National College of Ireland

Sector Based Stock Market Prediction In USA - Configuration Manual

MSc Research Project
Data Analytics

Muhammad Nizam Uddin
Student ID: x14127032

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Muhammad Nizam Uddin
Student ID:	x14127032
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Catherine Mulwa
Submission Due Date:	16/08/2021
Project Title:	Sector Based Stock Market Prediction In USA - Configuration Manual
Word Count:	587
Page Count:	25

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	18th September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Sector Based Stock Market Prediction In USA - Configuration Manual

Muhammad Nizam Uddin
x14127032

1 Overview

This configuration manual describes hardware specification, software requirements and different stages of implementation of research project and step by step guide to re-produce the project. - **"Sector Based Stock Market Prediction In USA"**

Chapter 2 will described System Features, Chapter 3 Tools used , Chapter 4 Data processing , Chapter 5 Project work flow process and Chapter 6 will give you troubleshoot guides.

2 System Features

Below Figure 1 shows the Dell laptop configuration used for the Project.

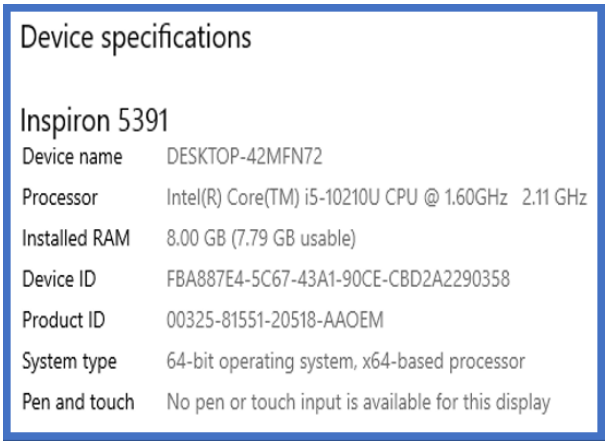


Figure 1: System Features

3 Tools Used

The tools used in this research are:

1. Microsoft Excel: The data is available in a .csv format, hence, excel was used to store the data.

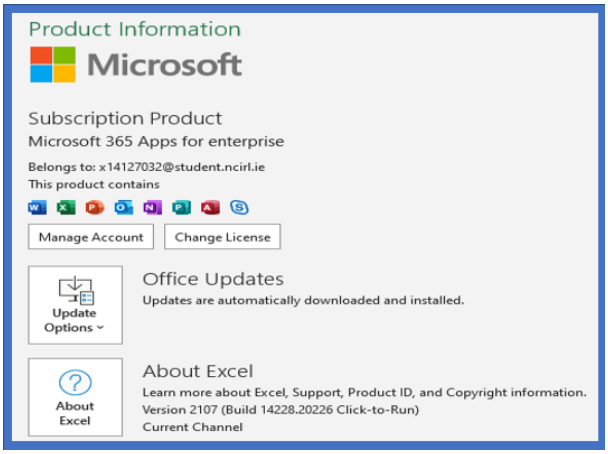


Figure 2: Microsoft Excel Version 2017.png

2. Python: All the programming has been done using Python [1]programming. So its important its has been installed. There are 2 version of python currently available 2 and 3. Please install python version 3.9.5 to run this project.(It will take approx. 15 minutes)

Figure 3: Python Version 3.9.5

For Anaconda Anaconda [2] Individual Edition 2020.11 has been used . Please install this version. (It will take approx. 20 to 25 minutes)

Libraries Imported : Anaconda Normally install with following libraries but Once the conda environment setup , it's best to check you have the updated version .

1. Numpy
2. Pandas
3. Matplotlib
4. Seaborn
5. SkLearn
6. TensorFlow

[1]https://www.python.org/downloads/
[2]https://www.anaconda.com/products/individual

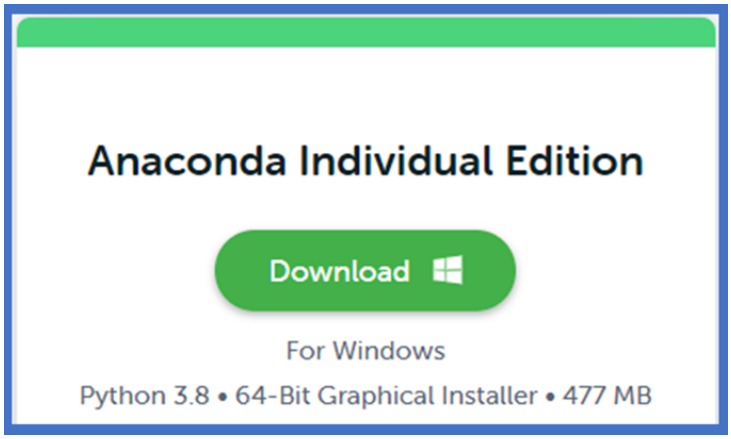


Figure 4: Anaconda Individual Edition 2020.11

7. Keras

```
In [1]: #Import libraries

        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import datetime as dt
        from statsmodels.graphics.tsaplots import plot_acf
        from statsmodels.tsa.stattools import acf
        from statsmodels.tsa.stattools import pacf
        from statsmodels.tsa.arima_model import ARIMA
        from statsmodels.tsa.arima_process import ArmaProcess
        from keras.models import Sequential
        from keras.layers import Dense, LSTM
        from sklearn.preprocessing import MinMaxScaler
        from pandas.plotting import register_matplotlib_converters
        from statsmodels.tsa import stattools
        from sklearn.metrics import r2_score
```

Figure 5: Python Libraries.png

4 Data Processing

For this research, we used Data from 3 different sources[3] [4] [5]. All the data sources that have been used in the research are CCO Certified for public use and the research is hence GDPR compliant. Certificate can be view able. [6] Please download the data from the web to your local folder and see the process shown below in the info-graphic visual in figure 6

Once data collection is completed run the following code step by step to clean and create sector based index and data. These step also will do some preliminary data exploration as well shown from figure 7 to figure 13

[3]https://datahub.io/core/nyse-other-listings#resource-nyse-listed
[4]https://www.kaggle.com/camnugent/sandp500
[5]https://www.kaggle.com/agailloty/fortune10,00?select=fortune1000.csv
[6]https://creativecommons.org/publicdomain/zero/1.0/

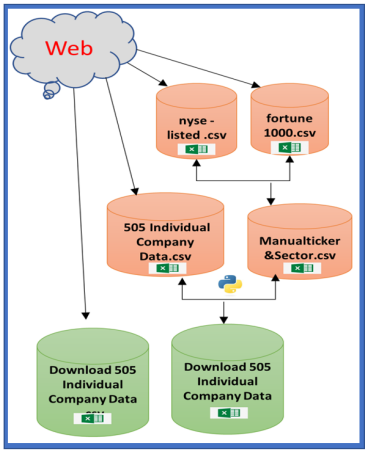


Figure 6: Data Processing Workflow

Run Code to create sector Based data and index

```
[1]: #To read the data frame we will need to import Pandas first.
     import pandas as pd

     # Visualization Imports
     import matplotlib.pyplot as plt
     from matplotlib.pyplot import *
     import seaborn as sns
     color = sns.color_palette()
     get_ipython().run_line_magic('matplotlib', 'inline')
     import plotly.offline as py
     py.init_notebook_mode(connected=True)
     import plotly.graph_objs as go
     import plotly.tools as tls
     import plotly.express as px
     import numpy as np

[2]: # read the file, create a data frame and store data
     dataset1 = pd.read_csv("all_stocks_5yr.csv")
     dataset2 = pd.read_csv("Sector & Ticker.csv")

[3]: #check the head of the data frame dataset1
     dataset1.head()
```

[3]:

	date	open	high	low	close	volume	Name
0	08/02/2013	15.07	15.12	14.63	14.75	8407500	AAL
1	11/02/2013	14.89	15.01	14.26	14.46	8882000	AAL
2	12/02/2013	14.45	14.51	14.10	14.27	8126000	AAL
3	13/02/2013	14.30	14.94	14.25	14.66	10259500	AAL
4	14/02/2013	14.94	14.96	13.16	13.99	31879900	AAL

```
[4]: #check the head of the data frame dataset2
     dataset2.head()
```

[4]:

	Name	Sector
0	AAP	Retailing
1	AAP	Retailing
2	AAP	Retailing
3	AAP	Retailing
4	AAP	Retailing

Figure 7: Preparation of Sector based data Step 1

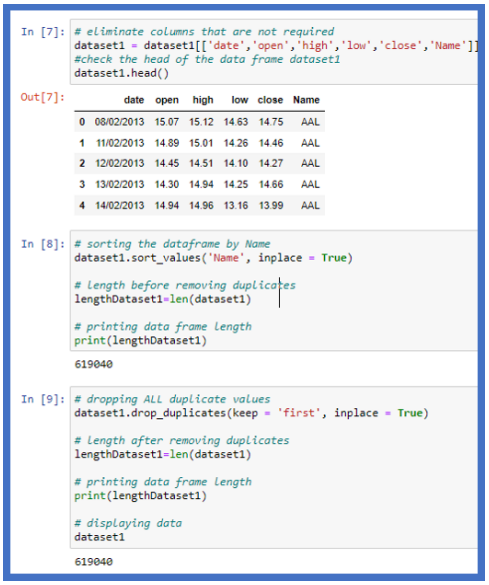
```
In [5]: # sorting the dataframe by Name
        dataset2.sort_values('Name', inplace = True)

        # length before removing duplicates
        lengthDataset2=len(dataset2)

        # printing data frame length
        print(lengthDataset2)
```

```
398608
```

```
In [6]: # dropping ALL duplicate values
        dataset2.drop_duplicates(keep = 'first', inplace = True)

        # length after removing duplicates
        lengthDataset2=len(dataset2)

        # printing data frame length
        print(lengthDataset2)

        # displaying data
        dataset2
```

```
319
```

Figure 8: Preparation of Sector based data Step 2



```
In [7]: # eliminate columns that are not required
dataset1 = dataset1[['date','open','high','low','close','Name']]
#check the head of the data frame dataset1
dataset1.head()
```

Out[7]:

	date	open	high	low	close	Name
0	08/02/2013	15.07	15.12	14.63	14.75	AAL
1	11/02/2013	14.89	15.01	14.26	14.46	AAL
2	12/02/2013	14.45	14.51	14.10	14.27	AAL
3	13/02/2013	14.30	14.94	14.25	14.66	AAL
4	14/02/2013	14.94	14.96	13.16	13.99	AAL

