Data Science Virtual Internship

Name:Nkululeko Freedom Mqadi

Batch Code: LISMU01

Submission date:28 July 2021

Submitted to:Data Glacier

Cloud and API deployment

Deployment of Machine learning model in Heroku using FLASK

Introduction

In this document I am going to explain a sequence of steps to deploy a machine learning model on Heroku. 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.

However there are 2 alternatives for doing an API Deployment on Heroku: **API Deployment on Heroku by GitHub** and **API Deployment on Heroku by Heroku GIT**. Therefore, I will do a demonstration explaining all the steps to deploy.

The first requirement is to deploy the model using Flask first. Therefore, I will first begin with the initial steps required to deploy a model using Flask before we deploy the model in Heroku.The full source code containing all the steps for deployment is available for review in the following link: https://github.com/Nkululeko353/Heroku-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 I will first begin with a sequence of steps 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]]))
```

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.

After we have created and saved the model,the next thing is to create an **main.py** file on PyCharm and insert the following code:

main.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>
```

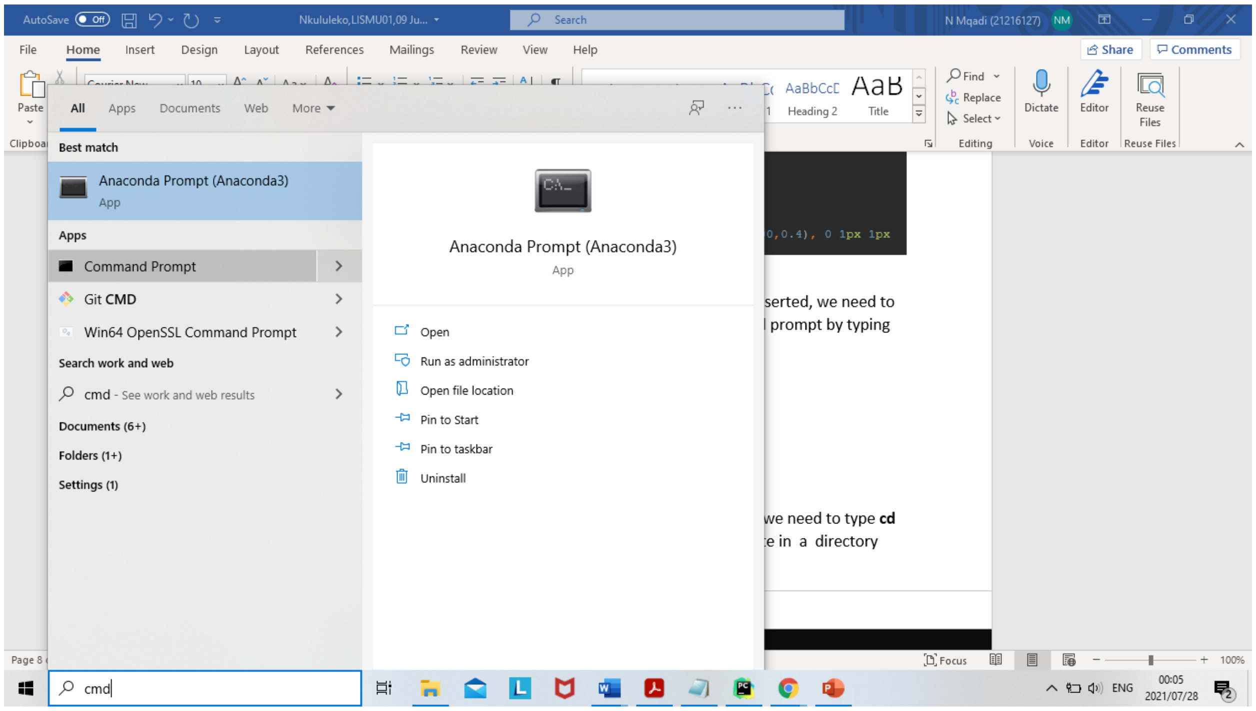
- To make an index.html file look attractive we can insert css files. Therefore, we need to create a new folder and renamed it as **static**,inside a folder we need to create a new subfolder and renamed it as **css** and inside the subfolder we need to insert a css file and rename is as **style.css**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

```
              @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; }
```

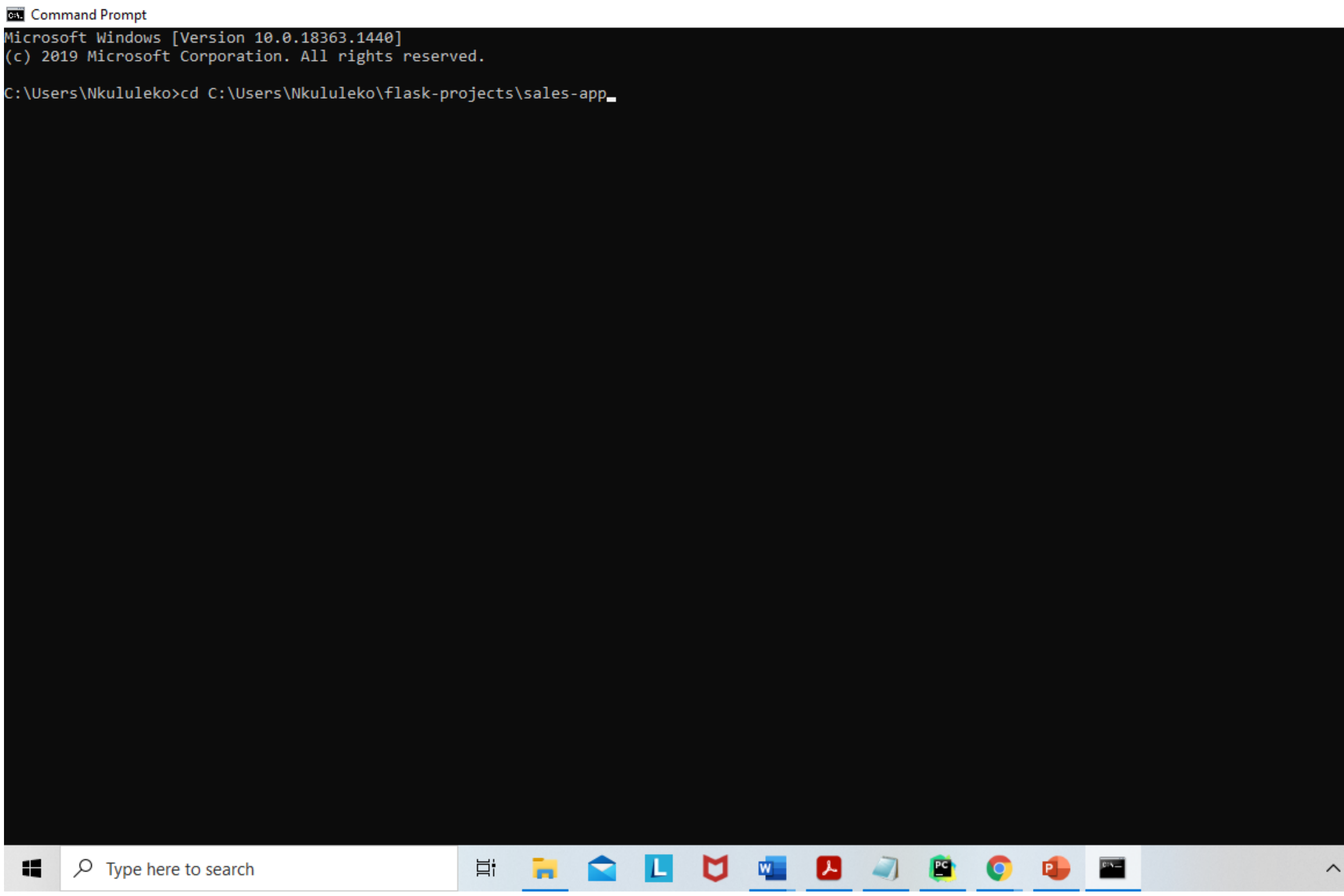
```
.btn-block { width: 100%; display:block; }

* { -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**.

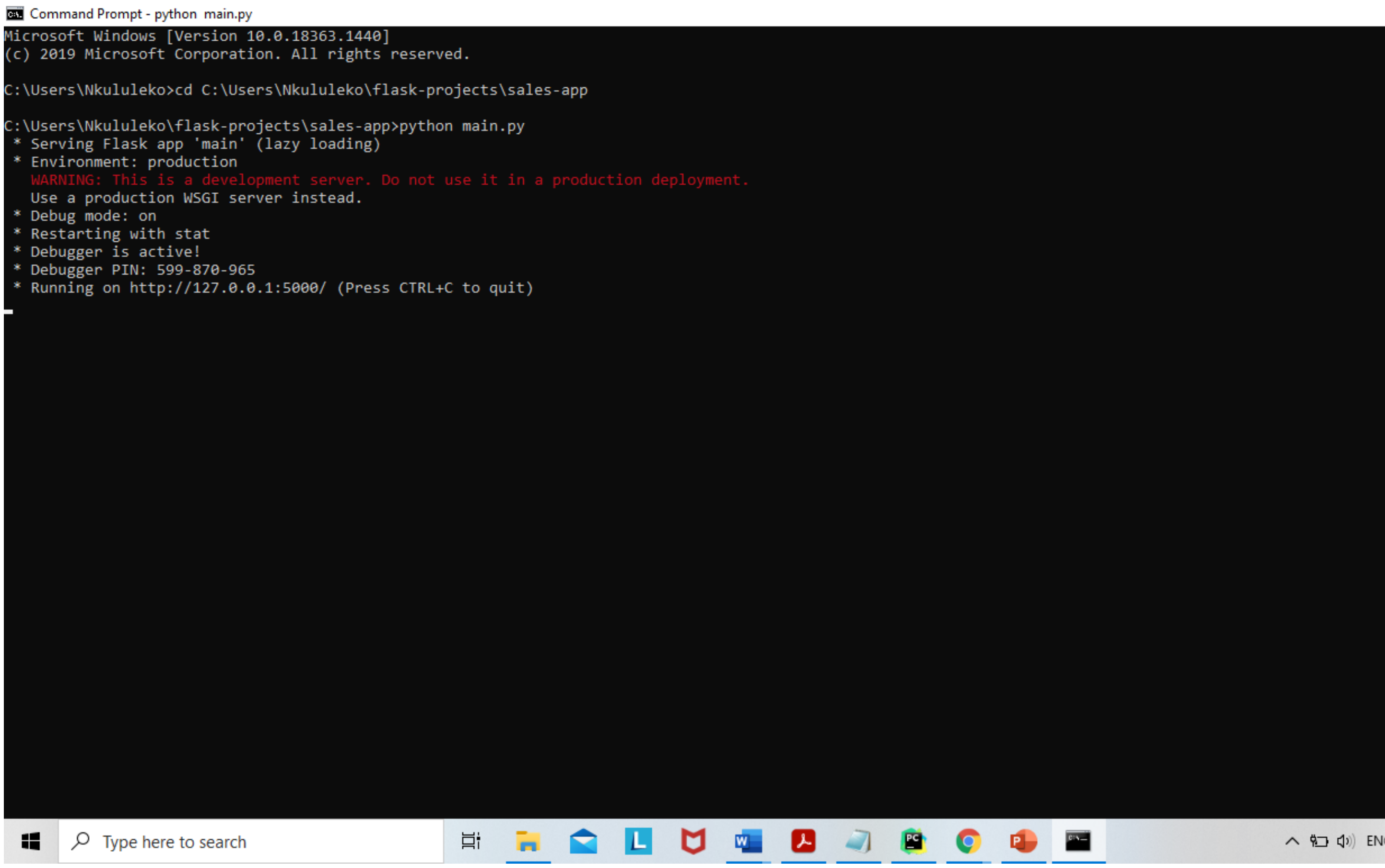


- After we typed **cmd** and opened the command prompt we need to type **cd C:\Users\Nkululeko\flask-projects\sales-app** to locate in a directory where all the project files are.



