Configuration Manual

MSc Research Project
Data Analytics

Yash Vijaywargiya
Student ID: 18136842

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Yash Vijaywargiya		
Student ID:	X18136842		
Programme:	Data Analytics	**Year:**	2019-2020
Module:	Research Project		
Supervisor:	Dr Catherine Mulwa		
Submission Due Date:	12-12-19		
Project Title:	Forecasting of Air Pollution in United Kingdom Using Deep Learning and Time series methods		
Word Count: 1038	……………………………………… **Page 10 Count**………………………………………….……..		

Signature: Yash Vijaywargiya

Date: 12-12-2019

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):	

Forecasting of Air Pollution in United Kingdom Using Deep Learning and Time series methods

Yash Vijaywargiya
X18136842

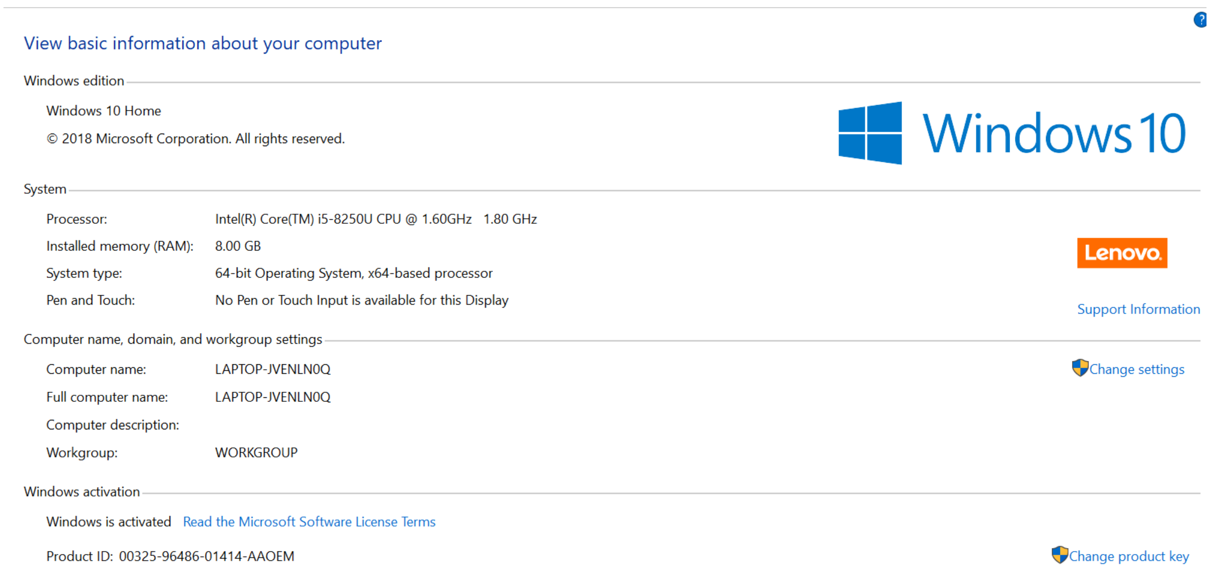
1 Introduction

The aim of this study is to forecast air pollution of the United Kingdom by taking data from the Europe site which is European Environment Agency. The data cleaning, pre-processing, exploration is then performed on this data which is having more than one lakh rows and seventeen columns. Different models were implemented like ARIMA, SARIMA, TBAT and neural networks. This manual is related to the important configurations that have been done for this forecasting project. It consists of all the information for the system configuration and various application's used for this research. The code of the program is been implemented.

2 System Specification

The project was been implemented on the laptop with the above specifications:

System configuration
LAPTOP Manufacturer: Lenovo ideapad 330s
OPERATING SYSTEM: Windows 10
RAM: 8GB
HARD DISC: 475Gb
PROCESSOR: i5 processor


View basic information about your computer

Windows edition
Windows 10 Home
© 2018 Microsoft Corporation. All rights reserved.

System

Processor:	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
Installed memory (RAM):	8.00 GB
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings

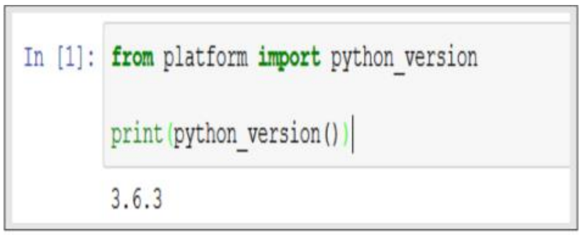
Computer name:	LAPTOP-JVENLN0Q
Full computer name:	LAPTOP-JVENLN0Q
Computer description:	
Workgroup:	WORKGROUP

Windows activation
Windows is activated Read the Microsoft Software License Terms
Product ID: 00325-96486-01414-AAOEM

Figure 1 System Specification

3 Software's involved

Python is the main software which is been used for running the models and getting the results. Python was been provided by the famous Anaconda tool. Excel and lucid chart has been used for making the model design and visualization of the graph. For setting up the software's mention above, the links are provided: For downloading Anaconda tool: https://www.anaconda.com/distribution/#download-section



```
In [1]: from platform import python_version

        print(python_version())

        3.6.3
```

