Data Science Virtual Internship

Name:Nkululeko Freedom Mqadi

Batch Code: LISMU01

Submission date:28 July 2021

Submitted to:Data Glacier

Introduction

In this document I am going to explain a sequence of steps to deploy a machine learning model on Flask.The tools and technologies that are required to accomplish this goal are Python 3.9.6, PyCharm 2021.1.3 for creating runtime and coding,Command Prompt for running commands and Google Chrome web browser. The full source code containing all the steps for deployment is available

for review in the following link: **https://github.com/Nkululeko353/Flask-deployment**

Data Understanding

It forms a crucial aspect to understand a dataset in terms of the features available in a dataset.

Here I will be working on an **Advertising.csv** data set and will deploy a **Linear Regression Model.**

What are the features?

TV: advertising dollars spent on TV for a single product in each market (in thousands of dollars).

Radio: advertising dollars spent on Radio

Newspaper: advertising dollars spent on Newspaper

What is the response?

Sales: Sales of a single product in each market (in thousands of items)

What else do we know?

Because the response variable is continuous, this is a regression problem.

There are 200 observations (represented by the rows), and each observation is a single market.

Now that the dataset is understood, now we can proceed to build and deploy our model(Linear Regression Model).Below is a sequence of steps required to deploy a model using Flask.

Model Deployment Using Flask

Steps:

- The first thing that is required is to create a new py file, rename it as **model.py** on **PyCharm** and insert the following code and run or debug it.

model.py file

```
            # Importing the libraries

import pandas as pd
import pickle

data = pd.read_csv('Advertising.csv')
data

data.shape
data.describe()


# What are the features?

# TV: advertising dollars spent on TV for a single product in a given market
#     (in thousands of dollars)
# Radio: advertising dollars spent on Radio
# Newspaper: advertising dollars spent on Newspaper

# What is the response?

# Sales: sales of a single product in a given market (in thousands of items)

# What else do we know?

# Because the response variable is continuous, this is a regression problem.
# There are 200 observations (represented by the rows), and each observation
is a single market.


# create a Python list of feature names
```

```
feature_cols = ['TV', 'radio', 'newspaper']

# use the list to select a subset of the original DataFrame
X = data[feature_cols]

# select a Series from the DataFrame
y = data['sales']

# We have to drop the variable Unnamed:0
data.drop(['Unnamed: 0'], axis=1)


#Splitting Training and Test Set
#Since we have a very small dataset, we will train our model with all
availabe data.

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

#Fitting model with trainig data
regressor.fit(X, y)

# Saving model to disk
pickle.dump(regressor, open('lr_model.pkl','wb'))

# Loading model to compare the results
model = pickle.load(open('lr_model.pkl','rb'))
print(model.predict([[2, 9, 6]]))
```

After we've run or debugged the code, we must ensure that the **lr_model.pkl** file is saved on the project directory.Therefore we can just copy the **lr_model.pkl** file from the disk and paste it to the project directory.

- The next thing is to create an **app1.py** file and insert the following code:

app1.py file

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('lr_model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Sales should be $
{}'.format(output))


@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=True)
```

- We need a page in which we a required to enter the values for **tv**,**radio** and **newspaper** in the textboxes to predict sales and therefore we need to create a new folder and rename it as **templates.**In the templates folder I will insert a new HTML file renamed as **index.html** with the following code inserted:

index.html file

```
            <!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css')
}}">

</head>

<body>
 <div class="login">
   <h1>Predict Sales Analysis</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
       <input type="text" name="TV" placeholder="TV Cost" required="required"
/>
        <input type="text" name="radio" placeholder="Radio Cost"
required="required" />
        <input type="text" name="newspaper" placeholder="Newspaper Cost"
required="required" />
        <button type="submit" class="btn btn-primary btn-block btn-
large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>