```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Nkululeko>cd C:\Users\Nkululeko\flask-projects\sales-app
```

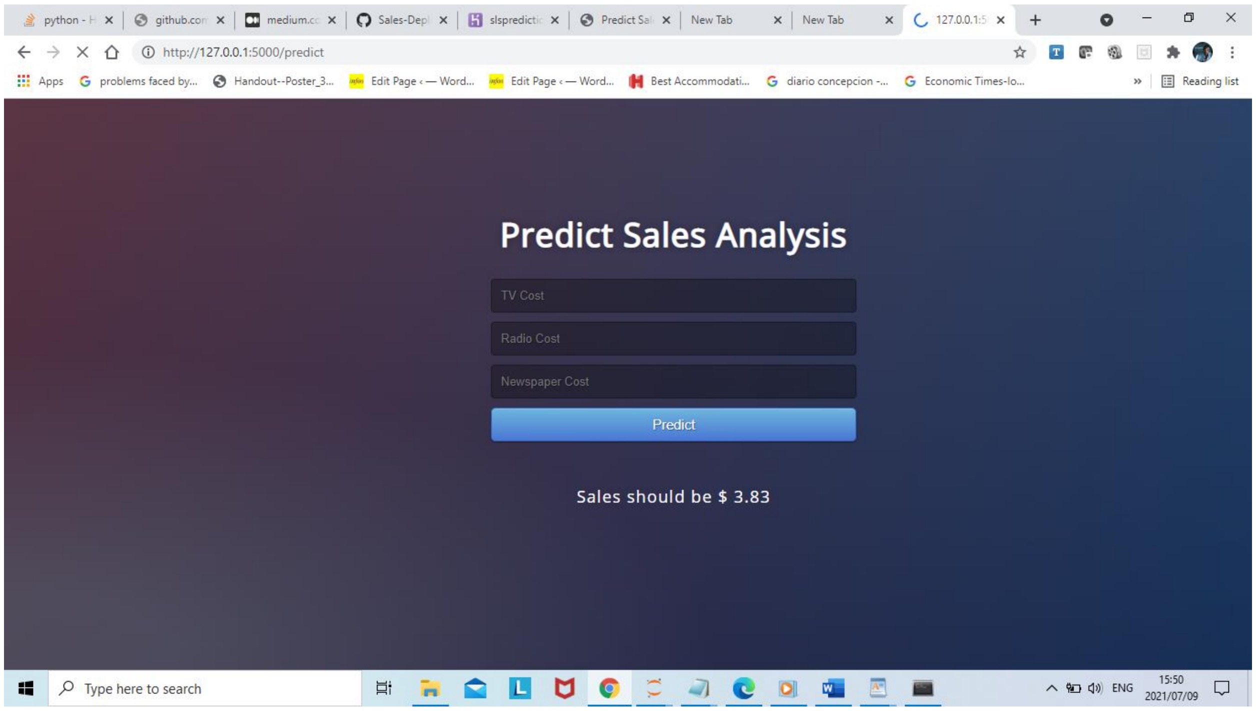
- Then we type **python main.py** to run the flask app and then copy the link http://127.0.0.1:5000/ and paste it in the browser.



Command Prompt - python main.py

```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Nkululeko>cd C:\Users\Nkululeko\flask-projects\sales-app

C:\Users\Nkululeko\flask-projects\sales-app>python main.py
 * Serving Flask app 'main' (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
 * Debugger is active!
 * Debugger PIN: 599-870-965
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- 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.



Sales should be $ 3.83

Now we've finished with the model deployment using flask, next is the **API deployment on Heroku.**

API deployment on Heroku

There can be 2 alternatives for API deployment on Heroku and are as follows:

- **API Deployment on Heroku by GitHub.**
- **API Deployment on Heroku by Heroku GIT.**

Before we can do an API deployment on **Heroku**, we must ensure that the following files are inserted in the project directory: **Procfile,requirements.txt** and **runtime.txt files.**

- The **Procfile** is always a simple text file that is named Procfile without a file extension in the root directory of the project, to explicitly declare what command should be executed to start your app. A Heroku app's web process type is special: it's the only process type that can receive external HTTP traffic from Heroku's routers. The first app refers to the filename app.py. The second app refers the instance of Flask which is inside app.py file.

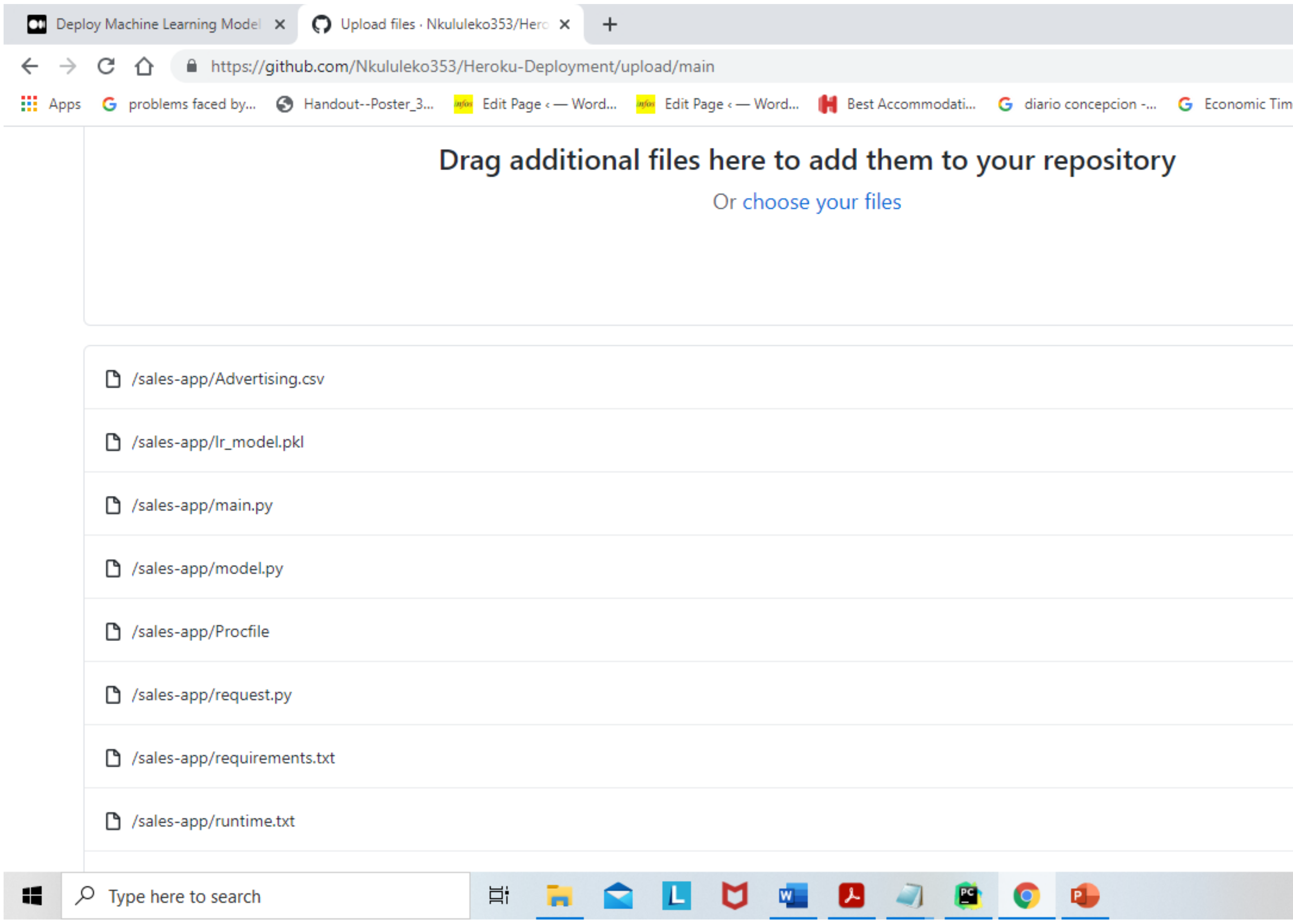
- The **requirements.txt** file lists all the app dependencies together. When an app is deployed, Heroku reads this file and installs the appropriate Python dependencies using the **pip install -r** command. We can run this with the following command**:**
pip freeze > requirements.txt.
- The **runtime.txt** file is used to specify the runtime version for deployment.

API Deployment on Heroku by GitHub

Steps:

I will explain a sequence of steps for **API Deployment on Heroku by GitHub.**

- I will start by uploading all the files required for deployment to GitHub and commit changes.



Drag additional files here to add them to your repository

Or choose your files

/sales-app/Advertising.csv

/sales-app/lr_model.pkl

/sales-app/main.py

/sales-app/model.py

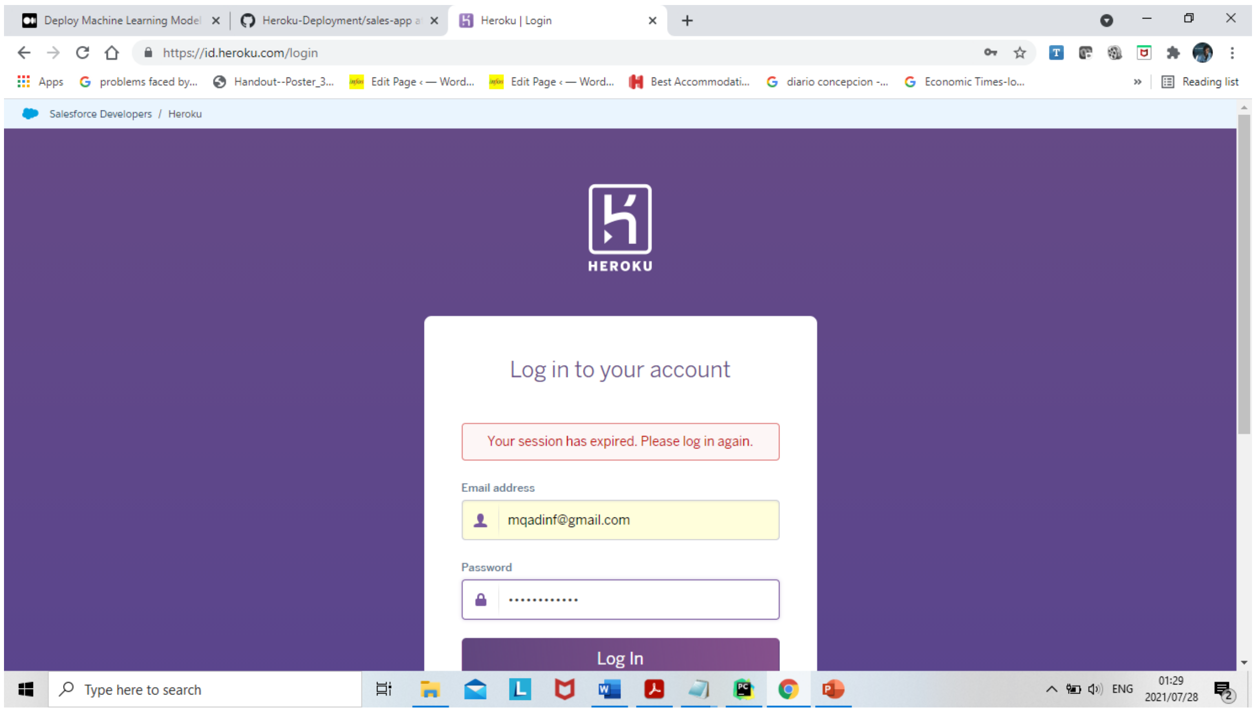
/sales-app/Procfile

/sales-app/request.py

/sales-app/requirements.txt

/sales-app/runtime.txt

- Then I will log in to Heroku by entering the login credentials.