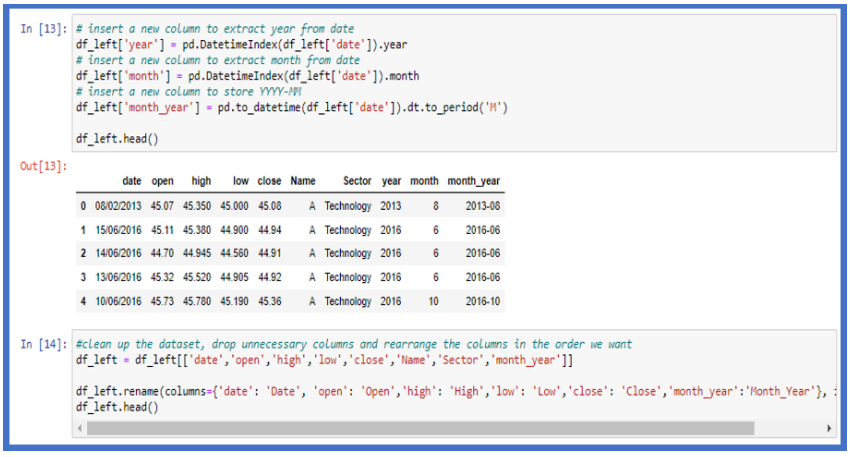
```
In [8]: # sorting the dataframe by Name
dataset1.sort_values('Name', inplace = True)

# length before removing duplicates
lengthDataset1=len(dataset1)

# printing data frame length
print(lengthDataset1)
```

```
619040
```

```
In [9]: # dropping ALL duplicate values
dataset1.drop_duplicates(keep = 'first', inplace = True)

# length after removing duplicates
lengthDataset1=len(dataset1)

# printing data frame length
print(lengthDataset1)

# displaying data
dataset1
```

```
619040
```

Figure 9: Preparation of Sector based data Step 3



```
In [13]: # insert a new column to extract year from date
df_left['year'] = pd.DatetimeIndex(df_left['date']).year
# insert a new column to extract month from date
df_left['month'] = pd.DatetimeIndex(df_left['date']).month
# insert a new column to store YYYY-MM
df_left['month_year'] = pd.to_datetime(df_left['date']).dt.to_period('M')

df_left.head()
```

Out[13]:

	date	open	high	low	close	Name	Sector	year	month	month_year
0	08/02/2013	45.07	45.350	45.000	45.08	A	Technology	2013	8	2013-08
1	15/06/2016	45.11	45.380	44.900	44.94	A	Technology	2016	6	2016-06
2	14/06/2016	44.70	44.945	44.560	44.91	A	Technology	2016	6	2016-06
3	13/06/2016	45.32	45.520	44.905	44.92	A	Technology	2016	6	2016-06
4	10/06/2016	45.73	45.780	45.190	45.36	A	Technology	2016	10	2016-10

```
In [14]: #clean up the dataset, drop unnecessary columns and rearrange the columns in the order we want
df_left = df_left[['date','open','high','low','close','Name','Sector','month_year']]

df_left.rename(columns={'date': 'Date', 'open': 'Open','high': 'High','low': 'Low','close': 'Close','month_year':'Month_Year'}, i
df_left.head()
```

Figure 10: Preparation of Sector based data Step 4

```
In [15]: #visualise the various sectors
dist = df_left['Sector'].value_counts()
colors = ['mediumturquoise', 'darkorange']
trace = go.Pie(values=(np.array(dist)),labels=dist.index)
layout = go.Layout(title='Sector Split')
data = [trace]
fig = go.Figure(trace,layout)
fig.update_traces(marker=dict(colors=colors, line=dict(color='#000000', width=2)))
fig.show()
```

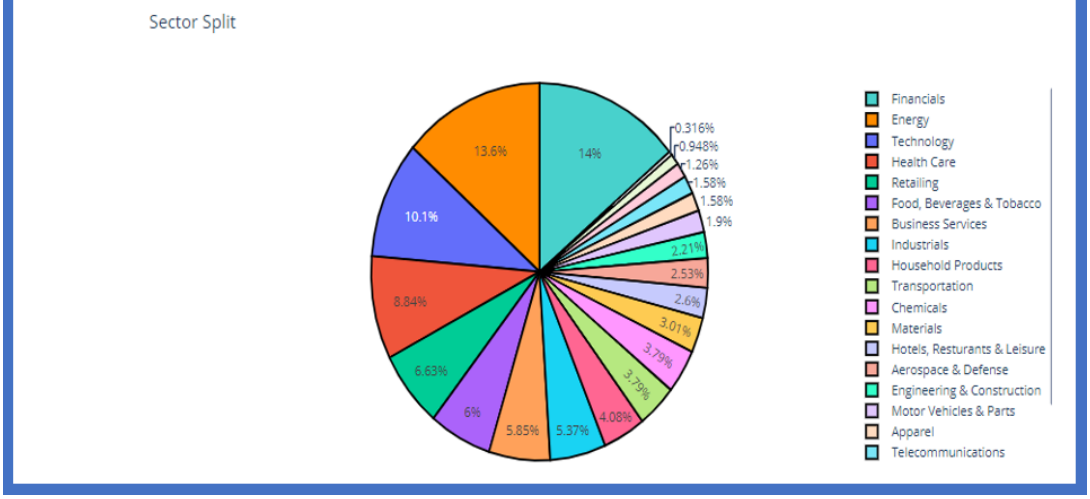


Figure 11: Preparation of Sector based data Step 5

```
In [17]: #visualize the close by Sector values
fig = px.box(df_left, x='Close', y='Sector')
fig.update_traces(marker_color="midnightblue",marker_line_color='rgb(8,48,107)',
                  marker_line_width=1.5)
fig.update_layout(title_text='Close values by sector')
fig.show()
```

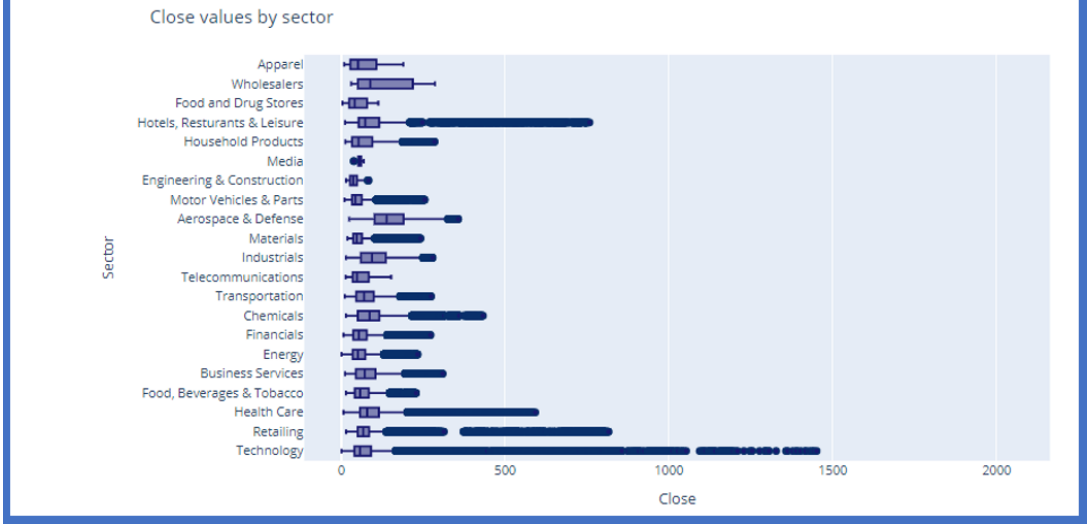


Figure 12: Preparation of Sector based data Step 6

```
In [23]: # export the clean data set to a comma separated file
         df_left.to_csv("CleanData.csv")

In [24]: ## group data by sector and month year
         df_sector= df_left.groupby(['Date','Sector'],as_index=False)[['Open','High','Low','Close']].sum()

In [25]: # insert a new column to store YYYY-MM
         df_sector['Year_Month'] = pd.to_datetime(df_left['Date']).dt.to_period('M')

         df_sector
```

Out[25]:

	Date	Sector	Open	High	Low	Close	Year_Month
0	01/02/2016	Aerospace & Defense	1150.350	1168.1000	1138.1751	1158.640	2013-08
1	01/02/2016	Apparel	317.920	325.2600	314.9950	322.405	2016-06
2	01/02/2016	Business Services	1518.540	1547.1550	1503.5151	1531.280	2016-06
3	01/02/2016	Chemicals	1001.610	1027.0795	988.8111	1021.380	2016-06
4	01/02/2016	Energy	2092.905	2125.7570	2064.5068	2104.430	2016-10
...	...	...	...	...	...	...	...
26434	31/12/2015	Retailing	2309.370	2323.8684	2292.0651	2294.830	2016-06
26435	31/12/2015	Technology	2982.900	2997.5738	2952.0700	2954.535	2016-06
26436	31/12/2015	Telecommunications	291.220	291.8600	288.9850	289.190	2016-06
26437	31/12/2015	Transportation	873.520	883.0600	867.9800	873.790	2016-08
26438	31/12/2015	Wholesalers	331.420	333.2648	328.9600	329.480	2015-12

26439 rows × 7 columns

```
In [26]: # export the clean data set to a comma separated file
         df_sector.to_csv("Sectorbased_data.csv")
```

Figure 13: Preparation of Sector based data Step 7

So we have our sector based data set ready to run the model for RNN, LSTM and ARIMA. For the individual data already processed , but we will do some checks and exploration while running the model.

5 A step by step guide to execute the Project

Below is step by step guide to execute the process flow. Visual info-graphic flow is the best way to understand the process quickly than explain in the word van der Aalst (2004).

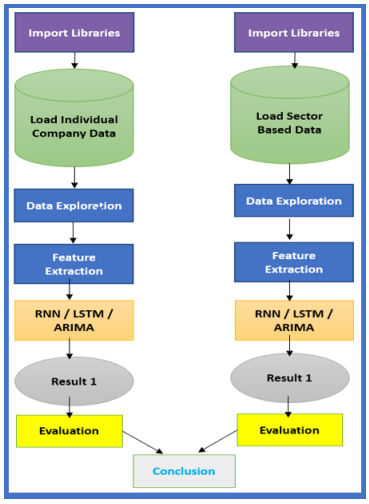


Figure 14: Project Analysis Flow

Now we will Run RNN, LSTM, and ARIMA Model on sector based data as well as

individual stock data. Below is the step by step guide .

