Contents

[Topic : Interactive Dashboards for Data Science 1](#_Toc29892906)

[Installation of anaconda 1](#_Toc29892907)

[Installing using pip and virtual environments 1](#_Toc29892908)

[Creating Script to run the flash framework 2](#_Toc29892909)

[First Method : Using Jupyter notebook 2](#_Toc29892910)

[Second Method : Using colab 14](#_Toc29892911)

# Topic: Interactive Dashboards for Data Science

In this tasted program, we are going to create an online Data Science Dashboard. It can conduct a powerful way of communicating the results of a Data Science project. A good Dashboard can:

* Summarize the main results of a data analysis.
* Enable the customers/company managers to test how varying some parameters can affect a certain outcome.
* Fetch continuously new data to update its graphs and summaries.
* Allow us to make predictions online using either pre-trained Machine Learning models or training them online.

## Step 0 : Installation of Anaconda Python Distribution

1. Install anaconda - <https://docs.google.com/document/d/1x9BrZ1dCKaLwnSLxj8Q8tSysiCUpb6BqybkEU54rhJI/edit?usp=sharing>

2. Open Anaconda Prompt

3. mkdir interactivedashboards-datascience

4. cd interactivedashboards-datascience

## Step 1 : Installing using pip and virtual environments

1. Create an requirements.txt

|  |
| --- |
| flask  dash  dash\_core\_components  dash\_html\_components  pandas  numpy  sklearn  statsmodels  gunicorn |

1. Run the following script in the Anaconda Prompt

pip install -r requirements.txt

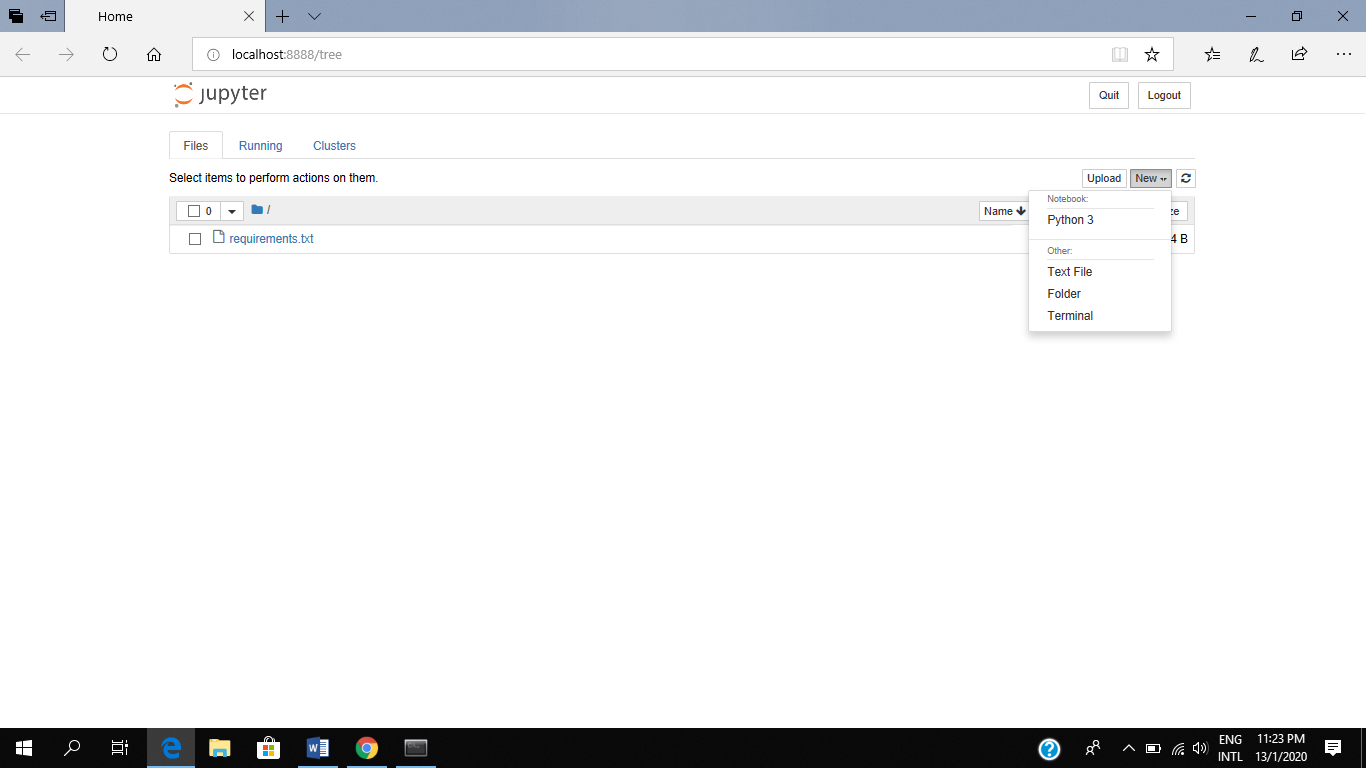
## Step 2 : Creating Script to run the flash framework

There are two method for conducting the .py script for running html code

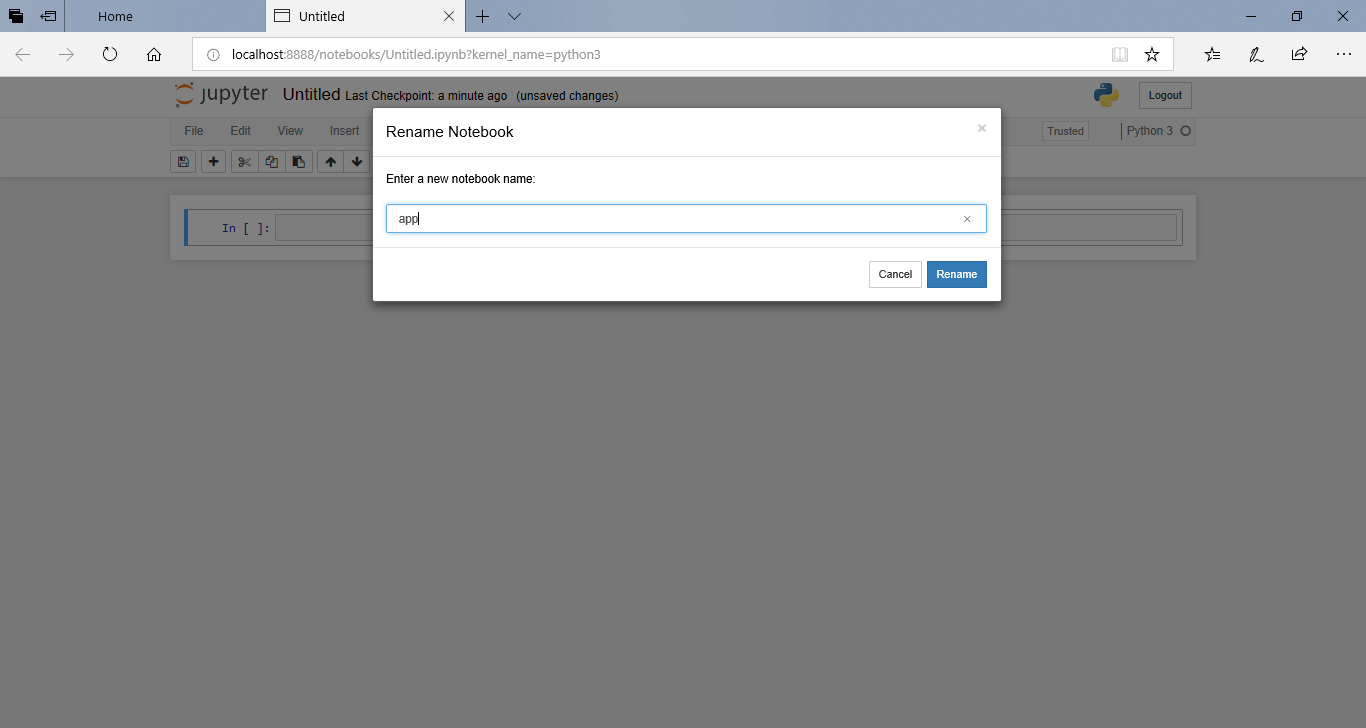
1. Using Jupyter notebook
2. Using Colab