</body>
</html>
```

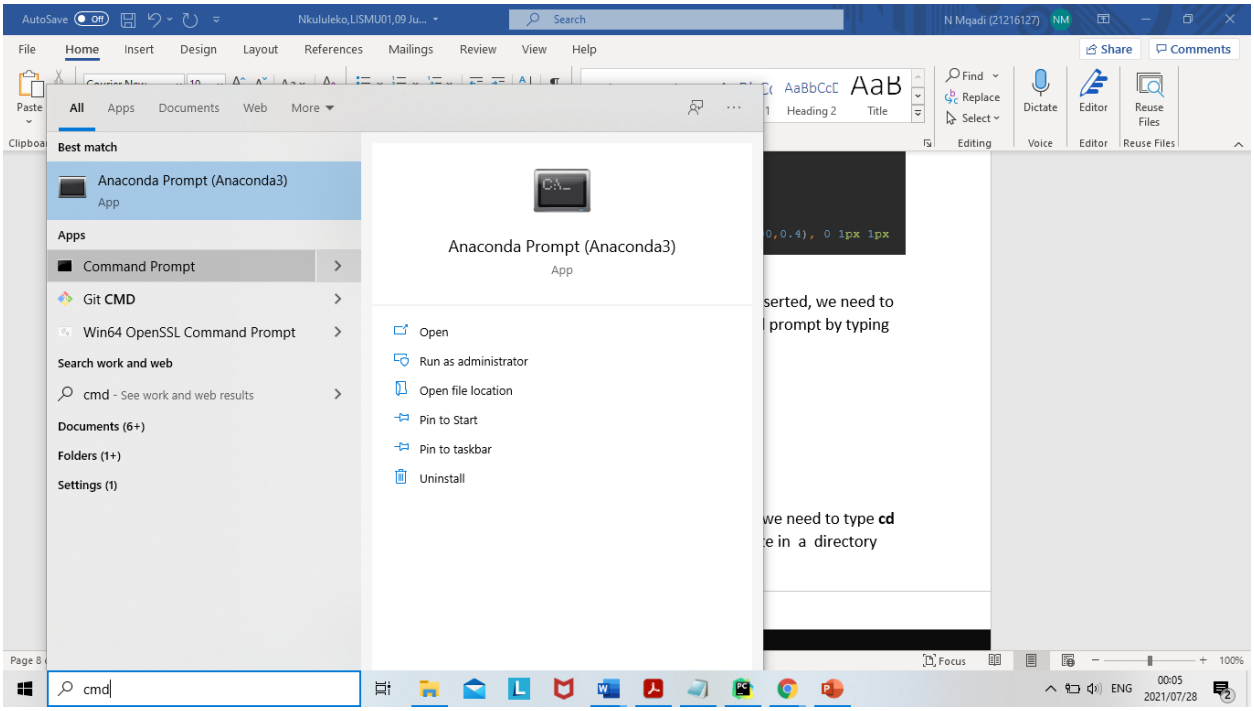
- To make an index.html file look attractive we can insert css files.Therefore we need to create a new folder and renamed it as **static**,inside a folder we need to create a new subfolder and renamed it as **css** and inside the subfolder we need to insert a css file and rename is as **style.css** and insert the following code:

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px
4px; margin-bottom: 0; font-size: 13px; line-height: 18px; color: #333333;
text-align: center;text-shadow: 0 1px 1px rgba(255, 255, 255, 0.75);
vertical-align: middle; background-color: #f5f5f5; background-image: -moz-
linear-gradient(top, #ffffff, #e6e6e6); background-image: -ms-linear-
gradient(top, #ffffff, #e6e6e6); background-image: -webkit-gradient(linear, 0
0, 0 100%, from(#ffffff), to(#e6e6e6)); background-image: -webkit-linear-
gradient(top, #ffffff, #e6e6e6); background-image: -o-linear-gradient(top,
#ffffff, #e6e6e6); background-image: linear-gradient(top, #ffffff, #e6e6e6);
background-repeat: repeat-x; filter:
progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff,
endColorstr=#e6e6e6, GradientType=0); border-color: #e6e6e6 #e6e6e6 #e6e6e6;
border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
border: 1px solid #e6e6e6; -webkit-border-radius: 4px; -moz-border-radius:
4px; border-radius: 4px; -webkit-box-shadow: inset 0 1px 0 rgba(255, 255,
255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05); -moz-box-shadow: inset 0 1px 0
rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05); box-shadow: inset 0
1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05); cursor:
pointer; *margin-left: .3em; }
.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] {
background-color: #e6e6e6; }
.btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -
webkit-border-radius: 5px; -moz-border-radius: 5px; border-radius: 5px; }
.btn:hover { color: #333333; text-decoration: none; background-color:
#e6e6e6; background-position: 0 -15px; -webkit-transition: background-
position 0.1s linear; -moz-transition: background-position 0.1s linear; -ms-
transition: background-position 0.1s linear; -o-transition: background-
position 0.1s linear; transition: background-position 0.1s linear; }
.btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
color: #ffffff; }
.btn-primary.active { color: rgba(255, 255, 255, 0.75); }
.btn-primary { background-color: #4a77d4; background-image: -moz-linear-
gradient(top, #6eb6de, #4a77d4); background-image: -ms-linear-gradient(top,
#6eb6de, #4a77d4); background-image: -webkit-gradient(linear, 0 0, 0 100%,
from(#6eb6de), to(#4a77d4)); background-image: -webkit-linear-gradient(top,
#6eb6de, #4a77d4); background-image: -o-linear-gradient(top, #6eb6de,
#4a77d4); background-image: linear-gradient(top, #6eb6de, #4a77d4);
background-repeat: repeat-x; filter:
progid:dximagetransform.microsoft.gradient(startColorstr=#6eb6de,
endColorstr=#4a77d4, GradientType=0);  border: 1px solid #3762bc; text-
shadow: 1px 1px 1px rgba(0,0,0,0.4); box-shadow: inset 0 1px 0 rgba(255, 255,
255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.5); }
.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-
primary.disabled, .btn-primary[disabled] { filter: none; background-color:
#4a77d4; }
.btn-block { width: 100%; display:block; }

* { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-
sizing:border-box; -o-box-sizing:border-box; box-sizing:border-box; }

html { width: 100%; height:100%; overflow:hidden; }

body {
```

