**Turning Machine Learning Models into APIs in Python**

Learn to how to create a simple API from a machine learning model in Python using Flask.

Consider the following situation:

You have built a super cool machine learning model that can predict if a particular transaction is fraudulent or not. But your friend found out that, you have coded your model in Python while your friend is building his application in Java. So? Won't it be possible to integrate your machine learning model into your friend's application?

Fortunately enough, you have the power of *APIs*. And the above situation is one of the many where the need of turning your machine learning models into APIs is extremely important. Many of the industries are now looking for Data Scientists who can do this. Now, wrapping a machine learning model into an API is not very difficult, and that is precisely what you will be doing in this tutorial - **Turn your machine learning model into an API**.

Specifically, you will be covering the following:

* Options to implement machine learning models
* What are APIs?
* Flask basics
* Creating a machine learning model
* Saving the machine learning model: Serialization & Deserialization
* Creating an API from a machine learning model using Flask
* Testing your API in Postman

The majority of the ML practitioners use R/Python for their experiments. But consumers of those ML models would be software engineers who use a completely different technology stack. There are two ways via which this problem can be solved:

* Rewriting the whole code in the language that the software engineering folks work.
* API-first approach –get the URL Endpoint from where the API is being served.

API is a contract between 2 softwares saying if the user software provides input in a pre-defined format, the later with extend its functionality and provide the outcome to the user software.

There are many popular ML APIs as well for example - *IBM Watson's* ML API which is capable of the following:

* Machine Translation - Helps translate text in different language pairs.
* Message Resonance – To find out the popularity of a phrase or word with a predetermined audience.
* Question and Answers - This service provides direct answers to the queries that are triggered by primary document sources.
* User Modelling – To make predictions about social characteristics of someone from a given text.

[Google Vision API](https://cloud.google.com/vision/) is also an excellent example which provides dedicated services for Computer Vision tasks. [Click here](https://github.com/GoogleCloudPlatform/cloud-vision/tree/master/python) to get an idea of what can be done using Google Vision API.

Popular examples of machine learning APIs suited explicitly for web development stuff are [DialogFlow](https://dialogflow.com/), [Microsoft's Cognitive Toolkit](https://www.microsoft.com/en-us/cognitive-toolkit/), [TensorFlow.js](https://js.tensorflow.org/), etc.

**Flask - A web services' framework in Python:**

Now, you might think what is a web service? Web service is a form of API only that assumes that an API is hosted over a server and can be consumed. Web API, Web Service - these terms are generally used interchangeably.

Coming to Flask, it is a web service development framework in Python. It is not the only one in Python, there couple others as well such as Django, Falcon, Hug, etc. But you will use Flask for this tutorial. For learning about Flask, you can refer to [these tutorials](https://www.tutorialspoint.com/flask).

If you downloaded the Anaconda distribution, you already have Flask installed. Otherwise, you will have to install it yourself with:

pip install flask

Flask is very minimal. Flask is favorite with Python developers for many reasons. Flask framework comes with an inbuilt light-weighted web server which needs minimal configuration, and it can be controlled from your Python code. This is one of the reasons why it is so popular.

Following code demonstrate Flask's minimality in a nice way. The code is used to create a simple Web-API which upon receiving a particular URL produces a specific output.

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

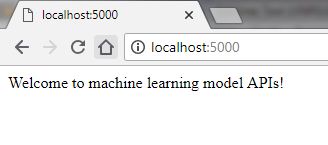
def hello():

return "Welcome to machine learning model APIs!"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Once executed, you can navigate to the web address (enter the address on a Web-Browser), which is shown on the terminal, and observe the result.



**Some points:**

* **Jupyter Notebooks are great for anything related to markdowns, R and Python. But when it comes to building a web server, it may show inconsistent behavior. So, it is a good idea to write the Flask codes in a text editor like *Sublime* and run the code from the terminal/command prompt**.
* Make sure you don't name the file as *flask.py*.
* Flask runs on port number 5000 by default. Sometimes, the Flask server starts on this port number successfully, but when you hit the URL (that the servers return on the terminal) in a web browser or any API-client like *Postman*, you may not get the output. Consider the following situation:

![Text

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDcRXhpZgAATU0AKgAAAAgABAE7AAIAAAAGAAAISodpAAQAAAABAAAIUJydAAEAAAAMAAAQyOocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFNheWFrAAAFkAMAAgAAABQAABCekAQAAgAAABQAABCykpEAAgAAAAMxNQAAkpIAAgAAAAMxNQAA6hwABwAACAwAAAiSAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAxODowOToxNyAyMjoyOToxNgAyMDE4OjA5OjE3IDIyOjI5OjE2AAAAUwBhAHkAYQBrAAAA/+ELGGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTgtMDktMTdUMjI6Mjk6MTYuMTU0PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPlNheWFrPC9yZGY6bGk+PC9yZGY6U2VxPg0KCQkJPC9kYzpjcmVhdG9yPjwvcmRmOkRlc2NyaXB0aW9uPjwvcmRmOlJERj48L3g6eG1wbWV0YT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgPD94cGFja2V0IGVuZD0ndyc/Pv/bAEMABwUFBgUEBwYFBggHBwgKEQsKCQkKFQ8QDBEYFRoZGBUYFxseJyEbHSUdFxgiLiIlKCkrLCsaIC8zLyoyJyorKv/bAEMBBwgICgkKFAsLFCocGBwqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKv/AABEIAEgCBAMBIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/AOW+OXhu41P4k6jqc11badpttbW0bXl4XCPIVGI0CKzO2PmIAOAMnHGfMb/wfrlj4mvNBXT5ry/s+ZI7JGmymAQ42jO0hlPIH3gDg8V7F8cNW06/1nUfDF/eppkyS22pW91NG7xSnyRG0bbFZlODuBwQcEHHBPmWnah4Y0PxtfvYap4kg0YRGK2udOlSK5kOUzuJ2jYSGIGAfu5Gc1U/iZMdjmTpl6ujpqpt3+wvO1sJxyvmBQxU+hwwIz15xnBxq6X4P1DVtHs9StprZYbzVk0mNZGYMJnUMGOFI24PXOfaoPD3iGXQbiZJIEvtNvFEV9YTEiO5TORyOVYHlXHKnn1B7fRdc8K2GnWul2usvDbWPiS31iOW9t5NzweUu6P92rfvFIKnOFOMg4PElHnWqafLpOsXmm3LI01nO8EjRklSyMVJGQDjI9Kq16f4f8Z6cusazctrj6NBf619tI8h0uPK3OwKyxhw5GcGGVGQg8EEkg8GeMtK0jR7qCK/ttPvn1pbtrq5tJo1lttpAHl2rclTk+Ux8v5yBnsAeYUV6R4e8YWOm+GvFvl6lbW1zeXkd3pVk9tKscbxyeZvVEDIh4TapYjMYDZUcz+HfHkF1fX97qeq22gSX2ow3NwljayRNhVILxyIJN5J5aKZGRtxIILEgA86OmXq6Omqm3f7C87WwnHK+YFDFT6HDAjPXnGcHGxoHgu+1/T471Luzsrea+TToHumf99cOMhAERiOMctgcjnrhml+Jm0fWL5liTUNKv2KXljNEIY7qPcSPkTIiYdVK/cPTIyD1Ohal4SsIbjTv7aebSpNRjmktNWtGeJ7Z4SHKCNGZLlCxXerIp2qQSOAAefXtnPp2oXFleR+XcW0rQypkHa6kgjI4PIPSoK9E8F+JPDvhr7bdWl5eQSrqcciw3Usy+fZJuIULAQrT9sS/u+R2LVe0W2g1a58Y6TpsEw8P63dI8WsWtqRbWKRSGbDiTywqhG55GNowGyKAPLa2LHw3cah4W1TXYLm28nS2jW4gYuJcSMFRh8u0gnP8WflPHTPpCfELQP+Es1nxBDdzW9xJqcDeU0tysd1Zwx7cxrFtBlYrnbN8gDAf3qztP8AEvhjQYvGTxtYapb6hqNvNY6YIJkjlhSfzCpygVQFfAB4yhG1lxuAPOjpl6ujpqpt3+wvO1sJxyvmBQxU+hwwIz15xnBxseHvB03iLSb3UY9W02xgsZYo5zevImzzG2oxKoRtLZGc8YJOBzWjqPixdP8AHOrz2+oP4m0fUmAu0vFMYvI8AgEYBRkPCuoG0qCoA+WtHR5/Cdt4e8U6PH4k+zWury2X2V7q0laWONHEj7wilSy7iowcMyE/KCDQBwup6Ze6Nqc+n6pbvbXdu2ySJ+oP8iCMEEcEEEcVVr2e3+J+kX17qd9bzQ6LeXOsQ3Hm3qXDeZapCIwD9nILsCpby3Oz5yMnrWB4R8V6dZXGo7taTRILnVluitvZvbyCIFiDG0fmDAzgwSqyYPDZJNAHH2Phu41DwtqmuwXNt5OltGtxAxcS4kYKjD5dpBOf4s/KeOmaNtpl7eWN5e21u8ltYqr3Mo+7GGYKuT6kngdeCegOO38K+KNE0Tw/4ne/FtqUl9eWrwabNbNELhIpt7FlQGOMYbhdxAKkEEYDalt4ysY7zxdpy+Nb9I9Vntp7PWHjlBTDqZRtQBg2zCDCqrCMD5RgUAeVUV6X4P8AFumafp5jk12awvB4iiv7q9njlEl/aAHKMYg5LZySjHbl+p5q94ZabU/Cfjy/0WSbSLFb6K606fy5FWz/AHjmQoIlYowhIVtg4UgH5eaAPOtE8P3XiD7VHpskL3kEXmpZFiJrlR94RDGGZRztyCRnAODWXXe6trPhfUfiZq3iUTzRWEEqT2lpbI0M19MAoyHUfulLguzkh8HgbjxueGL7WPEfg7xvrL3H9mytdW93BdwxSpFay5bzmj8pWZW8sgMwBYq2XOCTQB5NVq20y9vLG8vba3eS2sVV7mUfdjDMFXJ9STwOvBPQHHqy3dr4v1vxlbaVYXlvoOvS2+dYjtQILZrfDySSbmVQrfM5JYNyPlJbFRv8RbHVNY8ZQprtzocOqz2rWN7H5rLGsTKkjgIAys8ag4wM42swwDQB53Y+G7jUPC2qa7Bc23k6W0a3EDFxLiRgqMPl2kE5/iz8p46Zx67rQ/E1rpHg/wAXlNYd9V1Oe3NmZoWM77JSzSscMqNh8g7ywZSQchSewn8deGP+EgvJtF1hNKV9ftdQnuUt5oxe2ywqJI/kQsT5m8lXAUlycnJoA8Vor1LSPHmk6dp6pp19NpaS+MBfvbRI8e2wIGVYJ8pXgAoCenQ1594hmsrjxPqk2lBBYyXkz2wjTYojLkrhcDAxjjAxQBnUV3vw98R2ej6fLb3evf2SJL6GaUxwSJMUUH5kljVw+CcGKVChB4IJJE/gvxRomgfbb8300d62pxzPGwmt47i0XccRxW7bBLk/ckPlgEAHG6gDzuivUtFtoNWufGOk6bBMPD+t3SPFrFrakW1ikUhmw4k8sKoRueRjaMBsiryfELQP+Es1nxBDdzW9xJqcDeU0tysd1Zwx7cxrFtBlYrnbN8gDAf3qAPN7Hw3cah4W1TXYLm28nS2jW4gYuJcSMFRh8u0gnP8AFn5Tx0zj16fp/iXwxoMXjJ42sNUt9Q1G3msdMEEyRywpP5hU5QKoCvgA8ZQjay43YGpRaDr/AMQdVk1LxjMLCTMsWqT2MkrzMduIzGu0jAJGcAfJwACAADj6K9L+H1vqM3wz8WSW93NYLbS201re4l220gLec6GNWZW8sgMVGdrDd8taq3dr4v1vxlbaVYXlvoOvS2+dYjtQILZrfDySSbmVQrfM5JYNyPlJbFAHj9FdF4/16DxN481TVbMYt5pQsRyfnRFCBuQCNwUNgjjOO1c7QAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAem/tAHPxXn/687f8A9AFYvgXwbpfiqx1SXVdZTShaKhW5fPlw5YD95uUIQ2SF/eK2VPysOm5+0Iuz4t3A/wCnO3/9ArzeKyup7Se6gtppLe22+fMkZKRbjhdzDhcngZ61UtyY7Gx420K18N+Lr3S7GV5IYGwol3GRf94tGgJIw3ygrhhhm610vw9tYbzwR4nMg0eC4tpbP7Ne6nZxyJA0shRtzNGxCkADkEDOeOTXBXlldaddva6hbTWtxHjfDPGUdcjIyp5HBB/Gup8OWWq33gPXVttb03T9IjlhOpR3EDGRssPKO5ImYruHADcENkAHJko3/FfgGyvPEtxdaZv0m2uPEkWjLZvbY8svGrNMvzfdJJKqMDaRggcDNtvhvav532zW5oseIm0GLyrESb5O0jZlXap5yBkj3qr4m8Q+MtE8S6rput6gj3pnSWZhFE6iRYwsc0Xy/u22FcMoVumcEcZc3jfxBO4d71ARqK6p8ltEg+1KABJgKBnA5HQnJIJJNAG3p/w4ikt2Oq6w9tN/b/8AYKpbWgmUzYHz7mdCFzntnpx6HgTw5F/ws86Hfy2z31tPNAkc9iLu1lZEkDlwXQ4G3K8HnGcYrKtte8VXljeXttM8ltY6ius3Mohj2x3TMFWQnHUk8J04JxgHFXT/ABhrel63c6xZXMKajdStNJcvaQu4dt24qWU7M7mztxnP0oA1fDfgi11vQLXU7zVprT7XrCaRFHFZib946hg7EyLheTnAJ470+0+Hcr29wb7UUhmXWv7Ct0hiMiyXODyzEqUjzj5gGbk/LxzjjxhraRRRQ3MMEUN8moxxQWkMaJcIoVXCqoHQDjoepBJNb/hPxVqFkdV1yTT7+7kivItQvryy1BrRXzJgRzKFZHVnfhQob7/OM4AOW8QaZFoviO/0uC4e5WynaAzPEIy7KdrHaGbAyDjnkYPHQbfhrwTFrej2+pX2pPZw3erRaTbrDbCZjM67tzZdAqgEcgk9eOOcuJJfF3i6Z7m7sNOm1GeWeSe6lMVvGx3OQWOSBngdeSBRo/ivWNBt/I026RYfPW5WOa3jmVJlGBIokVgrY/iGD09BQB0Wn/DiKS3Y6rrD2039v/2CqW1oJlM2B8+5nQhc57Z6cenH6pp8uk6xeabcsjTWc7wSNGSVLIxUkZAOMj0q9B4s1u3t44Y9QcrHqI1RTIquxugMeYWYEk+xJB9Kzr28n1HULi9vJPMuLmVppXwBudiSTgcDknpQB2fhCy0+/wDhr4uOprbRC0azkivGtFkmh3ykOEYDcSwQKFLBcnkqCxq8/wAPdI0Wx8ZR69f3MlzojWqRzWtuGVUmZSrhS67mI+UqSAuSQXOMcRba9qNnoN5o1tMkdjfMr3KCFN0hUgrl8bsAjIGccn1Od/Q/FfiXVPGMtwNfs7K81LZ9outQWNLcmIBo2YFCoYbBtYLkHoeTQA/w74d8O3/g/wAUarfXd/nTWgSB0tlJRJJQA+zzAGYhSCpbCgk5Y4xR0rwxp+p6Dq+sHVLmK00meNZh9iVpHhkJWN1HmAbtwGVJAAOQzEYLNB1nxRYrqur6PPMyJ5c2oTyosyFjKDG7CQEF/MOVP3vvEfxGui8J6nPbfDjxjqPlQ388ssBvHuphI6kv+6fy5IXSTLsxbLZ4/hOGIBSi8FWulfEHUvD2uQaxqyWcW5ToMAeVs7GVmVgcLtfnrhiBk9TgeHvD0uvXEzyTpY6bZqJb6/mBMdsmcDgcsxPCoOWPHqRG/ifXH1u51gateR6jdZ865hmaN3Bx8uVx8vAwo4GBgcCqp1O9bR00o3D/AGFJ2uRAOF8wqFLH1OFAGenOMZOQDbttDsvEmvXkujq+k+HrJVa4vL1t/kRgBd74+9I5BIjXqzYHAyNzwzaeFtYuvFLWOkXKrZaLdXFk13chwuyNUDsgX/WElnzu2qWwB8oauSt/7ZvfDN1b2/nPpGnSi7uFGBHHJIViDN6scAAcnG4gY3GpNF8Var4ft54dKktoluVZJjJZQytIjABkLOhJU4Hy5x7UAdF8P5Ldrdkm0iw+zWl5Hd6rq9/Gk0cdoBjyAjo2Gc527PmY4HQZBo99Hqb3mi+EtBsIjJqMt6mo6lBDKLSyAxskMqOFVRhi2488DJPOdo3iPxTbaTfTaQIfsFldLqM5NhA0dvM7Kiuu5CEbOAoXGADgAA1lQ+JdWg1DU72K723GrRSw3r+Wh81JTmQYxhcn0xjtigC344v9I1PxpqF14bt0t9MdlWBI4REuFRVLBB0BIJ7HnkA5Fdh4I0axb4aXOuXEVtGLfVmF9dy20E7rapb7hHGs+VDPIyqMDJLrnIGK4Tw/4l1bwtqD3ug3f2W4kiMLP5aPlCQSMMCOqj8qNO8SanpenyWFtLC9nJKJmt7q1iuIxIARvCyKwVsHGRgkYz0FAGx8UNMstH+JWr2WmW6W1sjRusUfCqWiRmwOwyx4HA6DAro/BGjWLfDS51y4itoxb6swvruW2gndbVLfcI41nyoZ5GVRgZJdc5AxXH2XjjxHp/iO61611Nxqd4pSe4eNHLKSDjDAgD5VwABgAAYHFVNO8SanpenyWFtLC9nJKJmt7q1iuIxIARvCyKwVsHGRgkYz0FAGx8UNMstH+JWr2WmW6W1sjRusUfCqWiRmwOwyx4HA6DAqfwDp+kalZ+I/t1g897Z6Ld3UErygxIQgUHy9uSwLEhi2Bx8uQDWXZeOPEen+I7rXrXU3Gp3ilJ7h40cspIOMMCAPlXAAGAABgcVBovirVfD9vPDpUltEtyrJMZLKGVpEYAMhZ0JKnA+XOPagCfwfpen6trEsGq2Ot30KwF1j0SFZJg25RkhgRtwTk+pWt7wlLpt34Qi0ezuNHt/Et1rAET6lp32jfC0aqEDGGQDMmOuMc9M88fba1qVjqE17p17NYXE27e9k3kZBOSoCYAXIHyjjgccVb0fxXrGg2/kabdIsPnrcrHNbxzKkyjAkUSKwVsfxDB6egoA9B8OeH4oPBeq6prFrZ2t1Za7LHqs7WltMYoY4SxiiSQGNWaVgi7B1ZeoGK6LWfCeiaSNfv7Sxs7K1tNYVryZrWCcx2ws1lMcST5RWeWQKoUdXUYIUCvHLTxXrFpb3EAukuYbmf7TLHe28d0rTYIMmJVYBiCcsOT3zU7eN/EEj3bXV6l4LycXMyXttFcIZANoZUkVlU7fl+UDgAdAAAD1zWfCeiaSNfv7Sxs7K1tNYVryZrWCcx2ws1lMcST5RWeWQKoUdXUYIUCuI8X+C4Jvilrtrp4h0jRLDyp7u6ZT5NojxoxwO7MzELGvU8DAHHNN438QSPdtdXqXgvJxczJe20VwhkA2hlSRWVTt+X5QOAB0AAqap4m1nWopI9U1Ga5SW6a7dWIw0pULuIHoqhQOijgAZNAEGsy6bNq0zaHbTW1gNqwpO++QgKAWYjjcxBYgcAtgcAVv+CtMspLHW9curdNSn0WBJ4NLflZyzbTI4HLRx8Myjg5AJAznk6taZqd7o2pwahpdw9td27b45U6g/yIIyCDwQSDxQB3XgtobzwR4vvbtNHguLaW1lhu7zTY5EgaWQhwFETEKQAAoUgZ4A5NF54a8N