## Step 2.1 : Using Jupyter notebook

* + - 1. Run command “jupyter notebook”
      2. Click New -> Python 3



* + - 1. Rename to “app.ipynb”



* + - 1. Install the code to install dash and their related function

|  |
| --- |
| import dash  import dash\_core\_components as dcc  import dash\_html\_components as html  import pandas as pd  import plotly.graph\_objs as go  from dash.dependencies import Input, Output |

* + - 1. Install the code to run the dash and server function

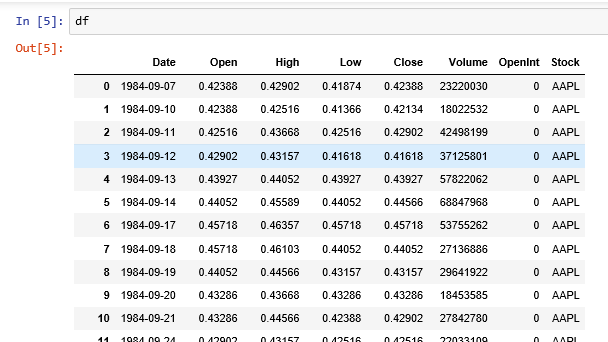
|  |
| --- |
| app = dash.Dash()  server = app.server |

* + - 1. Download two file stock\_data.csv and dataset\_Facebook.csv from <https://github.com/innoviai/InnoVi_Tech_Academy/tree/master/data_source>
      2. Upload those two file on the directory “C:/Users/[User]/interactivedashboards-datascience”
      3. Install the code for reading the data in csv format

|  |
| --- |
| df = pd.read\_csv("stock\_data.csv ")  df2 = pd.read\_csv("dataset\_Facebook.csv ",";") |

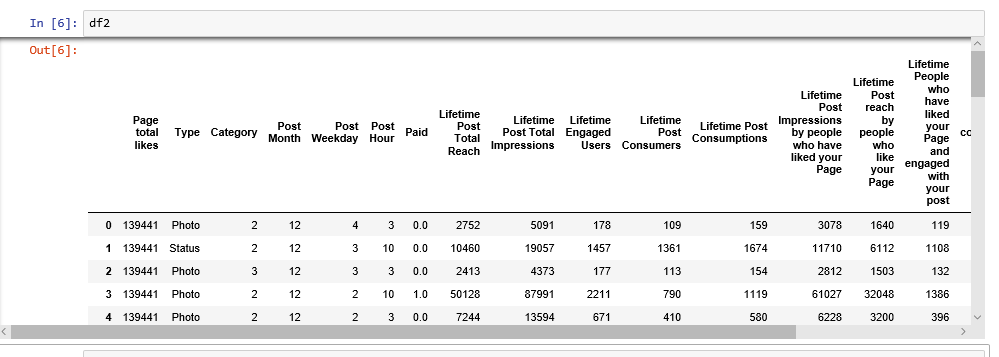
* + - 1. Run df to display the record from stock\_data.csv

|  |
| --- |
| df |



* + - 1. Run df2 to display the record from dataset\_Facebook.csv

|  |
| --- |
| df2 |



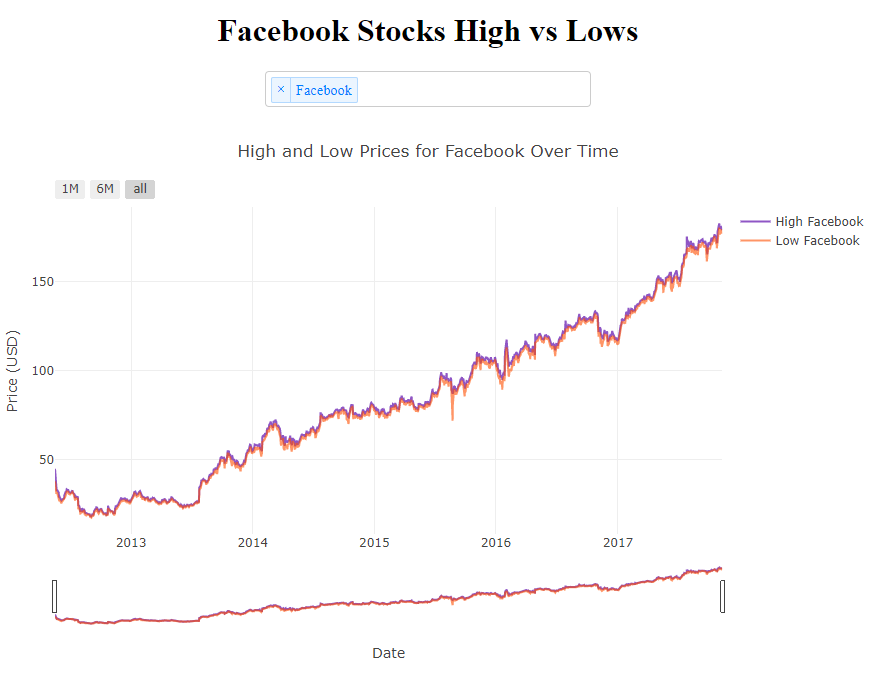
* + - 1. Set-up our App and its layout using the code shown below :