5.1 Recurrent Neural Networks - RNN

To run the RNN Model for sector based index and individual companies please Choose the "RNN_Student No -x14127032 - Sector Based Stock Market Prediction In USA" ipython notebook and follow the step from figure 15 to figure 33 .Please note to train the model,change the epoch number as required. code - model2.fit(x_train2, y_train2, batch_size=1, epochs=100)

```
In [273]: ### Msc Data Analytics
          ### "Sector Based Stock Market Prediction In USA"
          ### Student No: x14127032#Import libraries

          import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import seaborn as sns
          sns.set_style('whitegrid')
          plt.style.use("fivethirtyeight")
          %matplotlib inline
          # For time stamps
          from datetime import datetime

          # For reading stock data from yahoo
          #from pandas_datareader.data import DataReader

In [274]: #Import libraries

          import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import seaborn as sns
          import datetime as dt
          from statsmodels.graphics.tsaplots import plot_acf
          from statsmodels.tsa.stattools import acf
          from statsmodels.tsa.stattools import pacf
          from statsmodels.tsa.arima_model import ARIMA
          from statsmodels.tsa.arima_process import ArmaProcess
          from keras.models import Sequential
          from keras.layers import Dense, LSTM,GRU
          from sklearn.preprocessing import MinMaxScaler
          from pandas.plotting import register_matplotlib_converters
          from statsmodels.tsa import stattools
          from sklearn.metrics import r2_score

In [275]: sector_based_T = pd.read_csv('Sectorbased_data.csv')
```

Figure 15: RNN Model - Step 1

```
In [277]: sector_based_T = sector_based_T[sector_based_T['Sector'] == "Technology"]

In [278]: sector_based_T.head()

Out[278]:
```

	Date	Open	High	Low	Close	Volume	Name	Sector	Year
5036	08/02/2013	2.580	2.60	2.58	2.59	5971768	AMD	Technology	2013
5037	11/02/2013	2.590	2.70	2.59	2.67	22410941	AMD	Technology	2013
5038	12/02/2013	2.705	2.78	2.69	2.77	13675442	AMD	Technology	2013
5039	13/02/2013	2.810	2.83	2.73	2.75	11481985	AMD	Technology	2013
5040	14/02/2013	2.740	2.75	2.69	2.75	13283766	AMD	Technology	2013

```
In [279]: sector_based_T = sector_based_T.drop("Sector",axis=1)

In [280]: sector_based_T = sector_based_T.sort_values(by='Date', ascending=False)

In [281]: sector_based_T
```

Figure 16: RNN Model - Step 2

```
In [282]: sector_based_F = pd.read_csv('Sectorbased_data.csv')

In [283]: sector_based_F.head()
```

Out[283]:

	Date	Open	High	Low	Close	Volume	Name	Sector	Year
0	08/02/2013	38.76	39.03	38.51	38.79	13112320	AIG	Financials	2013
1	11/02/2013	38.89	39.56	38.65	39.45	14230893	AIG	Financials	2013
2	12/02/2013	39.50	39.90	38.50	38.63	25676629	AIG	Financials	2013
3	13/02/2013	38.93	39.18	38.56	38.87	16533791	AIG	Financials	2013
4	14/02/2013	38.64	39.26	38.50	39.21	18321181	AIG	Financials	2013

```
In [284]: sector_based_F = sector_based_F[sector_based_F['Sector'] == "Financials"]

In [285]: sector_based_F = sector_based_F.drop("Sector",axis=1)

In [286]: sector_based_F.head()
```

Out[286]:

	Date	Open	High	Low	Close	Volume	Name	Year
0	08/02/2013	38.76	39.03	38.51	38.79	13112320	AIG	2013
1	11/02/2013	38.89	39.56	38.65	39.45	14230893	AIG	2013
2	12/02/2013	39.50	39.90	38.50	38.63	25676629	AIG	2013
3	13/02/2013	38.93	39.18	38.56	38.87	16533791	AIG	2013
4	14/02/2013	38.64	39.26	38.50	39.21	18321181	AIG	2013

```
In [287]: sector_based_F = sector_based_F.sort_values(by='Date', ascending=False)

In [288]: sector_based_F=sector_based_F.reset_index(drop=True)
          sector_based_T=sector_based_T.reset_index(drop=True)

In [289]: sector_based_F = sector_based_F.drop("Year",axis=1)
          sector_based_T = sector_based_T.drop("Year",axis=1)
```

Figure 17: RNN Model - Step 3

```
In [287]: sector_based_F = sector_based_F.sort_values(by='Date', ascending=False)

In [288]: sector_based_F=sector_based_F.reset_index(drop=True)
          sector_based_T=sector_based_T.reset_index(drop=True)

In [289]: sector_based_F = sector_based_F.drop("Year",axis=1)
          sector_based_T = sector_based_T.drop("Year",axis=1)

In [290]:
          sector_based_F.columns =["date","open","high","low","close","volume","Name"]
          sector_based_T.columns =["date","open","high","low","close","volume","Name"]

In [291]: # The tech stocks we'll use for this analysis
          Financial_list = ['BAC','JPM','AAPL','AMZN',"Finan","TECH"]

In [292]: BAC= pd.read_csv('BAC_data.csv')
          JPM = pd.read_csv('JPM_data.csv')
          AAPL = pd.read_csv('AAPL_data.csv')
          AMZN = pd.read_csv('AMZN_data.csv')
          company_list = [BAC, JPM,AAPL,AMZN,sector_based_F, sector_based_T]
          company_name = ["BAC", "JPM","AAPL","AMZN", "Finance", "Technology"]

In [293]: df = pd.concat(company_list, axis=0)
          df.tail(10)
```

Figure 18: RNN Model - Step 4

In [302]:
```
AMZN.describe()
```

Out[302]:

	open	high	low	close	volume
count	1259.000000	1259.000000	1259.000000	1259.000000	1.259000e+03
mean	576.867264	582.017221	571.113517	576.880041	3.730465e+06
std	282.500019	284.417123	280.215237	282.500395	2.166506e+06
min	248.940000	252.930000	245.750000	248.230000	1.092970e+06
25%	325.870000	329.485000	322.185000	325.800000	2.511165e+06
50%	506.000000	512.330000	495.640000	503.820000	3.144719e+06
75%	777.620000	781.845000	770.720000	777.420000	4.220246e+06
max	1477.390000	1498.000000	1450.040000	1450.890000	2.385606e+07

In [303]:
```
for i, company in enumerate(company_list, 1):
  print(i,company)
```

Figure 19: RNN Model - Step 5

In [305]:
```
Financial_list
```

Out[305]:
```
['BAC', 'JPM', 'AAPL', 'AMZN', 'Finan', 'TECH']
```

In [306]:
```
# Let's see a historical view of the closing price


plt.figure(figsize=(15, 6))
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company_list, 1):
    plt.subplot(3, 2, i)
    company['close'].plot()
    plt.ylabel('close')
    plt.xlabel(None)
    plt.title(f"Closing Price of {Financial_list[i - 1]}")

plt.tight_layout()
```