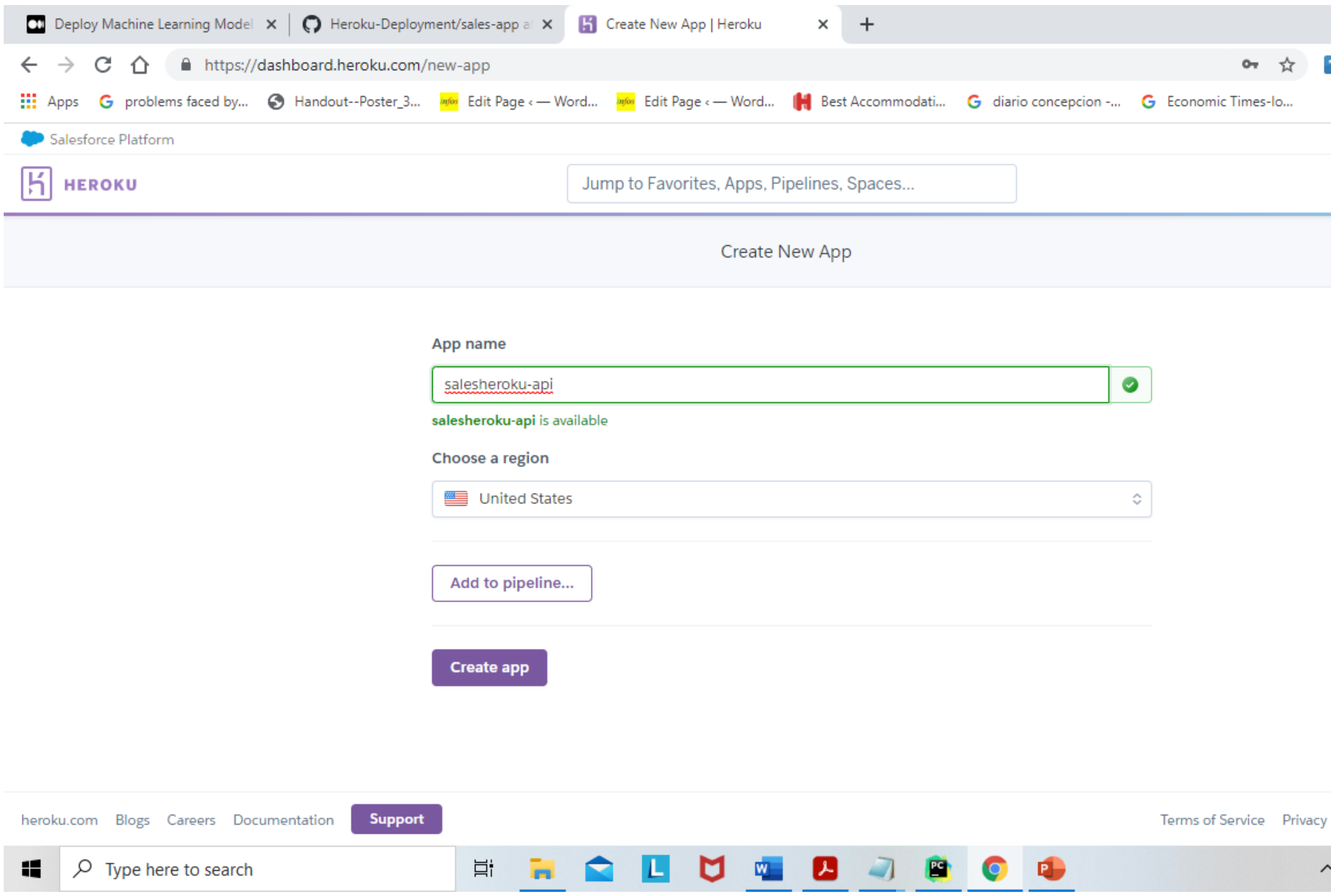

Log in to your account

Your session has expired. Please log in again.

Email address: mqadinf@gmail.com

Password: ••••••••••••

- I will create a new app by clicking the **Create New App** button. The app name is renamed as **salesheroku-api** as shown below.



Create New App

App name

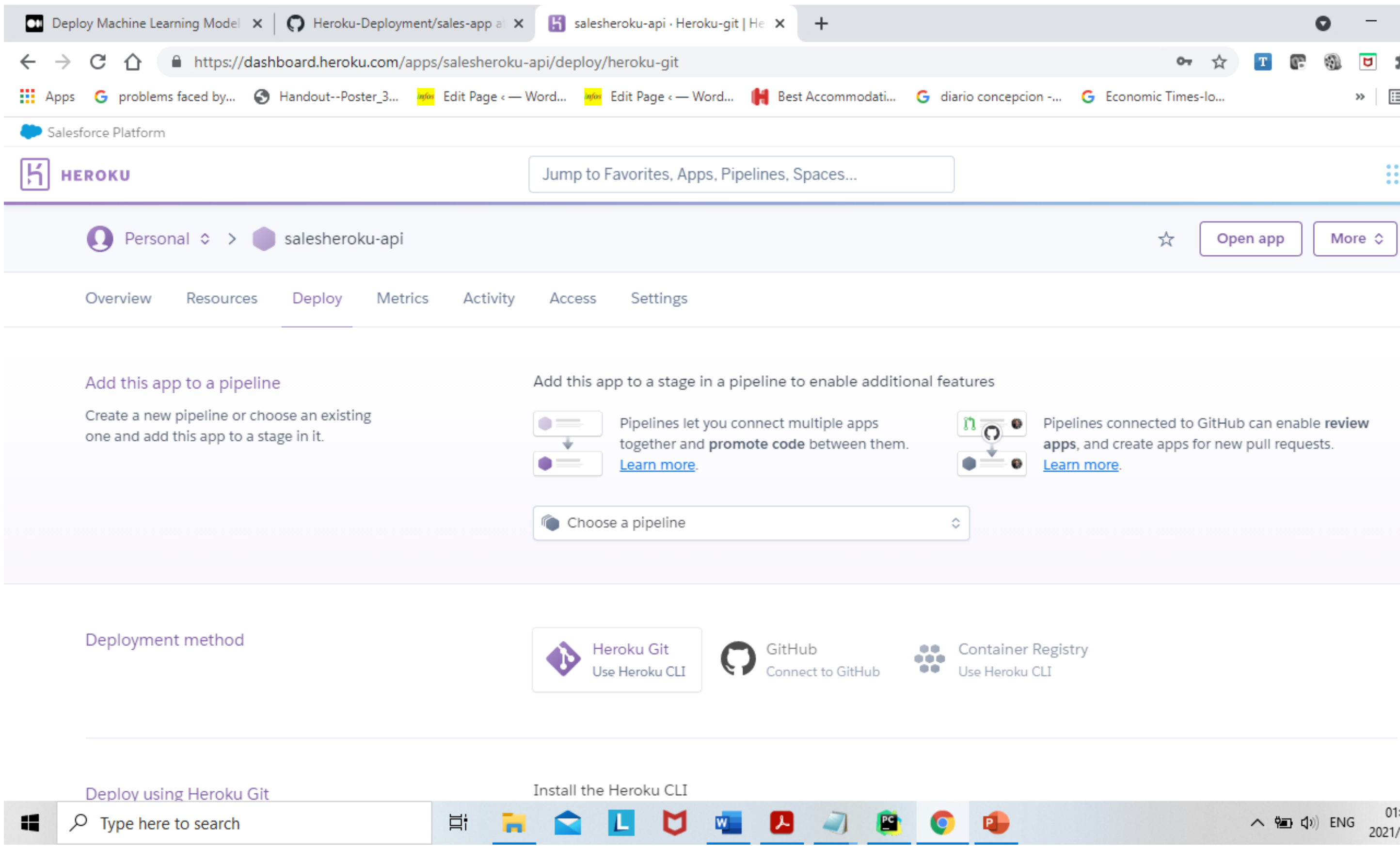
salesheroku-api

salesheroku-api is available

Choose a region

United States

- In Heroku I will click and select **Connect to GitHub.**



Deploy Machine Learning Model
Heroku-Deployment/sales-app
salesheroku-api · Heroku-git | He
https://dashboard.heroku.com/apps/salesheroku-api/deploy/heroku-git

Personal > salesheroku-api

Overview Resources Deploy Metrics Activity Access Settings

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

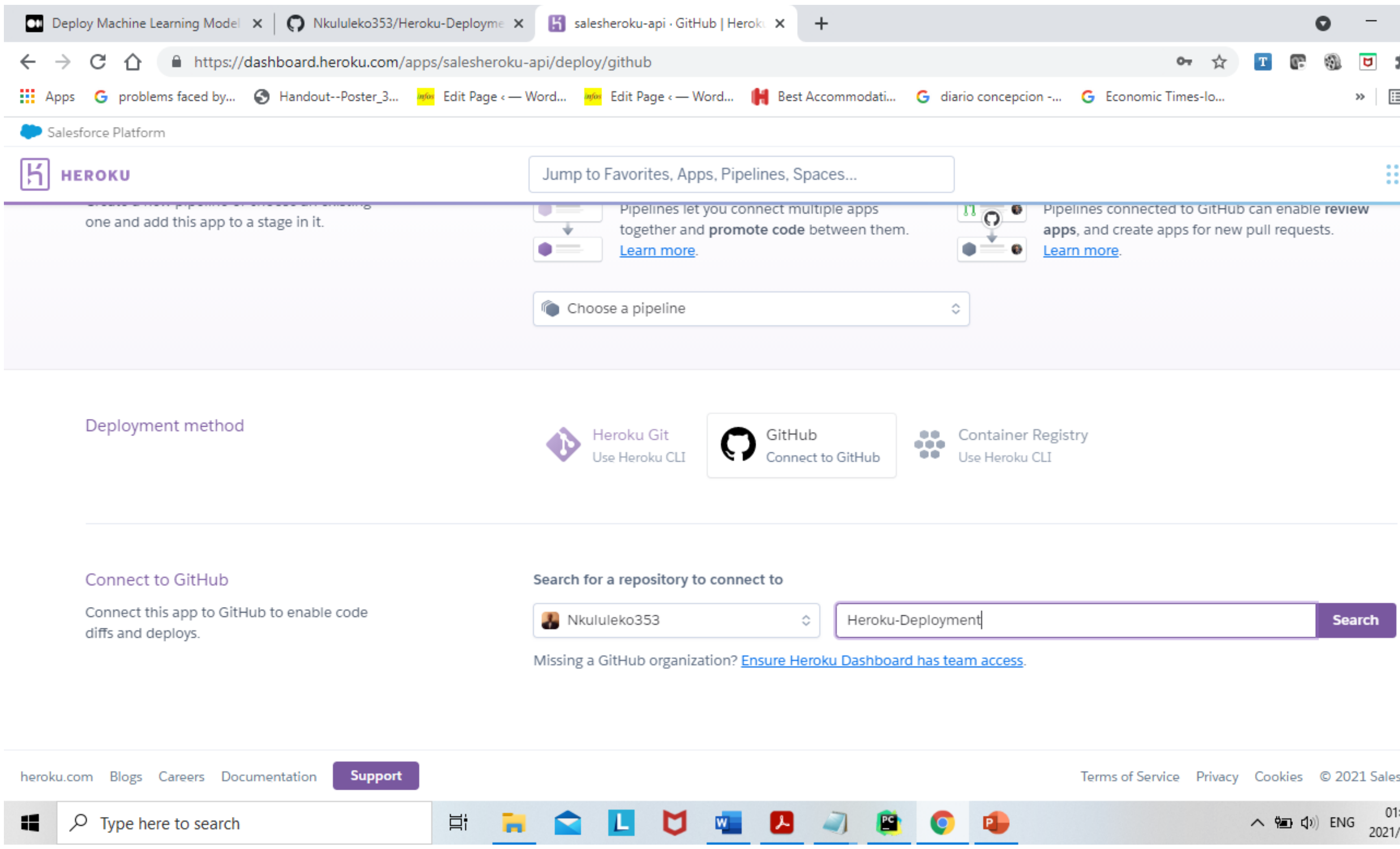
Deployment method

Heroku Git — Use Heroku CLI
GitHub — Connect to GitHub
Container Registry — Use Heroku CLI

Deploy using Heroku Git

Install the Heroku CLI

- I will then write repo-name and search.



https://dashboard.heroku.com/apps/salesheroku-api/deploy/github

one and add this app to a stage in it.