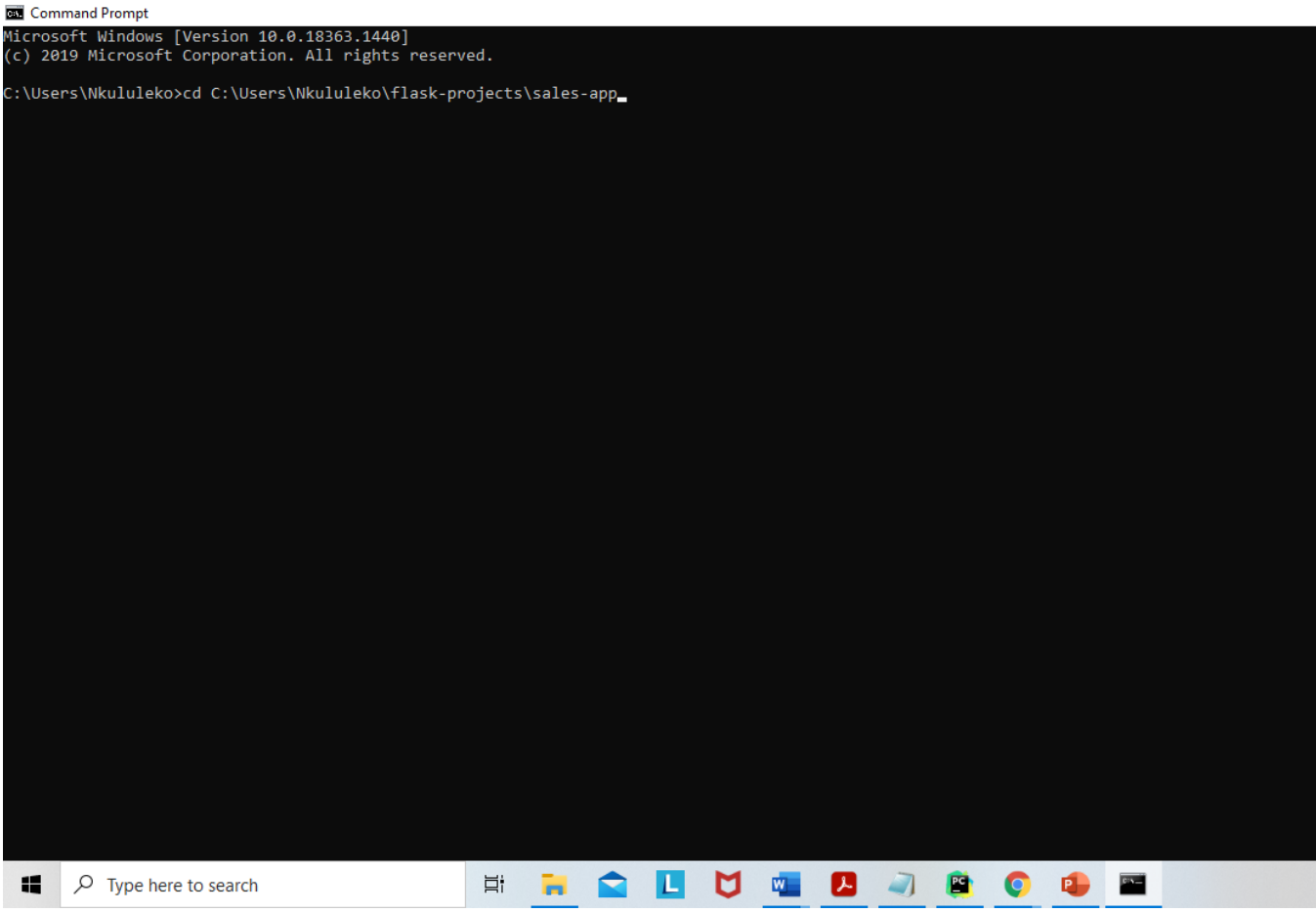
```
   width: 100%;
   height:100%;
   font-family: 'Open Sans', sans-serif;
   background: #092756;
   color: #fff;
   font-size: 18px;
   text-align:center;
   letter-spacing:1.2px;
   background: -moz-radial-gradient(0% 100%, ellipse cover,
rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%),-moz-linear-gradient(top,
rgba(57,173,219,.25) 0%, rgba(42,60,87,.4) 100%), -moz-linear-gradient(-
45deg,  #670d10 0%, #092756 100%);
   background: -webkit-radial-gradient(0% 100%, ellipse cover,
rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -webkit-linear-
gradient(top,  rgba(57,173,219,.25) 0%,rgba(42,60,87,.4) 100%), -webkit-
linear-gradient(-45deg,  #670d10 0%,#092756 100%);
   background: -o-radial-gradient(0% 100%, ellipse cover,
rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -o-linear-gradient(top,
rgba(57,173,219,.25) 0%,rgba(42,60,87,.4) 100%), -o-linear-gradient(-45deg,
#670d10 0%,#092756 100%);
   background: -ms-radial-gradient(0% 100%, ellipse cover,
rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -ms-linear-gradient(top,
rgba(57,173,219,.25) 0%,rgba(42,60,87,.4) 100%), -ms-linear-gradient(-45deg,
#670d10 0%,#092756 100%);
   background: -webkit-radial-gradient(0% 100%, ellipse cover,
rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), linear-gradient(to bottom,
rgba(57,173,219,.25) 0%,rgba(42,60,87,.4) 100%), linear-gradient(135deg,
#670d10 0%,#092756 100%);
   filter: progid:DXImageTransform.Microsoft.gradient(
startColorstr='#3E1D6D', endColorstr='#092756',GradientType=1 );

}
.login {
   position: absolute;
   top: 40%;
   left: 50%;
   margin: -150px 0 0 -150px;
   width:400px;
   height:400px;
}

.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-
spacing:1px; text-align:center; }

input {
   width: 100%;
   margin-bottom: 10px;
   background: rgba(0,0,0,0.3);
   border: none;
   outline: none;
   padding: 10px;
   font-size: 13px;
   color: #fff;
   text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
   border: 1px solid rgba(0,0,0,0.3);
   border-radius: 4px;
   box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px
```

```
rgba(255,255,255,0.2);
   -webkit-transition: box-shadow .5s ease;
   -moz-transition: box-shadow .5s ease;
   -o-transition: box-shadow .5s ease;
   -ms-transition: box-shadow .5s ease;
   transition: box-shadow .5s ease;
}
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px
rgba(255,255,255,0.2); }
```