Figure 2 Python Version

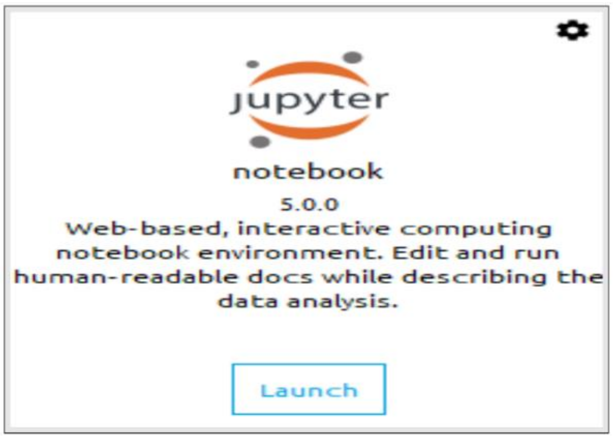
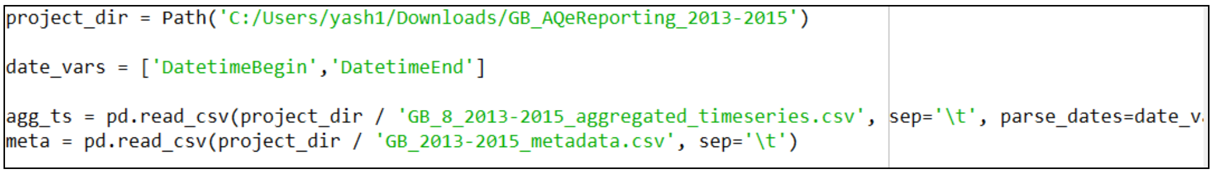

jupyter
notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Figure 3 Version of Jupyter

4 Data preparation and Feature Selection

1) **Load Dataset in Python:** The dataset was been downloaded from the very famous site of Europe which is European Environment Agency. The link is been given below: https://www.eea.europa.eu/data-and-maps/data/aqereporting-2/gb/gb-aqereporting-2014. The file contains data which is directly loaded into python for further processing.

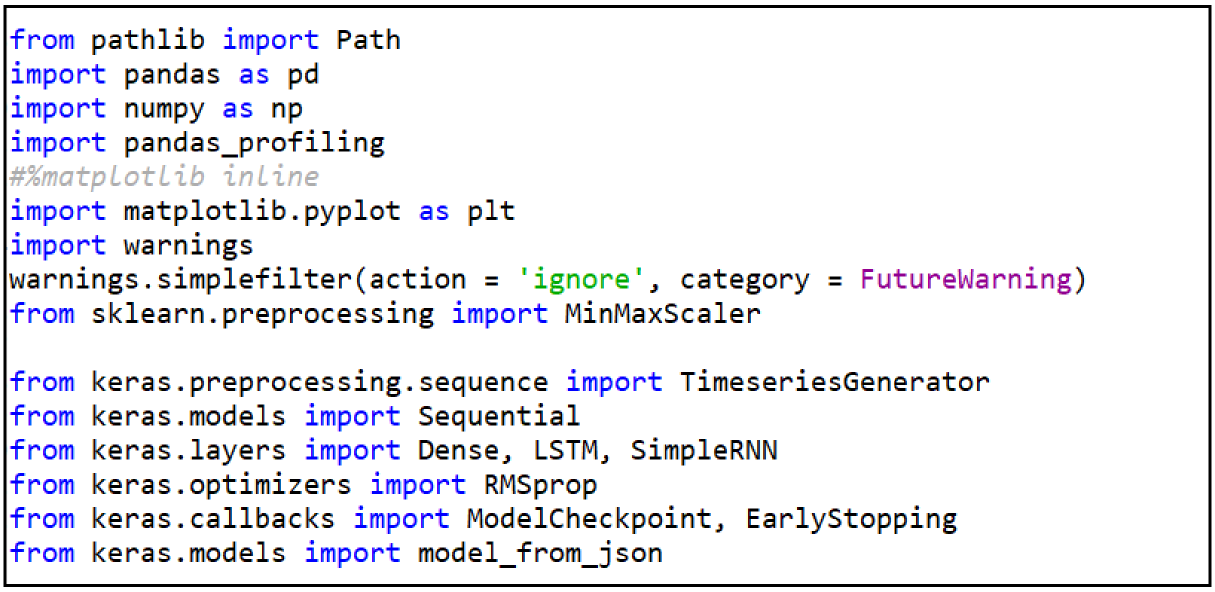


```
project_dir = Path('C:/Users/yash1/Downloads/GB_AQeReporting_2013-2015')

date_vars = ['DatetimeBegin','DatetimeEnd']

agg_ts = pd.read_csv(project_dir / 'GB_8_2013-2015_aggregated_timeseries.csv', sep='\t', parse_dates=date_v
meta = pd.read_csv(project_dir / 'GB_2013-2015_metadata.csv', sep='\t')
```

Figure 4 Loading

2) **Loading required libraries in Python:** All the above shown packages are been installed on the python software and libraries were run. The above figure includes all the required libraries for this project. Pandas library is used to handle dataframe for the pollution data exported in the file. Pumpy library has used here for

Converting data into n dims array of the pollutants no2. It has also used for performing operations like for calculating mean of DateofTime andDateofEnd from Air_index. Matplotlib library was used for making graph of the predicted pollution in United Kingdom by testing and training the dataset. The results are perfectly be seen here. Warning library is the major library used here which surpressed warning in console for all the errors produce by the code.