|  |
| --- |
| app.layout = html.Div([  # Setting the main title of the Dashboard  html.H1("Facebook Data Analysis", style={"textAlign": "center"}),  # Dividing the dashboard in tabs  dcc.Tabs(id="tabs", children=[  # Defining the layout of the first Tab  dcc.Tab(label='Stock Prices', children=[  html.Div([  html.H1("Facebook Stocks High vs Lows",  style={'textAlign': 'center'}),  # Adding the first dropdown menu and the subsequent time-series graph  dcc.Dropdown(id='my-dropdown',  options=[{'label': 'Tesla', 'value': 'TSLA'},  {'label': 'Apple','value': 'AAPL'},  {'label': 'Facebook', 'value': 'FB'},  {'label': 'Microsoft','value': 'MSFT'}],  multi=True,value=['FB'],  style={"display": "block", "margin-left": "auto",  "margin-right": "auto", "width": "60%"}),  dcc.Graph(id='highlow'),  html.H1("Facebook Market Volume", style={'textAlign': 'center'}),  # Adding the second dropdown menu and the subsequent time-series graph  dcc.Dropdown(id='my-dropdown2',  options=[{'label': 'Tesla', 'value': 'TSLA'},  {'label': 'Apple','value': 'AAPL'},  {'label': 'Facebook', 'value': 'FB'},  {'label': 'Microsoft','value': 'MSFT'}],  multi=True,value=['FB'],  style={"display": "block", "margin-left": "auto",  "margin-right": "auto", "width": "60%"}),  dcc.Graph(id='volume')  ], className="container"),  ]),  # Defining the layout of the second tab  dcc.Tab(label='Performance Metrics', children=[  html.H1("Facebook Metrics Distributions",  style={"textAlign": "center"}),  # Adding a dropdown menu and the subsequent histogram graph  html.Div([  html.Div([dcc.Dropdown(id='feature-selected1',  options=[{'label': i.title(),  'value': i} for i in df2.columns.values[1:]],  value="Type")],  className="twelve columns",  style={"display": "block", "margin-left": "auto",  "margin-right": "auto", "width": "60%"}),  ], className="row",  style={"padding": 50, "width": "60%",  "margin-left": "auto", "margin-right": "auto"}),  dcc.Graph(id='my-graph2'),  ])  ])  ]) |

* + - 1. Install first function “highlow” to take the selected values from the dropdowns, send them as input to the graph function and then take the variable returned by the function to pass it to the graph declared in the layout.

|  |
| --- |
| @app.callback(Output('highlow', 'figure'),  [Input('my-dropdown', 'value')])  def update\_graph(selected\_dropdown):  dropdown = {"TSLA": "Tesla","AAPL": "Apple","FB": "Facebook","MSFT": "Microsoft",}  trace1 = []  trace2 = []  for stock in selected\_dropdown:  trace1.append(  go.Scatter(x=df[df["Stock"] == stock]["Date"],  y=df[df["Stock"] == stock]["High"],  mode='lines', opacity=0.7,  name=f'High {dropdown[stock]}',textposition='bottom center'))  trace2.append(  go.Scatter(x=df[df["Stock"] == stock]["Date"],  y=df[df["Stock"] == stock]["Low"],  mode='lines', opacity=0.6,  name=f'Low {dropdown[stock]}',textposition='bottom center'))  traces = [trace1, trace2]  data = [val for sublist in traces for val in sublist]  figure = {'data': data,  'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',  '#FF7400', '#FFF400', '#FF0056'],  height=600,  title=f"High and Low Prices for Over Time",  xaxis={"title":"Date",  'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',  'step': 'month',  'stepmode': 'backward'},  {'count': 6, 'label': '6M',  'step': 'month',  'stepmode': 'backward'},  {'step': 'all'}])},  'rangeslider': {'visible': True}, 'type': 'date'},  yaxis={"title":"Price (USD)"})}  return figure |

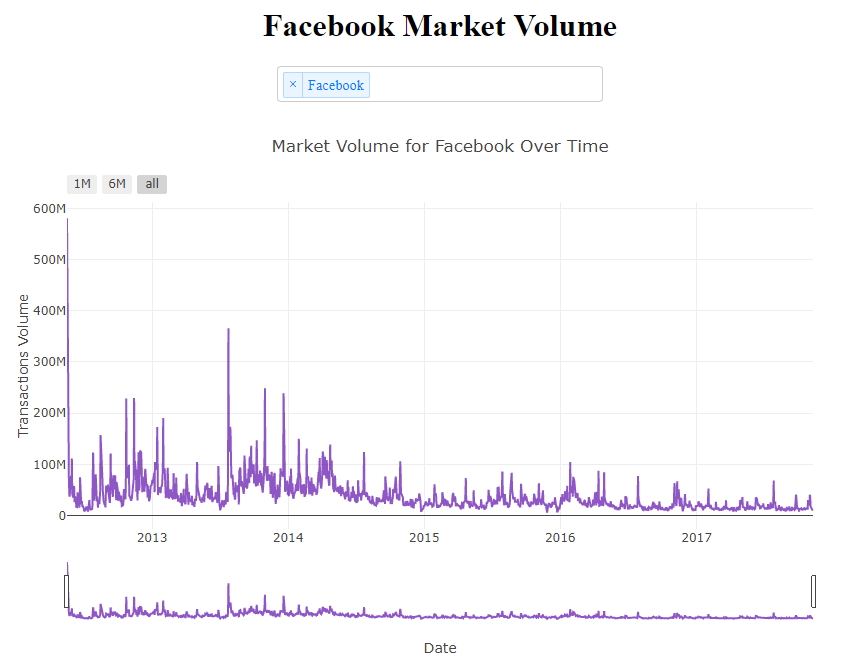
1. The first function will generate the graph shown below.



1. Install second function “volume” to take the selected values from the dropdowns, send them as input to the graph function and then take the variable returned by the function to pass it to the graph declared in the layout.

|  |
| --- |
| @app.callback(Output('volume', 'figure'),  [Input('my-dropdown2', 'value')])  def update\_graph(selected\_dropdown\_value):  dropdown = {"TSLA": "Tesla","AAPL": "Apple","FB": "Facebook","MSFT": "Microsoft",}  trace1 = []  for stock in selected\_dropdown\_value:  trace1.append(  go.Scatter(x=df[df["Stock"] == stock]["Date"],  y=df[df["Stock"] == stock]["Volume"],  mode='lines', opacity=0.7,  name=f'Volume {dropdown[stock]}', textposition='bottom center'))  traces = [trace1]  data = [val for sublist in traces for val in sublist]  figure = {'data': data,  'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',  '#FF7400', '#FFF400', '#FF0056'],  height=600,  title=f"Market Volume for Over Time",  xaxis={"title":"Date",  'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',  'step': 'month',  'stepmode': 'backward'},  {'count': 6, 'label': '6M',  'step': 'month',  'stepmode': 'backward'},  {'step': 'all'}])},  'rangeslider': {'visible': True}, 'type': 'date'},  yaxis={"title":"Transactions Volume"})}  return figure |