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

Deployment method

Heroku Git — Use Heroku CLI

GitHub — Connect to GitHub

Container Registry — Use Heroku CLI

Connect to GitHub

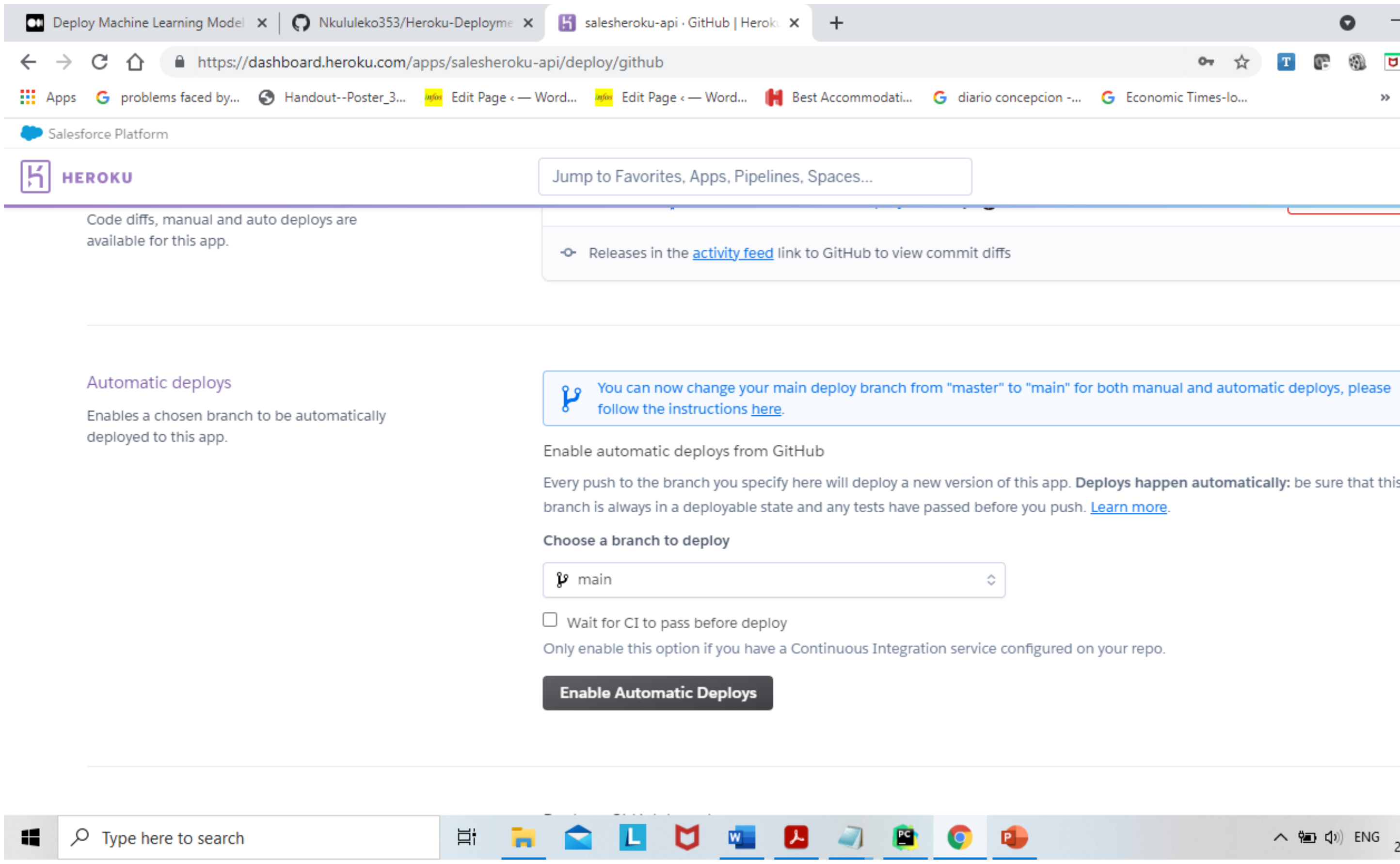
Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

Nkululeko353 | Heroku-Deployment | Search

Missing a GitHub organization? Ensure Heroku Dashboard has team access.

- Click in **Enable Automatic Deploys.**


Code diffs, manual and auto deploys are available for this app.

Releases in the activity feed link to GitHub to view commit diffs

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

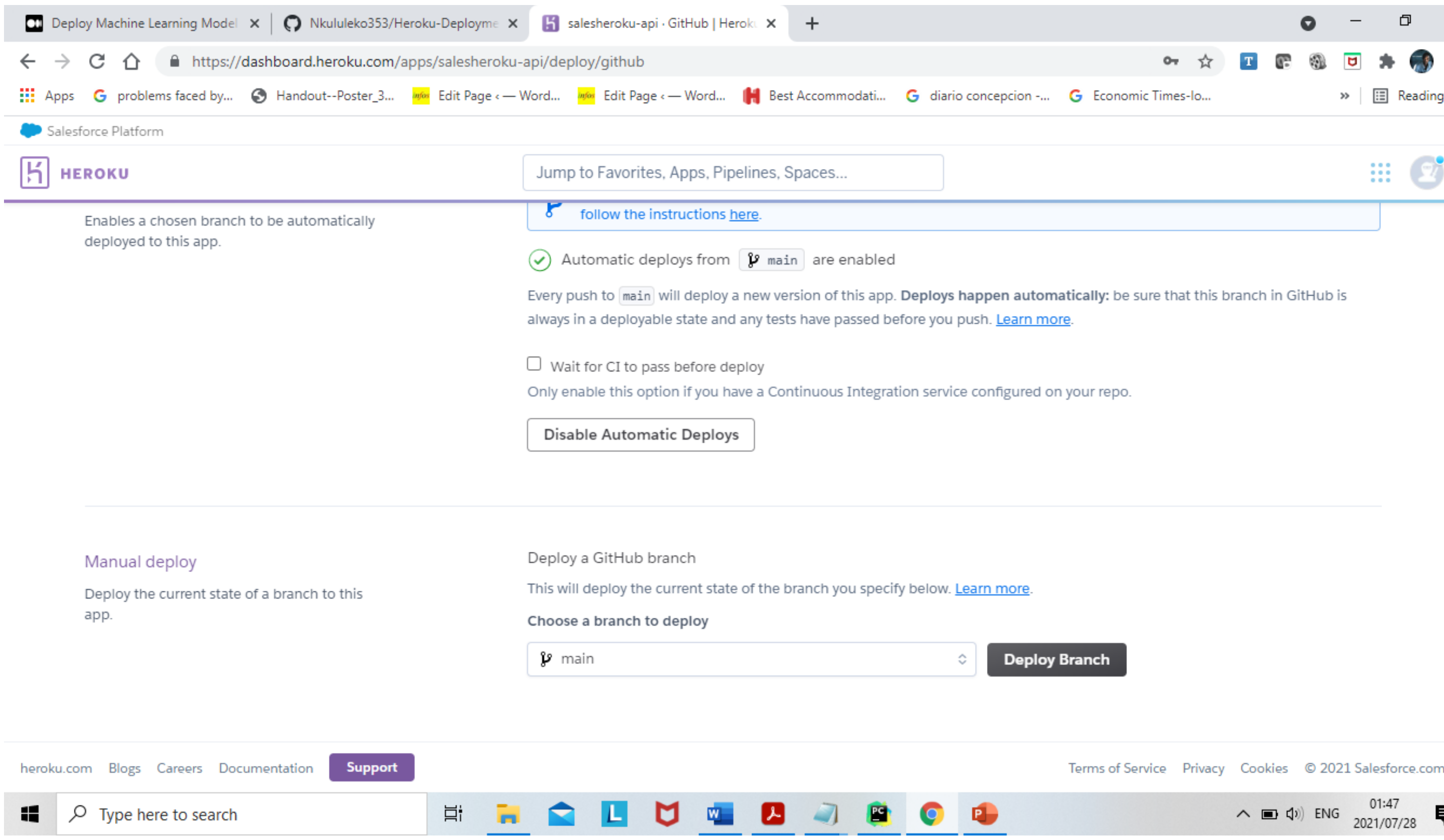
Choose a branch to deploy

main

Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

- Click in **Deploy Branch.**


Deploy Machine Learning Model · Nkululeko353/Heroku-Deployme · salesheroku-api · GitHub | Herok

https://dashboard.heroku.com/apps/salesheroku-api/deploy/github

Enables a chosen branch to be automatically deployed to this app.

follow the instructions here.

Automatic deploys from main are enabled

Every push to main will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. Learn more.

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Disable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

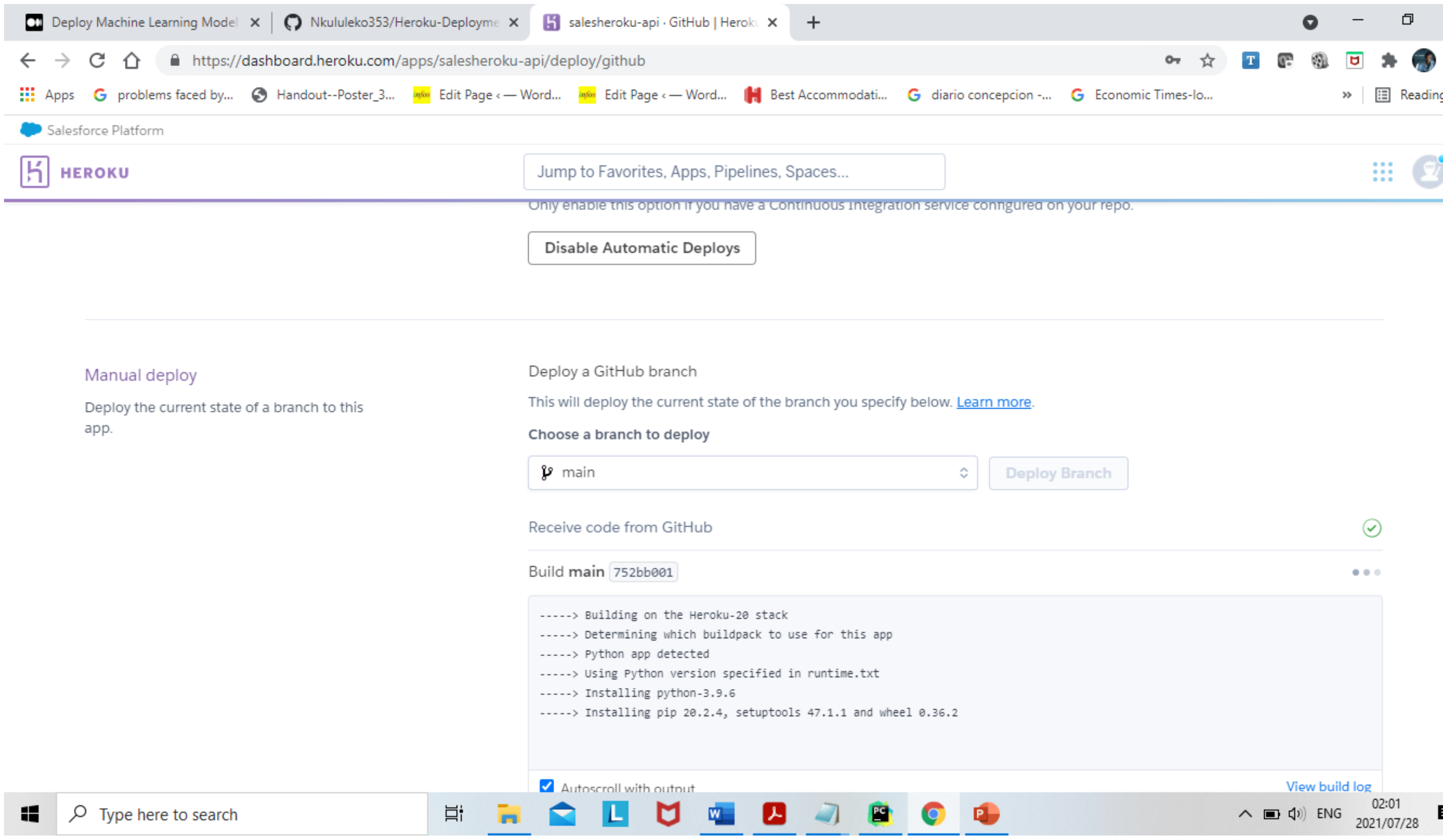
This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

