National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Harsh Chudasama
Student ID: X18187340

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Harsh Chudasama
Student ID:	X18187340
Programme:	Data Analytics
Year:	2020
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	17/08/2020
Project Title:	Configuration Manual
Word Count:	650
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:	
Date:	17th August 2020

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):	

Configuration Manual

Harsh Chudasama
X18187340

1 Introduction

This configuration manual specifies the hardware and software requirements and the code snippets explaining the implementation of the below research project in detail:

"Forecasting the Novel Coronavirus(COVID-19) using Time Series Model"

2 System Configurations

2.1 Hardware

- Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.60 GHz
- RAM: 8 GB
- System Type: Windows 10 (64 bit)
- GPU: Intel(R) UHD Graphics 620 , 2 GB
- Storage: 256 GB SSD

2.2 Software

- **Microsoft Excel 2019**
  I am currently using the latest version of MS Excel(2019) Version for the data handling for Comma Separated Values(CSV) and Plotting the graphs. Additionally, the NumXL Plugin[1] is installed for ARIMA and ARMA Forecasting.

- **Anaconda Distribution-Jupyter Notebook**
  Jupyter Notebook[2] is an interactive cell-based Integrated development environment(IDE).In this software, Python or R machine learning codes can be executed in a cell and output will be displayed. For this project, Exploratory data analysis, Data preprocessing, Data Transformation, Visualization, and Model Implementation is done using Jupyter Notebook ***(Python Version 3.8.3)***

[1]NumXL Tool Plugin : `https://www.numxl.com/products/numxl`
[2]Anaconda Distribution `https://www.anaconda.com/distribution/`

3 Project Development

The research project was executed successfully and all the four models(LSTM, ARMA, ARIMA, and Prophet) were deployed. Python language was used for scripting in the Jupyter Notebook IDE. The Project Development lifecycle consists of following steps Data preparation, Data Transformation, and Time Series Modelling.

3.1 Data preparation

For this project, Data(University; 2020) is downloaded and loaded in the Jupyter Notebook using Numpy libraries, and the Data manipulation task is done using the Pandas library.

3.1.1 Symptom Dataset

The main objective of this project was to extract the key symptoms for early prediction of COVID-19. The dataset was loaded successfully using Numpy CSV reader function and it consisted of 32 columns with 14,216 entries(Refer to Figure 1).

```
#Importing the dataset
open_line = pd.read_csv('COVID19_open_line_list.csv')
open_line = open_line.iloc[:, :-12]
# Check for missing information, datatypes and shape
open_line.info()
#Output
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14126 entries, 0 to 14125
Data columns (total 33 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   ID                         13173 non-null  float64
 1   age                        1349 non-null   object
 2   sex                        1264 non-null   object
 3   city                       10194 non-null  object
 4   province                   12906 non-null  object
 5   country                    13148 non-null  object
 6   wuhan(0)_not_wuhan(1)      13170 non-null  float64
 7   latitude                   13147 non-null  float64
 8   longitude                  13147 non-null  float64
 9   geo_resolution             13147 non-null  object
 10  date_onset_symptoms        746 non-null    object
 11  date_admission_hospital    730 non-null    object
 12  date_confirmation          13089 non-null  object
 13  symptoms                   493 non-null    object
 14  lives_in_Wuhan             565 non-null    object
 15  travel_history_dates       503 non-null    object
 16  travel_history_location    758 non-null    object
 17  reported_market_exposure   35 non-null     object
 18  additional_information     2412 non-null   object
 19  chronic_disease_binary     18 non-null     float64
 20  chronic_disease            13 non-null     object
 21  source                     12950 non-null  object
 22  sequence_available         1 non-null      object
 23  outcome                    184 non-null    object
 24  date_death_or_discharge    93 non-null     object
 25  notes_for_discussion       187 non-null    object
 26  location                   1024 non-null   object
 27  admin3                     1159 non-null   object
 28  admin2                     9068 non-null   object
 29  admin1                     12877 non-null  object
 30  country_new                13079 non-null  object
 31  admin_id                   13103 non-null  object
 32  data_moderator_initials    17 non-null     object
dtypes: float64(5), object(28)
```

Figure 1: Symptom Dataset

3.1.2 COVID-19 Dataset

This is the main dataset which consists of the Number of confirmed cases, deaths, and recovered cases that are mapped to their respective states(Refer to Figure 2).