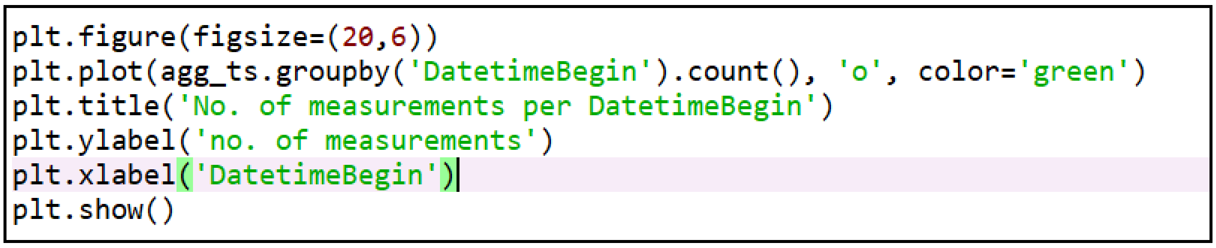


```
from pathlib import Path
import pandas as pd
import numpy as np
import pandas_profiling
#%matplotlib inline
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter(action = 'ignore', category = FutureWarning)
from sklearn.preprocessing import MinMaxScaler

from keras.preprocessing.sequence import TimeseriesGenerator
from keras.models import Sequential
from keras.layers import Dense, LSTM, SimpleRNN
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.models import model_from_json
```

Figure 5 Libraries

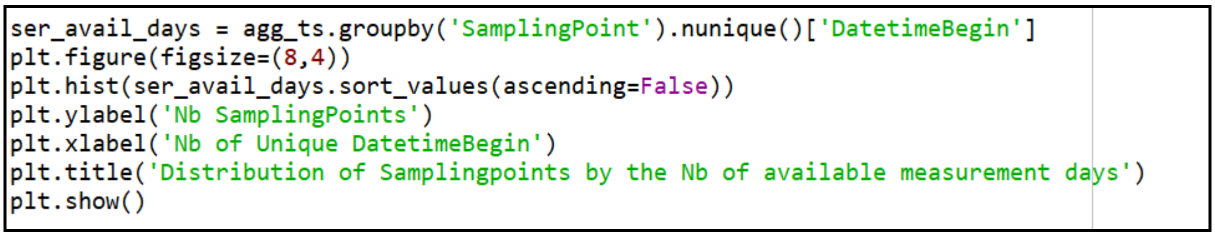
3) **Exploratory Data Analysis :** It was used for checking the quality of the data of Europe. The data accuracy is also been checked here and the , timestamp is right or not for Data

From fig 6 and fig 7, a graph is been plotted for the measurement of the set of attributes taken from the dataset from Datetime Begin.



```
plt.figure(figsize=(20,6))
plt.plot(agg_ts.groupby('DatetimeBegin').count(), 'o', color='green')
plt.title('No. of measurements per DatetimeBegin')
plt.ylabel('no. of measurements')
plt.xlabel('DatetimeBegin')
plt.show()
```

Figure 6 EDA1

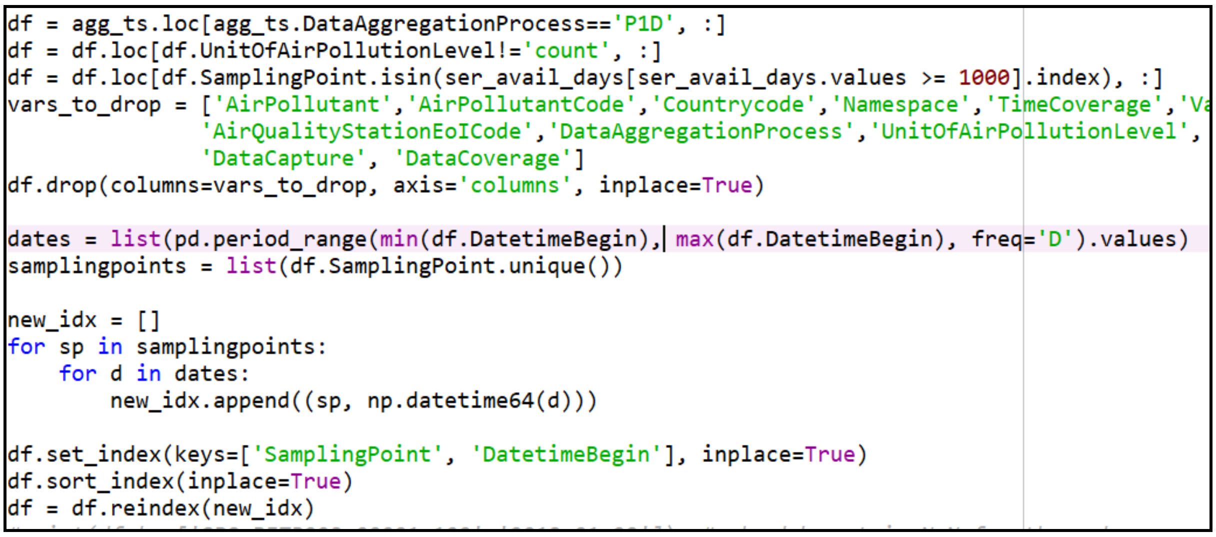


```
ser_avail_days = agg_ts.groupby('SamplingPoint').nunique()['DatetimeBegin']
plt.figure(figsize=(8,4))
plt.hist(ser_avail_days.sort_values(ascending=False))
plt.ylabel('Nb SamplingPoints')
plt.xlabel('Nb of Unique DatetimeBegin')
plt.title('Distribution of Samplingpoints by the Nb of available measurement days')
plt.show()
```

