NAME: MOHAMMAD SHEHZAR KHAN

BATCH CODE: LISUM26

SUBMISSION DATE: 28-10-2023

SUBMITTED TO: DATA GLACIER

TOPIC: MACHINE LEARNING MODEL DEPLOYMENT ON FLASK

STEP 1: Choosing a dataset. Here, I chose a dataset from Kaggle, which contains phone specs and prices.

Link to the dataset-

(https://www.kaggle.com/datasets/mohannapd/mobile-price-prediction/download?datasetVersionNumber=2)

STEP 2: Doing the preprocessing steps and make the data ready for Machine Learning. Splittng the data into train and test data. Training the model on training data. Saving the model using serialization operation in pickle. Saving the python file.

```
import pandas as pd
import pickle
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

phone_pricing = pd.read_csv("C:\\Users\\Shez  Khan\\Desktop\\Data Glacier Internship\\Week4\\Cellphone.csv")
phone_pricing.head()
phone_pricing.isna().sum()

#considering only most important features for model training
X = phone_pricing[['internal mem', 'ram', 'RearCam', 'Front_Cam', 'battery']]
y = phone_pricing["Price"]

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=26)

lm = LinearRegression()
lm.fit(Xtrain, ytrain)

pickle.dump(lm, open('price_model.pickle', 'wb'))
```

STEP 3: Creating the “app.py” file. This is a Python script that defines and configures the Flask application. It is the entry point for the Flask web application and contains the core logic for handling HTTP requests and defining the routes for our web application.

STEP 4: This file contains the following:

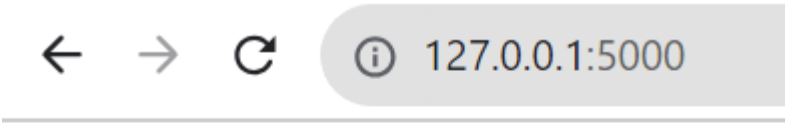
- IMPORT STATEMENTS: The script imports necessary libraries and functions using import statements. These include Flask for creating the web application, jsonify and request for handling HTTP requests and responses, pickle for loading a machine learning model, and pandas for data manipulation.
- CREATING A FLASK APP: An instance of a Flask application is created with app = Flask(__name__). This instance will be used to define routes and handle HTTP requests.
- ROUTE DEFINITIONS:
    - The script defines two routes using the @app.route decorator.
    - The / route (@app.route('/', methods=['GET', 'POST'])) is intended for both GET and POST requests.
    - The /predict/ route (@app.route('/predict/')) is intended for GET requests and is used for predicting phone prices. It loads a machine learning model (price_model.pickle) using pickle, retrieves parameters from the query string of the URL (e.g., internal mem, ram, RearCam, Front_Cam, and battery), creates a DataFrame with this data, and uses the model to make predictions. It then returns a JSON response containing the predicted phone price.
- DRIVER FUNCTION: The if __name__ == '__main__': block ensures that the app is only run when the script is executed directly (not when it's imported as a module). It starts the Flask development server with debugging enabled (app.run(debug=True)).

```python
#using flask to make an api
#import necessary libraries and functions
from flask import Flask, jsonify, request
import pickle
import pandas as pd

#creating a Flask app
app = Flask(__name__)

@app.route('/', methods = ['GET', 'POST'])
def home():
    if(request.method == 'GET'):
        data = "hello visitor"
        return jsonify({'data':data})

#'internal mem', 'ram', 'RearCam', 'Front_Cam', 'battery'
@app.route('/predict/')
def price_predict():
    model = pickle.load(open('price_model.pickle', 'rb'))
    internal_mem = request.args.get('internal mem')
    ram = request.args.get('ram')
    rear_cam = request.args.get('RearCam')
    front_cam = request.args.get('Front_Cam')
    battery = request.args.get('battery')

    test_df = pd.DataFrame({'internal mem':[internal_mem],
                            'ram':[ram],
                            'RearCam':[rear_cam],
                            'Front_Cam':[front_cam],
                            'battery':[battery]})

    pred_price = model.predict(test_df)
    return jsonify({'Phone Price': str(pred_price)})

#driver function
if __name__ == '__main__':
    app.run(debug = True)
```