In [1]:
```
#importing Libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
%matplotlib inline
import matplotlib # for data visualization
from matplotlib import cm # for data visualization
import geopandas as gpd #To plot Indian Country Border Map
import plotly.express as px # for data visualization
import seaborn as sns # for data visualization
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)    #THIS LINE IS MOST IMPORTANT AS THIS WILL DISPLAY PLOT ON
#NOTEBOOK WHILE KERNEL IS RUNNING
import plotly.graph_objects as go
from IPython.display import HTML,display
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
#importing main dataset
df = pd.read_csv('covid_19_india.csv', parse_dates=['Date'], dayfirst=True)

#importing tests dataset
tests = pd.read_csv('StatewiseTestingDetails.csv' )
#keeping only required columns
df = df[['Date', 'State/UnionTerritory','Cured','Deaths','Confirmed']]
#renaming column names
df.columns = ['date', 'state','cured','deaths','confirmed']
```

In [3]:
```
df.tail(5)
```

Out[3]:

	date	state	cured	deaths	confirmed
4841	2020-08-04	Telengana	49675	563	68946
4842	2020-08-04	Tripura	3675	28	5505
4843	2020-08-04	Uttarakhand	4538	90	7800
4844	2020-08-04	Uttar Pradesh	55393	1778	97362
4845	2020-08-04	West Bengal	54818	1731	78232

Figure 2: COVID-19 Dataset

3.2 Data Transformation

3.2.1 Symptom Dataset

Dataset was loaded successfully with 32 features consisting of 14,216 rows. But, not all features were required for the analysis, and hence a sneak peek of data was done with the first four rows of each column with its data types and respective values counter. Below are a few observations (Refer to Figure 3) :

- Feature 'sex' contains some ambiguities and requires unifying the values.
- Feature 'Wuhan(0)_not_wuhan(1)' indicates that all cases originate from outside of Wuhan the i.e. epicenter of the disease.
- Features **('age','additional_information', 'reported_market_exposure')** are currently not planning for the project scope.
- The following features are removed as they are inconsistent and provide ambiguous information:

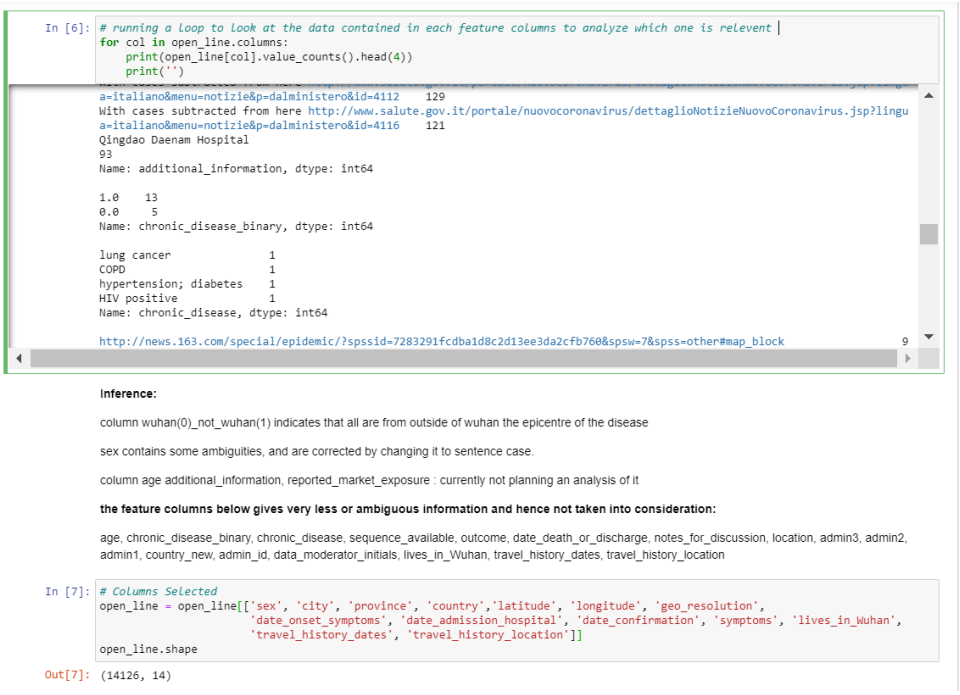

Figure 3: Symptom Dataset Analsis

age, chronic_disease_binary, chronic_disease, sequence_available, outcome, date_death_or_discharge, notes_for_discussion, location, admin3, admin2, admin1, country_new, admin_id, data_moderator_initials, lives_in_Wuhan, travel_history_dates, travel_history_location

- As the feature **'sex** 'consisted of ambiguity it was normalized and the graph was plotted to look at the date(Refer to Figure 4).

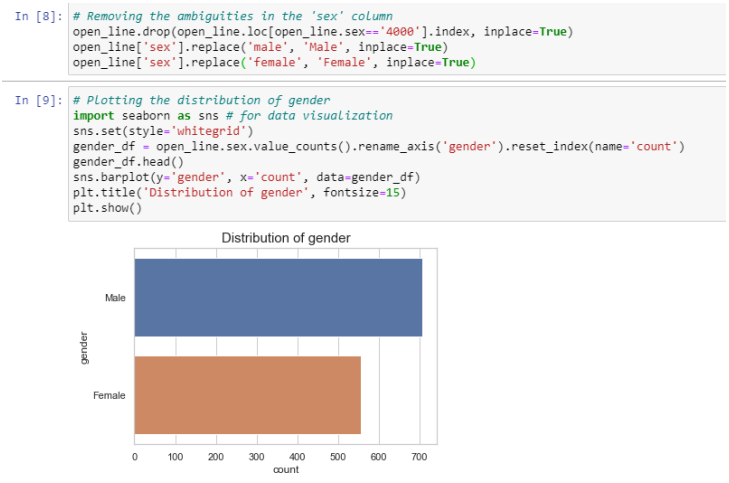

Figure 4: Normalized Feature 'Sex'

- Based on the feature 'Symptom', Text analysis was performed to extract the Top 10 Key Symptoms by defining the custom function(Refer to Figure 5).

```
In [11]: # A function to extract symptoms
         def find_symptoms(word):
             word_split = word.replace('()',',').split(',')
             word_split = [word.strip().rstrip(',') for word in word_split]
             key_symptoms.extend(word_split)

In [12]: # creating a dataframe of major symptoms (Top 10)
         key_symptoms = []
         symptoms_df['symptoms'].dropna().apply(find_symptoms)
         key_symptoms = pd.Series(key_symptoms)
         key_symptoms = key_symptoms[key_symptoms!='']
         major_symptoms = key_symptoms.value_counts()
         print("Top 10 Major Sympyoms identified are :")
         major_symptoms[:10]

         Top 10 Major Sympyoms identified are :

Out[12]: fever          290
         cough          158
         sore throat     27
         pneumonitis     19
         fatigue         17
         chills          16
         pneumonia       16
         headache        13
         runny nose      13
         malaise         12
         dtype: int64
```

Figure 5: Top 10 Key Symptoms Feature Extraction

- The Outcome of the key Symptom was summarized using a Word cloud package(Refer to Figure 6).

