



VIRTUAL INTERNSHIP

DATA SCIENCE

DEPLOYMENT OF FLASK

Marely Escalante Zárate

17 – march – 2021

Contents

I. INTRODUCTION	3
II. DATABASE MODEL.....	4
III. CREATING app.py.....	6
IV. CREATING index.html	7
V. CREATING style.css	8
VI. USING cmd.....	11
VII. USING Google Chrome.....	12
VIII. WARNING	13
IX. CONCLUSION	14

I. INTRODUCTION

This document contains the detailed report of all steps carried out in order to understand how to realize of Deployment on Flask.

I will use the IDE Visual Studio Code, cmd command interpreter, Google Collaboratory cloud service, Google Chrome web browser, html hypertext markup language, css graphic design language and python programming language as well as its different packages and librarys.

The hiring.csv database will be used which has some basic data about experience, test score, interview score and salary.

The following content will be found in this document: database model, creating app.py, index.html, style.css and using cmd and google chrome that will allow us to reach the objective will be defined.

II. DATABASE MODEL

In this segment, I will use Google Colaboratory, Python and dataset hiring.csv

1. Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle
```

2. Importing the dataset

```
dataset = pd.read_csv('/content/hiring.csv')
```

	experience	test_score(out of 10)	interview_score(out of 10)	salary(\$)
0	NaN	8.0	9	50000
1	NaN	8.0	6	45000
2	five	6.0	7	60000
3	two	10.0	10	65000
4	seven	9.0	6	70000
5	three	7.0	10	62000
6	ten	NaN	7	72000
7	eleven	7.0	8	80000

3. Cleaning the dataset

I don't need values in category type, so I change it to integer type and fill values NaN to number 0(zero).

```
fill=[0,0,5,2,7,3,10,11]          #this list to replace values on experience columns
for i in range(0,8):              #replacing to values in experience columns
    dataset['experience'][i]=fill[i]
dataset=dataset.fillna(0)          #replacing values NaN with 0 (zero)
```

	experience	test_score(out of 10)	interview_score(out of 10)	salary(\$)
0	0	8.0	9	50000
1	0	8.0	6	45000
2	5	6.0	7	60000
3	2	10.0	10	65000
4	7	9.0	6	70000
5	3	7.0	10	62000
6	10	0.0	7	72000
7	11	7.0	8	80000

4. Splitting the dataset into the Training set and Test set

```
x =dataset.iloc[:, :3]           #Train set
y =dataset.iloc[:, -1]          #Test sest
```

5. Fitting Simple Linear Regression

```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()      #linear regression
regressor.fit(x,y)                #creating the model
```

6. Importing to pickle file

```
pickle.dump(regressor, open('model.pkl','wb'))  #importing pickle file
```

I get three files from this segment: model.ipynb: , model.py and model.pkl (the extensions are: ipynb of Google Colaboratory, py of python and pkl of pickle respectivly)

III. CREATING app.py

In this segment, i will use Visual Studio Code, python and model.pkl

1. Importing libraries

```
import numpy as np
from flask import Flask, request, render_template
from sklearn.linear_model import LinearRegression
import pickle
```

2. Importing pickle file

First initialize the flask application and import model.pkl

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

3. Decorator home

Redirect API to homepage index.html

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

4. Decorator predict

Redirect API to predict result (salary)

```
@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='House price should be $ {}'.format(output))
```

5. Activate debug

```
if __name__ == "__main__":
    app.run(port=5000, debug=True)
```

IV. CREATING index.html

In this segment, I will use Visual Studio Code and html

Flask need to create a folder called templates where we save all html files

1. Using url_for to style.css file

- To doing reference a fonts urls
- url_for to call style.css file

```
<!DOCTYPE html>
<html>
  <!-- From 
```

2. Creating text_box to receving query of model.py

- url_for to call predict function of app.py
- prediction_text is the return of predict function