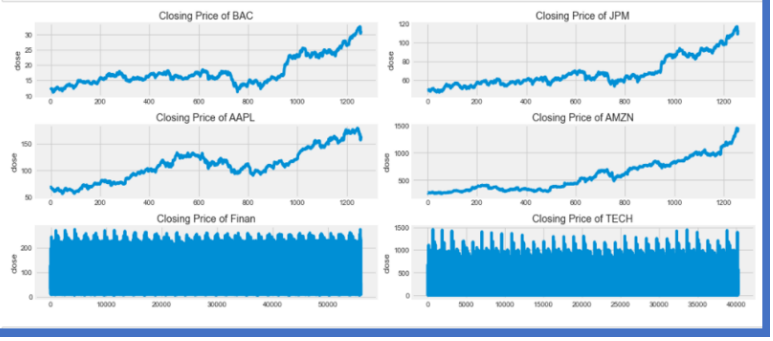


Figure 20: RNN Model - Step 6

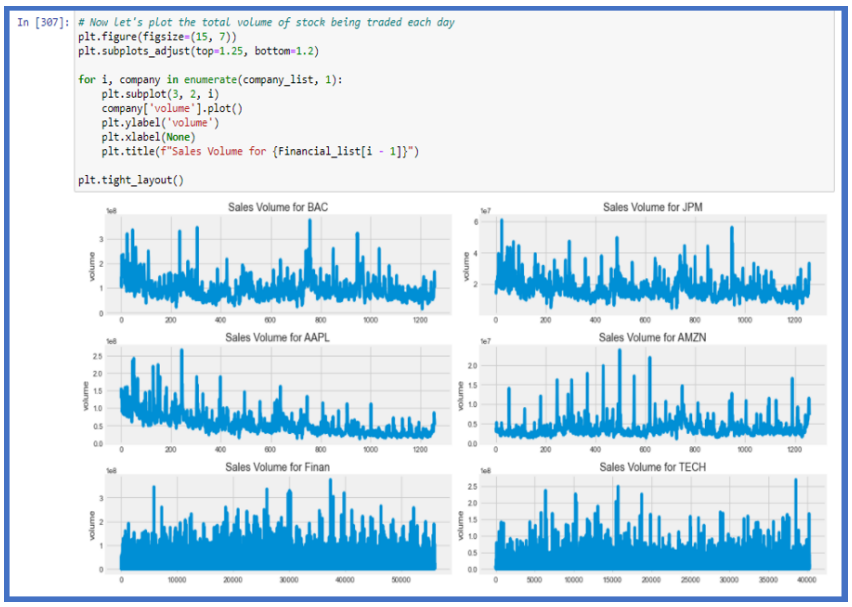


Figure 21: RNN Model - Step 7

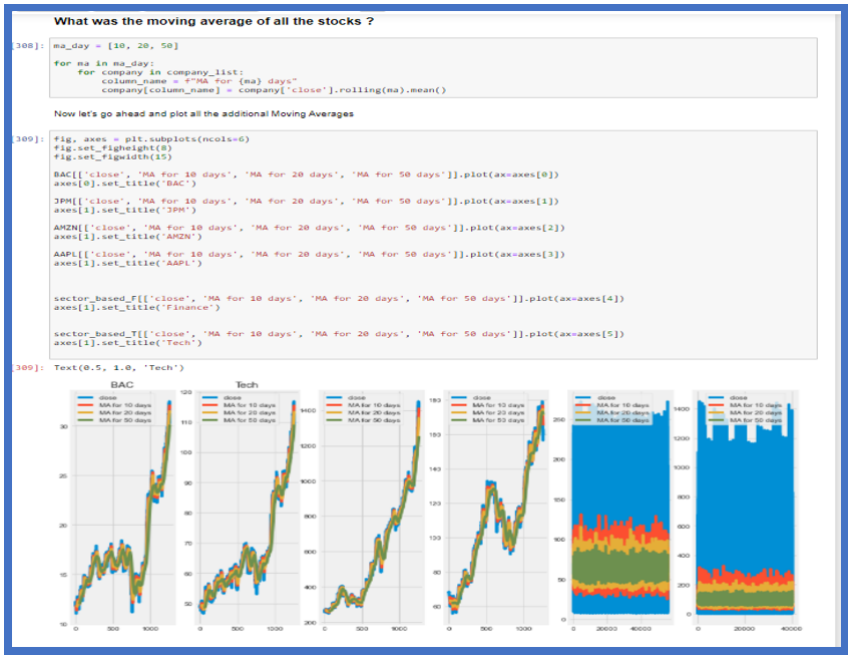


Figure 22: RNN Model - Step 8

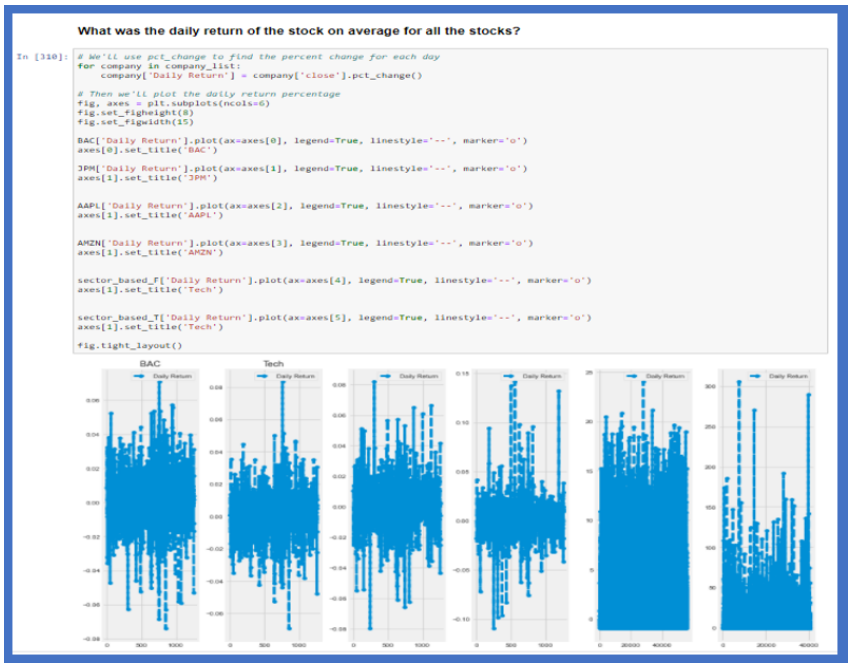
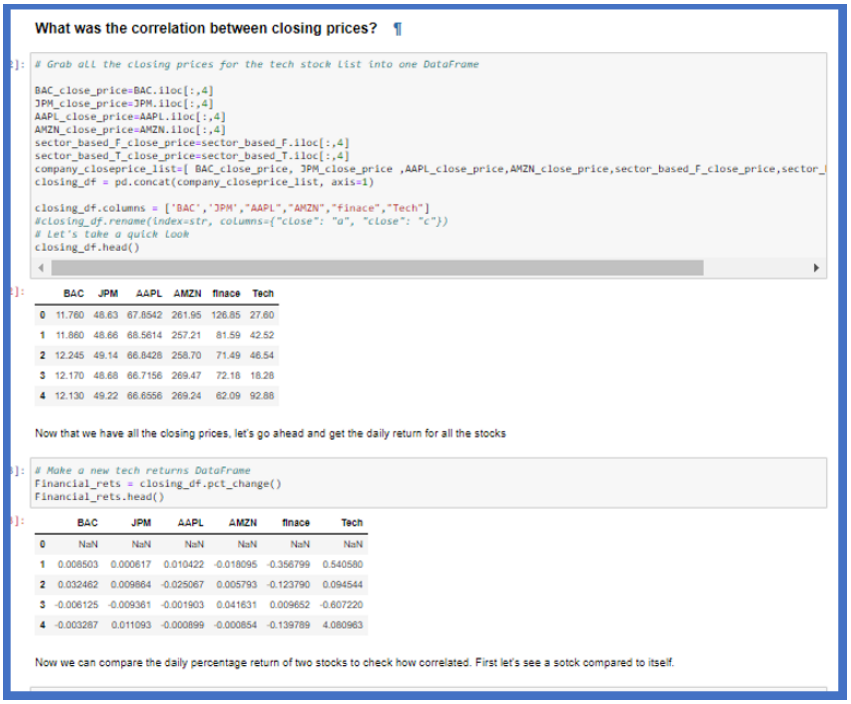


Figure 23: RNN Model - Step 9

Figure 24: RNN Model - Step 10



What was the correlation between closing prices? ¶

```
# Grab all the closing prices for the tech stock list into one DataFrame

BAC_close_price=BAC.iloc[:,4]
JPM_close_price=JPM.iloc[:,4]
AAPL_close_price=AAPL.iloc[:,4]
AMZN_close_price=AMZN.iloc[:,4]
sector_based_F_close_price=sector_based_F.iloc[:,4]
sector_based_T_close_price=sector_based_T.iloc[:,4]
company_closeprice_list=[ BAC_close_price, JPM_close_price ,AAPL_close_price,AMZN_close_price,sector_based_F_close_price,sector_
closing_df = pd.concat(company_closeprice_list, axis=1)

closing_df.columns = ['BAC','JPM',"AAPL","AMZN","finace","Tech"]
#closing_df.rename(index=str, columns={"close": "a", "close": "c"})
# Let's take a quick look
closing_df.head()
```

	BAC	JPM	AAPL	AMZN	finace	Tech
0	11.760	48.63	67.8542	261.95	126.85	27.60
1	11.860	48.66	68.5614	257.21	81.59	42.52
2	12.245	49.14	66.8428	258.70	71.49	46.54
3	12.170	48.68	66.7156	269.47	72.18	18.28
4	12.130	49.22	66.6556	269.24	62.09	92.88

Now that we have all the closing prices, let's go ahead and get the daily return for all the stocks

```
# Make a new tech returns DataFrame
Financial_rets = closing_df.pct_change()
Financial_rets.head()
```

	BAC	JPM	AAPL	AMZN	finace	Tech
0	NaN	NaN	NaN	NaN	NaN	NaN
1	0.008503	0.000617	0.010422	-0.018095	-0.356799	0.540580
2	0.032462	0.009864	-0.025067	0.005793	-0.123790	0.094544
3	-0.006125	-0.009361	-0.001903	0.041631	0.009652	-0.607220
4	-0.003287	0.011093	-0.000899	-0.000854	-0.139789	4.080963

Now we can compare the daily percentage return of two stocks to check how correlated. First let's see a sotck compared to itself.

Figure 25: RNN Model - Step 11

```
Out[314]: <seaborn.axisgrid.JointGrid at 0x209f4ec3df0>
```



```
In [315]: # Comparing Google to itself should show a perfectly linear relationship
          sns.jointplot('AAPL','AMZN', Financial_rets, kind='scatter', color='seagreen')

C:\Users\nizam\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword ar
gs: x, y, data. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(

Out[315]: <seaborn.axisgrid.JointGrid at 0x209c5ec00a0>
```



```
In [316]: # We'll use jointplot to compare the daily returns of Google and Microsoft
          sns.jointplot('BAC', 'JPM', Financial_rets, kind='scatter')
```