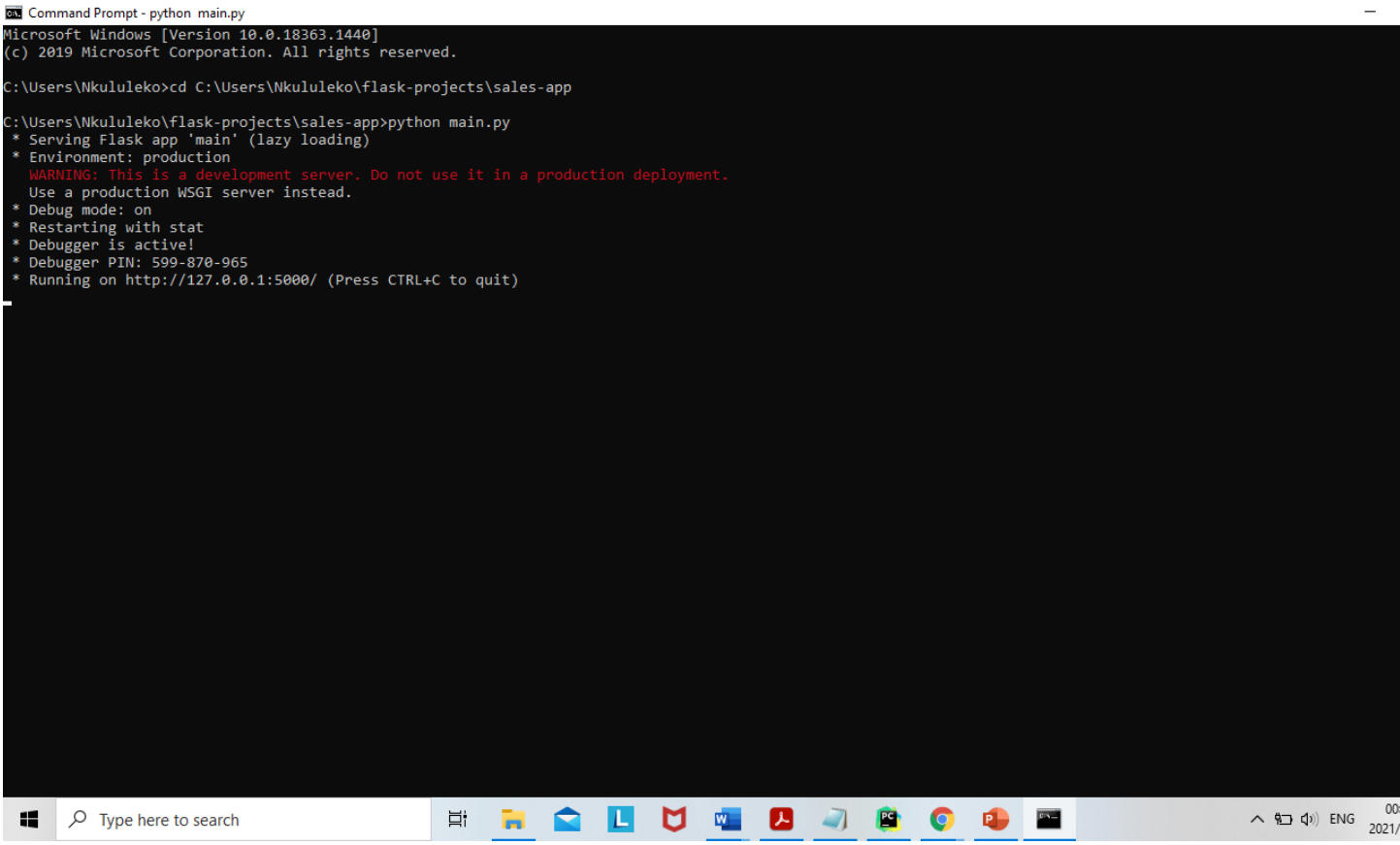
- Now that all the required files are created with codes inserted, we need to open the command prompt.We can open the command prompt by typing **cmd**.



