National College of Ireland

Configuration Manual

MSc Research Project
MSc. In Data Analytics

Yash Nilesh Mehta
Student ID: x18179916

School of Computing
National College of Ireland

Supervisor: Mr. Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Yash Nilesh Mehta
Student ID:	X18179916
Programme:	MSc Data Analytics **Year:** 2019-2020
Module:	MSc Research Project
Supervisor:	Mr. Hicham Rifai
Submission Due Date:	28th September 2020
Project Title:	"Impact Analysis of Market sentiments, Gold and Crude oil prices on DOW30 stocks"
Word Count: 936	**Page Count:** 12

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: Yash Nilesh Mehta

Date: 27th September 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

Yash Nilesh Mehta
X18179916

1 Introduction

This configuration manual provides the course of action required to be followed in order to replicate the proposed research and achieve desired results. The manual includes requirements of system configuration, steps for collecting and cleaning the datasets, steps and code snippets of implementing the models and lastly the steps and code snippets for evaluating the model.

2 System Configuration

All the required tools and software used for this research can be easily installed in any computer system having basic configuration listed below:

Operating System	Windows 10
RAM	8GB
Hard Disk	128GB+ SSD
Processor	Intel Core i5 8th gen

This research work used the basic tools that are given below:

a. Microsoft Office Suite
b. Python 3.7
c. Anaconda Jupyter Notebook

Microsoft excel and Microsoft Word are the two tools used from MS office suite. MS word is used for the purpose of reporting whereas, MS Excel is used for viewing the data and performing few merging operations. Python is the used for all the processes like data pre-processing, EDA, model building and evaluation. Python 3.7 is used which is open source and can be downloaded from the official website[1]. Jupyter Notebook is used as IDE for entire coding. Jupyter notebook is accessed from platform called Anaconda, which can be downloaded from official website[2].

1 https://www.python.org/downloads/

2 https://repo.anaconda.com/archive/Anaconda3-2019.10-Windows-x86_64.exe

3 Dataset Generation

Total four datasets are used for this research work, which are given as below:

1. DJIA stocks dataset:

   This dataset has been downloaded from Yahoo finance official website[3] in CSV format. Here the time period for which data is required can be selected, and the resultant data can be downloaded without any cost.

2. Gold Prices Dataset:

   This dataset has been downloaded from Yahoo finance official website[4] in CSV format. Here the time period for which data is required can be selected, and the resultant data can be downloaded without any cost.

3. Crude oil prices dataset:

   This dataset has been downloaded from Yahoo finance official website[5] in CSV format. Here the time period for which data is required can be selected, and the resultant data can be downloaded without any cost.

4. Twitter Sentiments data:

   This dataset is requested from one of the previous research works (Jain, 2019) done in the same field. This dataset already has the sentiment label as the author has originally done text processing and defined the sentiments for tweets. So there was no further requirement of text processing.

5. News Dataset:

   This dataset is downloaded from an opensource platform called Kaggle[6]. This is a public dataset and can directly be downloaded into csv format.

3 https://finance.yahoo.com/quote/%5EDJI?p=^DJI

4 https://finance.yahoo.com/quote/GC=F?p=GC=F

5 https://finance.yahoo.com/quote/CL=F?p=CL=F

6 https://www.kaggle.com/aaron7sun/stocknews

4 Importing python libraries

Below snippet shows the python code of importing all the required libraries for this research.

In [1]:
```
#importing Libraries

%matplotlib inline
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sb
import re
from statsmodels.tsa.stattools import grangercausalitytests
from sklearn.utils import resample
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
from functools import reduce
from statsmodels.tsa.api import VAR
from statsmodels.tsa.stattools import adfuller
from statsmodels.tools.eval_measures import rmse, aic
from sklearn.metrics import mean_absolute_error,mean_squared_error
import math
```