Figure 7 EDA 2

4) **Data Cleaning and Transformation**

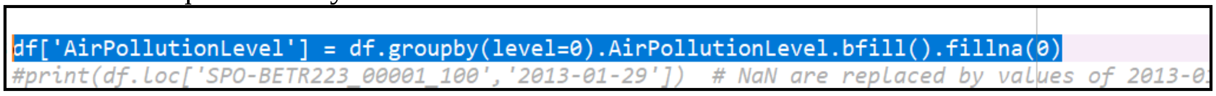
Data cleaning is used for dropping all 17 columns to the syntax df.drop. New timestamp is set and updated in np. datetime



```
df = agg_ts.loc[agg_ts.DataAggregationProcess=='P1D', :]
df = df.loc[df.UnitOfAirPollutionLevel!='count', :]
df = df.loc[df.SamplingPoint.isin(ser_avail_days[ser_avail_days.values >= 1000].index), :]
vars_to_drop = ['AirPollutant','AirPollutantCode','Countrycode','Namespace','TimeCoverage','Va
                'AirQualityStationEoICode','DataAggregationProcess','UnitOfAirPollutionLevel',
                'DataCapture', 'DataCoverage']
df.drop(columns=vars_to_drop, axis='columns', inplace=True)

dates = list(pd.period_range(min(df.DatetimeBegin), max(df.DatetimeBegin), freq='D').values)
samplingpoints = list(df.SamplingPoint.unique())

new_idx = []
for sp in samplingpoints:
    for d in dates:
        new_idx.append((sp, np.datetime64(d)))

df.set_index(keys=['SamplingPoint', 'DatetimeBegin'], inplace=True)
df.sort_index(inplace=True)
df = df.reindex(new_idx)
```

Figure 8 Data Cleaning

In the data transformation, air pollution level for the NA values are been splitted according to the models requirement by fillna.

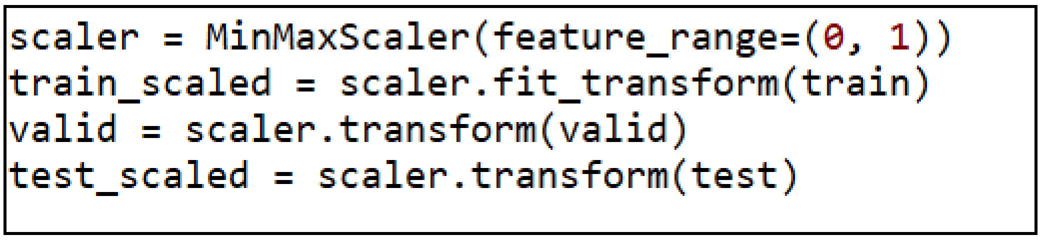


```
df['AirPollutionLevel'] = df.groupby(level=0).AirPollutionLevel.bfill().fillna(0)
#print(df.loc['SPO-BETR223_00001_100','2013-01-29'])  # NaN are replaced by values of 2013-0
```

Figure 9 Data transformation

5) Feature Scaling:

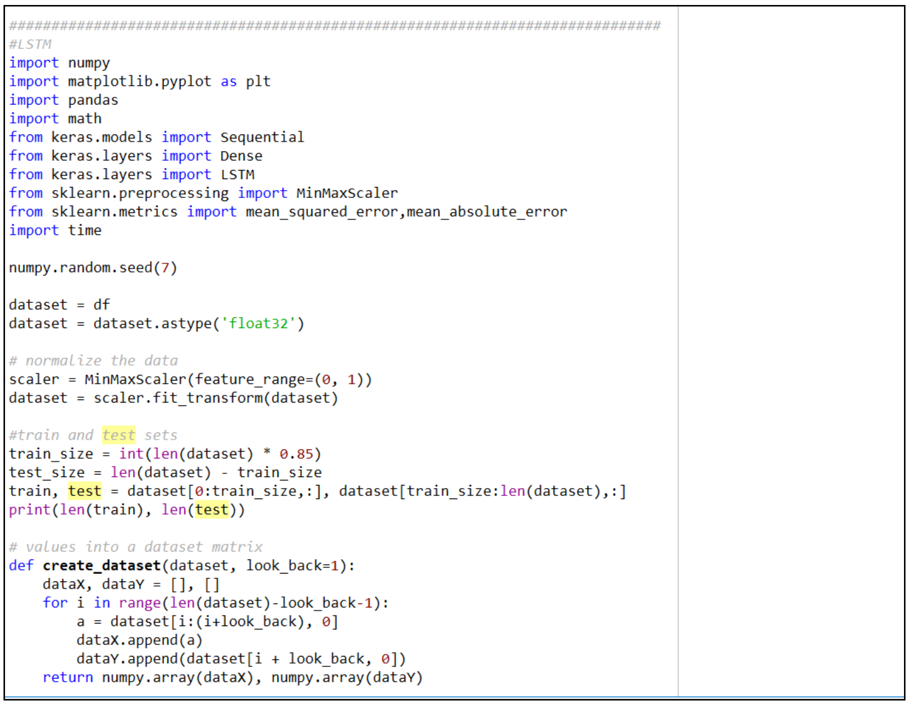
Feature scaling has been done using the function MinMaxScaler. All values has been converted In the range [0,1] Feature scaling has been done as models such as LSTM require scale data.



```
scaler = MinMaxScaler(feature_range=(0, 1))
train_scaled = scaler.fit_transform(train)
valid = scaler.transform(valid)
test_scaled = scaler.transform(test)
```