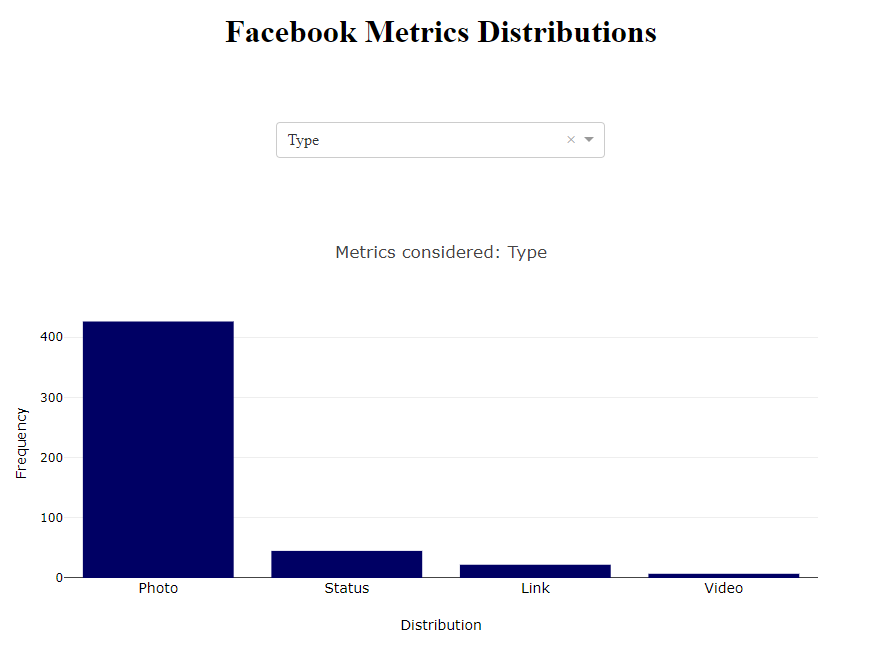
1. The second one will instead create the graph shown.



1. Install the last function “selected\_feature1” to take the selected values from the dropdowns, send them as input to the graph function and then take the variable returned by the function to pass it to the graph declared in the layout.

|  |
| --- |
| @app.callback(  dash.dependencies.Output('my-graph2', 'figure'),  [dash.dependencies.Input('feature-selected1', 'value')])  def update\_graph(selected\_feature1):  if selected\_feature1 == None:  selected\_feature1 = 'Type'  trace = go.Histogram(x= df2.Type,  marker=dict(color='rgb(0, 0, 100)'))  else:  trace = go.Histogram(x=df2[selected\_feature1],  marker=dict(color='rgb(0, 0, 100)'))  return {  'data': [trace],  'layout': go.Layout(title=f'Metrics considered: {selected\_feature1.title()}',  colorway=["#EF963B", "#EF533B"], hovermode="closest",  xaxis={'title': "Distribution",  'titlefont': {'color': 'black', 'size': 14},  'tickfont': {'size': 14, 'color': 'black'}},  yaxis={'title': "Frequency",  'titlefont': {'color': 'black', 'size': 14, },  'tickfont': {'color': 'black'}})} |

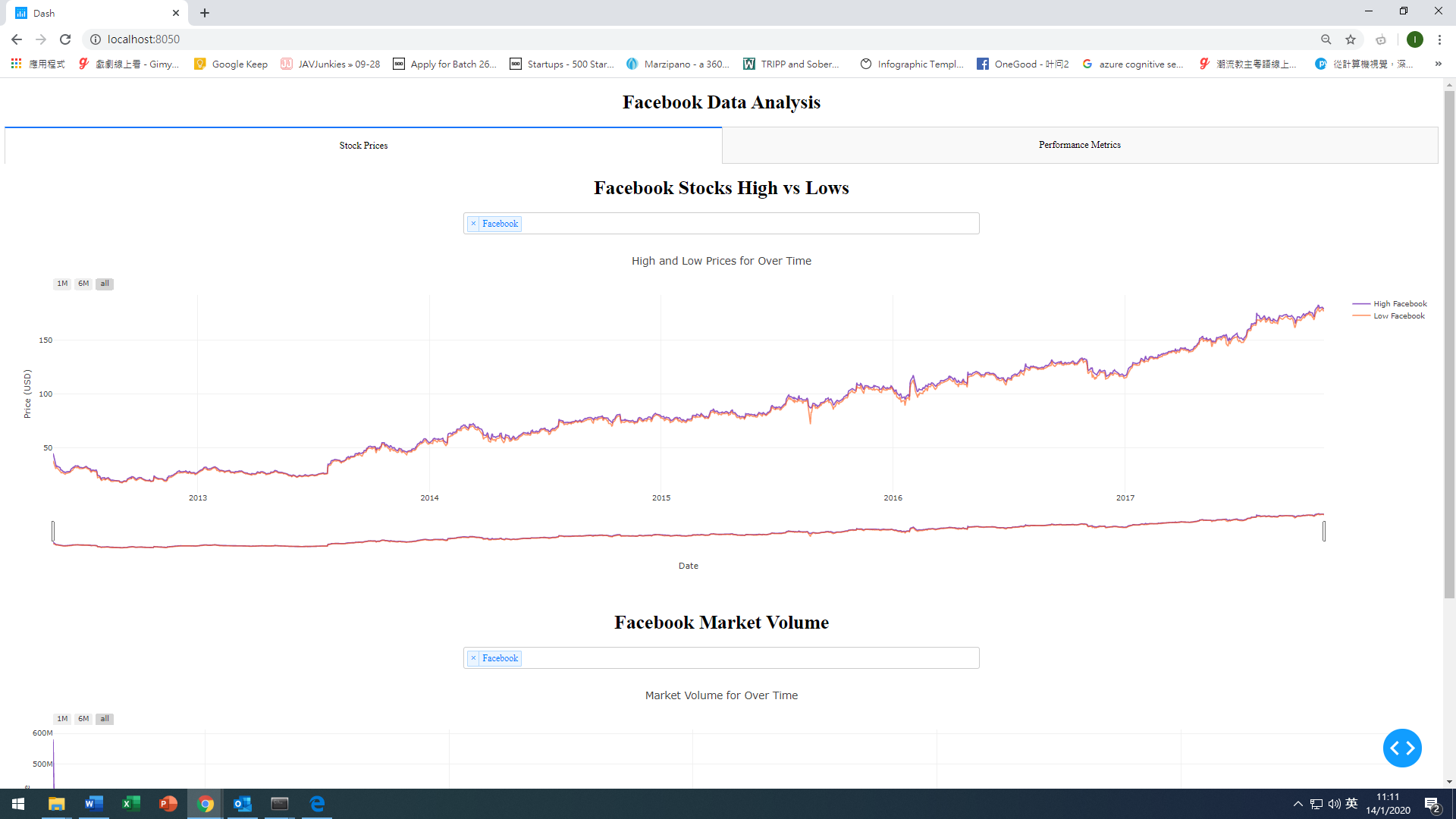
1. Lastly, the third function will generate the feature distribution histogram on the second tab of the Dashboard