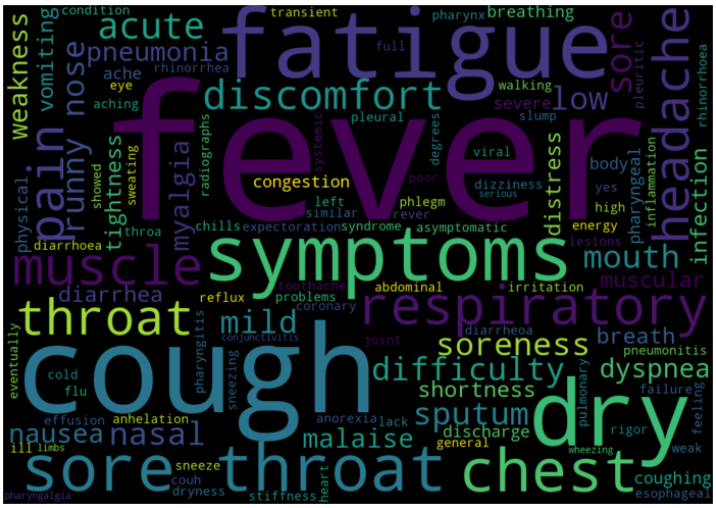

Figure 6: Word Cloud of Key Symptoms

3.2.2 COVID-19 Dataset

- Some discrepancies were observed in the state names and hence they were corrected and stored in new data frame 'merged'(Refer to Figure 7).
- Additional feature 'merged['dcratio']' was derived using the following formula and applied to each row of the data frame.

1.2 Covid-19 Exploratory Data Analysis

```
In [15]: #current date
         today = df[df.date == '2020-08-04']

In [16]: # Data Transformation
         gdf = gpd.read_file('Indian_States.shp')
         #renaming state names
         gdf['st_nm'].replace({"Andaman & Nicobar Island": "Andaman and Nicobar Islands",
                               "Arunanchal Pradesh": "Arunachal Pradesh",
                               'Dadara & Nagar Havelli':'Dadra and Nagar Haveli and Daman and Diu',
                               'Jammu & Kashmir':'Jammu and Kashmir',
                               'NCT of Delhi':'Delhi'}, inplace=True)
         merged = gdf.merge(today ,  left_on='st_nm', right_on='state')
         merged.state.replace({"Andaman and Nicobar Islands" : "A & N",
                               'Dadra and Nagar Haveli and Daman and Diu' : 'Daman & Diu',
                               'Jammu and Kashmir' : 'J & K'
                               }, inplace=True)

In [17]: merged.info()

         <class 'geopandas.geodataframe.GeoDataFrame'>
         Int64Index: 33 entries, 0 to 32
         Data columns (total 7 columns):
          #   Column     Non-Null Count  Dtype
         ---  ------     --------------  -----
          0   st_nm      33 non-null     object
          1   geometry   33 non-null     geometry
          2   date       33 non-null     datetime64[ns]
          3   state      33 non-null     object
          4   cured      33 non-null     int64
          5   deaths     33 non-null     int64
          6   confirmed  33 non-null     int64
         dtypes: datetime64[ns](1), geometry(1), int64(3), object(2)
         memory usage: 2.1+ KB
```

Figure 7: COVID-19 Dataset Transformation

merged['dcratio'] = merged['deaths'] / merged['confirmed'] * 100

- Seaborn library was used for plotting the graph of Fatality Rate Per State that required geopandas library for plotting the layout of India map (Refer to Figure 8).

3.3 Time Series Modelling

3.3.1 LSTM Model

- LSTM model was implemented which belongs to the following Recurrent Neural Network(RNN) class (Ayyoubzadeh et al.; 2020).
- Model Consisted of following layers deep neural network layer followed by dense and drop out layer(Refer to Figure 9).

3.3.2 Prophet Model

- The simplest and straight forward model to implement.
- Simple provide the input time series and give trend-setting information.(Refer to Figure 10)(Taylor and Letham; 2018).

3.3.3 ARIMA & ARMA Model

- These models were implemented using the MS Excel NumXL Plugin(Zhang; 2003).
- Initially, data were log-transformed and provided as input and the stationary test was performed.

