A Comprehensive Study to Forecast the Delhi and Bangalore Cities Air Pollution using Machine Learning Models

MSc Research Project
Data Analytics

Parth Darekar
Student ID: x19212739

School of Computing
National College of Ireland

Supervisor: Dr.Rashmi Gupta

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Parth Darekar
Student ID:	19212739
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr.Rashmi Gupta
Submission Due Date:	16/08/2021
Project Title:	A Comprehensive Study to Forecast the Delhi and Bangalore Cities Air Pollution using Machine Learning Models
Word Count:	1163
Page Count:	11

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:	Parth Darekar
Date:	10th October 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):	

A Comprehensive Study to Forecast the Delhi and Bangalore Cities Air Pollution using Machine Learning Models

Darekar
19212739

1 Introduction

Pollution is a big issue in today's world of rapid development. Increased traffic is a result of population growth, while tree loss increases NO2 and SO2 levels, resulting in air pollution. To anticipate the future, time series models like VAR,VARMAX,ARFIMA and SARIMA, as well as neural network model LSTM, have been used.This handbook pertains to the critical configurations completed for this forecasting project. It contains all of the data for the system settings and the numerous applications utilized in this study. The program's code has been showcased and explained in the below section

2 System Specification

The project was completed and executed on a laptop that met the following requirements:

Item	Value
OS Name	Microsoft Windows 10 Home Single Language
Version	10.0.19043 Build 19043
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	LAPTOP-37V2SFPU
System Manufacturer	LENOVO
System Model	81WJ
System Type	x64-based PC
System SKU	LENOVO_MT_81WJ_BU_idea_FM_IdeaPad S340-14IIL
Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz, 1190 ...
BIOS Version/Date	LENOVO CUCN62WW(V3.04), 03-01-2020
SMBIOS Version	3.2
Embedded Controll...	3.62
BIOS Mode	UEFI
BaseBoard Manufact...	LENOVO
BaseBoard Product	LNVNB161216
BaseBoard Version	SDK0Q55722 WIN

Figure 1: System Configuration

3 Software's Involved

Python is the primary software for running the models and obtaining the results. The well-known Anaconda tool was used to provide Python. The model creation and visualization of the graph were done using Excel and Lucid Chart. The following links will help you set up the software mentioned below:

url:-`https://www.anaconda.com/distribution/#download-section`

Figure 2: Python Version

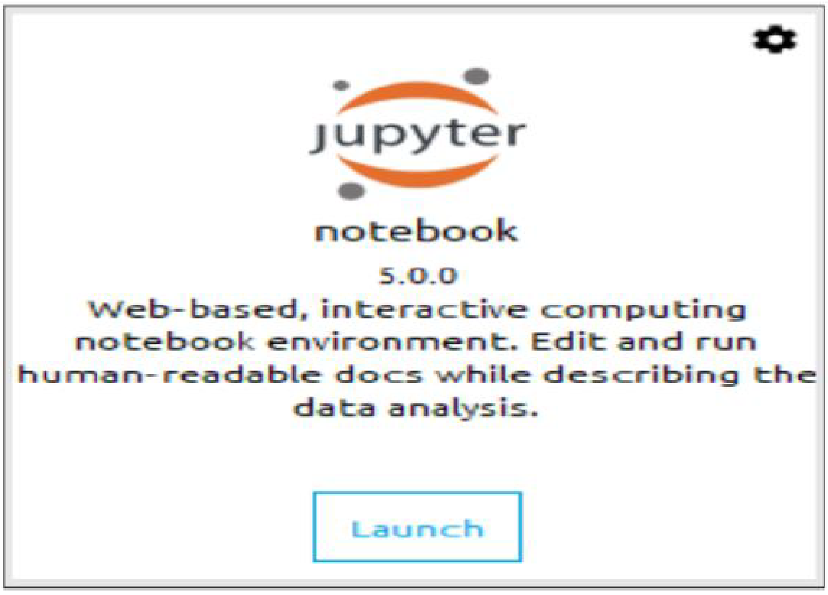

Figure 3: Jupyter Version

4 Data Preparation and Feature Selection

4.1 Importing the Dataset

The data was obtained from Kaggle a well-known website, for the student researcher project. The dataset was published by Central Pollution Control Board of India(CPCB). The dataset consist of different pollutants which will be used for predicting the air quality index of Delhi and Bangalore cities with the help of implementing different machine learning models on the dataset.The dataset was in .csv format,which was loaded in jupyter for further process.

```
# Loading the data
data = pd.read_csv("D:/Projects/ResearchProject/original/Final.csv",encoding="ISO-8859-1")
```