```
<body>
  <div class='login'>
    <h1>Predict Salary Analysis</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}" method="post">
      <input type="text" name="experience" placeholder="Experience" required="required"/>
      <input type="text" name="test_score" placeholder="Test Score" required="required"/>
      <input type="text" name="interview_score" placeholder="Interview Score" required="required"/>

      <button type="submit" class="btn btn-primary btn-block btn-Large">Predict</button>
    </form>

    <br>
    <br>
    {{ prediction_text }}

  </div>
</body>
</html>
```

V. CREATING style.css

In this segment, I will use Visual Studio Code and css.

Flask need to create a folder called static and into static create other folder called css where we save all css files

1. Bottom style

```
@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; }
```


2. Body style

```
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',Gradi-
entType=1 );
}
```

3. Login style

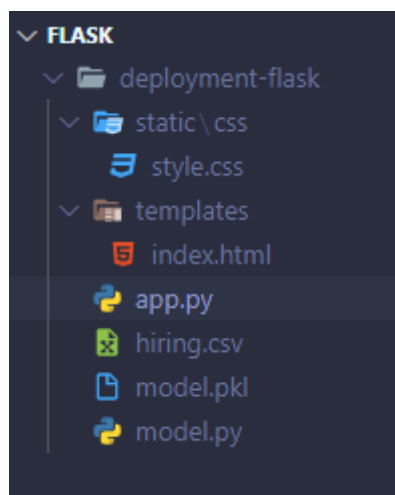
```
.login {
  position: absolute;
  top: 40%;
  left: 50%;
  margin: -150px 0 0 -150px;
  width: 400px;
  height: 400px;
}

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

4. Input style

```
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); }
```

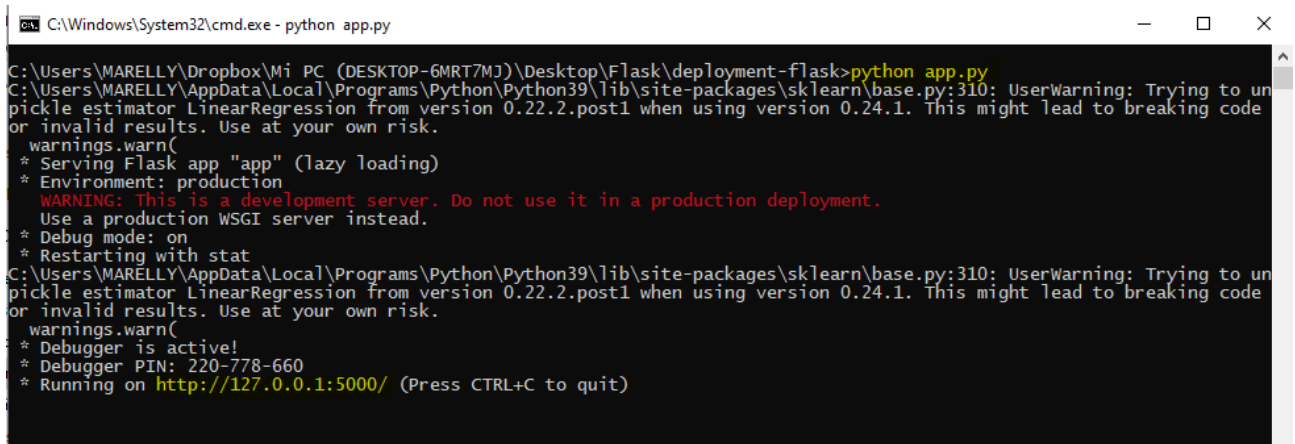
After to create app.py, index.html and style.css i have the file to this way:



VI. USING cmd

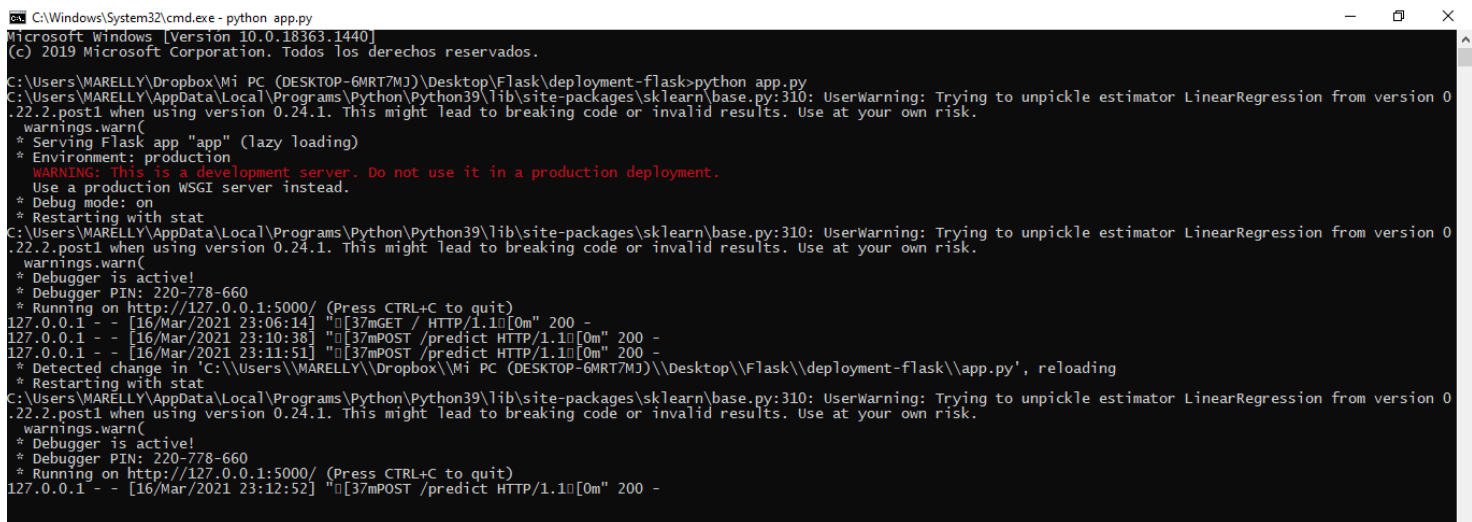
To this segment, I will use cmd of windows