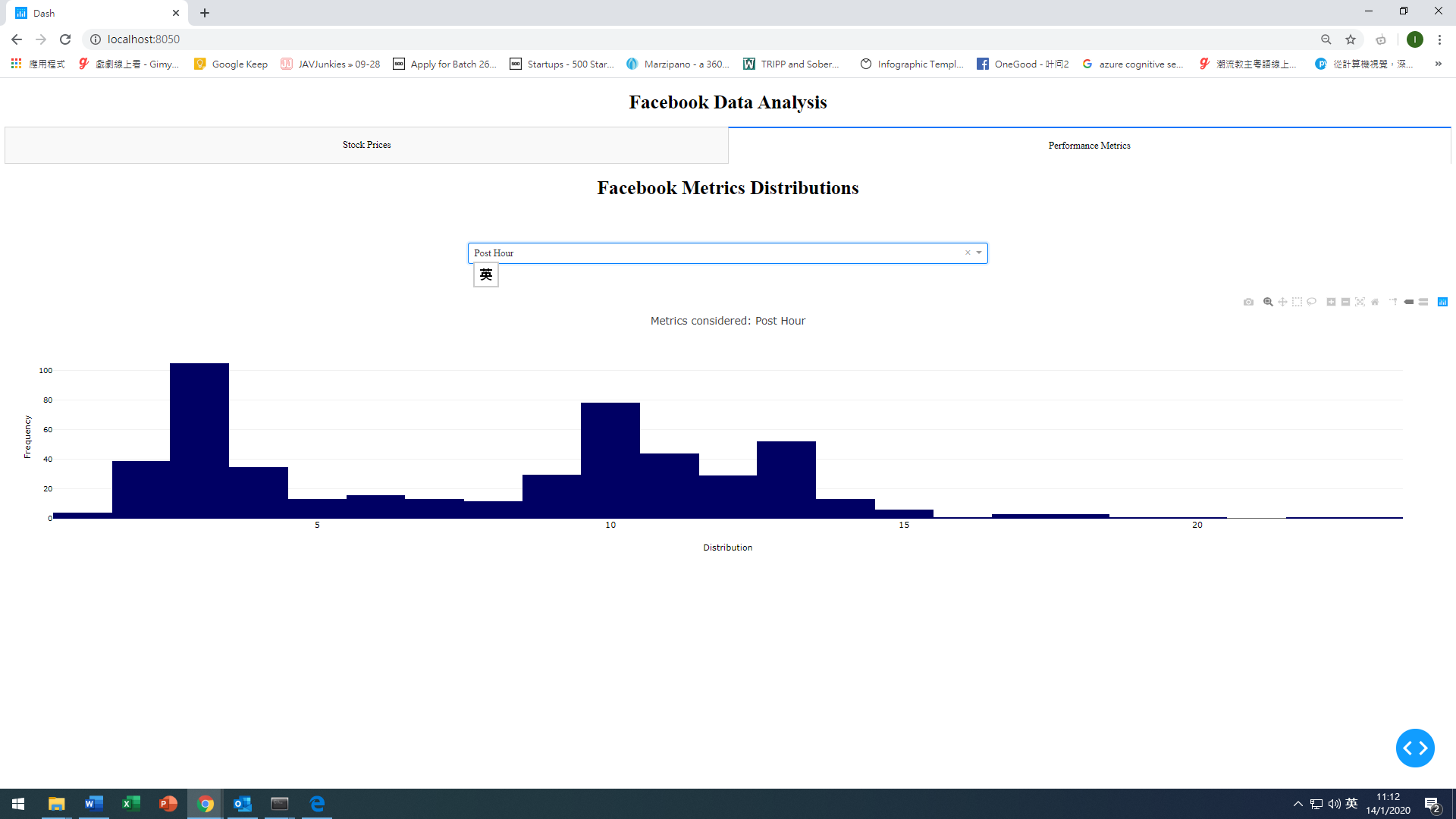


1. Click File -> Download as -> Python (,py) named app.py
2. Stored the app.py in the suitable folder C:\Users\[User[\interactivedashboards-datascience
3. Open the app.py in the C:\Users\[User[\interactivedashboards-datascience
4. Add the following two lines of code which start a local server and make our application run

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  app.run\_server(debug=True) |

1. Run the script “*python app.py*”
2. This will start a web server at <http://127.0.0.1:8050/>, going at this link we will see our final Python App running.
3. You can see the following screen

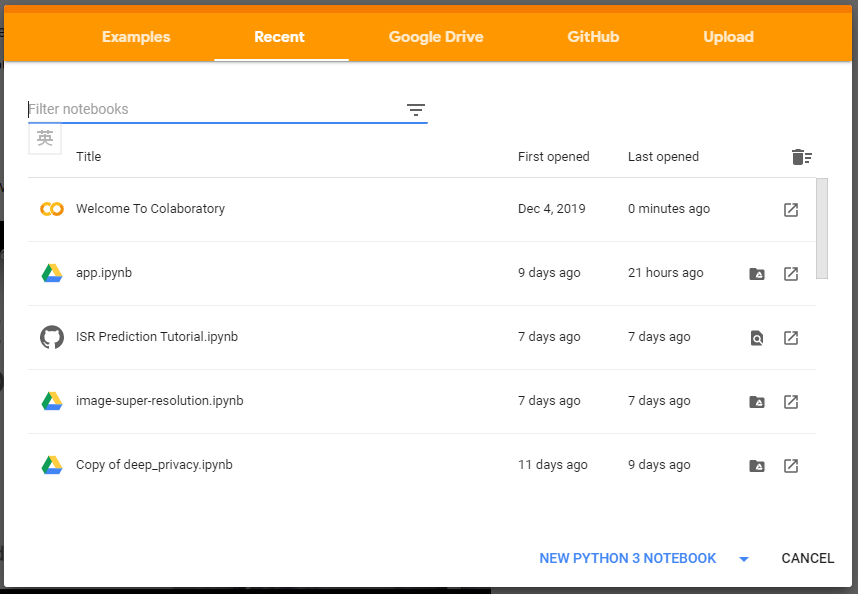




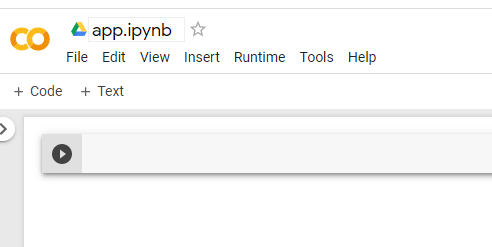
## 

## Step 2.2 : Using Colab

1. Open <https://colab.research.google.com/>
2. Login as gmail account
3. Create a New Python 3 Notebook



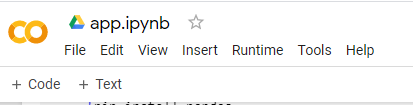
1. Named it “app.ipynb”



1. Install the necessary library including flask, dash, pandas, numpy, sklearn/

|  |
| --- |
| !pip install flask  !pip install dash  !pip install dash\_core\_components  !pip install dash\_html\_components  !pip install pandas  !pip install numpy  !pip install sklearn  !pip install statsmodels  !pip install gunicorn |

1. Add another line of code by pressing “+ code”



1. Install the code

|  |
| --- |
| import dash  import dash\_core\_components as dcc  import dash\_html\_components as html  import pandas as pd  import plotly.graph\_objs as go  from dash.dependencies import Input, Output |