+LNQ8S65oGof2TpWnXUBMb2oEYikIV5Ey4PUSMI9uSNqgZO0cnb+KtVtdM1HT7eS2jtNTYvdxCyhxIe2PkyoB5ULgKeVwajsPEmp6bol7pFnLCtlf/8AHzG1rE5kx0yzKWGOoweDyMHmgDrbb4Y2ji8ubnxVYDTYdRWxhvYGiaOTMYkLkvKigBWXKqXbIYAHbk5Wh+DLLVLi9t7jXUM9teC2WHTrf7XJMuW3Tou5C8Y25/dh2xklQOTl6P4r1jQbfyNNukWHz1uVjmt45lSZRgSKJFYK2P4hg9PQU/TvGOvaX5n2a+3+ZdC8b7VDHcYnGf3o8xW2vz94YJ4yeBQBe8OeErDXPtKya5+9iulgjgsbRriaVDnMyxsUZkGMkIGcDJKgDnmZ41iuJI45knVGKrLGCFcA/eG4A4PXkA+wrY07xjr2l+Z9mvt/mXQvG+1Qx3GJxn96PMVtr8/eGCeMngVjzzy3NxJPcyvNNKxeSSRizOxOSSTyST3oA7DSre30j4W3PiOG0trjUp9WXTla8gSdIYxF5pKI4K7iRgkg8dMZJrotDure61nxdBL4RsNIFrpN1qUVnd2SSy282yLbzImQoO5lTAUeYeCMY870rXtR0VLmOwmQQ3aqk8E0KTRShTuXcjgqSCMg4yOcdTV638b+ILbU9R1Bb1JbvU1KXctxbRTGVD1T51IC4wNowMADGAMAHWeC/J1nwv4vu5INBsLhbq1ktp7yxjaC0aaYqygsjlUwAoByF68cmtHXbPTLSx+Ja2GiW1othPaLai4sYzJCZW2SshYEqrY3Lg4AIKhc15pba9qNnoN5o1tMkdjfMr3KCFN0hUgrl8bsAjIGccn1OdR/iF4kme/ee7tpzqTI9352n28gmKABNwaMjChRgdAcnqSaAL3g7wIureO4fD/iad9MkKh2teRcOGhMi7fkZVIG0kOQcHHXgT21hZeHvhxea9ZGw1S+l1ZdOS5lt/Ohjj8kSkxxzIPmJ+UllPAOMZyeS1nWb/xBq02p6vP9ovJ9vmSbFXdtUKOFAHQDtUmla9qOipcx2EyCG7VUngmhSaKUKdy7kcFSQRkHGRzjqaAJ/FjrJ4pvZE0N9ADsH/s585gJUE8EAgEncBgABgBxitjwv4f0PUfBHiTVtXuLyOfTvs6oYIVcRiSQDcAXXexwy4JAAOfmONvM6nqd7rOpz6hqlw9zd3Db5JX6k/yAAwABwAABxVrQ/EureHJZX0e78jztnmI0aSIxRgyMVcEblYZDYyOxGaANzQPBFrrOnx6hLq01vZ3WsJpFoVsw8jyONweRfMARcFejMck+mTzOqafLpOsXmm3LI01nO8EjRklSyMVJGQDjI9K0dO8Y69pfmfZr7f5l0LxvtUMdxicZ/ejzFba/P3hgnjJ4FY888tzcST3MrzTSsXkkkYszsTkkk8kk96AOm0DwX/bOgR6nLf8A2f7VqaaVaRrDv3zuu4GQ7hsTlfmAY8n5eOdGb4cRK4EGsO6/8JIugEvaBTuwN0uA54ByAvcAHIzgY/hTxjeeGbuAMv2vTo7pLtrJxGQZUBCsrOj+W3YsoDEDGRwQ+/8AHGoyanqk2ku9laaheG++zSlJzBOc5kjcoCjAscMoVgNvJIBoA2PBfhu1X4ry6FPcQz3FrdT28Au9OFxbz7FkDGRDKpXhcgDdzj05peF/D+h6j4I8Satq9xeRz6d9nVDBCriMSSAbgC672OGXBIABz8xxtzrTxtrtjr1xrVrcW0epXLb5Ln7DAWDYIJXKYUkE5K43Z5zUen+MNb0rULi8065htnufL82OO0hELFCCjeVt2BlKghguQcnPJyAXvBcdre+KhoLaf/alhq0oti7QhLiFd3E6MNxjZR8zDJUgENkcjA1S0isNYvLO2ukvIbed4o7mPG2ZVYgOMEjBAz1PXrU9vr+qWqaiIbxw2pqUvJSA0sqk7mBkI3AMfvAEbu+azqACiiigAooooAKKKKAPUv2if+SvXP8A152//oFcz4K8fXXghLx7Gxtri5nULFJMi4jyVLbiF8xgQowodVBycE10/wC0WMfGC5H/AE6Qf+gVz/gXwbpfiqx1SXVdZTShaKhW5fPlw5YD95uUIQ2SF/eK2VPysOlS3JjsYnirXh4l8R3OqLapaLOxYQqseVySxyyIm85J+ZhuxgEnGa3/AAf4ksPDvgjxGs0tnPf3stp9nsLq1aZJVik3PvBXZtKscfMDwcYO01j+NtCtfDfi690uxleSGBsKJdxkX/eLRoCSMN8oK4YYZutb3gqeG0+HnirUZNN029n02WykgN7ZRzY3y7XUlhnaVUDGeOSME5qSjrV8e+HrbVdXfSdbSOa51qK9+338V0fNt/KAMYEO13Eb7gscmE2kDnqOd8I+K9OsrjUd2tJokFzqy3RW3s3t5BECxBjaPzBgZwYJVZMHhskmruufCjTIPEGqyLrCaTpMOoxWcPnNGxQvCsrFmkkjG1Q4AALuQDwcZOX4e0vTB4E8bW+rtbSrps9qsWoWdrHcSKWmKs0TkqWVti/xAYJIHJBADwj4stbPSvEukS+ILnRhqU8M1pexwNGsW2XMhEcOdjMmPlX5fl2lgAKf4L8U6RoP23UJtUvH1FtTjnlN1JcJ9stV3McLC+GnLE8SsU+Yc/eNM1H4Uy6ZdJBPqyMz6/Do4KQEjbLGsiy8t1w4ynrn5jVWP4c5+2ebqm3Zrp0K022+7zp+cNJ8w8tPu8jeeTwccgG3Y+KtI0XTfGkel61ZgXd9Hd6LbCC4EcbJJ5oYL5YVWHyKAeC0YDAoATBofxBGoQeJ4dS1X/hG7jWZbaeK4sY5RDA6OBMyqhLB3UZP94g5YcVnN8OIt92F1hyqa0NCtCbQAy3OOS43/u484G4b2xztHSp7f4ZWjvqMt14iS1sbLVjpQuJoooy0ijMjsHmUBR22szEZO0YxQBH8OPEltonjyTVNT8STW+nGV2uBOJjJfblkCsyIHBYMwY7m4J4Jq94a8V6f4X8N2GmQa/8AvY/E0N3cyWazqkln5Sh85RSy5GCpGTjoRg1iaN4HtdT03XLq416GAaJdRRXEscYkgeJ5NhlSTeC2ArsFC5bCgZLYGBr+lxaLr11p9tqFtqUMLAR3dq4aOVSAQQRkZwcEZOCCMnGaAPUvCMtvqlxqNpotjcrpk/ilZWn02BHWa1YsFt54tyukG3JyVZOWGMjB838aXv8AaPjrW7oXP2pJL6byphJvDRhyEw3ddoAGOMAYrV+GGmWureN7S1neE3D7/Ihu7AXVvJiNy3mKZEIwBkYzzjpiqnhbwtZeIbeV7jVngnWeOJbO1tftFwysCWlEe5SyrjkR72AySoAyQDS8OazoM/gC58PeIJ/s23U1u4X2SPv8yIws2FHHlZEuCfn27cc5rotK8eae/wAaNR8ST61/ZujPLteD9+32xFhaKNtqJg8gNh8bd3GSDXlM8axXEkccyTqjFVljBCuAfvDcAcHryAfYV03g/wAFnxRl7i4msbdrqK0iufLiMbSvn5cySxksBg7UDtg9OmQDY8IeILTQ9G8QeHn8UvYC9a2ktdTtFnMUZVwZSoChwxQ4+6A2zBIGCYND8TWukeD/ABeU1h31XU57c2ZmhYzvslLNKxwyo2HyDvLBlJByFJqx+BoLPT7y98R6x9ht7XWDo5e1tTcfvVBLuQWQhAB2yx5+X1f4d8N6DeeD/FGpane3LSaW0EcU1pBvQLJKFEiqzIWJ2kYbbtBzyeFAKPxE1LS9Y8f6nqGguj2NwyOjpEYwzeWu87SAcl9xJI5OT3rY+HviOz0fT5be717+yRJfQzSmOCRJiig/MksauHwTgxSoUIPBBJIq/DfS9P1Tx/BZrMkjM0gtEv8ATVnhmURuSZU80bTgAgAt83fjJq6J4L/tjQNP1P7f5P23XY9H8vyd2zeobzM7hnGfu4/GgDY8I+LLWz0rxLpEviC50YalPDNaXscDRrFtlzIRHDnYzJj5V+X5dpYACn+C/FOkaD9t1CbVLx9RbU455TdSXCfbLVdzHCwvhpyxPErFPmHP3jUFj8M4Zf7Sm1LXobKzstYfSVndY13MuS0jCSVAFxg4Uux544yZ/h4vhHTNQ8TWvjKOz1D7PauIJhIrRyqhIcQszLl3+UoRg4BwR3ALel+KdKsZvGGkWPiJ9IstTvIrjTb20imWK3TztzhUVQyt5ZC4AAOzG4DBPm97L5+oXE32ia58yVm8+cYklySdzDJ+Y9TyeT1PWoKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAPVP2juPjFdf9ekH/oFeZRWV1PaT3UFtNJb223z5kjJSLccLuYcLk8DPWvTv2j/+SyXf/XpB/wCgVy3grx9deCEvHsbG2uLmdQsUkyLiPJUtuIXzGBCjCh1UHJwTTe4lsc1eWV1p129rqFtNa3EeN8M8ZR1yMjKnkcEH8a7bwEPEK+HtSuND17R9Ks7e6gN39vtwxDs4ELlzC/yhxxz8pBbjqea8Va8PEviO51RbVLRZ2LCFVjyuSWOWRE3nJPzMN2MAk4zW/wCD7zRE8EeI9J1jW4dNl1aW0WPdbzSlEik3uxCIR0PAzyRzgc0hma2veKvCWp3en3Uzw3aXgupor2GO4K3A6TL5gYBiDnevUYOSMVb0afxDJ4O1zVrPU7N7W2uoptRt7qASy3DyHajPvjYSLuZiFZjhgWwDgnt7f4n6RfXup31vNDot5c6xDcebepcN5lqkIjAP2cguwKlvLc7PnIyetczoPiaxstG8VpaawmjXmtXkX2NoIZYBbIjvIWIjD7FIbYFVnIJwfl+agDnW8ceI3e7eXU3ma8nFzI0saOUmA2iSPcD5TAcBk2kAKBjAxqaF8Q5dN1O41K/trma+uLyO7lnsL02Qn2ZzHLGqmN1J5Pyhslssd1dbqfjvwvfXtzJZXH2SK18TQayitbsv2uNIVV/LCg4dnQt8+wHeCSDnFrw9cRazqeqRaVHc3Nje+KUup73TohILi3k3H7PcxsUkWHBbLMrJneMHHIBxGh/EE6XrF7qlzZXJu7vURfyNp+oyWqyfMzGGRCHV48t0wGGSNxBrOj8d69BqF5dW1xDH9rvjqDQtbxzRxTkk741kDbGGcZHOAMk4Fel6Jeya7resQQJeXVvP4rVp76w2XIubX5kWCdQ4YW+0cMQyY3DHHNWz8d6bZ6nrwGr21vfSeJGuRfXAuJEmtFyqKptzmQLjiNyIyG/IA80t/FesW2majYJdI8GqMXvfOt45XnbrlnZS2QeQc8EkjBOax69I8PeMLHTfDXi3y9Stra5vLyO70qye2lWON45PM3qiBkQ8JtUsRmMBsqOeI1/WpfEOvXWrXNtbW010weSO1QpHuwASASTkkZJzySTQA/QfEmp+Gbs3WjSwwXB6TPaxSunBHys6krkMQcYz3rY8N+I9dudZSytNV0rTFuJxch722gjtYJo0O2RVMZWNsDG5VBJxmt/4fW+ozfDPxZJb3c1gttLbTWt7iXbbSAt5zoY1ZlbyyAxUZ2sN3y1e0rxpokPxo1HxBb6z/ZegyS5lg8qZftp8lk3+XGpB/eEv8+D82euQADymeeW5uJJ7mV5ppWLySSMWZ2JySSeSSe9aOneJdW0nT5LKwu9lvJKJwjRo/lygECSMsCY3AP31w3A54Fdv4a8V6f4X8N2GmQa/+9j8TQ3dzJZrOqSWflKHzlFLLkYKkZOOhGDXC+IZrK48T6pNpQQWMl5M9sI02KIy5K4XAwMY4wMUAaMXj3xFDdz3KXkJe4ulvHD2cLIJ1GBKqFCqP6soBPUk1V0rxZrei3lzdWGoOJrtleczKswldX3q7BwQWDDIbqDnB5NdL4P8R2en+EDp9vr3/CO6oNYiu5b3yJH823EZXZ+7U79rZOx8Kd3Xk1e8H+LNJ03TzCNX/sm4HiKK/uLj7M8P2yzAIMW2AMBzk+WcIN3BPNAHM+DtX1tfF0K6TqNtbanqM4Rb69t1mbzGyPvsjspYsQSOu75uOQS+IfEfhTxHrNtZ6gllcuz2d39giSKJ9hKZVVUBTwcMAGGSeCTXVaH410TT7e9k065fRm/4SQamYo4mja6sQGxbDywRkdNjkJ83XrjR0zxvoen3etfYNWs7e4uvER1AX00F2EltWBIXEW13ZGJzHJhDuPXOaAOB8L6v4og3WHhiOa72yreC3SyW78qROBKqsjbGGQNwwenPSs6z1/VLC4v57a8fztSgkt7uSQCRpkkOXBLAnJI69fetjQ9c0q08X6jqF5d6xptnceaYjoO22kG6QMF2liFTGflDHBC8nGa5agDY0fwrq+u2d5e2No/2KyglnnunUiJRGm4ruxgsRjCjnkHgZINK8N3Gp6Zc6nNc22nabbMsbXl4XCPIekaBFZnbHzEAHAGTjjO34AvdK0y316TVdYtrJtQ0m40+GKSKZ23uF2udkbALwR1zx0rV8IeItE0/wwuhalf2AWDX0vJmu7FriG6tdmxxGDExDHbwSqHDdRyAAcfo/hXV9ds7y9sbR/sVlBLPPdOpESiNNxXdjBYjGFHPIPAyRj13vhTUfDtjqHiq4/tKHTLPUrG8sLC1njmeSNZCpjLFEcbQBg/MTkdD1rlrbWLnR/OtbP8As26iErETS6dDPv7ZVpoywU4BAOOvQEmgC3p/hSS98MNr9zqthp9iLz7EDc+cWMmwP0jjbjB6+xqPTfCWq6vFqFzpsPn2GnxTSy34VlgKxLuIDMoJYjGFwD8wJAGSOi8LeJ4ZIki1zWdNtbIanFc3OmXGiRvBNHtCu8Zjiba+1duNqj5s7sk0eFNR8O2OoeKrj+0odMs9SsbywsLWeOZ5I1kKmMsURxtAGD8xOR0PWgDmtH8K6vrtneXtjaP9isoJZ57p1IiURpuK7sYLEYwo55B4GSMeu38DXWkaP/wkX9o67Zw/bdMudNg/c3Db2fbtk4iOEOD1+b/ZrnLbWLnR/OtbP+zbqISsRNLp0M+/tlWmjLBTgEA469ASaAJNK8N3Gp6Zc6nNc22nabbMsbXl4XCPIekaBFZnbHzEAHAGTjjNrQPBd9r+nx3qXdnZW818mnQPdM/764cZCAIjEcY5bA5HPXHU+FvGOlSeG00/XJ9NtXGuxX1zHcaWrwTW3lBHWNI4mVX+XrtX733uTVrwZ418P6Lb3VtaXKadbPr63f2fUYpZo3sgDtVAofbMhCtu4yVTLMBgAHJ6d8P9VvvM+03Fnp2zUxpK/anZvMujn92PLV+mPvHC8jBPNc7e2c+nahcWV5H5dxbStDKmQdrqSCMjg8g9K73wr4xsNO8TaxqF1ff8ShLqXUbPS76Fp5LmfD+ViQq/lOMjMm7JIGdwyRwV7eT6jqFxe3knmXFzK00r4A3OxJJwOByT0oAgooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA9R/aIYt8XLhmOSbO35/4BXl1FFVP4mTD4UFFFFSUFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAH/2Q==)