Figure 26: RNN Model - Step 12

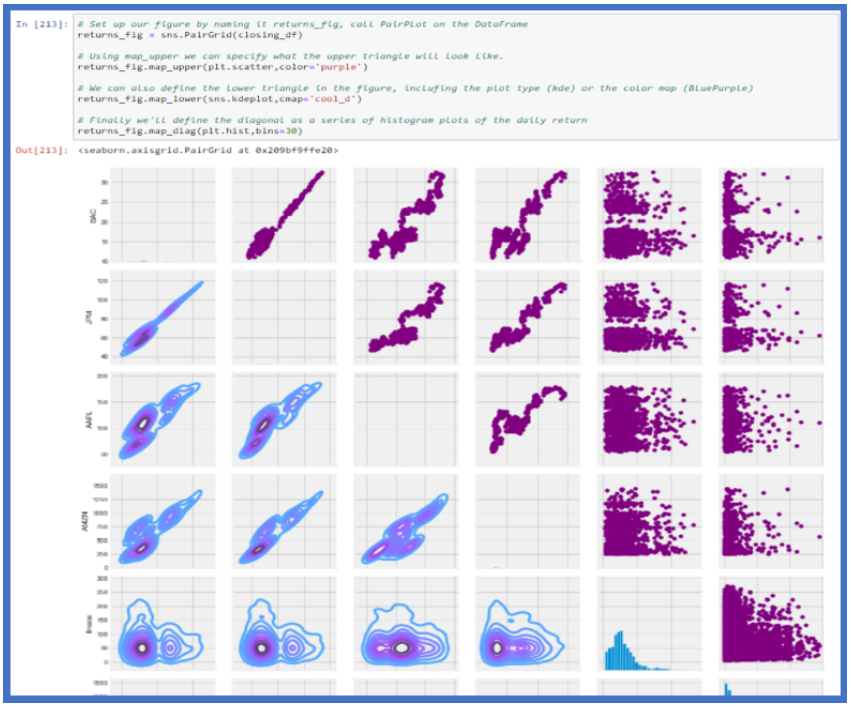


Figure 27: RNN Model - Step 13

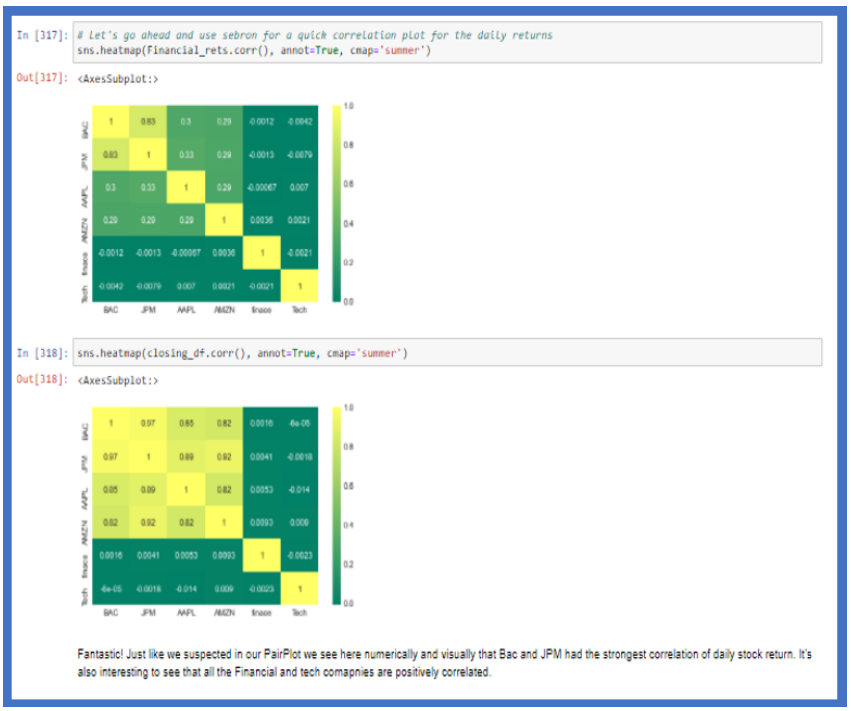


Figure 28: RNN Model - Step 14

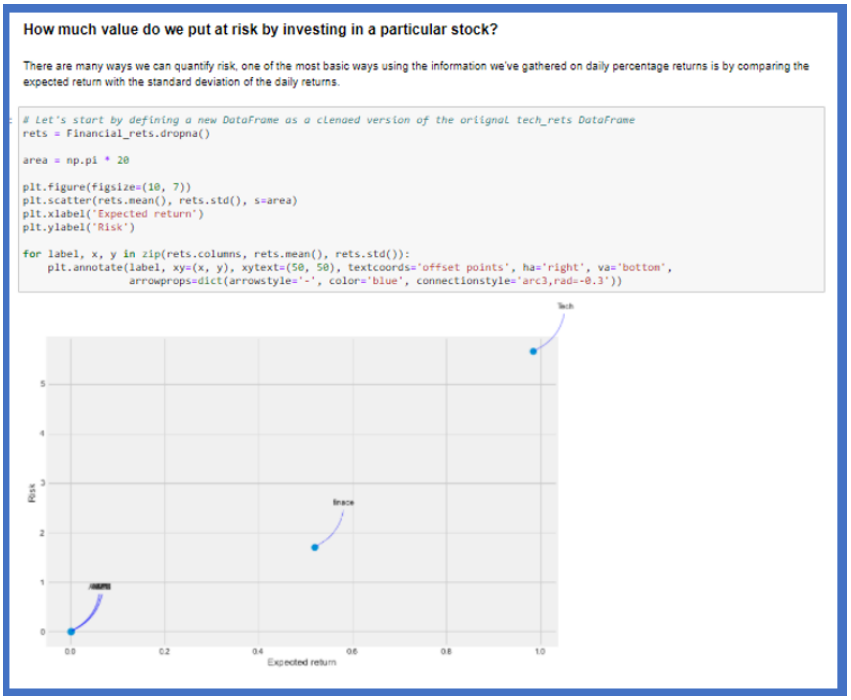


Figure 29: RNN Model - Step 15

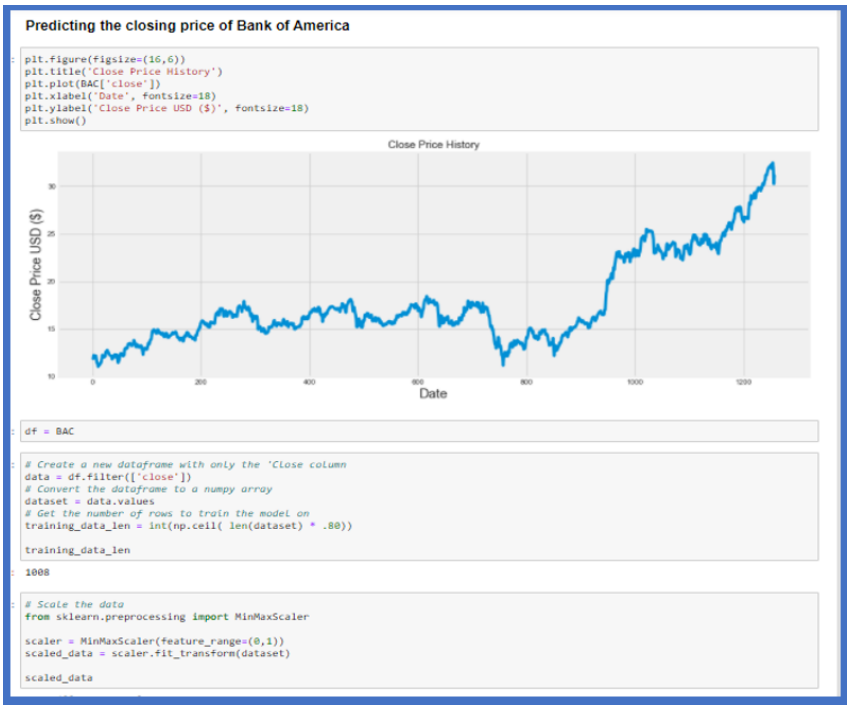
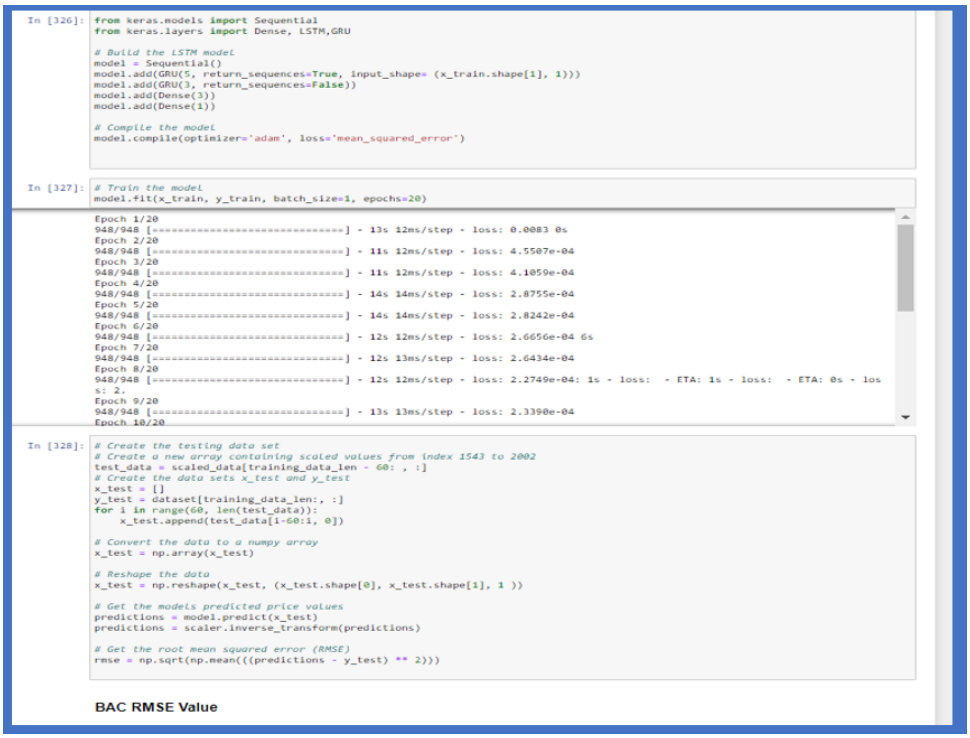


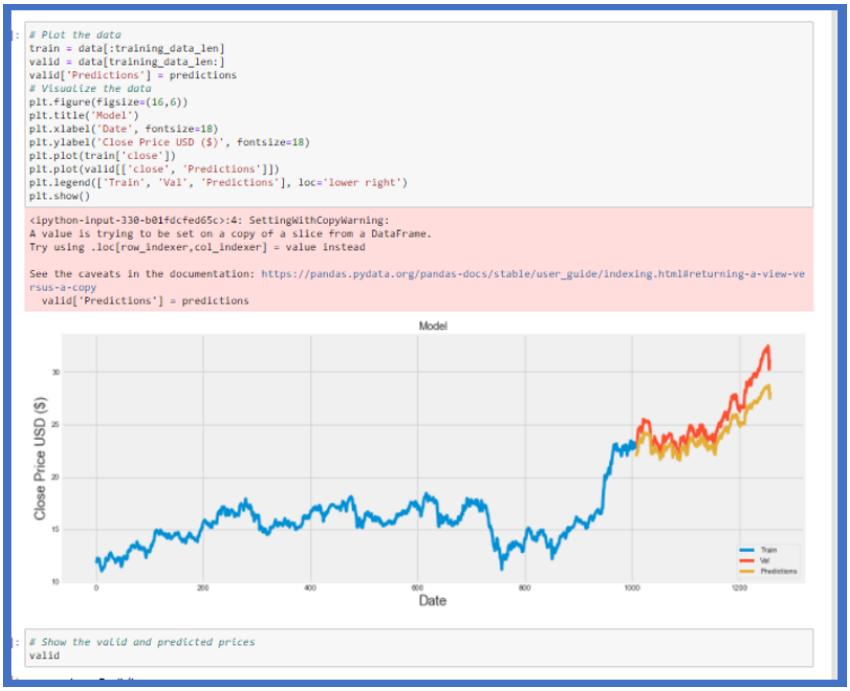
Figure 30: RNN Model - Step 16



```
In [326]:
from keras.models import Sequential
from keras.layers import Dense, LSTM,GRU

# Build the LSTM model
model = Sequential()
model.add(GRU(5, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(GRU(3, return_sequences=False))
model.add(Dense(3))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

In [327]:
# Train the model
model.fit(x_train, y_train, batch_size=1, epochs=20)

Epoch 1/20
948/948 [==============================] - 13s 12ms/step - loss: 0.0083 0s
Epoch 2/20
948/948 [==============================] - 11s 12ms/step - loss: 4.5507e-04
Epoch 3/20
948/948 [==============================] - 11s 12ms/step - loss: 4.1059e-04
Epoch 4/20
948/948 [==============================] - 14s 14ms/step - loss: 2.8755e-04
Epoch 5/20
948/948 [==============================] - 14s 14ms/step - loss: 2.8242e-04
Epoch 6/20
948/948 [==============================] - 12s 12ms/step - loss: 2.6656e-04 6s
Epoch 7/20
948/948 [==============================] - 12s 13ms/step - loss: 2.6434e-04
Epoch 8/20
948/948 [==============================] - 12s 12ms/step - loss: 2.2749e-04: 1s - loss:  - ETA: 1s - loss:  - ETA: 0s - los
s: 2.
Epoch 9/20
948/948 [==============================] - 13s 13ms/step - loss: 2.3390e-04
Epoch 10/20

In [328]:
# Create the testing data set
# Create a new array containing scaled values from index 1543 to 2002
test_data = scaled_data[training_data_len - 60: , :]
# Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

# Convert the data to a numpy array
x_test = np.array(x_test)

# Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

# Get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

# Get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
```

BAC RMSE Value

Figure 31: RNN Model - Step 17



```
# Plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
# Visualize the data
plt.figure(figsize=(16,6))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['close'])
plt.plot(valid[['close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()

<ipython-input-330-b01fdcfed65c>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  valid['Predictions'] = predictions
```



```
# Show the valid and predicted prices
valid
```

Figure 32: RNN Model - Step 18

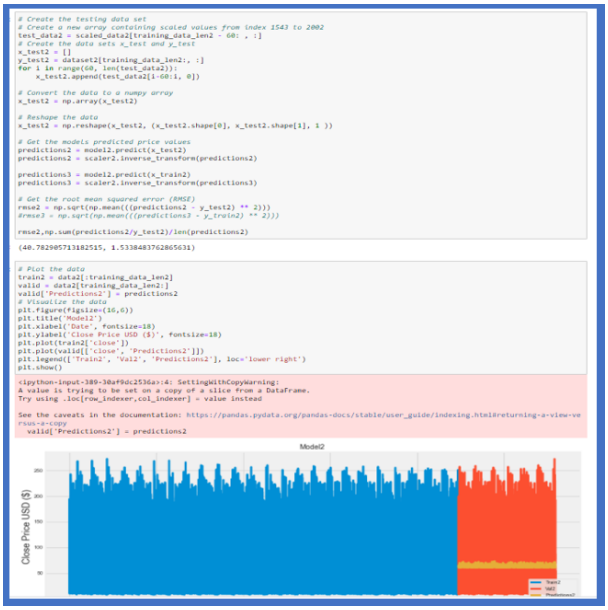


Figure 33: RNN Model - Step 19

5.2 Long Short-Term Memory - LSTM

To run the LSTM Model for sector based index and individual companies please Choose the "LSTM_Student No -x14127032 - Sector Based Stock Market Prediction In USA" ipython notebook and follow the step from figure 34 to figure 41 . Please note to train the model,change the epoch number as required. code - model2.fit(x_train2, y_train2, batch_size=1, epochs=500)