- We need to localize into the directory, in my case the directory where I have my api is:
C:\Users\MARELLY\Dropbox\Mi PC (DESKTOP-6MRT7MJ)\Desktop\Flask\deployment-flask>
- Write python app.py to run our APP
- Finally, copy the url <http://127.0.0.1:5000/> into chrome without close the cmd.



```
C:\Windows\System32\cmd.exe - python app.py
C:\Users\MARELLY\Dropbox\Mi PC (DESKTOP-6MRT7MJ)\Desktop\Flask\deployment-flask>python app.py
C:\Users\MARELLY\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:310: UserWarning: Trying to unpickle estimator LinearRegression from version 0.22.2.post1 when using version 0.24.1. This might lead to breaking code or invalid results. Use at your own risk.
  warnings.warn(
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
C:\Users\MARELLY\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:310: UserWarning: Trying to unpickle estimator LinearRegression from version 0.22.2.post1 when using version 0.24.1. This might lead to breaking code or invalid results. Use at your own risk.
  warnings.warn(
* Debugger is active!
* Debugger PIN: 220-778-660
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

If I do some changes into the app.py and google chrome is registered on the cmd



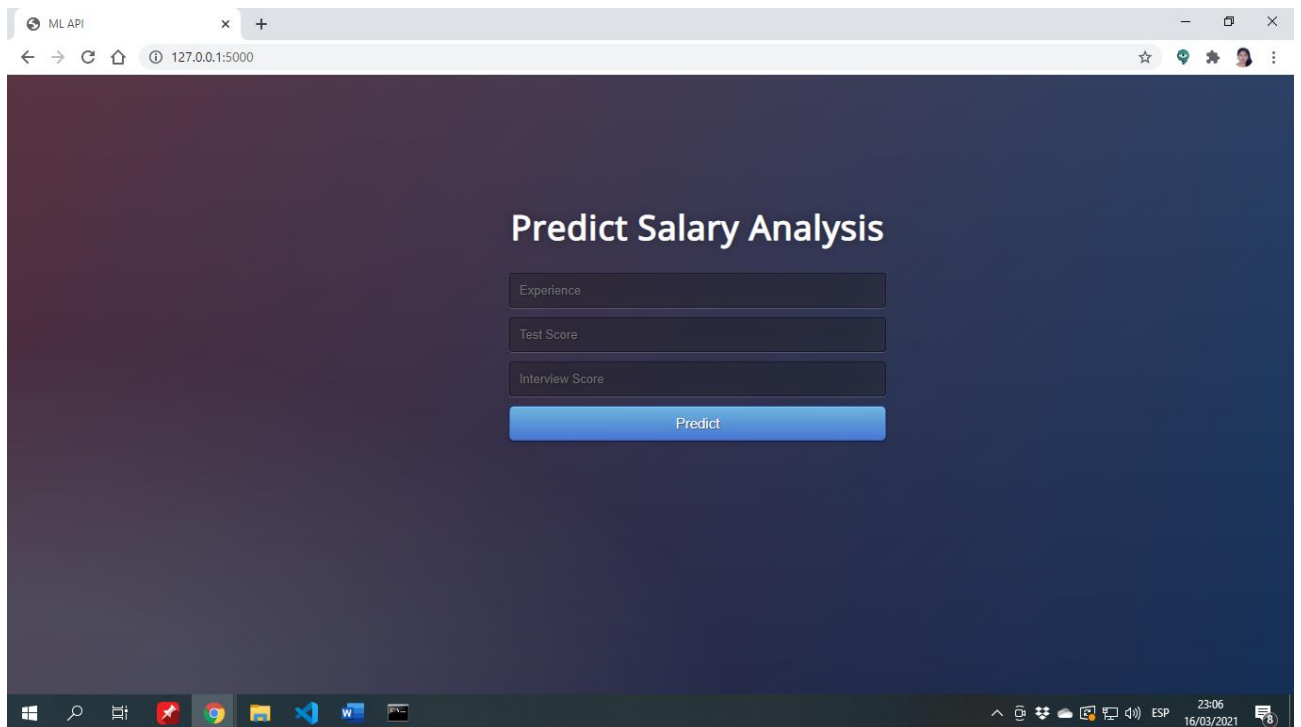
```
C:\Windows\System32\cmd.exe - python app.py
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\MARELLY\Dropbox\Mi PC (DESKTOP-6MRT7MJ)\Desktop\Flask\deployment-flask>python app.py