Figure 4: Loading the dataset

4.2 Requires libraries

```
# Importing the dependencies
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import *
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
import math
from sklearn.metrics import mean_absolute_error
```

Figure 5: Libraries required for implementing the model

All of the essential libraries for this project are shown in the diagram above. To handle the dataframe for the pollution data exported in the file, the Pandas library is used. The Numpy library was used to turn data into an n-dimensional array for the pollutants n02 and s02.The Matplotlib package is being used to construct a graph of expected pollution in India by testing and training the dataset. Here's a good example of the finished product. The Warning library is the major library used here, and it suppresses all warnings in the console for all problems caused by the code.

4.3 Data Cleaning and Data Transformation

In this section the data had many cities in the in the city column.But for our research work we will be working on Delhi and Bangalore cities as shown inf figure(6).For that we used data.loc and data.unique function to segregate the data only for Bangalore and Delhi cities.Further the date column was converted into required data and time format as shown in below figure(8).At the end we also checked for if there exist any null values and the missing values in the dataset as shown in Figure(7) and Figure(9)

```
[12] # Storing the new data into a dataframe
     dataframe  = data.loc[(data["location"] == 'Delhi') | (data["location"] == "Bangalore")]
```

```
[13] dataframe.shape
```

```
(15210, 13)
```

```
[14] dataframe.head()
```

	stn_code	sampling_date	state	location	agency	type	so2	no2	rspm	spm	location_monitoring_station	pm2_5	date
25	144	January - M011992	Delhi	Delhi	NaN	Residential, Rural and other Areas	28.2	38.8	NaN	365.0	NaN	NaN	01-01-1992
26	146	January - M011992	Delhi	Delhi	NaN	Residential, Rural and other Areas	76.8	72.4	NaN	735.0	NaN	NaN	01-01-1992
27	145	January - M011992	Delhi	Delhi	NaN	Industrial Area	48.7	41.2	NaN	NaN	NaN	NaN	01-01-1992
28	56	January - M011992	Delhi	Delhi	Central Pollution Control Board	Industrial Area	17.3	40.1	NaN	186.0	NaN	NaN	01-01-1992
29	58	January - M011992	Delhi	Delhi	Central Pollution Control Board	Industrial Area	7.9	29.4	NaN	357.0	NaN	NaN	01-01-1992

Figure 6: Segregating data for Delhi and Bangalore

```
[15] dataframe['location'].unique()
```

```
array(['Delhi', 'Bangalore'], dtype=object)
```

```
[16] # Checking for the null values
     data.isnull().sum()
```

```
stn_code                        144074
sampling_date                        0
state                                0
location                             0
agency                          149478
type                              5390
so2                              34643
no2                              16230
rspm                             40219
spm                             237380
location_monitoring_station      27488
pm2_5                           426421
date                                 0
dtype: int64
```

Figure 7: Checking Null values

```
[17] #converting date column into Date-time format
     dataframe['date']=pd.to_datetime(dataframe['date'])

     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: 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

[18] # dropping the columns which are not useful
     dataframe.drop(['stn_code','sampling_date','location_monitoring_station','agency'],axis=1,inplace=True)

     /usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4174: SettingWithCopyWarning:
     A value is trying to be set on a copy of a slice from a DataFrame

     See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
       errors=errors,

[19] dataframe.info()

     <class 'pandas.core.frame.DataFrame'>
     Int64Index: 15210 entries, 25 to 435708
     Data columns (total 9 columns):
      #   Column    Non-Null Count  Dtype
     ---  ------    --------------  -----
      0   state     15210 non-null  object
      1   location  15210 non-null  object
      2   type      15210 non-null  object
```