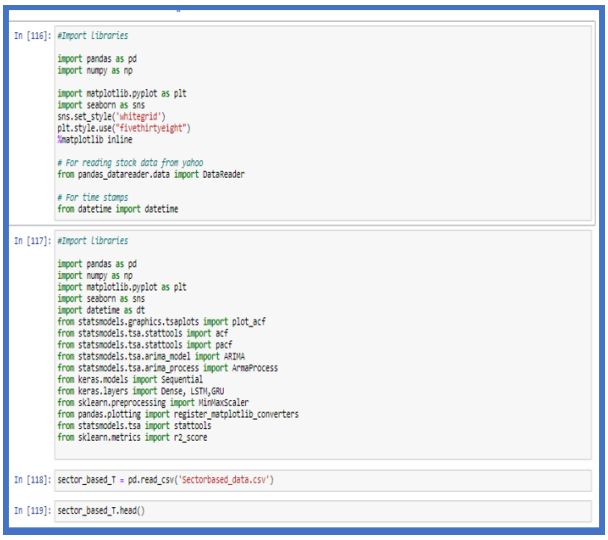


Figure 34: LSTM Model - Step 1

```
In [122]: sector_based_T = sector_based_T.drop("Sector",axis=1)

In [123]: sector_based_T = sector_based_T.sort_values(by='Date', ascending=False)

In [124]: sector_based_T
```

Out[124]:

	Date	Open	High	Low	Close	Volume	Name	Year
45623	31/12/2015	27.60	27.7800	27.4300	27.60	2834070	JNPR	2015
95682	31/12/2015	42.68	42.8148	42.1600	42.52	1576097	XRX	2015
50659	31/12/2015	47.65	47.8500	46.5400	46.54	1158091	MCHP	2015
33038	31/12/2015	18.35	18.5200	18.2600	18.28	7590076	GLW	2015
29262	31/12/2015	92.91	93.6800	92.7350	92.88	1739511	DHR	2015
...	...	...	...	...	...	...	...	...
85630	01/02/2016	52.30	53.2700	52.3000	52.83	6336663	TXN	2016
54456	01/02/2016	54.88	55.0900	54.4950	54.71	44208542	MSFT	2016
44384	01/02/2016	87.05	89.0983	86.6801	88.41	500152	IT	2016
9562	01/02/2016	578.15	581.8000	570.3100	574.81	6355123	AMZN	2016
94443	01/02/2016	49.91	50.2580	49.4300	49.90	3148166	XLNX	2016

40270 rows × 8 columns

```
In [125]: sector_based_F = pd.read_csv('Sectorbased_data.csv')

In [126]: sector_based_F.head()
```

Out[126]:

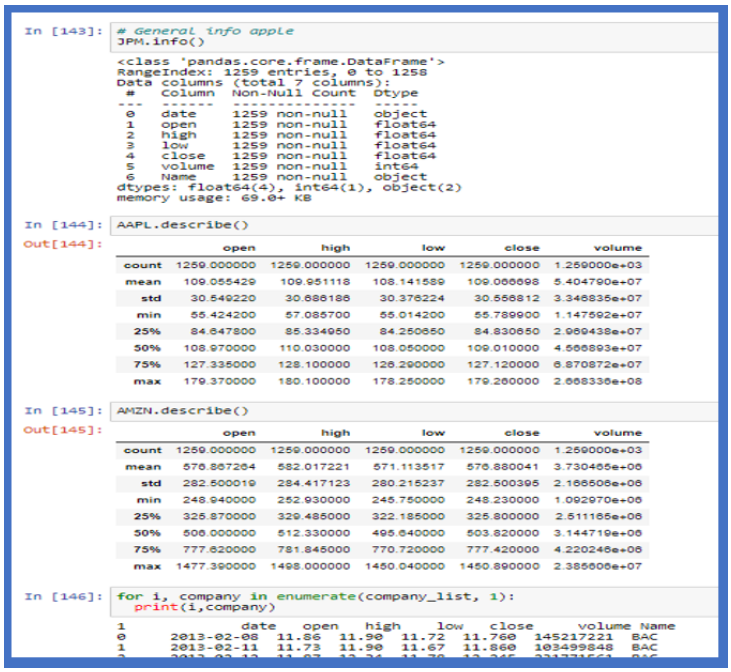
	Date	Open	High	Low	Close	Volume	Name	Sector	Year
0	08/02/2013	38.76	39.03	38.51	38.79	13112320	AIG	Financials	2013
1	11/02/2013	38.89	39.56	38.65	39.45	14230893	AIG	Financials	2013
2	12/02/2013	39.50	39.90	38.50	38.63	25676629	AIG	Financials	2013
3	13/02/2013	38.93	39.18	38.56	38.87	16533791	AIG	Financials	2013
4	14/02/2013	38.64	39.26	38.50	39.21	18321181	AIG	Financials	2013

```
In [127]: sector_based_F = sector_based_F[sector_based_F['Sector'] == "Financials"]

In [128]: sector_based_F = sector_based_F.drop("Sector",axis=1)

In [129]: sector_based_F.head()
```

Figure 35: LSTM Model - Step 2

```
In [130]: sector_based_F = sector_based_F.sort_values(by='Date', ascending=False)

In [131]: sector_based_F=sector_based_F.reset_index(drop=True)
          sector_based_T=sector_based_T.reset_index(drop=True)

In [132]: sector_based_F = sector_based_F.drop("Year",axis=1)
          sector_based_T = sector_based_T.drop("Year",axis=1)

In [133]: sector_based_F.columns =["date","open","high","low","close","volume", "Name"]
          sector_based_T.columns =["date","open","high","low","close","volume", "Name"]

In [134]: # The tech stocks we'll use for this analysis
          Financial_list = ['BAC','JPM','AAPL','AMZN',"Finan","TECH"]

In [135]: BAC= pd.read_csv('BAC_data.csv')
          JPM = pd.read_csv('JPM_data.csv')
          AAPL = pd.read_csv('AAPL_data.csv')
          AMZN = pd.read_csv('AMZN_data.csv')
          company_list = [BAC, JPM,AAPL,AMZN,sector_based_F, sector_based_T]
          company_name = ["BAC", "JPM","AAPL","AMZN", "Finance", "Technology"]

In [136]: df = pd.concat(company_list, axis=0)
          df.tail(10)
```

Figure 36: LSTM Model - Step 3



```
In [143]: # General info apple
          JPM.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    1259 non-null   object
 1   open    1259 non-null   float64
 2   high    1259 non-null   float64
 3   low     1259 non-null   float64
 4   close   1259 non-null   float64
 5   volume  1259 non-null   int64
 6   Name    1259 non-null   object
dtypes: float64(4), int64(1), object(2)
memory usage: 69.0+ KB

In [144]: AAPL.describe()
```

Out[144]:

	open	high	low	close	volume
count	1259.000000	1259.000000	1259.000000	1259.000000	1.259000e+03
mean	109.055429	109.951118	108.141589	109.066698	5.404790e+07
std	30.549220	30.686186	30.376224	30.556812	3.346836e+07
min	55.424200	57.085700	55.014200	55.789900	1.147592e+07
25%	84.647800	85.334950	84.250650	84.830650	2.989438e+07
50%	108.970000	110.030000	108.050000	109.010000	4.566893e+07
75%	127.335000	128.100000	126.290000	127.120000	6.870872e+07
max	179.370000	180.100000	178.250000	179.260000	2.668336e+08

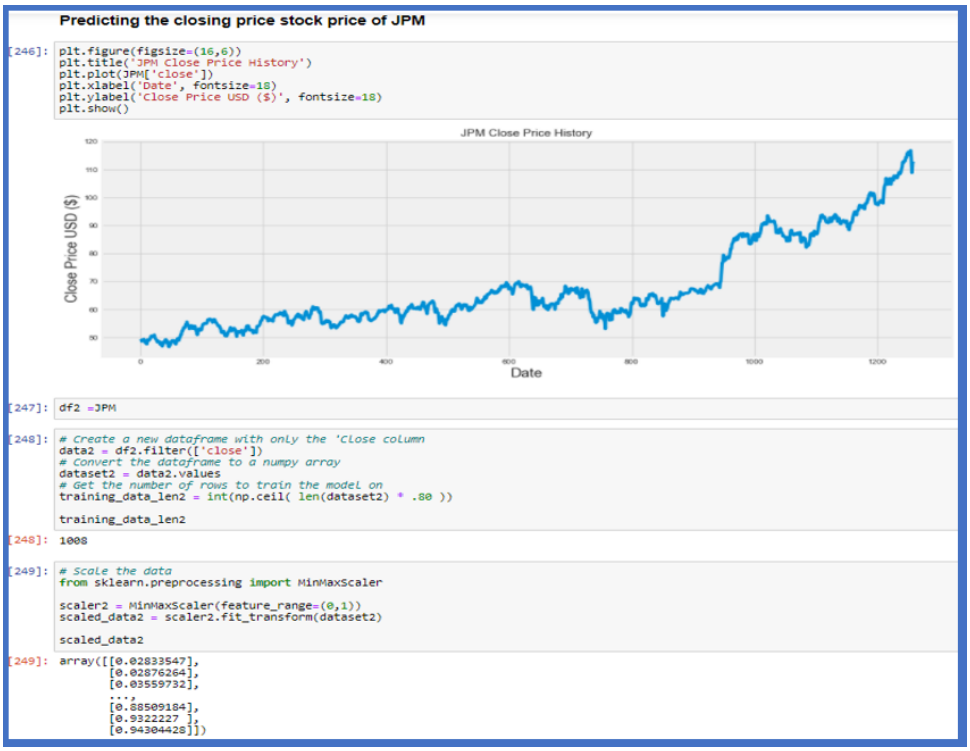
```
In [145]: AMZN.describe()
```

Out[145]:

	open	high	low	close	volume
count	1259.000000	1259.000000	1259.000000	1259.000000	1.259000e+03
mean	576.867264	582.017221	571.113517	576.880041	3.730465e+06
std	282.500019	284.417123	280.215237	282.500395	2.166506e+06
min	248.940000	252.930000	245.750000	248.230000	1.092970e+06
25%	325.870000	329.485000	322.185000	325.800000	2.511165e+06
50%	506.000000	512.330000	495.640000	503.820000	3.144719e+06
75%	777.620000	781.845000	770.720000	777.420000	4.220246e+06
max	1477.390000	1498.000000	1450.040000	1450.890000	2.385606e+07

```
In [146]: for i, company in enumerate(company_list, 1):
              print(i,company)

1
          date   open   high    low   close     volume Name
0   2013-02-08  11.86  11.90  11.72  11.760  145217221  BAC
1   2013-02-11  11.73  11.90  11.67  11.860  103499848  BAC
```

Figure 37: LSTM Model - Step 4



Predicting the closing price stock price of JPM

```
[246]: plt.figure(figsize=(16,6))
       plt.title('JPM Close Price History')
       plt.plot(JPM['close'])
       plt.xlabel('Date', fontsize=18)
       plt.ylabel('Close Price USD ($)', fontsize=18)
       plt.show()
```



```
[247]: df2 =JPM

[248]: # Create a new dataframe with only the 'Close column
       data2 = df2.filter(['close'])
       # Convert the dataframe to a numpy array
       dataset2 = data2.values
       # Get the number of rows to train the model on
       training_data_len2 = int(np.ceil( len(dataset2) * .80 ))

       training_data_len2

[248]: 1008

[249]: # Scale the data
       from sklearn.preprocessing import MinMaxScaler

       scaler2 = MinMaxScaler(feature_range=(0,1))
       scaled_data2 = scaler2.fit_transform(dataset2)

       scaled_data2

[249]: array([[0.02833547],
              [0.02876264],
              [0.03559732],
              ...,
              [0.88509184],
              [0.9322227 ],
              [0.94304428]])
```