main

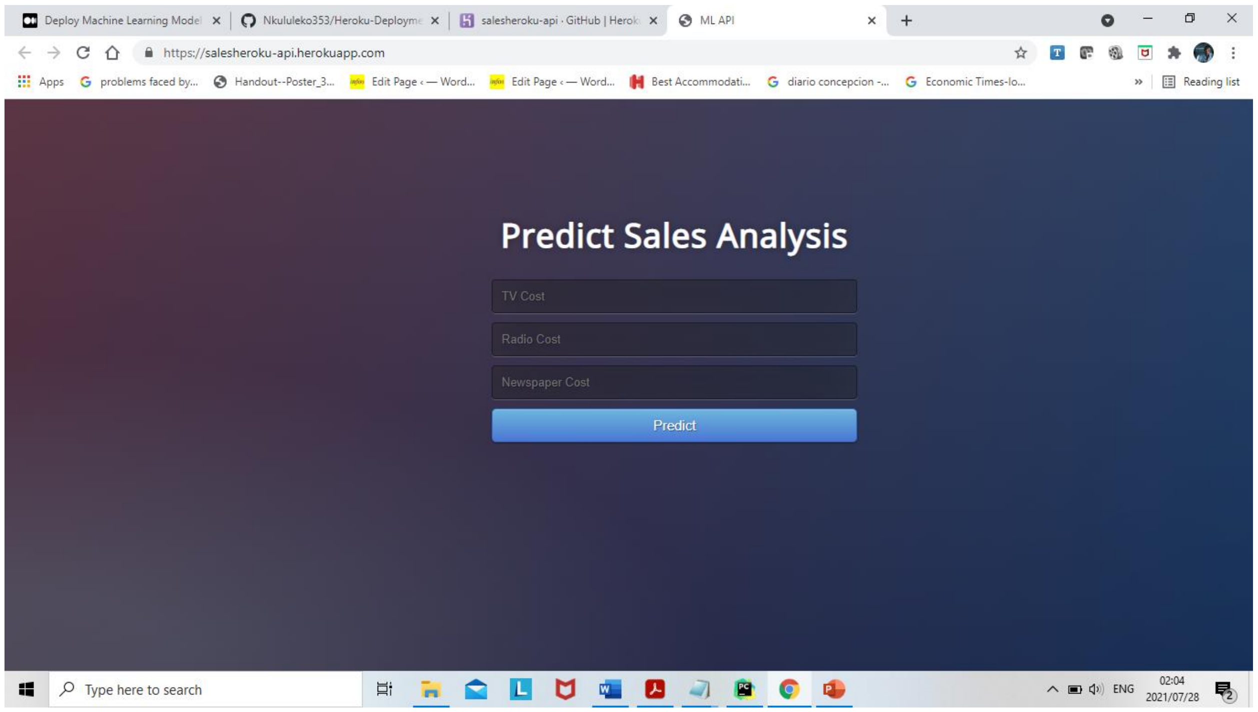
Deploy Branch

- Its then start to build.

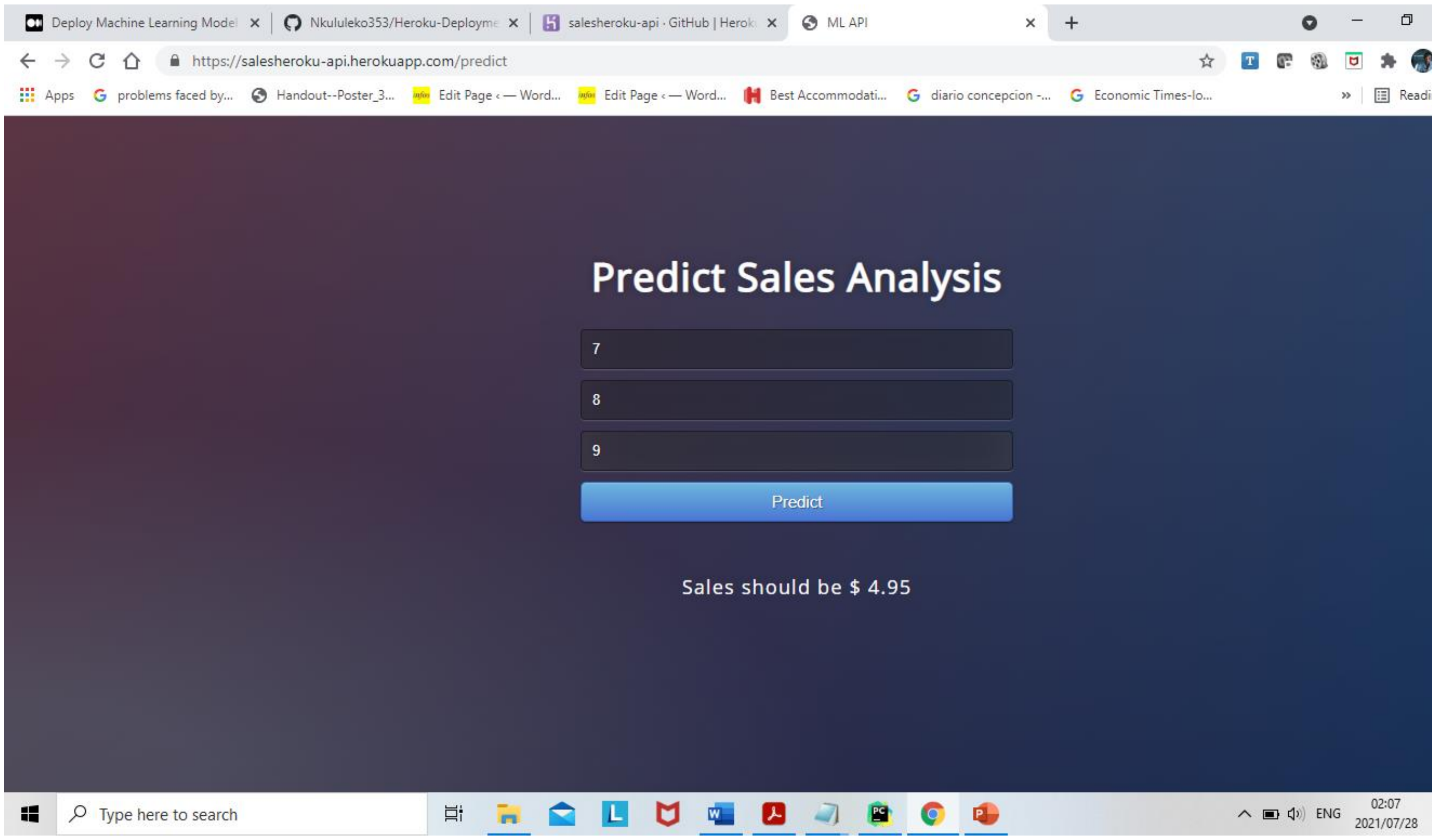

salesheroku-api · GitHub | Heroku
https://dashboard.heroku.com/apps/salesheroku-api/deploy/github
HEROKU
Only enable this option if you have a Continuous Integration service configured on your repo.
Disable Automatic Deploys
Manual deploy
Deploy the current state of a branch to this app.
Deploy a GitHub branch
This will deploy the current state of the branch you specify below. Learn more.
Choose a branch to deploy
main
Deploy Branch
Receive code from GitHub
Build main 752bb001
Autoscroll with output
View build log

```
-----> Building on the Heroku-20 stack
-----> Determining which buildpack to use for this app
-----> Python app detected
-----> Using Python version specified in runtime.txt
-----> Installing python-3.9.6
-----> Installing pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
```

- The web app of Predict Sales Analysis deployed with the following url: https://salesheroku-api.herokuapp.com

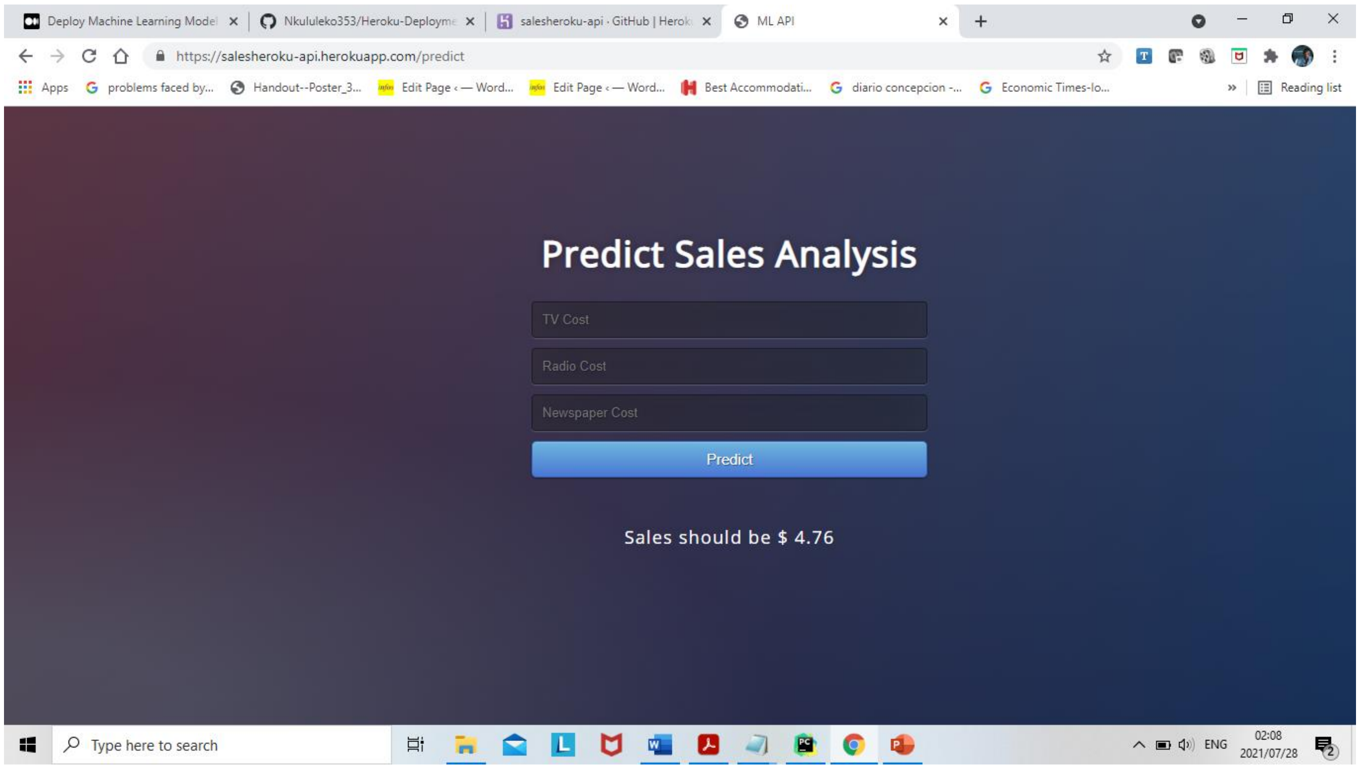


- We can test the web app functionality.



Sales should be $ 4.95

- Its working perfectly and able to predict, we can confirm this as shown below.