Figure 8: Converting Date column into Date-Time format

```
[21] # Filling the null values with the mean of that particular feature
     dataframe.fillna(dataframe.mean(),inplace = True)

     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: DataFrame.mean and DataFrame.median with numeric_only=None will include datetime64 and

     /usr/local/lib/python3.7/dist-packages/pandas/core/series.py:4536: SettingWithCopyWarning:
     A value is trying to be set on a copy of a slice from a DataFrame

     See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
       downcast=downcast,

[22] # Here we observe there are no missing values
     dataframe.isnull().sum()

     state       0
     location    0
     type        0
     so2         0
     no2         0
     rspm        0
     spm         0
     pm2_5       0
     date        0
     dtype: int64
```

Figure 9: Filling the null values with the mean of the features selected

4.4 Feature Selection and Splitting the Data

As we are working with time-series model which handles a univariate variable so we will choose so2 as the feature.We will split the data into 80:20 ratio 80% into training and 20% into testing.Similarly for LSTM model we will split the data into 95:5 ratio,where 95% will be training and 5% will be testing as shown in Figure(11).Hence we will be taking no2, so2, spm, rspm and pm2.5 as our feature and for the label part we will be sending the so2 data into label. Here in our dataset, we will split the data by considering the past 1 day data of the above 5 features and will try to predict the 2nd day so2 air pollutant volume.

```
[28] # Splitting the data into train and test as we are working with time-series model which handles a univariate variable so we will choose so2 as the feature
     # We will split the data into 80:20 ratio 80% into training and 20% into testing
     split = len(dataframe) - int(0.2*len(dataframe))
     train, test = dataframe['so2'][0:split], dataframe['so2'][split:]
```

Figure 10: Feature Selection and splitting of Data for Timeseries models

```
[40] # TRANSFORMING OUR DATASET TO A STANDARD SCALER

     Scaler = StandardScaler()

     Scaler = Scaler.fit(df_for_training)

[42] df_for_training_scaled = Scaler.transform(df_for_training)

[43] len(df_for_training_scaled)

     15210

[44] x_train = []
     y_train = []

[45] # We will split the data into 95% training and 5% testing
     training_size = int(len(df_for_training_scaled)*0.95)
     test_size = len(df_for_training_scaled)-training_size
     train_data,test_data=df_for_training_scaled[0:training_size,:],df_for_training_scaled[training_size:len(df_for_training_scaled),:7]

[46] print(train_data.shape,test_data.shape)

     (14449, 4) (761, 4)
```

Figure 11: Feature Selection and splitting of Data for LSTM

4.5 Parameter Calibration to find the p,q,d values for our time-series model

Parmeter clibrtin is a tehnique in which we must analyze the proprite ,q,d vlus that must be ssed when implementing our univrite time series mdels like SARIMA, VAR and ARFIMA,which we shall lk into further.We had performed the Augmented-Dickey-Fuller(ADF) test to check our data is stationary or not.From the ADF we got the p-value less than 0.05,ie 0.00 as shown in figure(12),which specifies that our data is stationary and we can proceed for implementing the time series models.We also performed hust test to check wheter our data is stationary or not.From the Figure(13) we can specify the thurst value is between 0 and 1,i.e 0.2746,which indicates that our data is stationary.

```
# The next step is to check whether the time series we are dealing with is stationary or not using the ADF test (Augmented Dickey-Fuller)

from statsmodels.tsa.stattools import adfuller

result = adfuller(train)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

ADF Statistic: -7.350382
p-value: 0.000000
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
```

Figure 12: Augmented Dickey Fuller Test

```
[35] # We can also confirm the stationarity and non-stationarity of the time-series feature using hurst exponent
     import hurst
     H, c,data = hurst.compute_Hc(train)
     print("H = {:.4f}, c = {:.4f}".format(H,c))

     H = 0.2746, c = 1.0803
```

Figure 13: Augmented Dickey Fuller Test

5 Implementation and Evaluation of the Models

5.1 SARIMA

The below Figure(14) and Figure(15) showcase the SARIMA model.From the Figure(15)Quantile Quantile plot is showing the almost the same distribution with the predicted line,which showcased that our model is fitted and forecasting better predictions.

```
# Fitting our model on a time-series model Named SARIMAX
import statsmodels.api as sm
model = sm.tsa.statespace.SARIMAX(train,order=(0,0,1), seasonal_order=(0,2,0,12))
results = model.fit()
print(results.summary().tables[1])

# Use plot diagnostics with results calculated from above
results.plot_diagnostics(figsize =(15,8))
plt.show()

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:219: ValueWarning: A date index has been provi
  ' ignored when e.g. forecasting.', ValueWarning)
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1          0.2389      0.005     44.831      0.000       0.228       0.249
sigma2       208.5893      1.228    169.828      0.000     206.182     210.997
==============================================================================
```