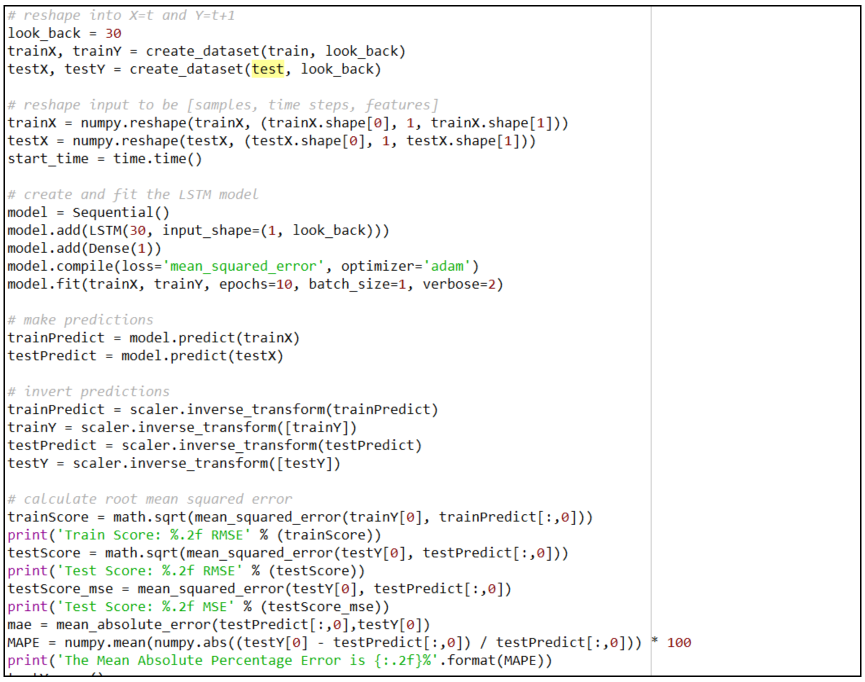
6) Implementation and Evaluation of the Air pollution forecasting:

i) LSTM: It is the LSTM architecutre . Train and test split of the dataset is mentioned and the basic transformation for LSTM to take as an input layer is showcase in the piece of code listed below.



```python
################################################################################
#LSTM
import numpy
import matplotlib.pyplot as plt
import pandas
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error,mean_absolute_error
import time

numpy.random.seed(7)

dataset = df
dataset = dataset.astype('float32')

# normalize the data
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)

#train and test sets
train_size = int(len(dataset) * 0.85)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
print(len(train), len(test))

# values into a dataset matrix
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

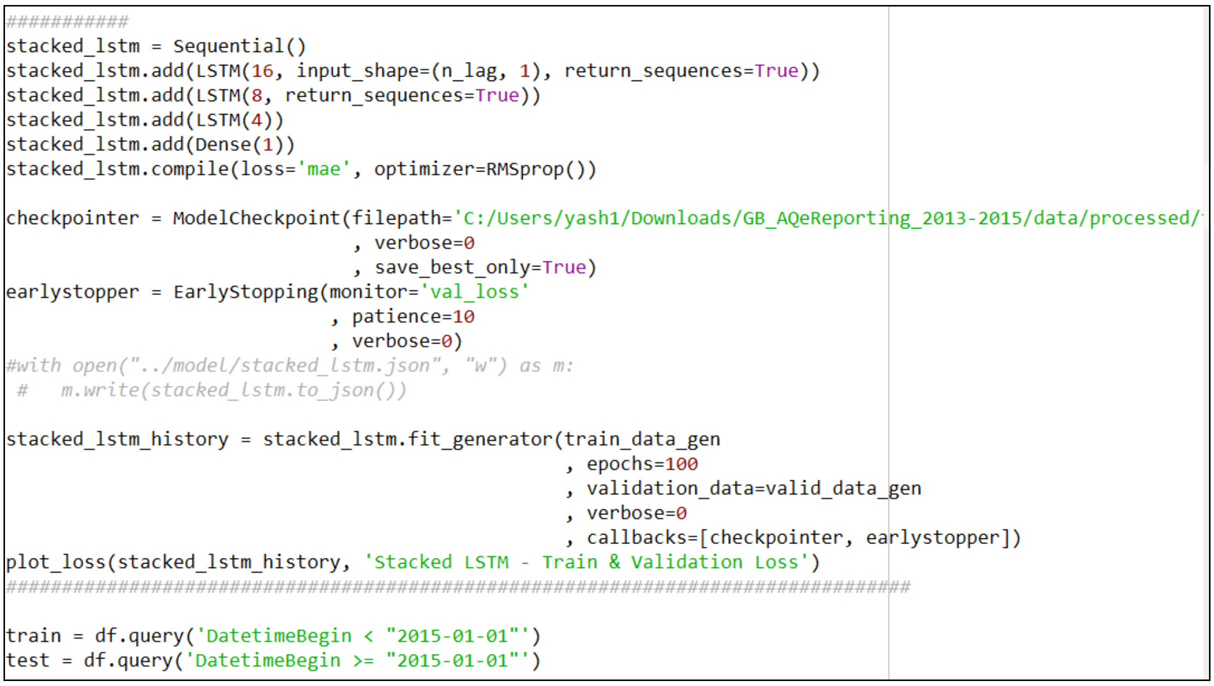
Figure 10 LSTM 1