1. Install the code to run the dash and server function

|  |
| --- |
| app = dash.Dash()  server = app.server |

1. Upload “stock\_data.csv” from local drive to colab

|  |
| --- |
| from google.colab import files  uploaded = files.upload() |

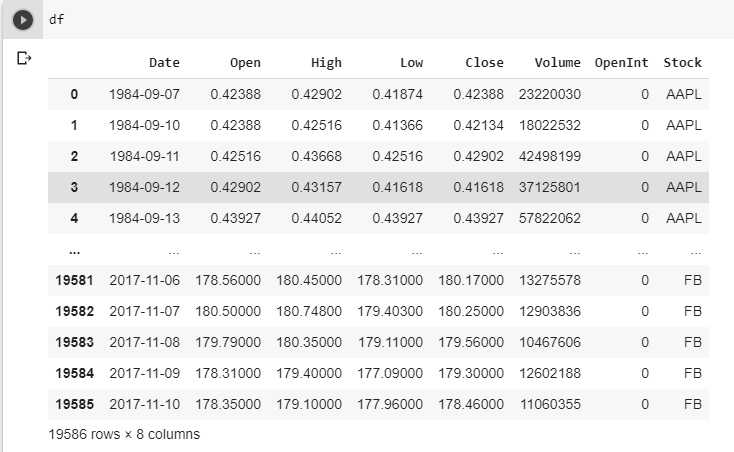


1. Install the code for reading the data in csv format

|  |
| --- |
| import io  df = pd.read\_csv(io.BytesIO(uploaded['stock\_data.csv'])) |

1. Run df

|  |
| --- |
| df |



1. Install the code for uploading “dataset\_Facebook.csv” from local drive to colab

|  |
| --- |
| from google.colab import files  uploaded = files.upload() |

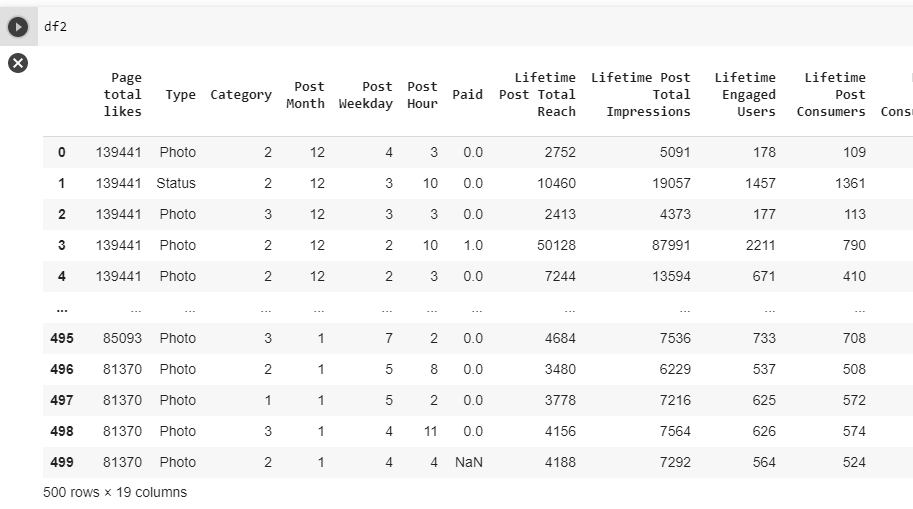


1. Install the code for reading the csv dataset\_Facebook.csv

|  |
| --- |
| import io  df2 = pd.read\_csv(io.BytesIO(uploaded['dataset\_Facebook.csv']),";") |