STEP 5: Then we open the terminal and go to the directory where our .py files are located. Then we enter the activate the FLASK in our virtual environment using ".\flask\Scripts\activate". Then we run python app.py, this initializes and starts the Flask web application, making it accessible via a web browser or by sending HTTP requests. It allows you to interact with the defined routes and view functions, which handle and respond to HTTP requests according to your application's logic.

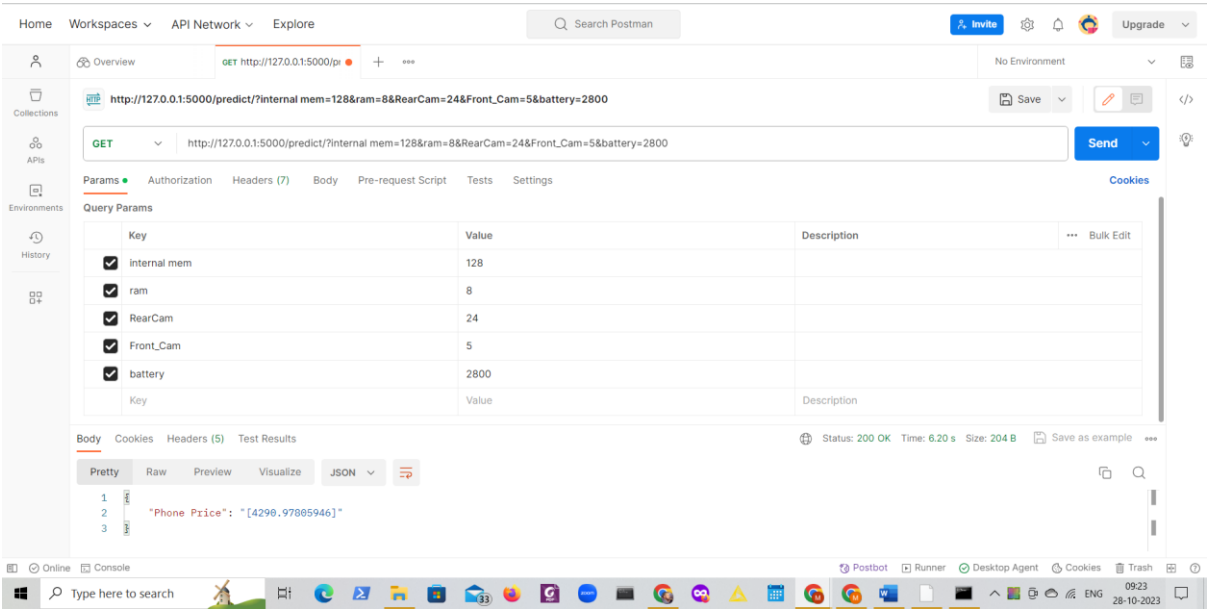
```
C:\Windows\System32\cmd.exe - python  app.py
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shez  Khan\Desktop\Data Glacier Internship\Week4>.\flask\Scripts\activate

(flask) C:\Users\Shez  Khan\Desktop\Data Glacier Internship\Week4>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 274-280-473
```

STEP 6: Then we go to our homepage by using the URL in the above picture.




```
{
  "data": "hello visitor"
}
```


STEP 7: Then we open the postman app to get prediction from our ML model. We paste our URL followed by "/predict/" and provide the key and values for which we want to predict the price. In simple words, provide the phone specification and model will predict the price.



We can clearly see that our model made a prediction of approximately 4290 price units when we give internal memory as 128GB, RAM as 8GB, Back Camera as 24MP, Front Camera as 5MP, Battery as 2800Mah.

I hope I was able to explain each of my steps well.

Thank you!