* According to Flask, its server has started successfully on port 5000, but when the URL was fired in the browser, it didn't return anything. So, this can be a possible case of port number conflict. In this case, changing the default port 5000 to your desired port number would be a good choice. You can do that just by doing the following:

app.run(debug=True,port=12345)

* In that case, the Flask server would look something like the following:

Text

Description automatically generated

Now, let's go through step by step of the code that you wrote:

* You created an instance of the Flask class and passed in the "**name**" variable (which is filled by Python itself). This variable will be "**main**", if this file is being directly run through Python as a script. If you imported the file instead, the value of "**name**" would be the name of the file which you imported. For example, if you had test.py and run.py, and you imported test.py into run.py the "**name**" value of test.py will be test (app = Flask(test)).
* Above hello() method definition, there is @app.route("/"). route() is a [decorator](https://jeffknupp.com/blog/2013/11/29/improve-your-python-decorators-explained/) that tells Flask what URL should trigger the function defined as hello().
* The hello() method is responsible for producing an output (Welcome to machine learning model APIs!) whenever your API is properly hit (or consumed). In this case, hitting a web-browser with localhost:5000/ will produce the intended output (provided the flask server is running on port 5000).

You will now study some of the factors that you will need to keep in mind if you are turning your machine learning models (built using scikit-learn) into a Flask API.