1. Display the value from df2

|  |
| --- |
| df2 |



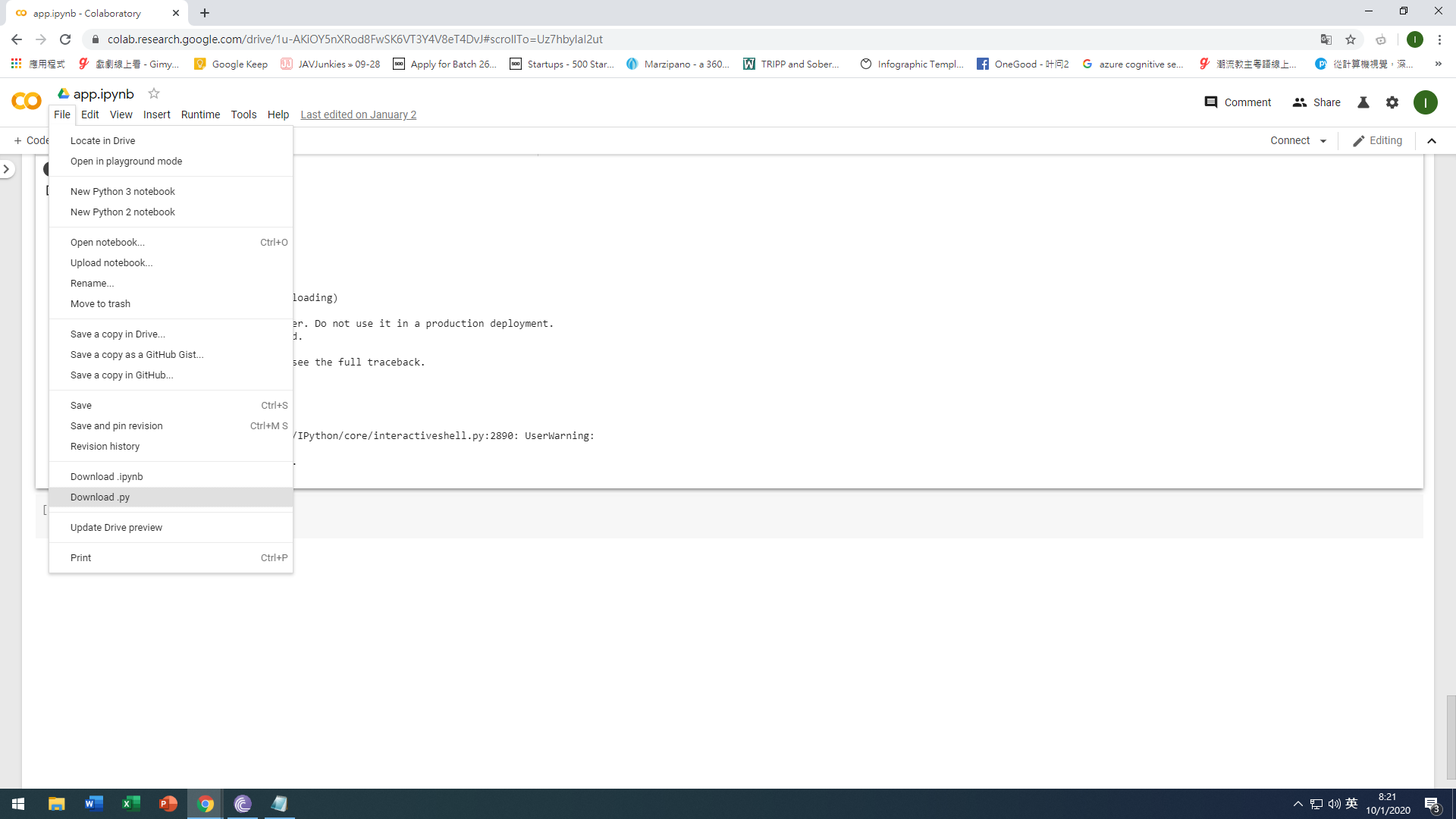
1. Install the code

|  |
| --- |
| app.layout = html.Div([  # Setting the main title of the Dashboard  html.H1("Facebook Data Analysis", style={"textAlign": "center"}),  # Dividing the dashboard in tabs  dcc.Tabs(id="tabs", children=[  # Defining the layout of the first Tab  dcc.Tab(label='Stock Prices', children=[  html.Div([  html.H1("Facebook Stocks High vs Lows",  style={'textAlign': 'center'}),  # Adding the first dropdown menu and the subsequent time-series graph  dcc.Dropdown(id='my-dropdown',  options=[{'label': 'Tesla', 'value': 'TSLA'},  {'label': 'Apple','value': 'AAPL'},  {'label': 'Facebook', 'value': 'FB'},  {'label': 'Microsoft','value': 'MSFT'}],  multi=True,value=['FB'],  style={"display": "block", "margin-left": "auto",  "margin-right": "auto", "width": "60%"}),  dcc.Graph(id='highlow'),  html.H1("Facebook Market Volume", style={'textAlign': 'center'}),  # Adding the second dropdown menu and the subsequent time-series graph  dcc.Dropdown(id='my-dropdown2',  options=[{'label': 'Tesla', 'value': 'TSLA'},  {'label': 'Apple','value': 'AAPL'},  {'label': 'Facebook', 'value': 'FB'},  {'label': 'Microsoft','value': 'MSFT'}],  multi=True,value=['FB'],  style={"display": "block", "margin-left": "auto",  "margin-right": "auto", "width": "60%"}),  dcc.Graph(id='volume')  ], className="container"),  ]),  # Defining the layout of the second tab  dcc.Tab(label='Performance Metrics', children=[  html.H1("Facebook Metrics Distributions",  style={"textAlign": "center"}),  # Adding a dropdown menu and the subsequent histogram graph  html.Div([  html.Div([dcc.Dropdown(id='feature-selected1',  options=[{'label': i.title(),  'value': i} for i in df2.columns.values[1:]],  value="Type")],  className="twelve columns",  style={"display": "block", "margin-left": "auto",  "margin-right": "auto", "width": "60%"}),  ], className="row",  style={"padding": 50, "width": "60%",  "margin-left": "auto", "margin-right": "auto"}),  dcc.Graph(id='my-graph2'),  ])  ])  ]) |

1. Install the code

|  |
| --- |
| @app.callback(Output('highlow', 'figure'),  [Input('my-dropdown', 'value')])  def update\_graph(selected\_dropdown):  dropdown = {"TSLA": "Tesla","AAPL": "Apple","FB": "Facebook","MSFT": "Microsoft",}  trace1 = []  trace2 = []  for stock in selected\_dropdown:  trace1.append(  go.Scatter(x=df[df["Stock"] == stock]["Date"],  y=df[df["Stock"] == stock]["High"],  mode='lines', opacity=0.7,  name=f'High {dropdown[stock]}',textposition='bottom center'))  trace2.append(  go.Scatter(x=df[df["Stock"] == stock]["Date"],  y=df[df["Stock"] == stock]["Low"],  mode='lines', opacity=0.6,  name=f'Low {dropdown[stock]}',textposition='bottom center'))  traces = [trace1, trace2]  data = [val for sublist in traces for val in sublist]  figure = {'data': data,  'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',  '#FF7400', '#FFF400', '#FF0056'],  height=600,  title=f"High and Low Prices for {', '.join(str(dropdown[i]) for i in selected\_dropdown)} Over Time",  xaxis={"title":"Date",  'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',  'step': 'month',  'stepmode': 'backward'},  {'count': 6, 'label': '6M',  'step': 'month',  'stepmode': 'backward'},  {'step': 'all'}])},  'rangeslider': {'visible': True}, 'type': 'date'},  yaxis={"title":"Price (USD)"})}  return figure  @app.callback(Output('volume', 'figure'),  [Input('my-dropdown2', 'value')])  def update\_graph(selected\_dropdown\_value):  dropdown = {"TSLA": "Tesla","AAPL": "Apple","FB": "Facebook","MSFT": "Microsoft",}  trace1 = []  for stock in selected\_dropdown\_value:  trace1.append(  go.Scatter(x=df[df["Stock"] == stock]["Date"],  y=df[df["Stock"] == stock]["Volume"],  mode='lines', opacity=0.7,  name=f'Volume {dropdown[stock]}', textposition='bottom center'))  traces = [trace1]  data = [val for sublist in traces for val in sublist]  figure = {'data': data,  'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',  '#FF7400', '#FFF400', '#FF0056'],  height=600,  title=f"Market Volume for {', '.join(str(dropdown[i]) for i in selected\_dropdown\_value)} Over Time",  xaxis={"title":"Date",  'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',  'step': 'month',  'stepmode': 'backward'},  {'count': 6, 'label': '6M',  'step': 'month',  'stepmode': 'backward'},  {'step': 'all'}])},  'rangeslider': {'visible': True}, 'type': 'date'},  yaxis={"title":"Transactions Volume"})}  return figure  @app.callback(  dash.dependencies.Output('my-graph2', 'figure'),  [dash.dependencies.Input('feature-selected1', 'value')])  def update\_graph(selected\_feature1):  if selected\_feature1 == None:  selected\_feature1 = 'Type'  trace = go.Histogram(x= df2.Type,  marker=dict(color='rgb(0, 0, 100)'))  else:  trace = go.Histogram(x=df2[selected\_feature1],  marker=dict(color='rgb(0, 0, 100)'))  return {  'data': [trace],  'layout': go.Layout(title=f'Metrics considered: {selected\_feature1.title()}',  colorway=["#EF963B", "#EF533B"], hovermode="closest",  xaxis={'title': "Distribution",  'titlefont': {'color': 'black', 'size': 14},  'tickfont': {'size': 14, 'color': 'black'}},  yaxis={'title': "Frequency",  'titlefont': {'color': 'black', 'size': 14, },  'tickfont': {'color': 'black'}})} |