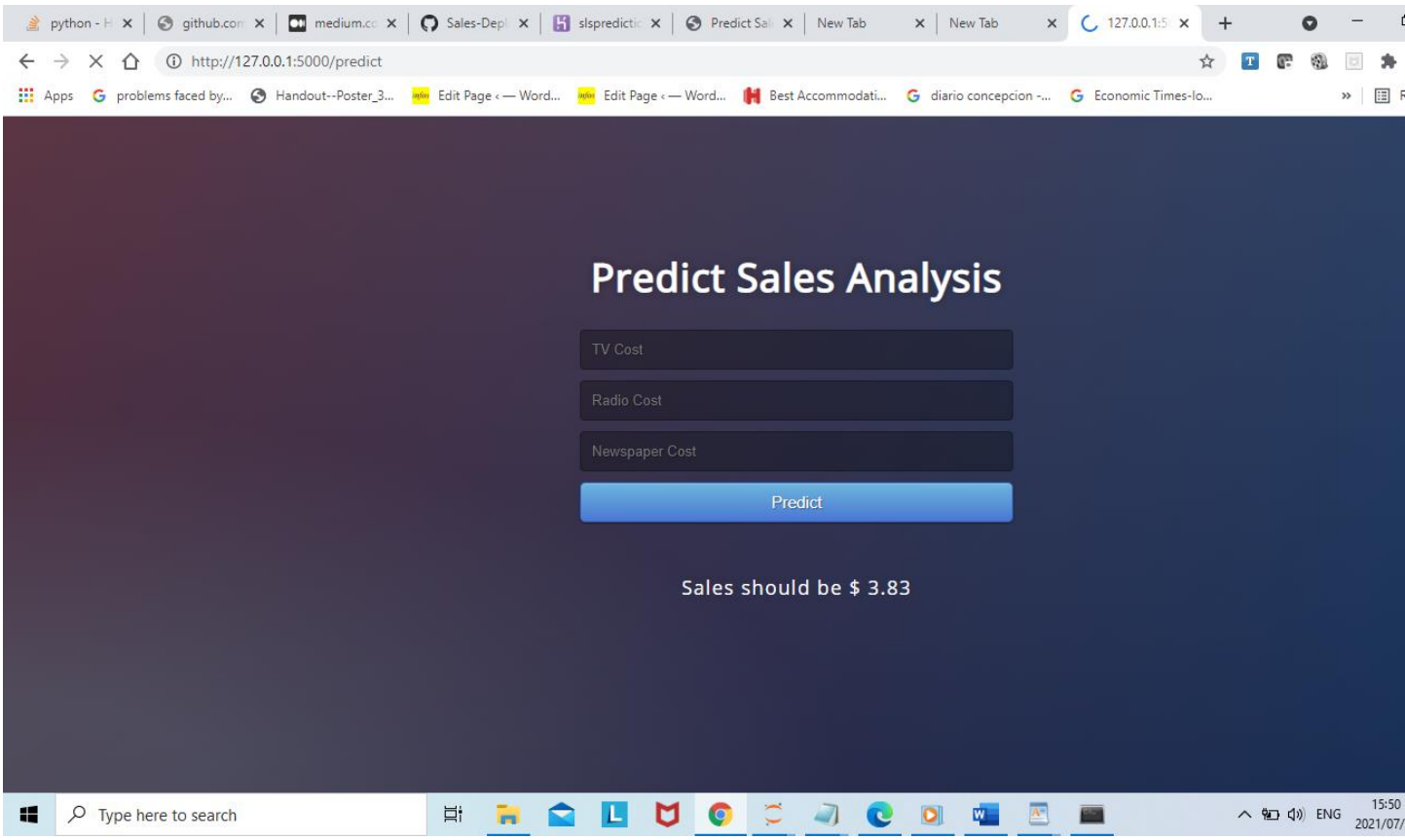
- After we typed cmd and opened the command prompt ,we need to type **cd C:\Users\Nkululeko\flask-projects\sales-app** to locate in a directory where all the project files are located.



- Then we type **python main.py** to run the flask app and then copy the link **http://127.0.0.1:5000/** and paste it in the browser.



- As we can see that the app is deployed using flask,we can now enter any values and press the Predict button to predict the sales value and the predicted sales value will be displayed.



We now have finshed with Deployment using Flask.

The full code is available for review in the following link:
https://github.com/Nkululeko353/Flask-deployment