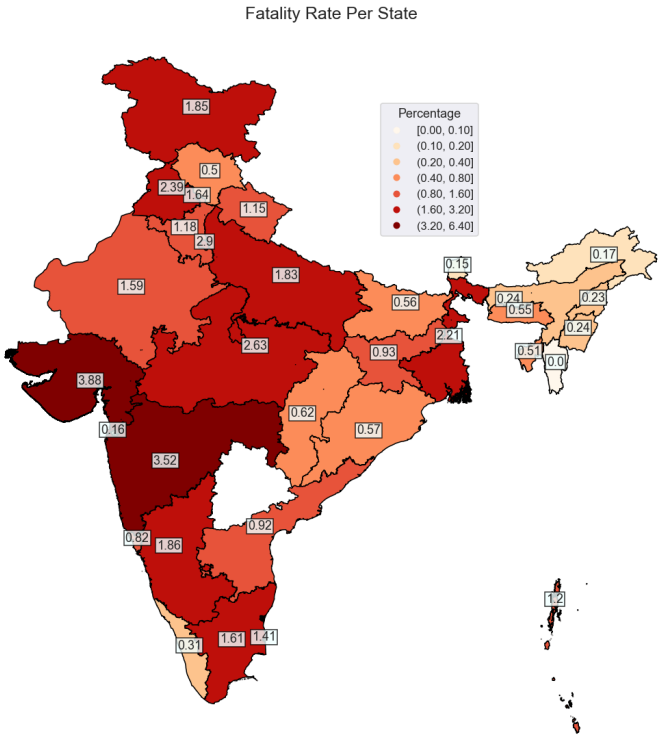

Figure 8: Fatality Rate Per State

- Stationary test results proved that model shows ARCH Effect i.e. lags are not constant rather exponential(Refer to Figure 11)(Kelvin et al.; 2020).
- In conclusion, The model is aligned dependent on beginning segments and coefficients, and the result is anticipated with the ideal strides to conjecture(Refer to Figure 12).

References

Ayyoubzadeh, S. M., Ayyoubzadeh, S. M., Zahedi, H., Ahmadi, M. and R Niakan Kalhori, S. (2020). Predicting covid-19 incidence through analysis of google trends data in iran: Data mining and deep learning pilot study, *JMIR Public Health Surveill* **6**(2): e18828.
URL: *https://doi.org/10.2196/18828*

Kelvin, D., Rubino, S. and Kelvin, N. (2020). Similarity in case fatality rates (cfr) of covid-19/sars-cov-2 in italy and china, *J Infect Dev Ctries* **14**(2): 125–128.
URL: *https://jidc.org/index.php/journal/article/view/32146445*

Taylor, S. J. and Letham, B. (2018). Forecasting at Scale, *The American Statistician* **72**(1): 37–45.
URL: *https://ideas.repec.org/a/taf/amstat/v72y2018i1p37-45.html*

University, J. H. (2020). Covid-19 dataset.
URL: *https://github.com/CSSEGISandData/COVID-19*

In [172]:

```
#Importing the required Package
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

#Scaling the date
scaler.fit(train_data)
scaled_train_data = scaler.transform(train_data)
scaled_test_data = scaler.transform(test_data)

#Creating Dataframe of Scaled Data as Model Input
data = pd.DataFrame(columns = ['ds','y'])
data['ds'] = train_data.index
data['y'] = scaled_train_data
#Creating Model
#from keras.preprocessing.sequence import TimeseriesGenerator
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
import tensorflow as tf

n_input = 12 # Number of Forecasting Days
n_features= 1 #Input Columns

#generating batches of temporal data
generator = TimeseriesGenerator(scaled_train_data, #Scaled Input Train Data
                                scaled_train_data, #Target Data
                                length=n_input, #Length as per input size=12
                                batch_size=1) #Batches generated

#Starting of Model in sequential manner
lstm_model = Sequential()
#First LSTM Layer of size (12,1) with 500 hidden layers
lstm_model.add(LSTM(500, activation='tanh', input_shape=(n_input, n_features)))
#Drop out layer
lstm_model.add(Dropout(0.10))
#Adding Dense Layer
lstm_model.add(Dense(1))
#Compiling the model with Optimizer as adam
lstm_model.compile(optimizer='adam', loss='mse')
#Displaying Summary of the model
lstm_model.summary()
#Generating the model with 50 Epochs
lstm_model.fit_generator(generator,epochs=50)
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_19 (LSTM)               (None, 500)               1004000
_________________________________________________________________
dropout_10 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 501
=================================================================
Total params: 1,004,501
Trainable params: 1,004,501
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: Code for LSTM Model

In [130]:
```
#importing fbprophet
from fbprophet import Prophet