C:\Users\MARELLY\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:310: UserWarning: Trying to unpickle estimator LinearRegression from version 0.22.2.post1 when using version 0.24.1. This might lead to breaking code or invalid results. Use at your own risk.
  warnings.warn(
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
C:\Users\MARELLY\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:310: UserWarning: Trying to unpickle estimator LinearRegression from version 0.22.2.post1 when using version 0.24.1. This might lead to breaking code or invalid results. Use at your own risk.
  warnings.warn(
* Debugger is active!
* Debugger PIN: 220-778-660
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [16/Mar/2021 23:06:14] "[37mGET / HTTP/1.1[0m" 200 -
127.0.0.1 - - [16/Mar/2021 23:10:38] "[37mPOST /predict HTTP/1.1[0m" 200 -
127.0.0.1 - - [16/Mar/2021 23:11:51] "[37mPOST /predict HTTP/1.1[0m" 200 -
* Detected change in 'C:\Users\MARELLY\Dropbox\Mi PC (DESKTOP-6MRT7MJ)\Desktop\Flask\deployment-flask\app.py', reloading
* Restarting with stat
C:\Users\MARELLY\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\base.py:310: UserWarning: Trying to unpickle estimator LinearRegression from version 0.22.2.post1 when using version 0.24.1. This might lead to breaking code or invalid results. Use at your own risk.
  warnings.warn(
* Debugger is active!
* Debugger PIN: 220-778-660
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [16/Mar/2021 23:12:52] "[37mPOST /predict HTTP/1.1[0m" 200 -