Figure 38: LSTM Model - Step 5

```
In [250]: # Create the training data set
          # Create the scaled training data set
          train_data2 = scaled_data2[0:int(training_data_len2), :]
          # Split the data into x_train and y_train data sets
          x_train2 = []
          y_train2 = []

          for i in range(60, len(train_data2)):
              x_train2.append(train_data2[i-60:i, 0])
              y_train2.append(train_data2[i, 0])
              if i<= 61:
                  print(x_train2)
                  print(y_train2)
                  print()
```

Figure 39: LSTM Model - Step 6

```
In [251]: # Convert the x_train and y_train to numpy arrays
          x_train2, y_train2 = np.array(x_train2), np.array(y_train2)

          # Reshape the data
          x_train2 = np.reshape(x_train2, (x_train2.shape[0], x_train2.shape[1], 1))
          # x_train.shape

In [252]: from keras.models import Sequential
          from keras.layers import Dense, LSTM,RNN

          # Build the LSTM model
          model2 = Sequential()
          model2.add(LSTM(5, return_sequences=True, input_shape= (x_train2.shape[1], 1)))
          model2.add(LSTM(3, return_sequences=False))
          model2.add(Dense(3))
          model2.add(Dense(1))

          # Compile the model
          model2.compile(optimizer='adam', loss='mean_squared_error')

          # Train the model

In [254]: model2.fit(x_train2, y_train2, batch_size=1, epochs=100)

          Epoch 1/100
          948/948 [==============================] - 12s 13ms/step - loss: 5.3788e-04
          Epoch 2/100
          948/948 [==============================] - 13s 14ms/step - loss: 4.8200e-04
          Epoch 3/100
          948/948 [==============================] - 12s 13ms/step - loss: 4.5099e-04
          Epoch 4/100
          948/948 [==============================] - 13s 13ms/step - loss: 4.2808e-04 0s - loss: 4.3 - ETA: 0s - loss: 4.3
          Epoch 5/100
          948/948 [==============================] - 14s 14ms/step - loss: 3.9860e-04
          Epoch 6/100
          948/948 [==============================] - 13s 13ms/step - loss: 3.8219e-04
          Epoch 7/100
          948/948 [==============================] - 13s 13ms/step - loss: 3.5839e-04
          Epoch 8/100
          948/948 [==============================] - 13s 13ms/step - loss: 3.6089e-04
          Epoch 9/100
          948/948 [==============================] - 14s 14ms/step - loss: 3.0749e-04
          Epoch 10/100
          948/948 [==============================] - 14s 14ms/step - loss: 2.7383e-04
          Epoch 11/100
```

Figure 40: LSTM Model - Step 7

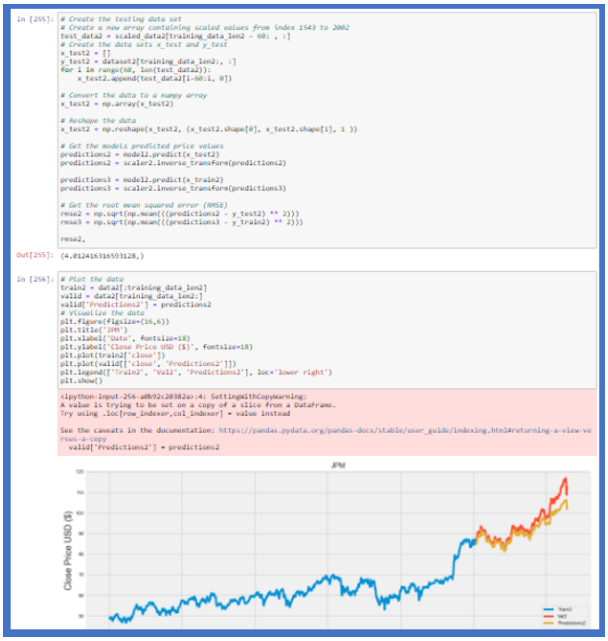


Figure 41: LSTM Model - Step 8

5.3 Time Series ARIMA Model

To run the Time Series ARIMA Model for sector based index and individual companies please Choose the "Arima Time Series -Student No -x14127032 - Sector Based Stock Market Prediction In USA" ipython notebook and follow the step from figure 42 to figure 46 .

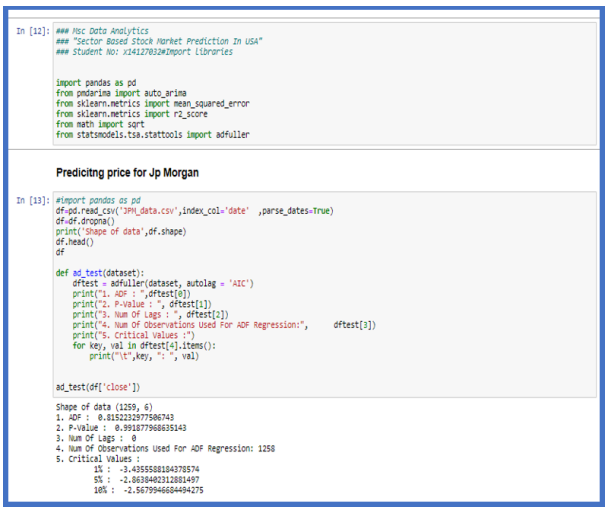
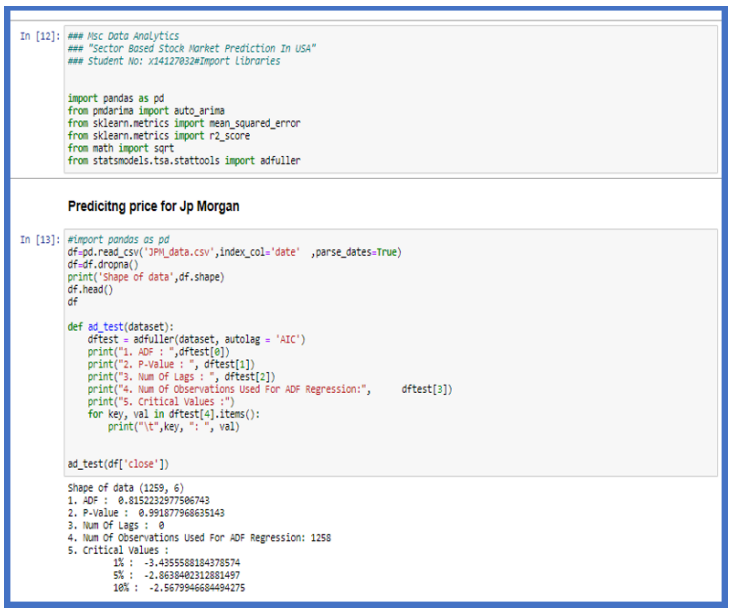


Figure 42: Time Series ARIMA Model - Step 1

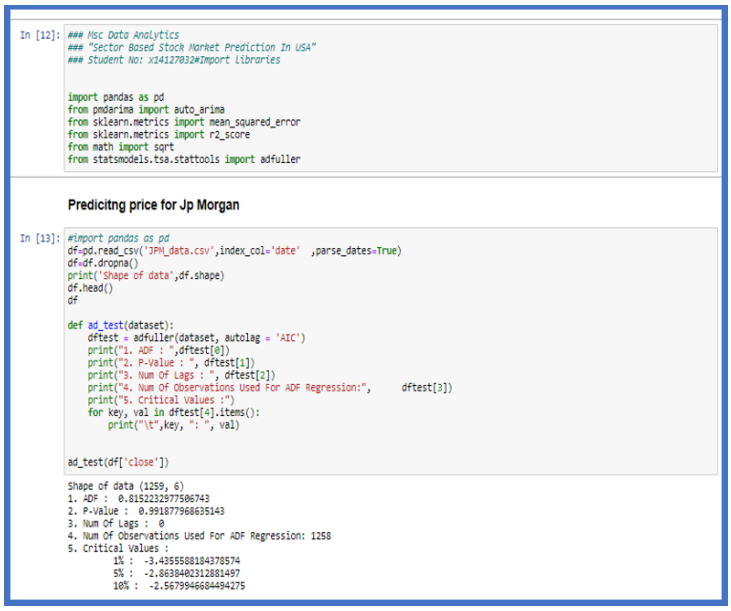


```
In [12]: ### Msc Data Analytics
         ### "Sector Based Stock Market Prediction In USA"
         ### Student No: x14127032#Import libraries


         import pandas as pd
         from pmdarima import auto_arima
         from sklearn.metrics import mean_squared_error
         from sklearn.metrics import r2_score
         from math import sqrt
         from statsmodels.tsa.stattools import adfuller
```

Predicitng price for Jp Morgan

```
In [13]: #import pandas as pd
         df=pd.read_csv('JPM_data.csv',index_col='date'  ,parse_dates=True)
         df=df.dropna()
         print('Shape of data',df.shape)
         df.head()
         df

         def ad_test(dataset):
             dftest = adfuller(dataset, autolag = 'AIC')
             print("1. ADF : ",dftest[0])
             print("2. P-Value : ", dftest[1])
             print("3. Num Of Lags : ", dftest[2])
             print("4. Num Of Observations Used For ADF Regression:",      dftest[3])
             print("5. Critical Values :")
             for key, val in dftest[4].items():
                 print("\t",key, ": ", val)


         ad_test(df['close'])

         Shape of data (1259, 6)
         1. ADF :  0.8152232977506743
         2. P-Value :  0.991877968635143
         3. Num Of Lags :  0
         4. Num Of Observations Used For ADF Regression: 1258
         5. Critical Values :
                 1% :  -3.4355588184378574
                 5% :  -2.8638402312881497
                 10% :  -2.5679946684494275
```

Figure 43: Time Series ARIMA Model - Step 2

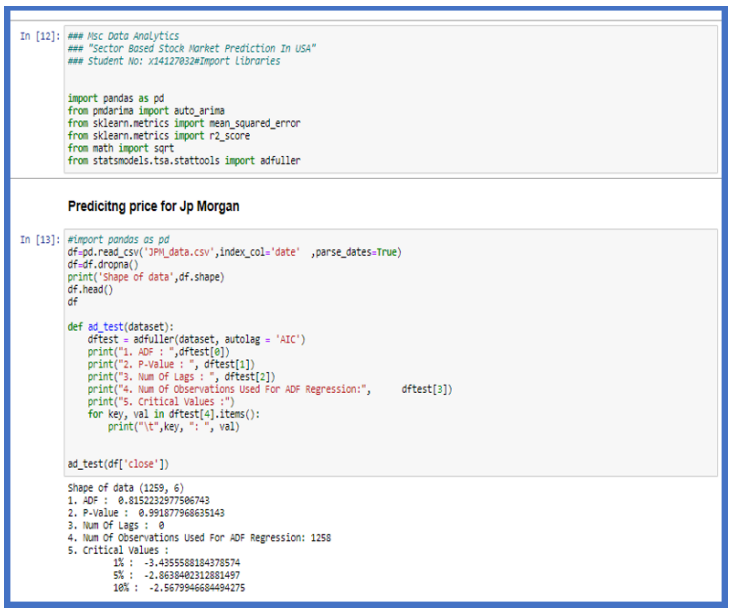


```
In [12]: ### Msc Data Analytics
         ### "Sector Based Stock Market Prediction In USA"
         ### Student No: x14127032#Import libraries


         import pandas as pd
         from pmdarima import auto_arima
         from sklearn.metrics import mean_squared_error
         from sklearn.metrics import r2_score
         from math import sqrt
         from statsmodels.tsa.stattools import adfuller
```