#model
m = Prophet()

#fitting the model
m.fit(df3)

#forecasting Future dates
future = m.make_future_dataframe(periods= 12)
future.tail(12)

#rename the column
fb_res.columns = ['ds','FBProphet']
fb_res['FBProphet'] = fb_res['FBProphet'].astype(int)
result = pd.concat([df2,fb_res],axis=1)
del result['deaths']
del result['cured']
del result['ds']
result.FBProphet = result.FBProphet.replace(np.nan, 0)
out = result.tail(12)
result.tail(12)
```

Out[130]:

	date	confirmed	FBProphet
176	2020-07-24	1287945	1148408.0
177	2020-07-25	1336861	1171513.0
178	2020-07-26	1385522	1195112.0
179	2020-07-27	1435453	1218749.0
180	2020-07-28	1483156	1242036.0
181	2020-07-29	1531669	1265485.0
182	2020-07-30	1583792	1288950.0
183	2020-07-31	1638870	1307538.0
184	2020-08-01	1695988	1330644.0
185	2020-08-02	1750723	1354242.0
186	2020-08-03	1803695	1377879.0
187	2020-08-04	1855745	1401167.0

Figure 10: Code for FB Prophet Model

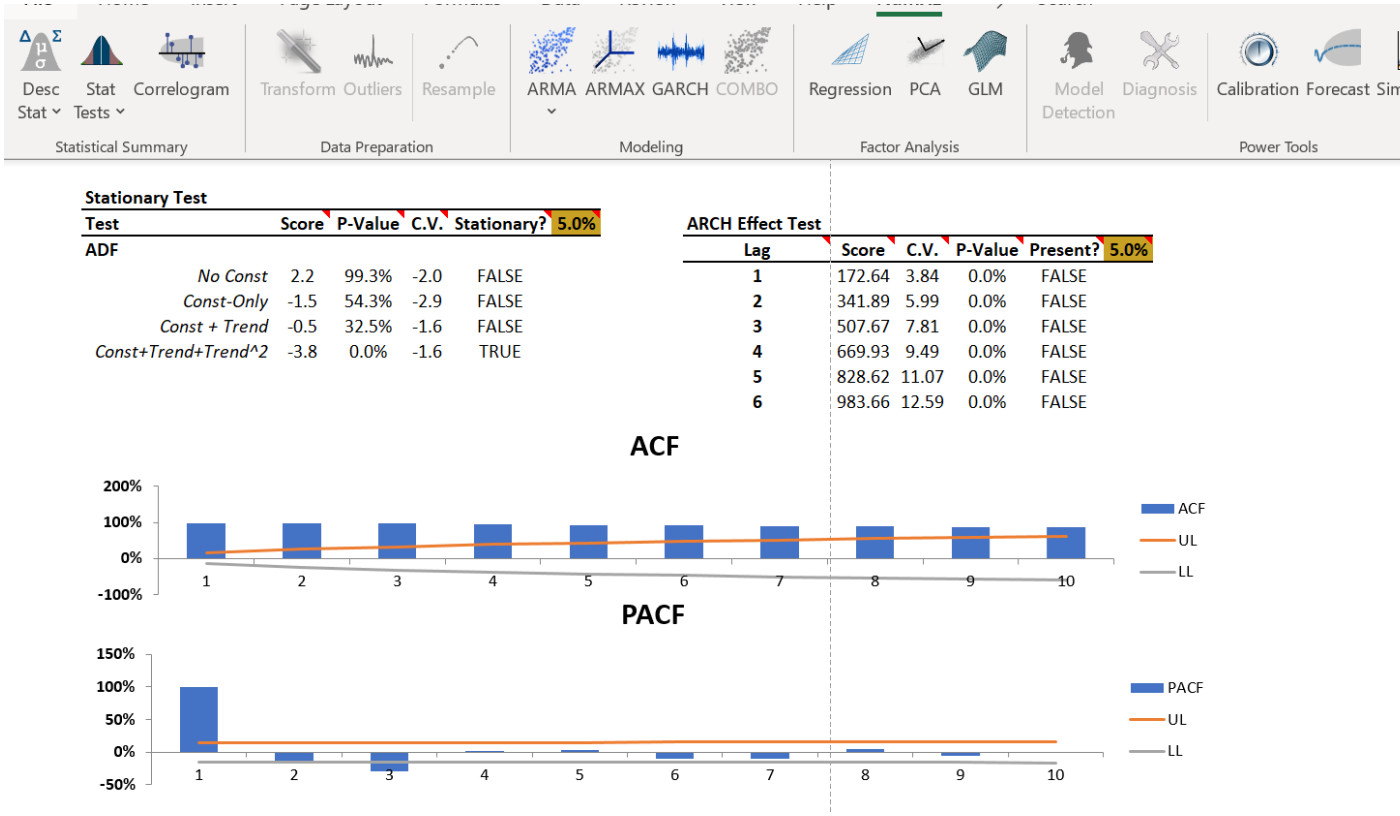

Figure 11: Stationary Test Check for ARIMA & ARMA Model

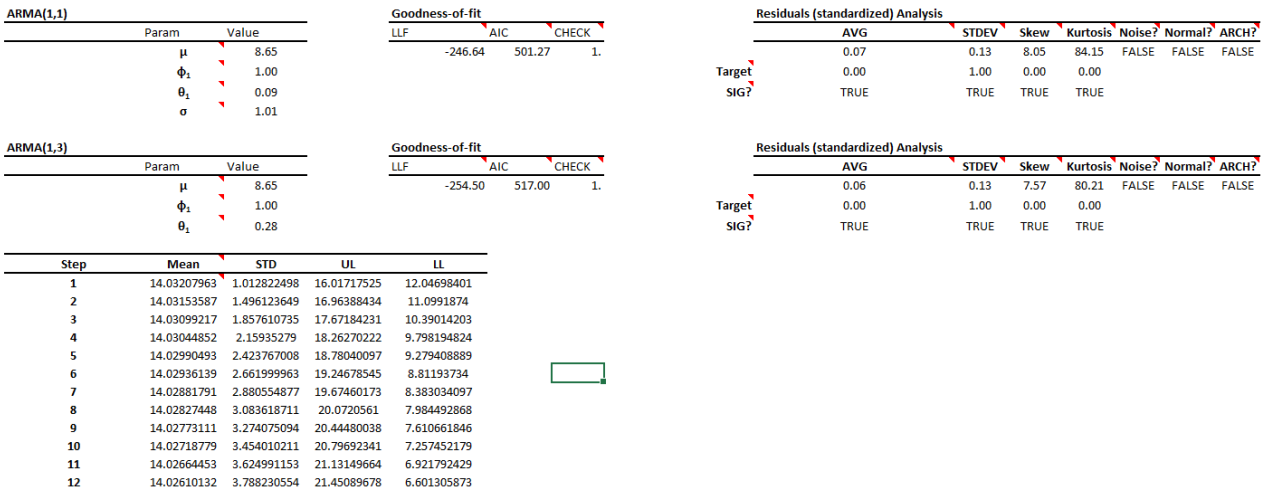

Figure 12: ARMA Model Predictions

Zhang, G. (2003). Time series forecasting using a hybrid arima and neural network model, *Neurocomputing* **50**: 159 – 175.
URL: *http://www.sciencedirect.com/science/article/pii/S0925231201007020*