```

VII. USING Google Chrome

In this segment, I will use Google Chrome

Copy the url <http://127.0.0.1:5000/> and we have this, now I can introduce the numbers that I want to predict.



I input:

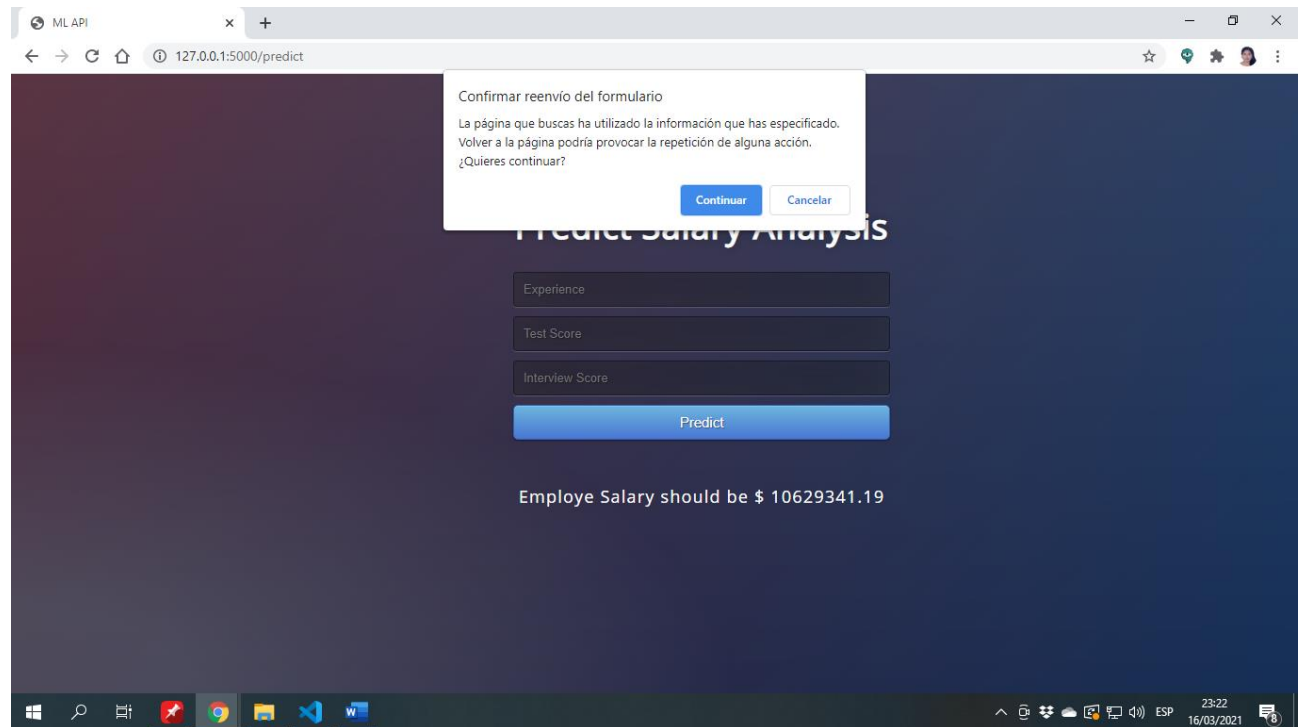
- Experience = 1
- Test Score = 9
- Interview Score = 500

A close-up view of the input fields in the 'Predict Salary Analysis' application. The fields are labeled 'Experience', 'Test Score', and 'Interview Score'. The values entered are '1', '9', and '5000' respectively. A blue 'Predict' button is at the bottom.A close-up view of the output result in the 'Predict Salary Analysis' application. The title 'Predict Salary Analysis' is at the top. Below the input fields is a blue 'Predict' button. At the bottom, the text reads 'Employee Salary should be \$ 10629341.19'.

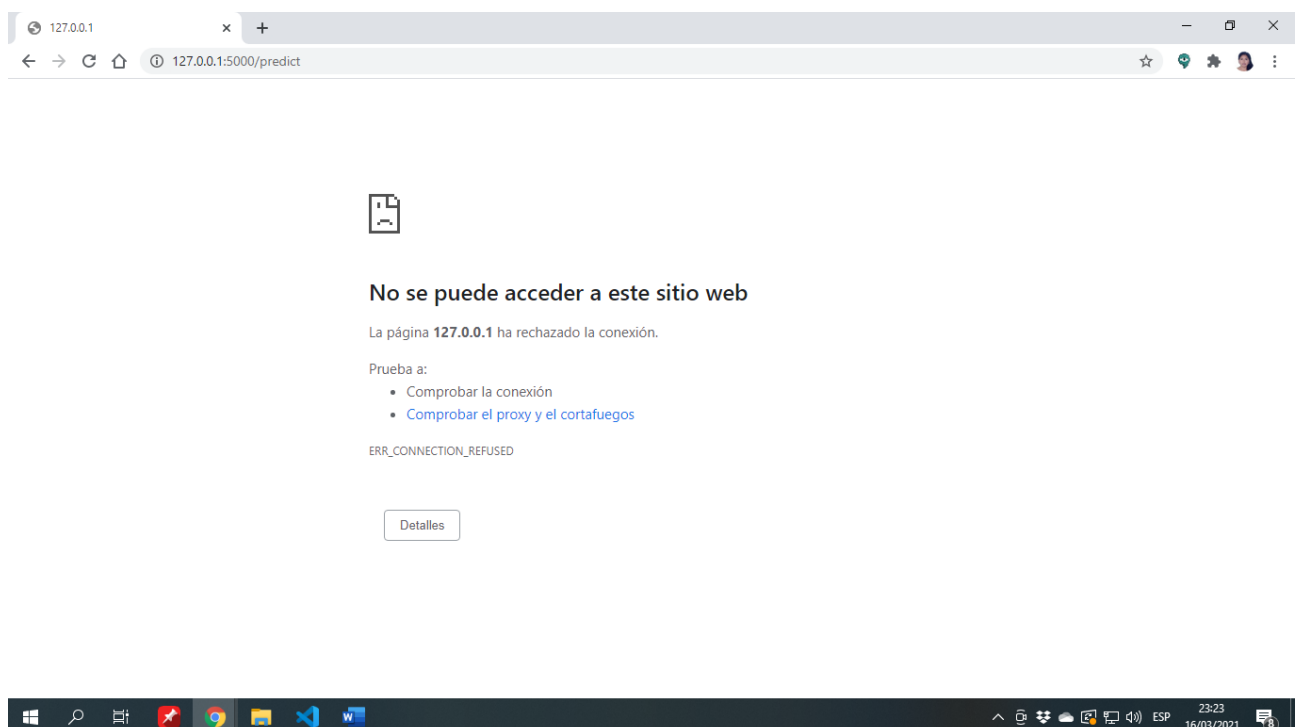
VIII. WARNING

If I close the cmd and i refresh, i load the page `http://127.0.0.1:5000/` again, I obtained this message.

Confirm form resubmission the page you are looking for has used the information you have specified. Returning to the page could cause a repeat action. Do you want to continue?



If I Confirm this message, this can't find the connection then have Can't access this website



IX. CONCLUSION

In this document i show step by step the simple way to do a small application web using flask, run it on a development server and allow the user to provide custom data through URL parameters and web forms.