```python
# reshape into X=t and Y=t+1
look_back = 30
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)

# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
start_time = time.time()

# create and fit the LSTM model
model = Sequential()
model.add(LSTM(30, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=10, batch_size=1, verbose=2)

# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)

# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])

# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
testScore_mse = mean_squared_error(testY[0], testPredict[:,0])
print('Test Score: %.2f MSE' % (testScore_mse))
mae = mean_absolute_error(testPredict[:,0],testY[0])
MAPE = numpy.mean(numpy.abs((testY[0] - testPredict[:,0]) / testPredict[:,0])) * 100
print('The Mean Absolute Percentage Error is {:.2f}%'.format(MAPE))
```

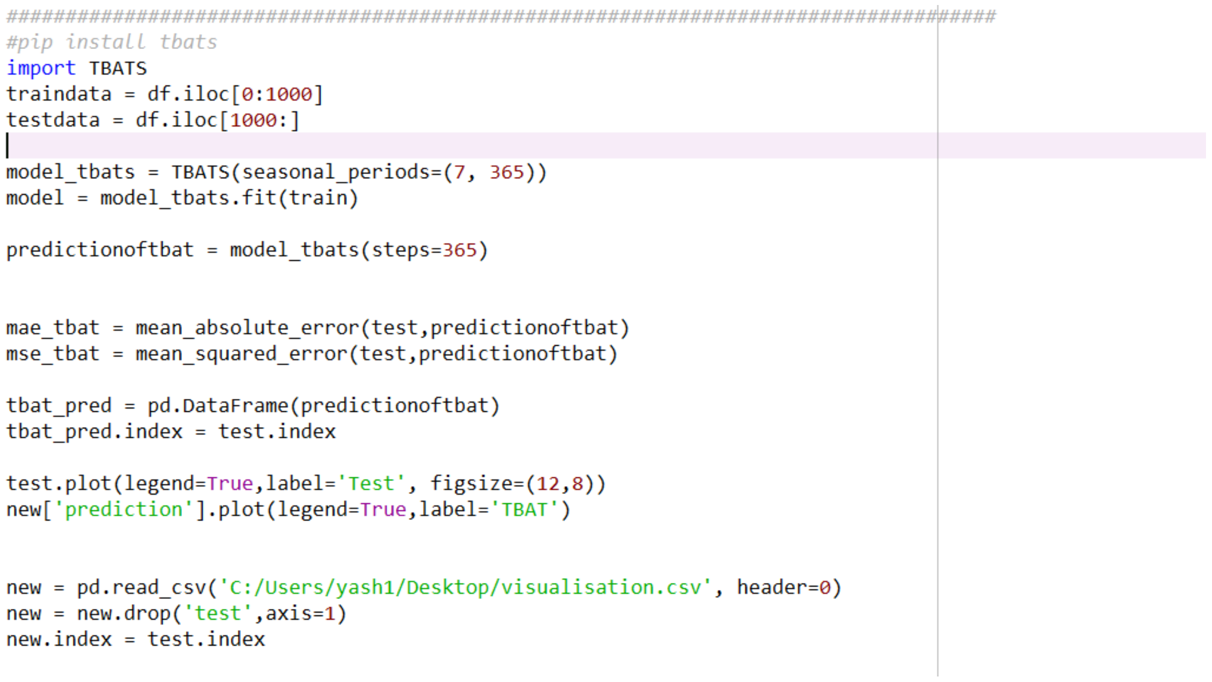
Figure 11 LSTM 2

Stacked LSTM: It is the stacked LSTM architecutre . Train and test split of the dataset is mentioned and the basic transformation for stacked LSTM to take as an input layer is showcase in the piece of code listed below.



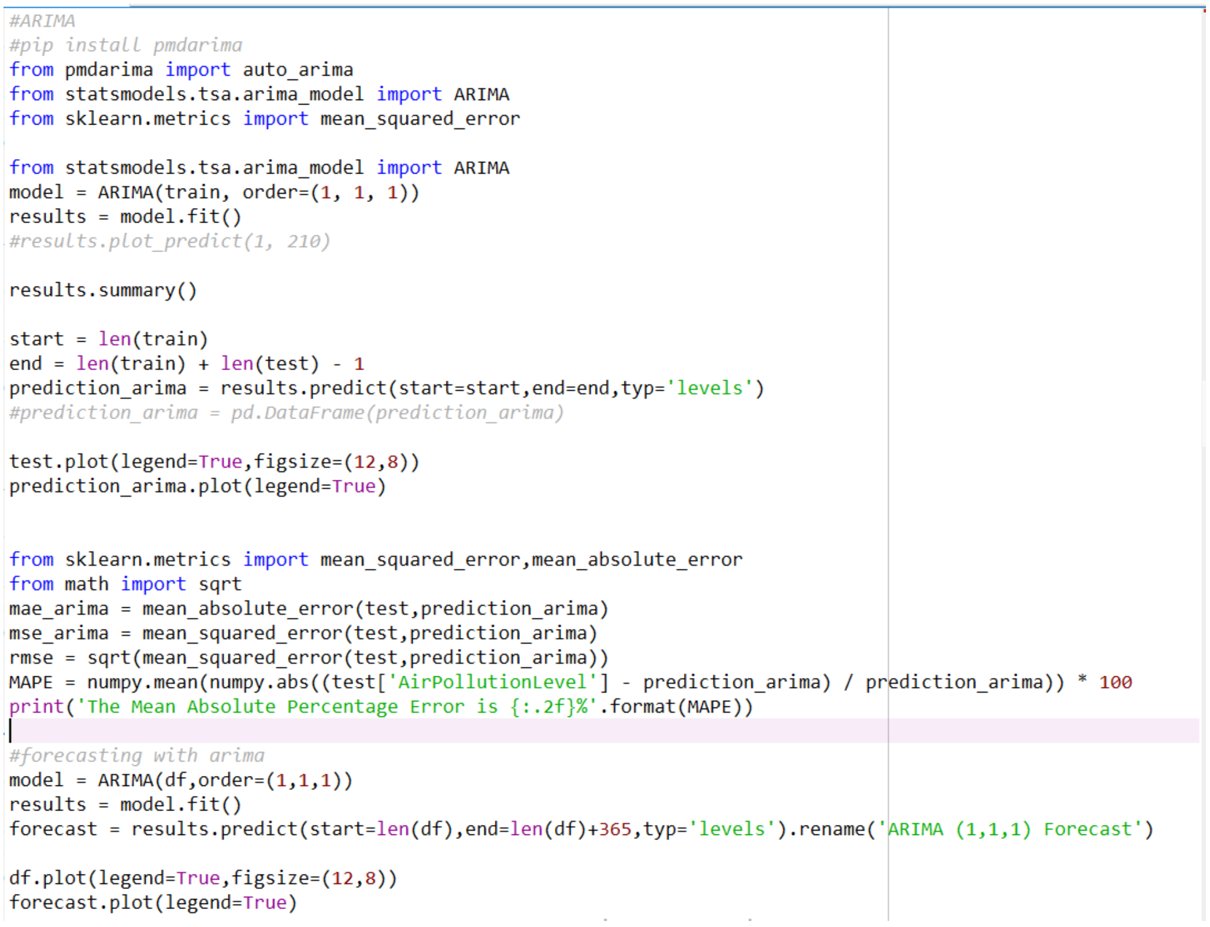
```
###########
stacked_lstm = Sequential()
stacked_lstm.add(LSTM(16, input_shape=(n_lag, 1), return_sequences=True))
stacked_lstm.add(LSTM(8, return_sequences=True))
stacked_lstm.add(LSTM(4))
stacked_lstm.add(Dense(1))
stacked_lstm.compile(loss='mae', optimizer=RMSprop())