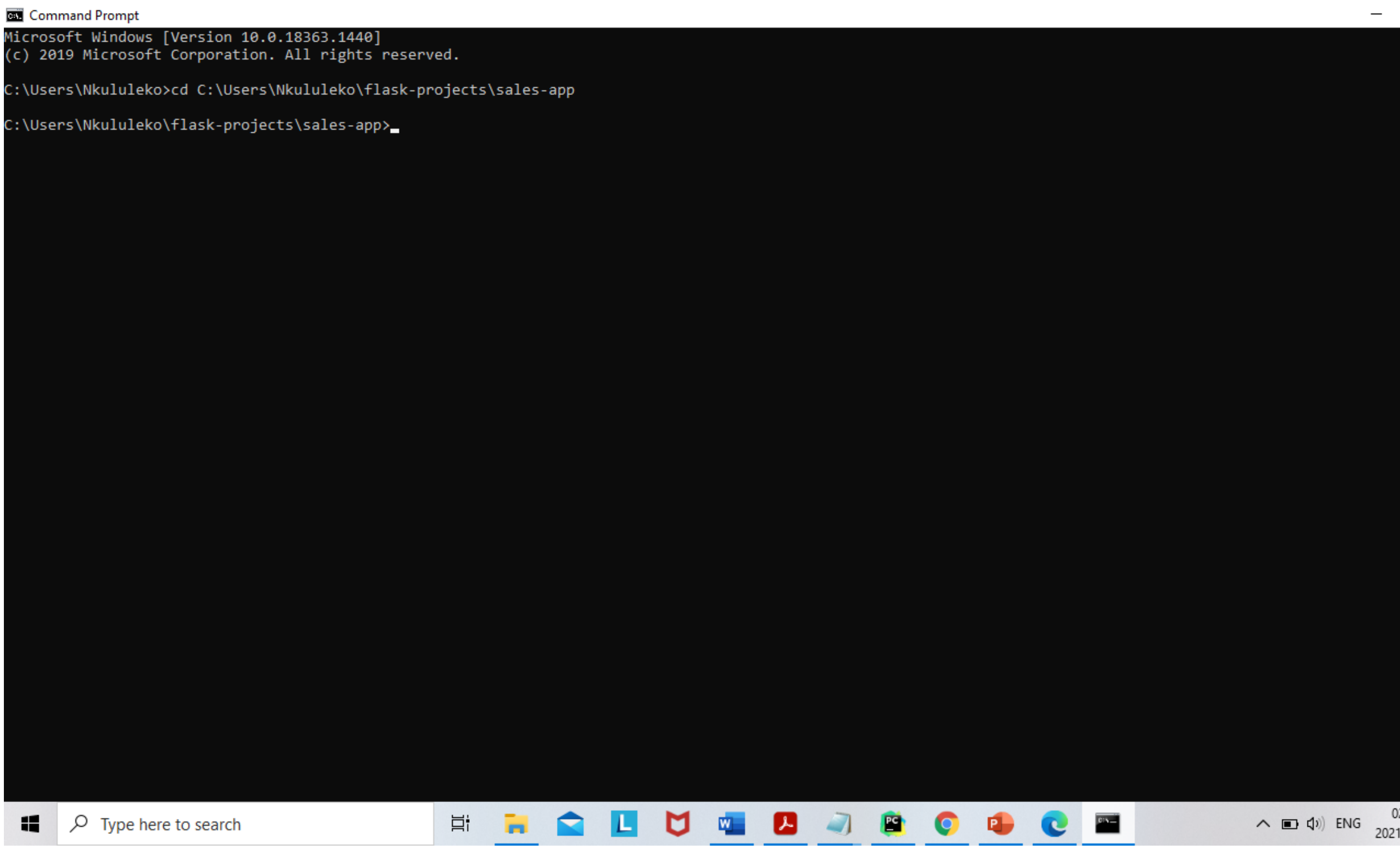
We've shown the first alternative of **API deployment by GitHub**, now let's go **to API deployment by Heroku GIT.**

API deployment by Heroku GIT.

To accomplish this, the requirement ensures that Heroku CLI is installed.

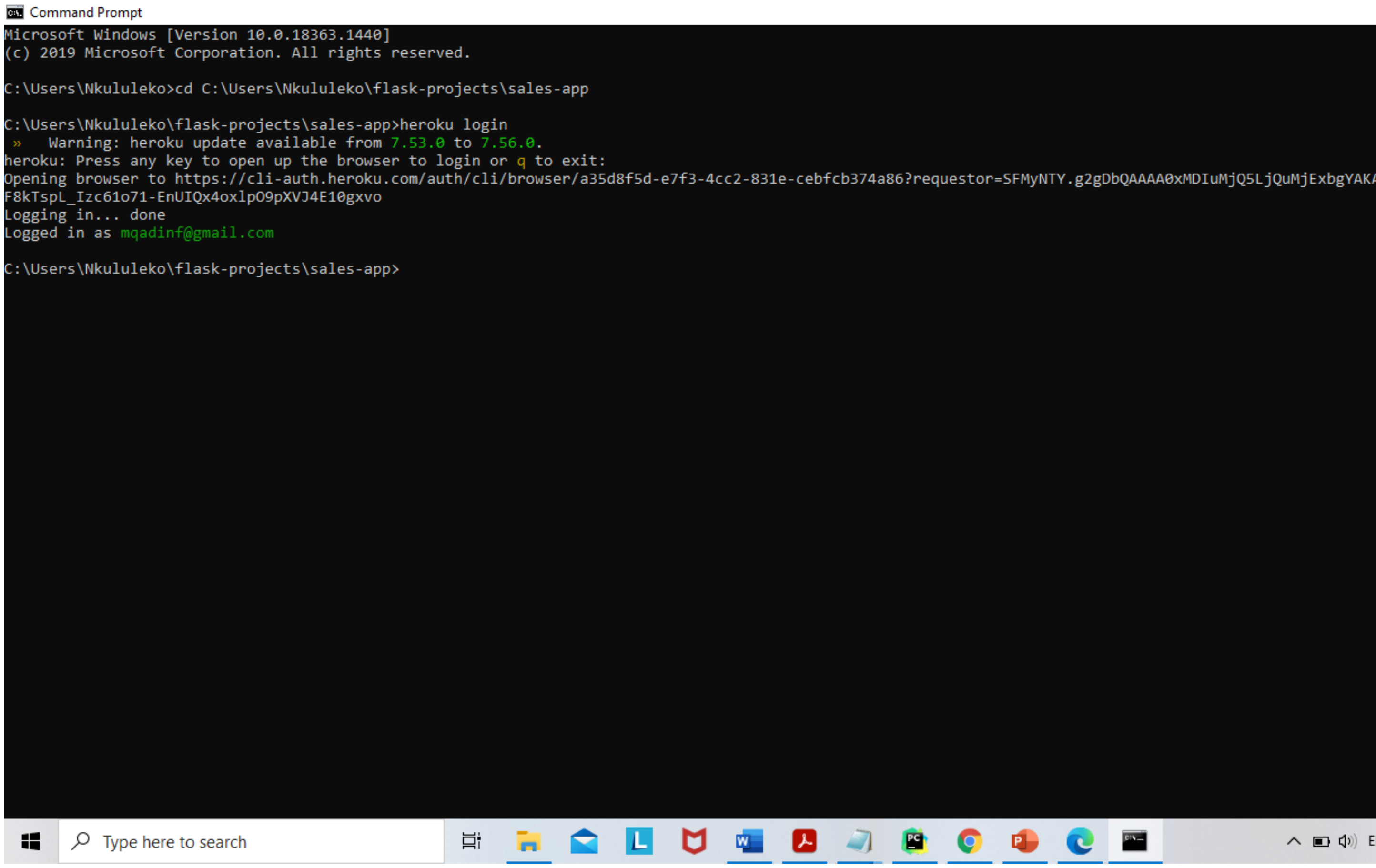
Let's begin with the steps for API deployment by Heroku GIT.

- **The first thing before logging in to Heroku, we must first locate in the project directory.**



```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Nkululeko>cd C:\Users\Nkululeko\flask-projects\sales-app

C:\Users\Nkululeko\flask-projects\sales-app>_
```

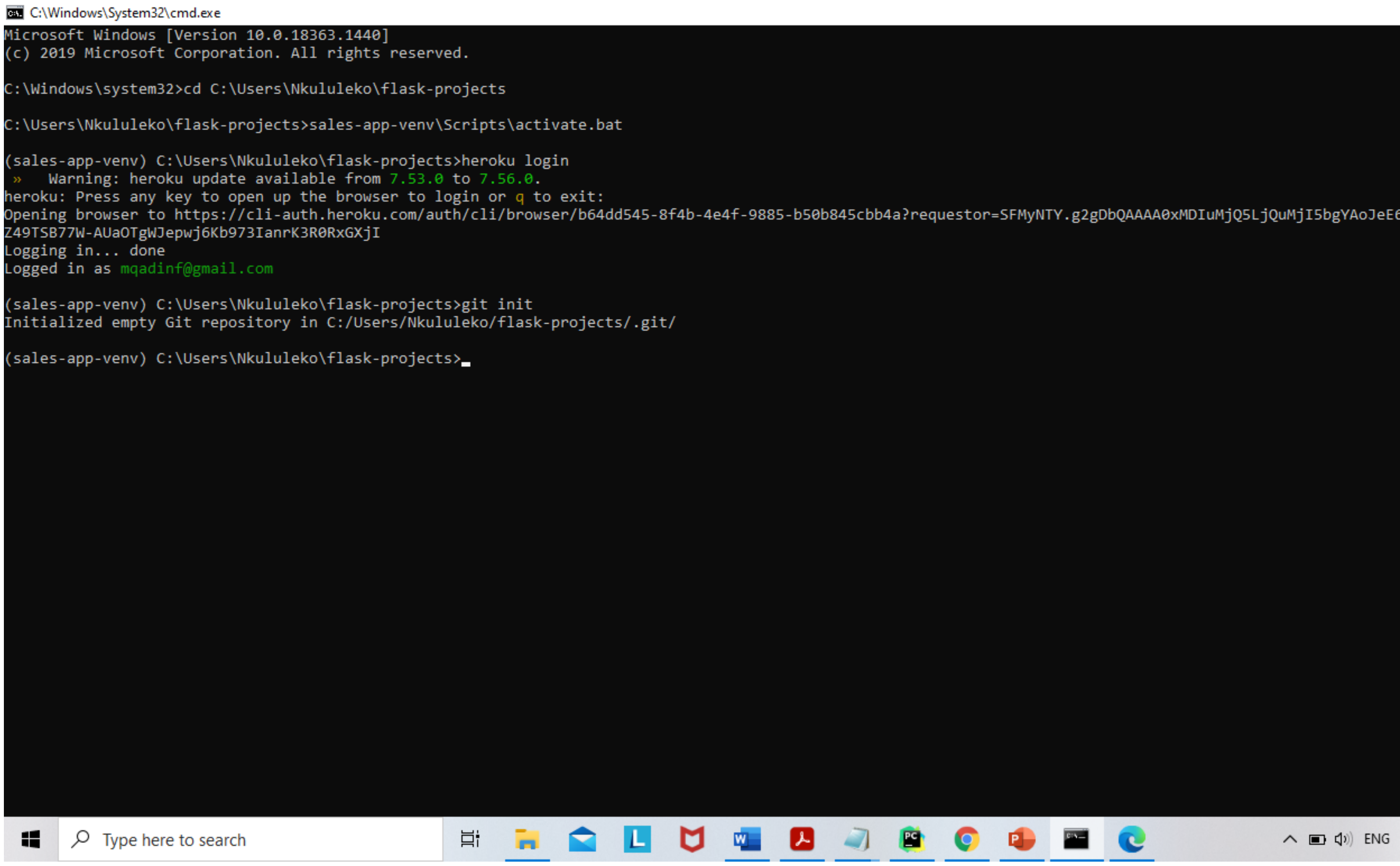
- Then we log in to Heroku by entering **heroku login** command. It will ask you to enter email id and password to login. After successful login next screen will show like below:



Command Prompt

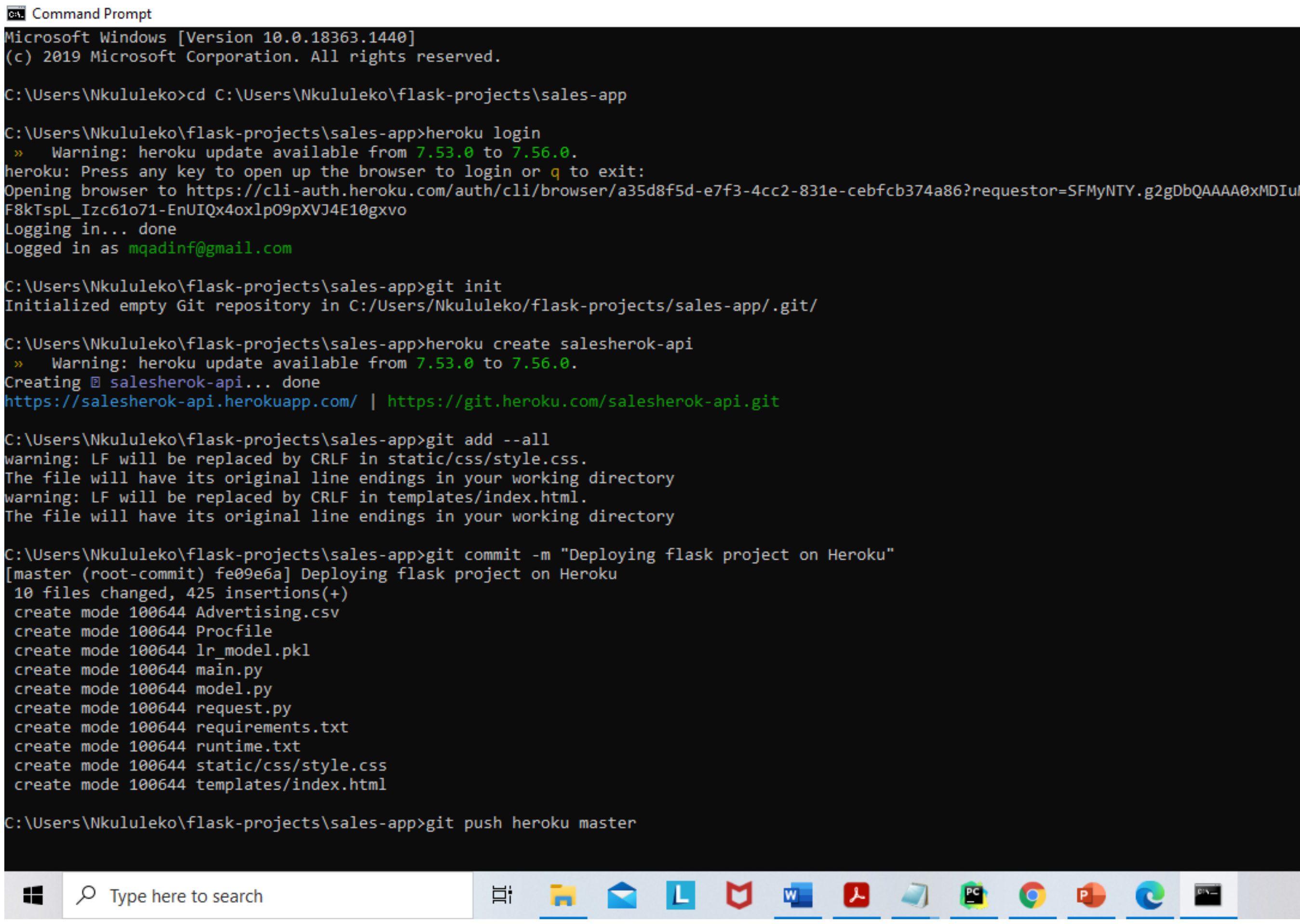
```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Nkululeko>cd C:\Users\Nkululeko\flask-projects\sales-app

C:\Users\Nkululeko\flask-projects\sales-app>heroku login
 »   Warning: heroku update available from 7.53.0 to 7.56.0.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/a35d8f5d-e7f3-4cc2-831e-cebfcb374a86?requestor=SFMyNTY.g2gDbQAAAA0xMDIuMjQ5LjQuMjExbgYAKA
F8kTspL_Izc61o71-EnUIQx4oxlpO9pXVJ4E10gxvo
Logging in... done
Logged in as mqadinf@gmail.com

C:\Users\Nkululeko\flask-projects\sales-app>
```

- Next, we initialize git with the command **git init.**



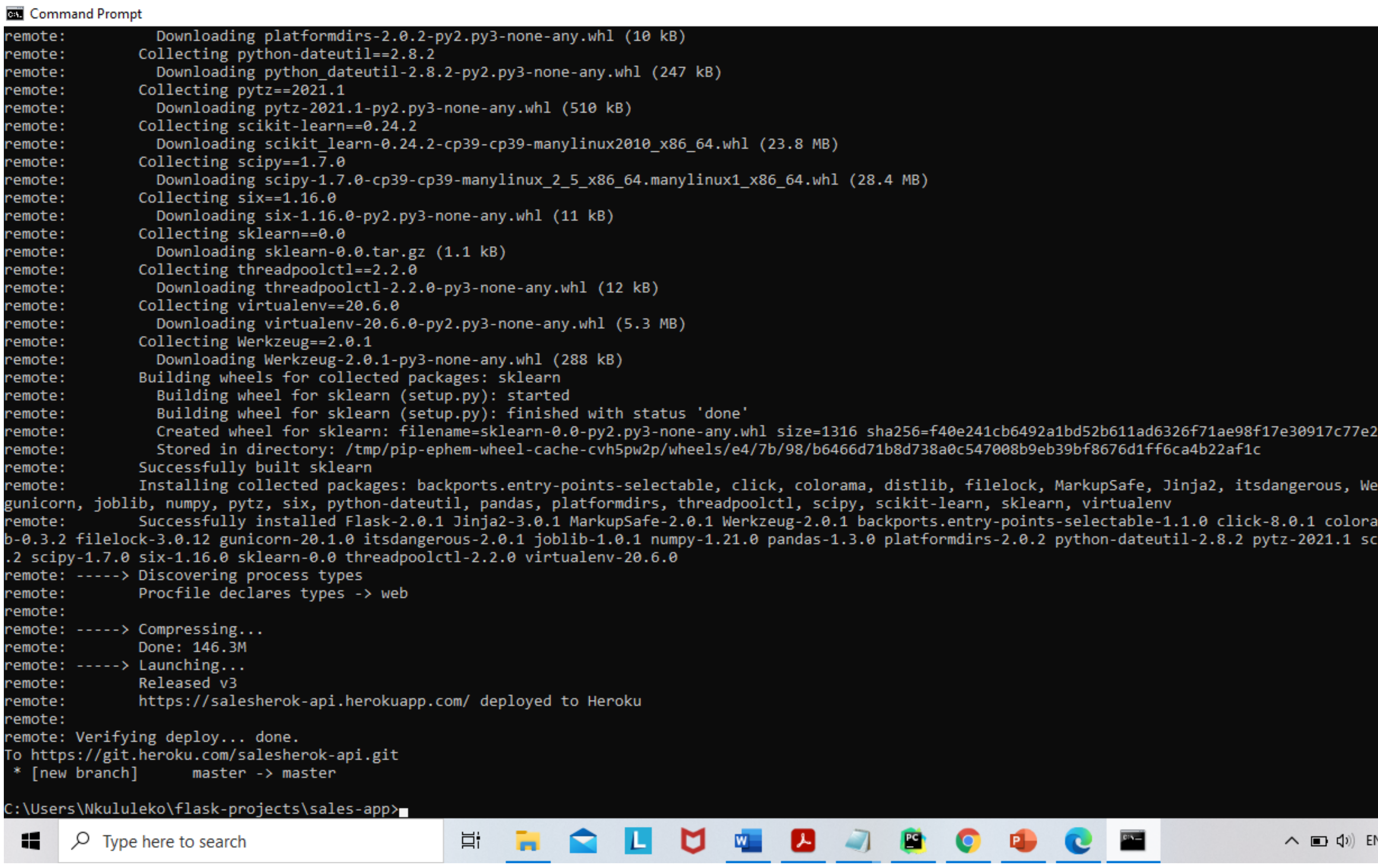
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\Nkululeko\flask-projects

C:\Users\Nkululeko\flask-projects>sales-app-venv\Scripts\activate.bat

(sales-app-venv) C:\Users\Nkululeko\flask-projects>heroku login
 »   Warning: heroku update available from 7.53.0 to 7.56.0.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/b64dd545-8f4b-4e4f-9885-b50b845cbb4a?requestor=SFMyNTY.g2gDbQAAAA0xMDIuMjQ5LjQuMjI5bgYAoJeE6
Z49TSB77W-AUaOTgWJepwj6Kb973IanrK3R0RxGXjI
Logging in... done
Logged in as mqadinf@gmail.com

(sales-app-venv) C:\Users\Nkululeko\flask-projects>git init
Initialized empty Git repository in C:/Users/Nkululeko/flask-projects/.git/

(sales-app-venv) C:\Users\Nkululeko\flask-projects>_
```

- Then we create Heroku app, add files to GIT and deploy.



```
Command Prompt
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Nkululeko>cd C:\Users\Nkululeko\flask-projects\sales-app

C:\Users\Nkululeko\flask-projects\sales-app>heroku login
 »   Warning: heroku update available from 7.53.0 to 7.56.0.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/a35d8f5d-e7f3-4cc2-831e-cebfcb374a86?requestor=SFMyNTY.g2gDbQAAAA0xMDIu
F8kTspL_Izc61o71-EnUIQx4oxlpO9pXVJ4E10gxvo
Logging in... done
Logged in as mqadinf@gmail.com

C:\Users\Nkululeko\flask-projects\sales-app>git init
Initialized empty Git repository in C:/Users/Nkululeko/flask-projects/sales-app/.git/

C:\Users\Nkululeko\flask-projects\sales-app>heroku create salesherok-api
 »   Warning: heroku update available from 7.53.0 to 7.56.0.
Creating ⬡ salesherok-api... done
https://salesherok-api.herokuapp.com/ | https://git.heroku.com/salesherok-api.git

C:\Users\Nkululeko\flask-projects\sales-app>git add --all
warning: LF will be replaced by CRLF in static/css/style.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in templates/index.html.
The file will have its original line endings in your working directory

C:\Users\Nkululeko\flask-projects\sales-app>git commit -m "Deploying flask project on Heroku"
[master (root-commit) fe09e6a] Deploying flask project on Heroku
 10 files changed, 425 insertions(+)
 create mode 100644 Advertising.csv
 create mode 100644 Procfile
 create mode 100644 lr_model.pkl
 create mode 100644 main.py
 create mode 100644 model.py
 create mode 100644 request.py
 create mode 100644 requirements.txt
 create mode 100644 runtime.txt
 create mode 100644 static/css/style.css
 create mode 100644 templates/index.html