Figure 14: Implementing SARIMA

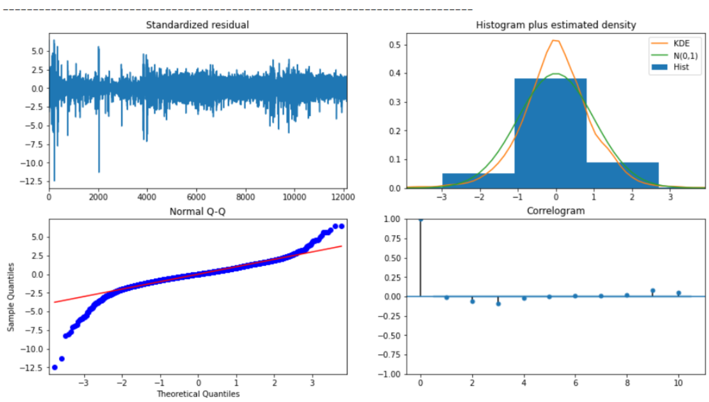

Figure 15: SARIMA Model Prediction

```
[ ] from sklearn.metrics import mean_squared_error,mean_absolute_error
    from math import sqrt
    mae_sarima = mean_absolute_error(test,prediction_sarima)
    mse_sarima = mean_squared_error(test,prediction_sarima)
    print(mae_sarima)
    print(mse_sarima)
    rms = sqrt(mean_squared_error(test, prediction_sarima))
    print(rms)

    549.5979384832791
    844773.181070534
    919.1154340291181
```

Figure 16: SARIMA Evaluation

5.2 LSTM Model

LSTM mdel reles the hidden lyer neurns f the RNN by unique set f memry ells, nd the stte f the memry ells t s its key. The LSTM mdel mintins nd udtes the stte f memry ells using the gte struture by filtering infrmtin.In below Figure(17) we have used relu activation for executing the model.From Figure(17),we also build a sequential LSTM architecture which consists of 3 hidden LSTM layers and in each layer there are 60 LSTM units and 2 fully connected layer,where one layer consists of 64 hidden neurons and the output layer consist of only 1 neuron as we have to predict only 1 outcome.The Figure(18) specifies that for compiling the LSTM model we have used adam optimizer,as well as for calculating the loss function and the metrics we have used MAE(Mean Absolute Error) and MSE(Mean Squared Error).The Figure(19) showcased the trained model executed for 20 epochs and batch size of 32.After running 20 epochs we see that the Mae is 0.70.

```
[ ] model_unilstm = Sequential()
    model_unilstm.add(LSTM(60, activation='relu',input_shape=(x_train.shape[1], x_train.shape[2]), return_sequences=True,dropout = 0.2, recurrent_dropout = 0.2))
    model_unilstm.add(LSTM(60, activation='relu', return_sequences=True,dropout = 0.2, recurrent_dropout = 0.2))
    model_unilstm.add(LSTM(60,activation = 'relu',return_sequences = False))
    model_unilstm.add(Dense(64,activation = 'relu'))
    model_unilstm.add(Dense(1,activation ='relu'))

[ ] model_unilstm.summary()

    Model: "sequential_1"
    _________________________________________________________________
    Layer (type)                Output Shape              Param #
    =================================================================
    lstm (LSTM)                 (None, 365, 60)           15600
    _________________________________________________________________
    lstm_1 (LSTM)               (None, 365, 60)           29040
    _________________________________________________________________
    lstm_2 (LSTM)               (None, 60)                29040
    _________________________________________________________________
    dense (Dense)               (None, 64)                3904
    _________________________________________________________________
    dense_1 (Dense)             (None, 1)                 65
    =================================================================
    Total params: 77,649
    Trainable params: 77,649
    Non-trainable params: 0
    _________________________________________________________________
```