checkpointer = ModelCheckpoint(filepath='C:/Users/yash1/Downloads/GB_AQeReporting_2013-2015/data/processed/
                               , verbose=0
                               , save_best_only=True)
earlystopper = EarlyStopping(monitor='val_loss'
                             , patience=10
                             , verbose=0)
#with open("../model/stacked_lstm.json", "w") as m:
 #   m.write(stacked_lstm.to_json())

stacked_lstm_history = stacked_lstm.fit_generator(train_data_gen
                                                  , epochs=100
                                                  , validation_data=valid_data_gen
                                                  , verbose=0
                                                  , callbacks=[checkpointer, earlystopper])
plot_loss(stacked_lstm_history, 'Stacked LSTM - Train & Validation Loss')
################################################################################

train = df.query('DatetimeBegin < "2015-01-01"')
test = df.query('DatetimeBegin >= "2015-01-01"')
```

TBATS: The below is the implementation of TBAT model. Prediction is done for 1 year and the MSE and Mae is been calculated using the forecasting and the test data we have.



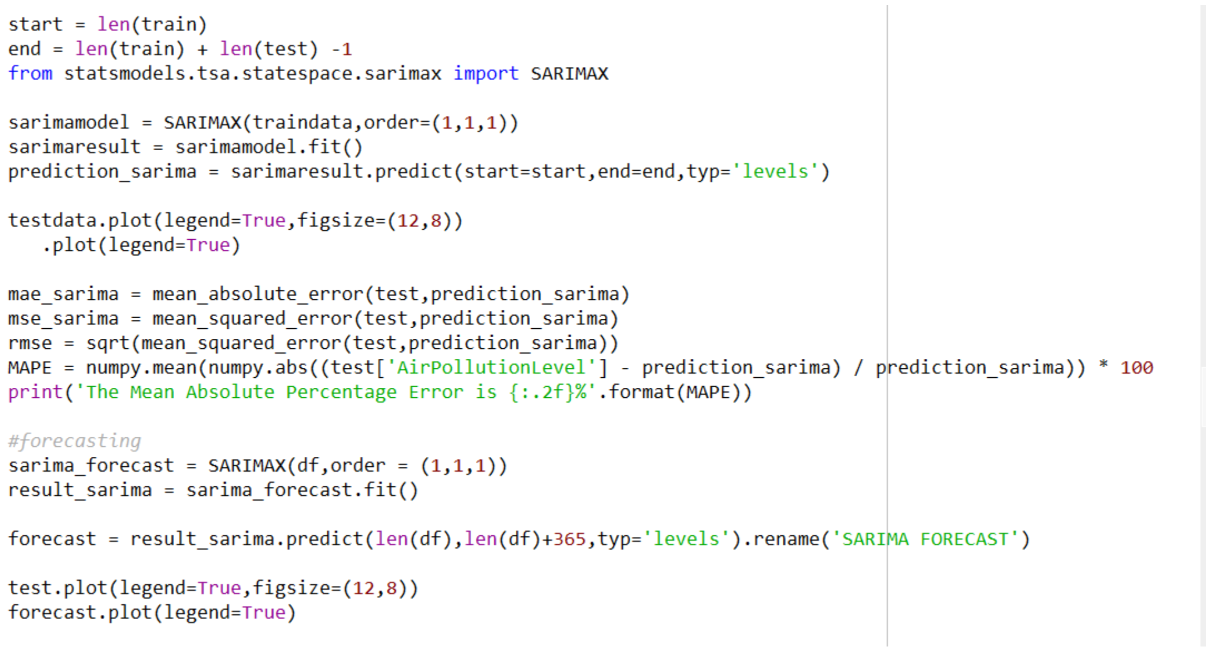
```
##########################################################################################
#pip install tbats
import TBATS
traindata = df.iloc[0:1000]
testdata = df.iloc[1000:]

model_tbats = TBATS(seasonal_periods=(7, 365))
model = model_tbats.fit(train)

predictionoftbat = model_tbats(steps=365)


mae_tbat = mean_absolute_error(test,predictionoftbat)
mse_tbat = mean_squared_error(test,predictionoftbat)

tbat_pred = pd.DataFrame(predictionoftbat)
tbat_pred.index = test.index

test.plot(legend=True,label='Test', figsize=(12,8))
new['prediction'].plot(legend=True,label='TBAT')


new = pd.read_csv('C:/Users/yash1/Desktop/visualisation.csv', header=0)
new = new.drop('test',axis=1)
new.index = test.index
```

ARIMA: The below is the implementation of ARIMA model. Model is imported by the function statsmodel.tsa. Prediction is done for 1 year and the MSE and MAE is been calculated using the forecasting and the test data we have.



```
#ARIMA
#pip install pmdarima
from pmdarima import auto_arima
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error

from statsmodels.tsa.arima_model import ARIMA
model = ARIMA(train, order=(1, 1, 1))
results = model.fit()
#results.plot_predict(1, 210)

results.summary()

start = len(train)
end = len(train) + len(test) - 1
prediction_arima = results.predict(start=start,end=end,typ='levels')
#prediction_arima = pd.DataFrame(prediction_arima)

test.plot(legend=True,figsize=(12,8))
prediction_arima.plot(legend=True)


from sklearn.metrics import mean_squared_error,mean_absolute_error
from math import sqrt
mae_arima = mean_absolute_error(test,prediction_arima)
mse_arima = mean_squared_error(test,prediction_arima)
rmse = sqrt(mean_squared_error(test,prediction_arima))
MAPE = numpy.mean(numpy.abs((test['AirPollutionLevel'] - prediction_arima) / prediction_arima)) * 100
print('The Mean Absolute Percentage Error is {:.2f}%'.format(MAPE))

#forecasting with arima
model = ARIMA(df,order=(1,1,1))
results = model.fit()
forecast = results.predict(start=len(df),end=len(df)+365,typ='levels').rename('ARIMA (1,1,1) Forecast')

df.plot(legend=True,figsize=(12,8))
forecast.plot(legend=True)
```

SARIMA: The below is the implementation of SARIMA model. Model is imported by the values (p,d,q)=(1,1,1). Prediction is done for 1 year and the MSE and MAE is been calculated using the forecasting and the test data we have.



```
start = len(train)
end = len(train) + len(test) -1
from statsmodels.tsa.statespace.sarimax import SARIMAX

sarimamodel = SARIMAX(traindata,order=(1,1,1))
sarimaresult = sarimamodel.fit()
prediction_sarima = sarimaresult.predict(start=start,end=end,typ='levels')

testdata.plot(legend=True,figsize=(12,8))
   .plot(legend=True)

mae_sarima = mean_absolute_error(test,prediction_sarima)
mse_sarima = mean_squared_error(test,prediction_sarima)
rmse = sqrt(mean_squared_error(test,prediction_sarima))
MAPE = numpy.mean(numpy.abs((test['AirPollutionLevel'] - prediction_sarima) / prediction_sarima)) * 100
print('The Mean Absolute Percentage Error is {:.2f}%'.format(MAPE))

#forecasting
sarima_forecast = SARIMAX(df,order = (1,1,1))
result_sarima = sarima_forecast.fit()

forecast = result_sarima.predict(len(df),len(df)+365,typ='levels').rename('SARIMA FORECAST')

test.plot(legend=True,figsize=(12,8))
forecast.plot(legend=True)
```