C:\Users\Nkululeko\flask-projects\sales-app>git push heroku master
```

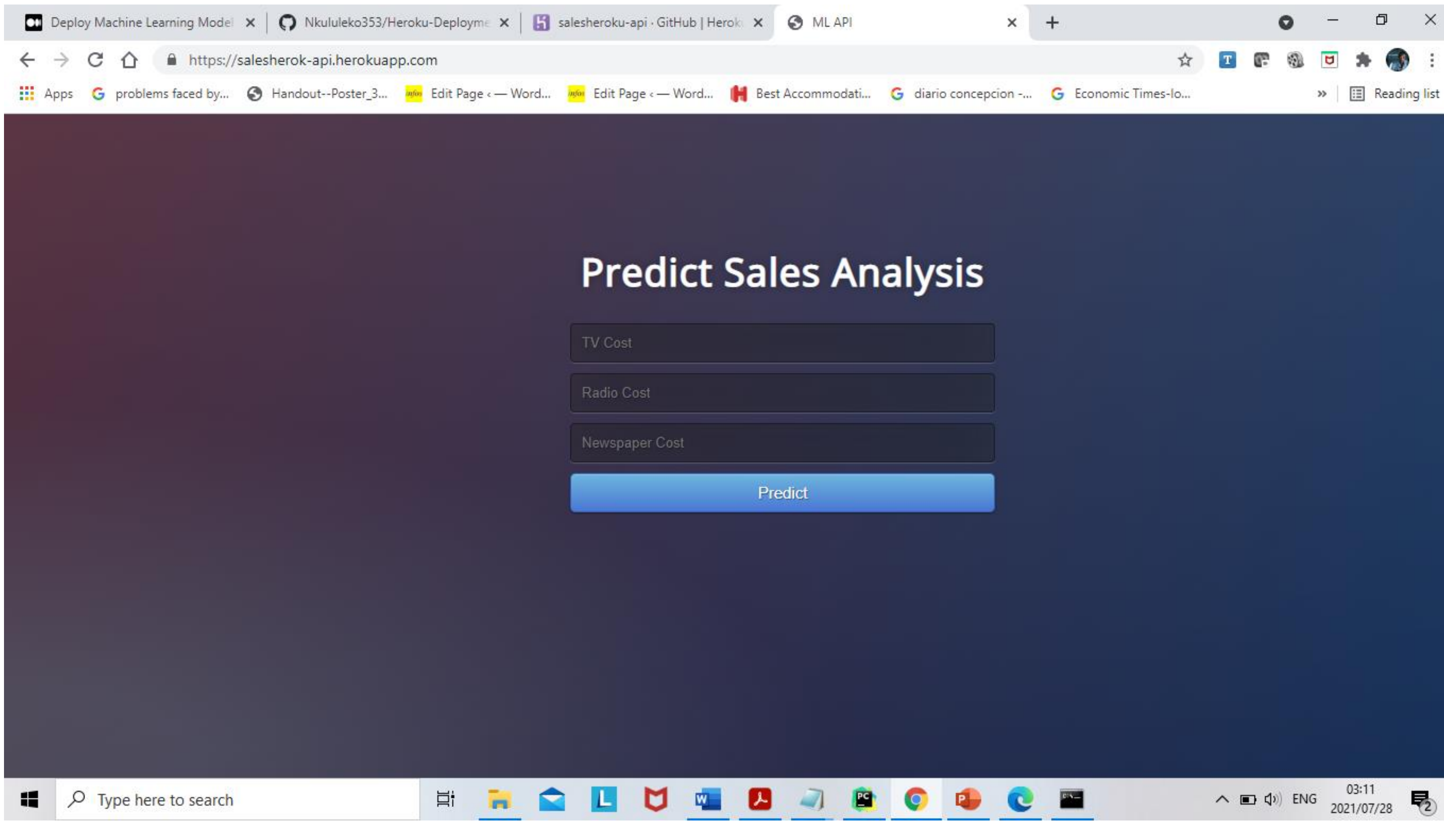
- **Finally, we get our deployed url as follows: https://salesherok-api.herokuapp.com/**



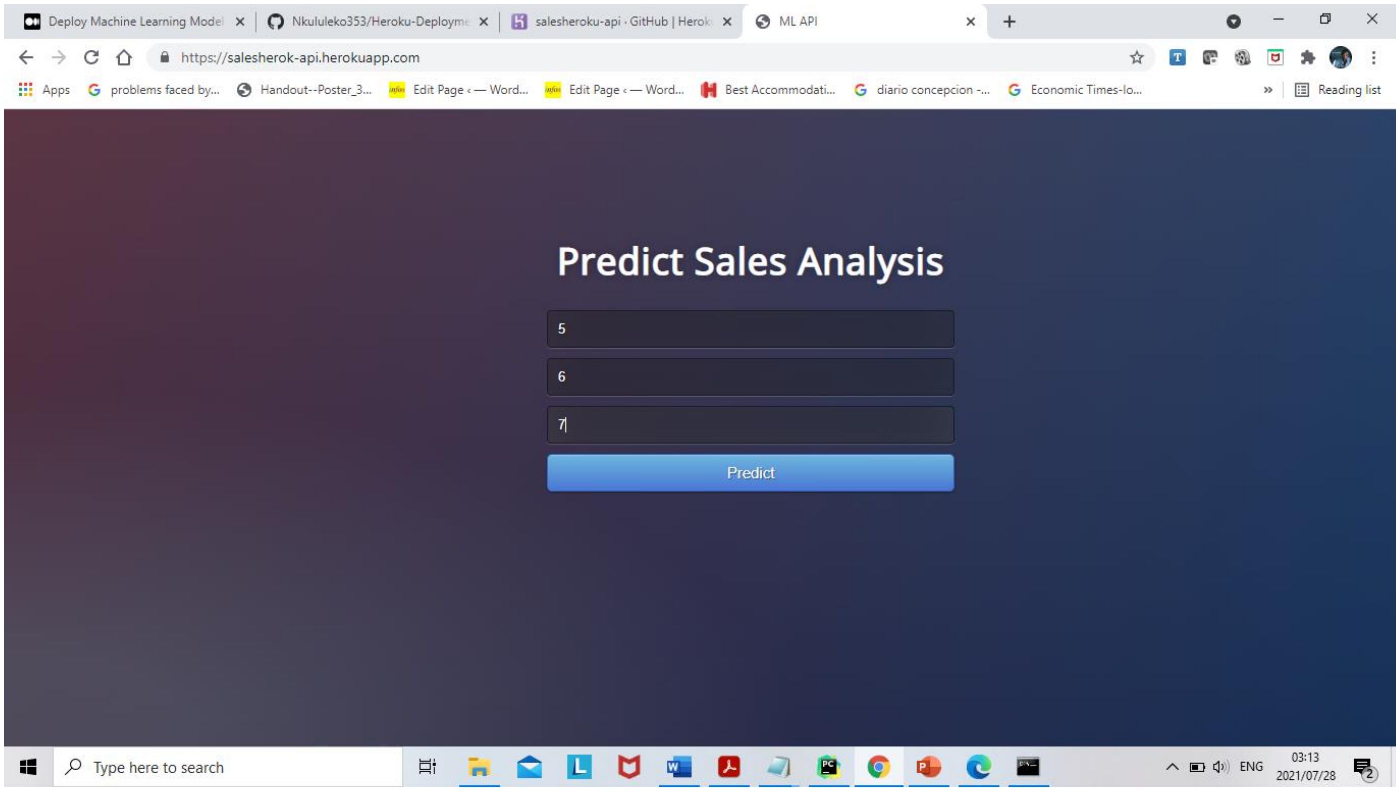
Command Prompt

```
remote:          Downloading platformdirs-2.0.2-py2.py3-none-any.whl (10 kB)
remote:        Collecting python-dateutil==2.8.2
remote:          Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
remote:        Collecting pytz==2021.1
remote:          Downloading pytz-2021.1-py2.py3-none-any.whl (510 kB)
remote:        Collecting scikit-learn==0.24.2
remote:          Downloading scikit_learn-0.24.2-cp39-cp39-manylinux2010_x86_64.whl (23.8 MB)
remote:        Collecting scipy==1.7.0
remote:          Downloading scipy-1.7.0-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.whl (28.4 MB)
remote:        Collecting six==1.16.0
remote:          Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
remote:        Collecting sklearn==0.0
remote:          Downloading sklearn-0.0.tar.gz (1.1 kB)
remote:        Collecting threadpoolctl==2.2.0
remote:          Downloading threadpoolctl-2.2.0-py3-none-any.whl (12 kB)
remote:        Collecting virtualenv==20.6.0
remote:          Downloading virtualenv-20.6.0-py2.py3-none-any.whl (5.3 MB)
remote:        Collecting Werkzeug==2.0.1
remote:          Downloading Werkzeug-2.0.1-py3-none-any.whl (288 kB)
remote:        Building wheels for collected packages: sklearn
remote:          Building wheel for sklearn (setup.py): started
remote:          Building wheel for sklearn (setup.py): finished with status 'done'
remote:          Created wheel for sklearn: filename=sklearn-0.0-py2.py3-none-any.whl size=1316 sha256=f40e241cb6492a1bd52b611ad6326f71ae98f17e30917c77e2
remote:          Stored in directory: /tmp/pip-ephem-wheel-cache-cvh5pw2p/wheels/e4/7b/98/b6466d71b8d738a0c547008b9eb39bf8676d1ff6ca4b22af1c
remote:        Successfully built sklearn
remote:        Installing collected packages: backports.entry-points-selectable, click, colorama, distlib, filelock, MarkupSafe, Jinja2, itsdangerous, We
gunicorn, joblib, numpy, pytz, six, python-dateutil, pandas, platformdirs, threadpoolctl, scipy, scikit-learn, sklearn, virtualenv
remote:        Successfully installed Flask-2.0.1 Jinja2-3.0.1 MarkupSafe-2.0.1 Werkzeug-2.0.1 backports.entry-points-selectable-1.1.0 click-8.0.1 colora
b-0.3.2 filelock-3.0.12 gunicorn-20.1.0 itsdangerous-2.0.1 joblib-1.0.1 numpy-1.21.0 pandas-1.3.0 platformdirs-2.0.2 python-dateutil-2.8.2 pytz-2021.1 sc
.2 scipy-1.7.0 six-1.16.0 sklearn-0.0 threadpoolctl-2.2.0 virtualenv-20.6.0
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 146.3M
remote: -----> Launching...
remote:        Released v3
remote:        https://salesherok-api.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/salesherok-api.git
 * [new branch]      master -> master

C:\Users\Nkululeko\flask-projects\sales-app>
```

- **This is how the web app looks as shown below.**



- We can test the web app functionality.


Deploy Machine Learning Model
Nkululeko353/Heroku-Deployme
salesheroku-api · GitHub | Heroku
ML API
https://salesherok-api.herokuapp.com
Predict

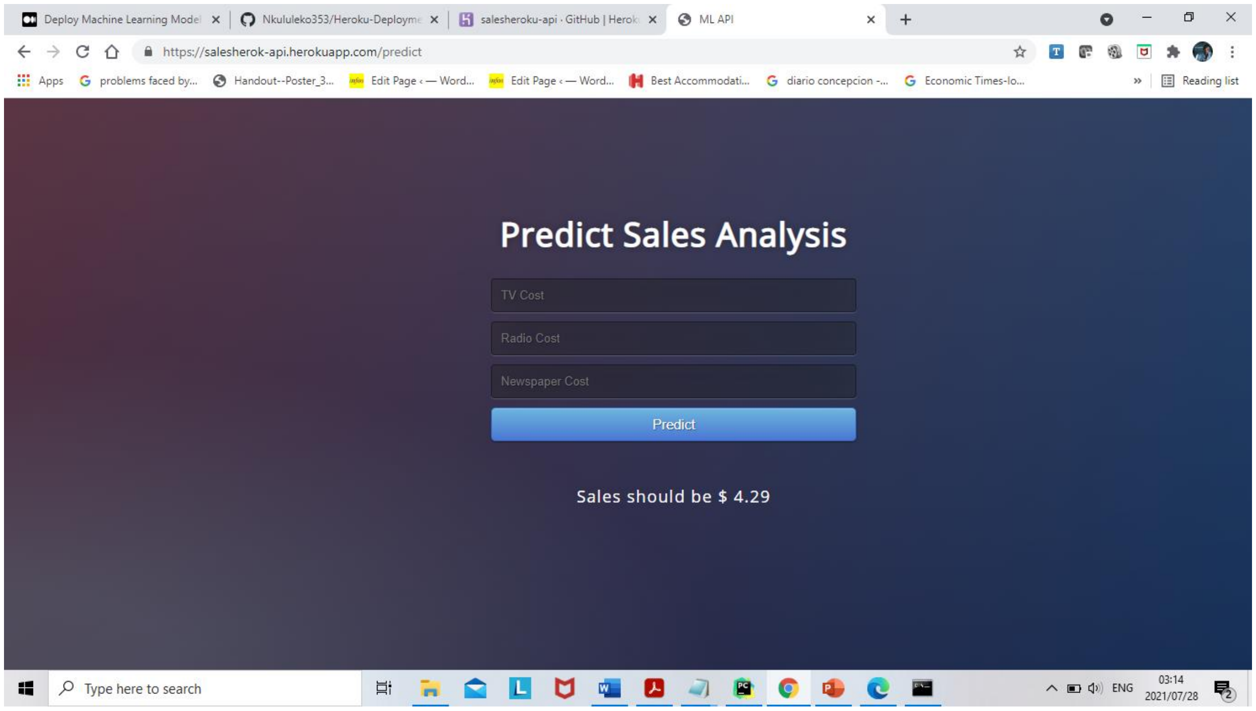
Predict Sales Analysis

5

6

7

- Its working perfectly and able to predict, we can confirm this as shown below.


Deploy Machine Learning Model
Nkululeko353/Heroku-Deployme
salesheroku-api · GitHub | Heroku
ML API
https://salesherok-api.herokuapp.com/predict
TV Cost
Radio Cost
Newspaper Cost
Predict

Predict Sales Analysis

Sales should be $ 4.29

Now I’ve finished with the **API Deployment on Heroku.**

The full source code is available for review on the following link:
[**https://github.com/Nkululeko353/Heroku-Deployment**](https://github.com/Nkululeko353/Heroku-Deployment)

The End!