Predicitng price for Jp Morgan

```
In [13]: #import pandas as pd
         df=pd.read_csv('JPM_data.csv',index_col='date'  ,parse_dates=True)
         df=df.dropna()
         print('Shape of data',df.shape)
         df.head()
         df

         def ad_test(dataset):
             dftest = adfuller(dataset, autolag = 'AIC')
             print("1. ADF : ",dftest[0])
             print("2. P-Value : ", dftest[1])
             print("3. Num Of Lags : ", dftest[2])
             print("4. Num Of Observations Used For ADF Regression:",      dftest[3])
             print("5. Critical Values :")
             for key, val in dftest[4].items():
                 print("\t",key, ": ", val)


         ad_test(df['close'])

         Shape of data (1259, 6)
         1. ADF :  0.8152232977506743
         2. P-Value :  0.991877968635143
         3. Num Of Lags :  0
         4. Num Of Observations Used For ADF Regression: 1258
         5. Critical Values :
                 1% :  -3.4355588184378574
                 5% :  -2.8638402312881497
                 10% :  -2.5679946684494275
```

Figure 44: Time Series ARIMA Model - Step 3

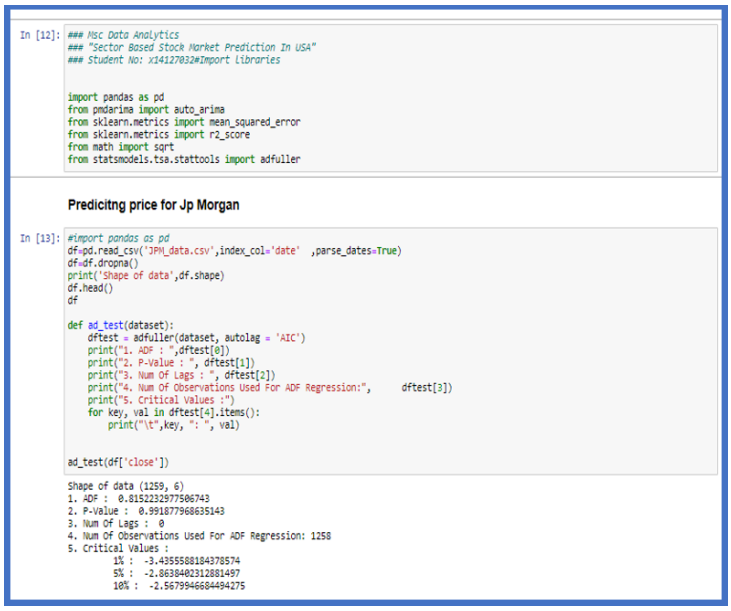


```
In [12]: ### Msc Data Analytics
         ### "Sector Based Stock Market Prediction In USA"
         ### Student No: x14127032#Import libraries


         import pandas as pd
         from pmdarima import auto_arima
         from sklearn.metrics import mean_squared_error
         from sklearn.metrics import r2_score
         from math import sqrt
         from statsmodels.tsa.stattools import adfuller
```

Predicitng price for Jp Morgan

```
In [13]: #import pandas as pd
         df=pd.read_csv('JPM_data.csv',index_col='date'  ,parse_dates=True)
         df=df.dropna()
         print('Shape of data',df.shape)
         df.head()
         df

         def ad_test(dataset):
             dftest = adfuller(dataset, autolag = 'AIC')
             print("1. ADF : ",dftest[0])
             print("2. P-Value : ", dftest[1])
             print("3. Num Of Lags : ", dftest[2])
             print("4. Num Of Observations Used For ADF Regression:",      dftest[3])
             print("5. Critical Values :")
             for key, val in dftest[4].items():
                 print("\t",key, ": ", val)


         ad_test(df['close'])

         Shape of data (1259, 6)
         1. ADF :  0.8152232977506743
         2. P-Value :  0.991877968635143
         3. Num Of Lags :  0
         4. Num Of Observations Used For ADF Regression: 1258
         5. Critical Values :
                 1% :  -3.4355588184378574
                 5% :  -2.8638402312881497
                 10% :  -2.5679946684494275
```

Figure 45: Time Series ARIMA Model - Step 4



```
In [12]: ### Msc Data Analytics
         ### "Sector Based Stock Market Prediction In USA"
         ### Student No: x14127032#Import libraries


         import pandas as pd
         from pmdarima import auto_arima
         from sklearn.metrics import mean_squared_error
         from sklearn.metrics import r2_score
         from math import sqrt
         from statsmodels.tsa.stattools import adfuller
```

Predicitng price for Jp Morgan

```
In [13]: #import pandas as pd
         df=pd.read_csv('JPM_data.csv',index_col='date'  ,parse_dates=True)
         df=df.dropna()
         print('Shape of data',df.shape)
         df.head()
         df

         def ad_test(dataset):
             dftest = adfuller(dataset, autolag = 'AIC')
             print("1. ADF : ",dftest[0])
             print("2. P-Value : ", dftest[1])
             print("3. Num Of Lags : ", dftest[2])
             print("4. Num Of Observations Used For ADF Regression:",      dftest[3])
             print("5. Critical Values :")
             for key, val in dftest[4].items():
                 print("\t",key, ": ", val)


         ad_test(df['close'])

         Shape of data (1259, 6)
         1. ADF :  0.8152232977506743
         2. P-Value :  0.991877968635143
         3. Num Of Lags :  0
         4. Num Of Observations Used For ADF Regression: 1258
         5. Critical Values :
                 1% :  -3.4355588184378574
                 5% :  -2.8638402312881497
                 10% :  -2.5679946684494275
```

Figure 46: Time Series ARIMA Model - Step 5

6 Troubleshoots Guide

While implementing the project you may come across below error. For the figure 8 error Please use tensor flow 2.2 or higher version. if you get figure 9 error , Please add python

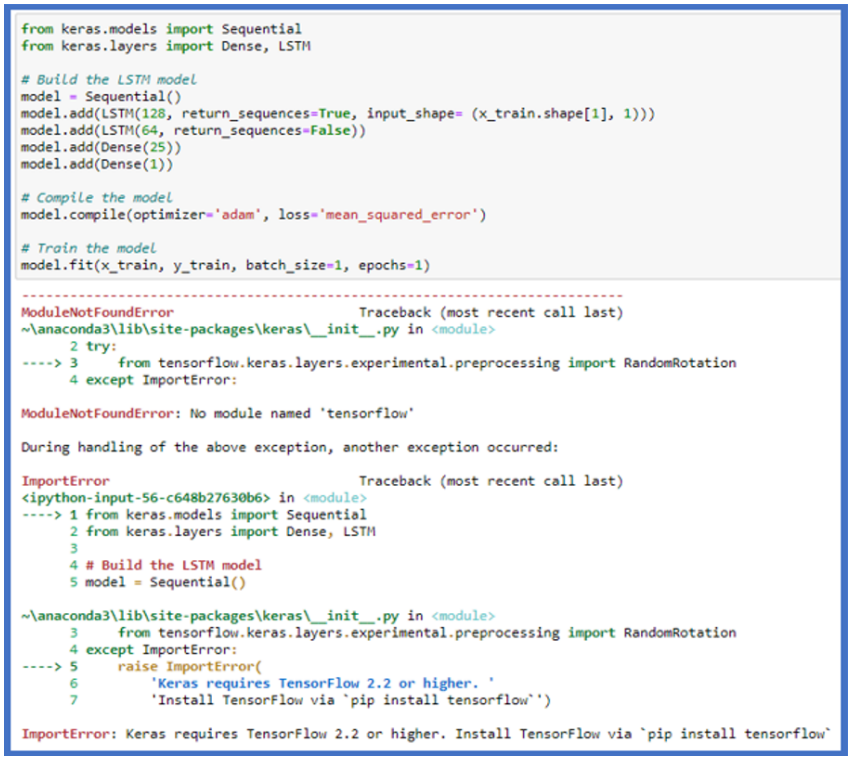


Figure 47: Keras Installation Error

in the file path.

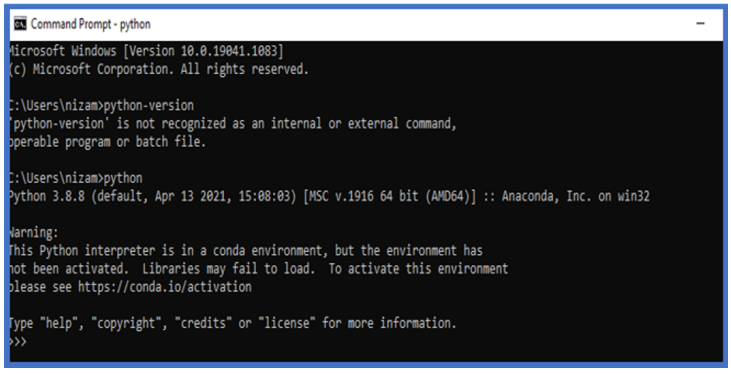


Figure 48: Python Path Error

To join the data fuzzy_match was implemented but because of its only 11% output matched , it was not used in the research. However it just attached for note only.

```
In [1]: import pandas as pd
        dataset1 = pd.read_csv("nyse-listed.csv")
        dataset3 = pd.read_csv("fortune1000.csv")

        import pandas as pd
        import numpy as np

        # We will now import libraries for data matching using levenshtein distance in fuzzywuzzy modules
        import fuzzywuzzy
        from fuzzywuzzy import process
        import chardet
        from fuzzywuzzy import fuzz
        compare = pd.MultiIndex.from_product([dataset1['Company Name'],
                                              dataset3['Company']]).to_series()


        def metrics(tup):
            return pd.Series([fuzz.ratio(*tup),
                              fuzz.token_sort_ratio(*tup)],
                             ['ratio', 'token'])

        compare.apply(metrics)
```

Out[1]:

		ratio	token
Company Name	Company		
Agilent Technologies, Inc. Common Stock	Walmart	13	18
	Exxon Mobil	16	29
	Apple	14	14
	Berkshire Hathaway	21	18
	McKesson	21	22
...	...	...	...
China Zenix Auto International Limited American Depositary Shares, each representing four ordinary shares.	New York Community Bancorp	24	28
	Portland General Electric	26	26
	Portland General Electric	26	26
	Wendy's	9	11
	Briggs & Stratton	16	18

3298000 rows × 2 columns

Figure 49: Fuzzy Match code

References

van der Aalst, W. M. (2004). Why workflow is not just a pi-process, *BP Trends* pp. 02–04.