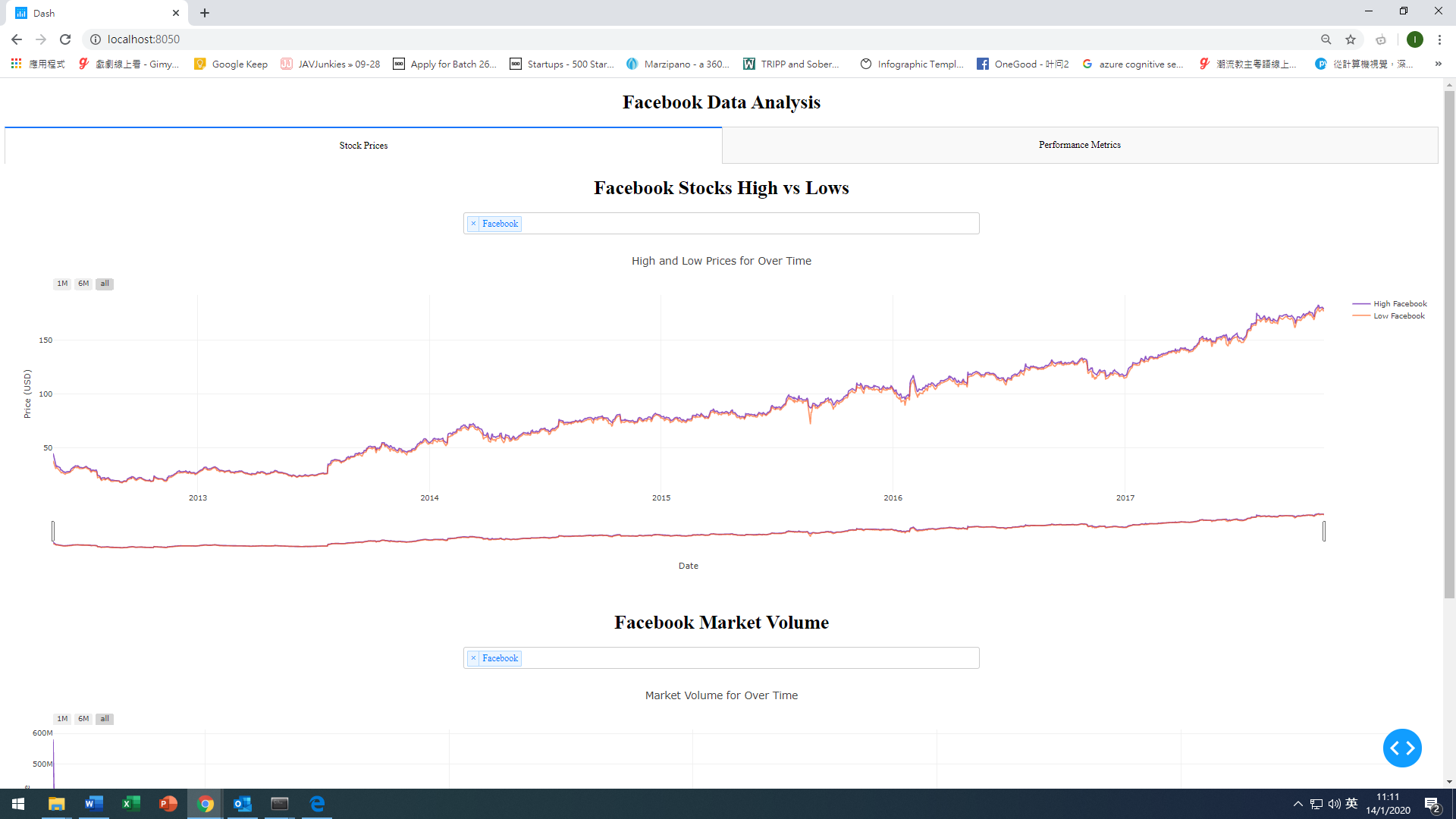
1. Download the file as app.py and stored in C:\Users\[User[\interactivedashboards-datascience

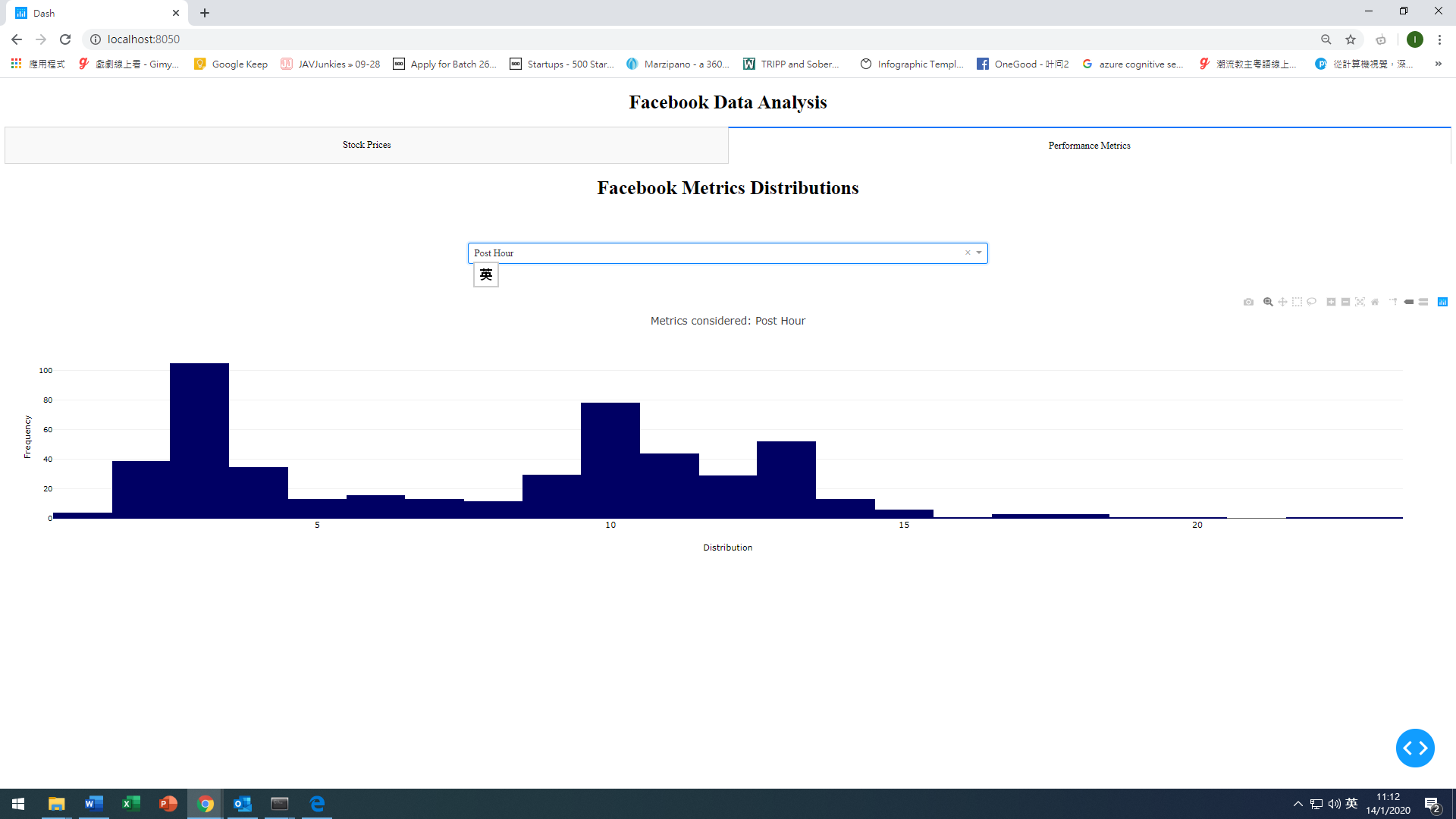


1. Open the app.py in the C:\Users\[User[\interactivedashboards-datascience
2. Add the following two lines of code which start a local server and make our application run

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  app.run\_server(debug=True) |

1. Run the script “*python app.py*”
2. This will start a web server at <http://127.0.0.1:8050/>, going at this link we will see our final Python App running.
3. You can see the following screen





## 