Figure 17: Building of LSTM Model

```
[54] # Compiling the model
     model_unilstm.compile(optimizer = 'adam',loss = 'mse',metrics = ['mae'])
```

Figure 18: LSTM Model

```
[52] %%time
     # FITTING THE MODEL FOR TRAINING
     history = model_unilstm.fit(x_train,y_train , epochs=20, batch_size = 32)

     Epoch 1/20
     452/452 [==============================] - 8s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 2/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 3/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 4/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 5/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 6/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 7/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 8/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 9/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 10/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 11/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 12/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 13/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 14/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 15/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 16/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 17/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
     Epoch 18/20
     452/452 [==============================] - 3s 6ms/step - loss: 0.8915 - mae: 0.7045
```

Figure 19: LSTM Model

5.3 ARFIMA Model

Arfima models was implemented using 'Rugarch' library.As we can see in Figure(20) the likelihood for the optimal parameter section is less than 0.5 which is -21.6,this justifies that our novel model has done a decent work.Whereas the Weighted Lunj box test specifies that p-value is less than the statistical value.The overall performance of the model was decent.

```
*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model      : sGARCH(1,1)
Mean Model       : ARFIMA(1,d,0)
Distribution     : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
mu      6.297010    7.919500   0.79513  0.42654
ar1    -0.377230    0.013276 -28.41399  0.00000
arfima  1.000000          NA        NA       NA
omega   1.123942    0.195624   5.74541  0.00000
alpha1  0.083278    0.008190  10.16822  0.00000
beta1   0.903503    0.009595  94.16709  0.00000

Robust Standard Errors:
        Estimate  Std. Error   t value Pr(>|t|)
mu      6.297010   28.863370   0.21817 0.827300
ar1    -0.377230    0.013907 -27.12527 0.000000
arfima  1.000000          NA        NA       NA
omega   1.123942    0.510384   2.20215 0.027655
alpha1  0.083278    0.021390   3.89330 0.000099
beta1   0.903503    0.026455  34.15301 0.000000

LogLikelihood : -21166.87

Information Criteria
------------------------------------

Akaike       6.8385
Bayes        6.8439
Shibata      6.8385
Hannan-Quinn 6.8403

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic   p-value
Lag[1]                       39.7 2.955e-10
Lag[2*(p+q)+(p+q)-1][2]     368.5 0.000e+00
Lag[4*(p+q)+(p+q)-1][5]     618.7 0.000e+00
d.o.f=1
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic p-value
Lag[1]                      2.134  0.1441
Lag[2*(p+q)+(p+q)-1][5]     3.249  0.3635
Lag[4*(p+q)+(p+q)-1][9]     5.116  0.4125
```

Figure 20: Arfima Model

5.4 VAR Model

We performed Vector Auto Regressive Model(VAR) model our second novel approach to forecast the air pollution of Delhi and Bangalore cities.The code implementation for model is shown in Figure(21).As we can see VAR model in Figure(22) it was attempting to capture the underlying time series pattern for the so2.Similarly we performed VARMA (Vector Auto regression Moving-Average) another variant of VAR,by using similarpackage which was used for VAR model as well with the help of statsmodels package.The model prediction graph is shown in Figure(23).

```
[56] from statsmodels.tsa.vector_ar.var_model import VAR
     from random import random

[57] # Common code for display result
     def show_graph(df1,df2,title):
         data = pd.concat([df1, df2])
         data.reset_index(inplace=True, drop=True)
         for col in data.columns:
             if col.lower().startswith('pred'):
                 data[col].plot(label=col,linestyle="dotted")
             else:
                 data[col].plot(label=col)
         plt.title(title)
         plt.legend()
         plt.show()

[58] def VAR_model(train,test):
         # fit model
         model = VAR(train)
         model_fit = model.fit()
         # make prediction
         yhat = model_fit.forecast(model_fit.y, steps=len(test))
         res=pd.DataFrame({"Pred1":[x[0] for x in yhat], "Pred2":[x[1] for x in yhat],
                           "Act1":test["Act1"].values, "Act2":test["Act2"].values})
         return res

     df_train = pd.DataFrame({'Act1':[x + random()*10 for x in range(0, 100)],
                              'Act2':50+np.sin(np.linspace(0, 2*np.pi, 100))*50})
     df_test = pd.DataFrame({'Act1':[x + random()*10 for x in range(101, 201)],
                              'Act2':50+np.sin(np.linspace(0, 2*np.pi, 100))*50})
     df_ret = VAR_model(df_train, df_test)
     show_graph(df_train, df_ret, "Vector Autoregression (VAR)")
```

