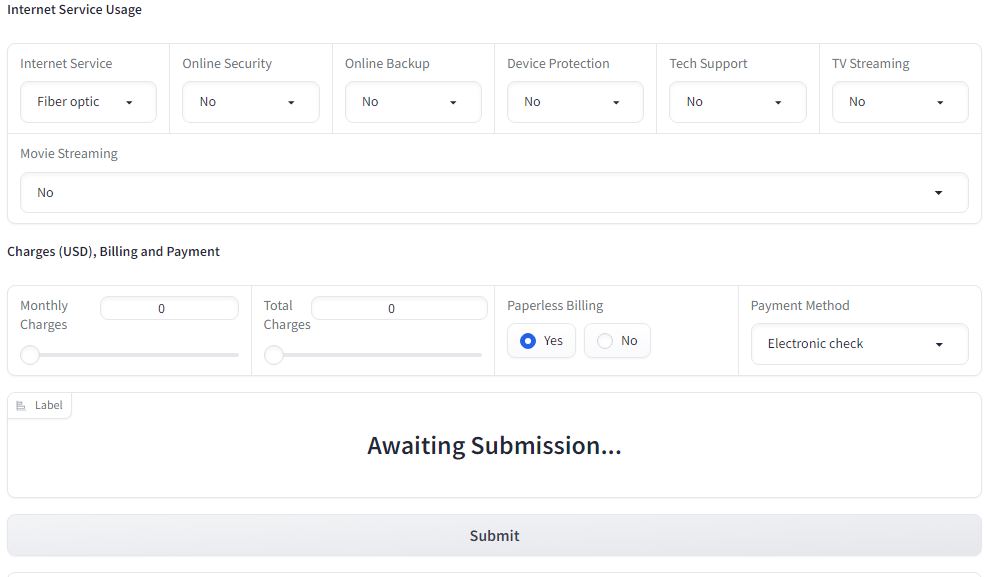


**Gradio App Deployment**

There are different ways to deploy a Gradio app, depending on your preferences and needs. Here are some of the most common methods:

1. You can use the share parameter in the launch() method to generate a public, shareable link that you can send to anyone. For example, you can write app.launch (share=True) to create a link like XXXXX.gradio.app. This method is easy and convenient, but the link will expire after 72 hours and anyone can access it. Also, the processing will happen on your device, so you need to keep it on and connected to the internet. (This is the one that we used.)
2. You can use Hugging Face Spaces to host your Gradio app for free and get a permanent link. You can either use the Gradio CLI (gradio deploy) in your app directory, drag and drop your app folder on the Spaces website, or connect Spaces with your Git repository. This method is more secure and reliable, but you need to create a Hugging Face account and follow some steps to deploy your app
3. You can use your own web server or cloud service provider (such as AWS, Azure, or Google Cloud) to host your Gradio app. You can either use Docker to create a containerized app or use Nginx to configure a reverse proxy for your app. This method gives you more control and flexibility, but you need to have some technical skills and pay for the hosting service.





**Final Notes**

Thank you for reading this far, I hope the article was helpful to you.

You may find the project repository here: GitHub