Figure 21: VAR Model Implementation

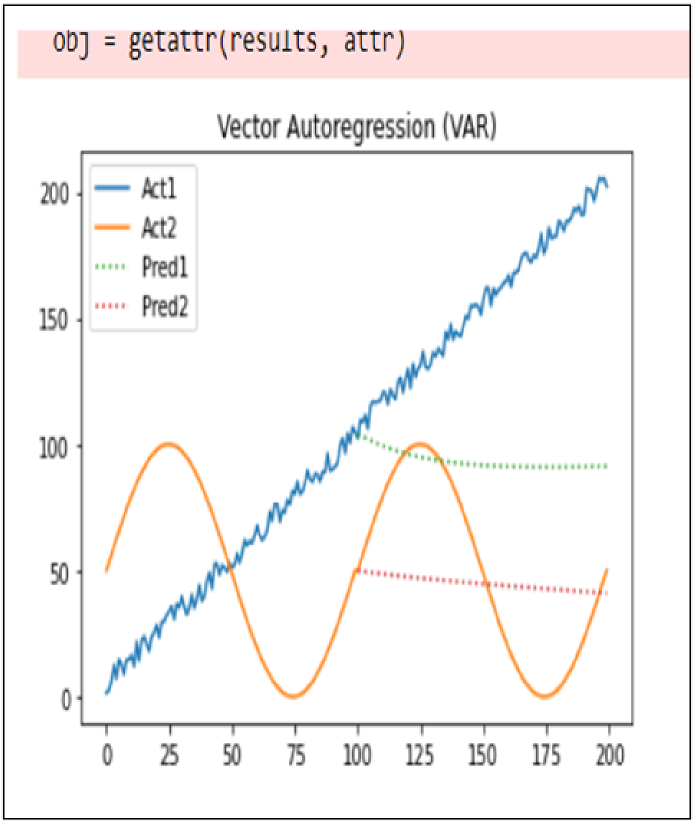

Figure 22: VAR Model Output

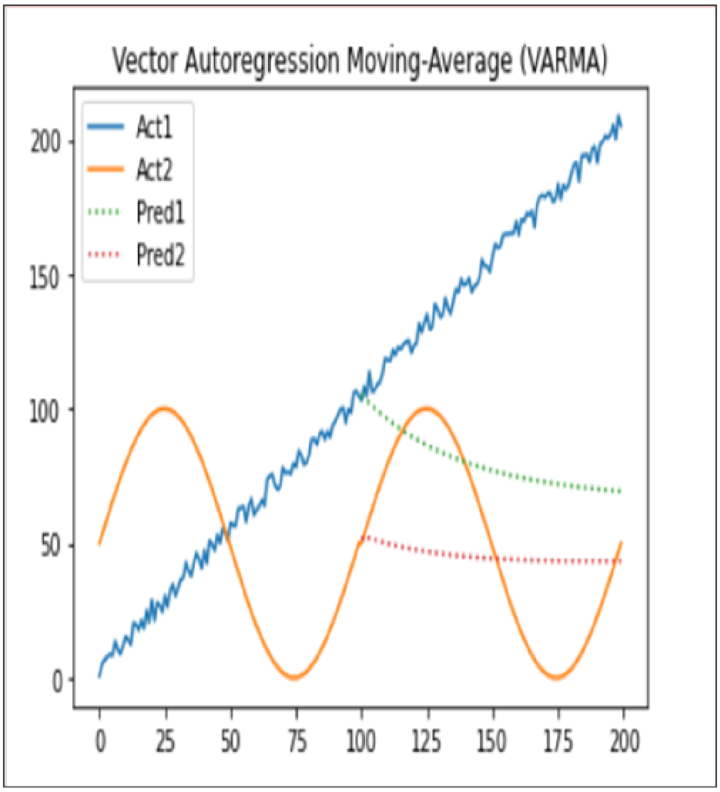

Figure 23: VARMA Model Output