5 Importing the datasets and Checking for missing values

All the datasets were imported into python environment and checks for NA values in the datasets were performed. Below image shows the code snippet for importing the data and checking for NA values in data.

```
In [2]: # Reading stock market dataset into a dataframe
        df = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/DJIA.xlsx', index_col = 0)
        print("Overview of DJIA stocks Dataset")
        print(df.info())

        # Checking for NA values in stocks data
        print("NA values in DJIA stocks Dataset are: ")
        print(df.isna().sum())

        # Reading news dataset into a dataframe
        dfnews = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/News_sentiments.xlsx', index_col = 0)
        print("Overview of news Dataset")
        print(dfnews.info())

        # Checking for NA values in news data
        print("NA values in news Dataset are: ")
        print(dfnews.isna().sum())

        # import twitter data file
        twitter_data = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/Twitter_sentiments.xlsx')
        print("Overview of Twitter Dataset")
        print(twitter_data.info())

        # Checking for NA values in Twitter data
        print("NA values in Twitter Dataset are: ")
        print(twitter_data.isna().sum())

        # import Crude oil data file
        dfCrude = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/Crudeoil.xlsx', index_col = 0)
        print("Overview of Crude oil Dataset")
        print(dfCrude.info())

        # Checking for NA values in Crude oil data
        print("NA values in Crude oil Dataset are: ")
        print(dfCrude.isna().sum())

        # import Gold data file
        dfGold = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/Gold.xlsx', index_col = 0)
        print("Overview of Gold Dataset")
        print(dfGold.info())

        # Checking for NA values in gold data
        print("NA values in Gold Dataset are: ")
        print(dfGold.isna().sum())
```

6 Data Pre-Processing

At different levels of study datasets were combined as per the requirement. Also, positive ratio was calculated on Twitter sentiments data. Code snippets for the same are given below:

```
#Grouping -ve and +ve sentiments in twitter data by date

dfDate = pd.DataFrame()
dfDate['Sentiments_Count'] = None
dfDate['Sentiments_Count'] = twitter_data.groupby(['Date', 'Sentiments'])['Sentiments'].count()
dfDate = dfDate.reset_index()
neg = dfDate[dfDate['Sentiments']=='Negative']
pos = dfDate[dfDate['Sentiments']=='Positive']
negcount = neg.groupby('Date').sum().reset_index()
poscount = pos.groupby('Date').sum().reset_index()

# Counting total number of positive and negative sentiments on each date
allcount = negcount.merge(poscount,on=['Date'],how='outer')
allcount = allcount.set_index('Date')
allcount.fillna(0,inplace=True)
allcount = allcount.rename(columns={'Sentiments_Count_x':'Negative','Sentiments_Count_y':'Positive'})
print(allcount.isna().sum())
print(allcount.head())
```

```
# Calculating Positive ratio for tweets and storing it into a CSV
allcount['Ratio'] = allcount['Positive']/(allcount['Positive']+allcount['Negative'])
allcount['Ratio'] = allcount['Ratio'].round(2)
allcount = allcount.sort_index()
allcount = allcount.reset_index()
#allcount.to_csv('C:/Users/yash8/Desktop/Research project/Datasets/PosRatioTweets.csv')#



# Defining Sentiment for each date depending on maximum sentimens count for corresponding date

'''Eg: If there are 3 positive tweets and 1 negative tweet on a particular date, than the sentiment assigned to that date
   will be positive'''

for i in range(0,len(allcount)):
    if(allcount.loc[i,'Positive']>allcount.loc[i,'Negative']):
        allcount.loc[i,'sentiment'] = 1
    else:
        allcount.loc[i,'sentiment'] = 0

allcount = allcount.set_index('Date')

print(allcount.groupby('sentiment').count())

allcount = allcount.reset_index()
```

```
# Reading Stockmarket Data into a dataframe
df = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/DJIA.xlsx', index_col = 0)
dfGC = pd.DataFrame()
dfGC['StockPrice'] = df.Close
dfg = dfGC.reset_index()


# Defining the trend (0 = uptrend, 1 = downtrend)
for i in range(0, len(dfg)-1):
    value = dfg.loc[i+1, 'StockPrice'] - dfg.loc[i, 'StockPrice']
    if(value > 1):
        dfg.loc[i+1,'trend'] = 0
    else:
        dfg.loc[i+1,'trend'] = 1



# Merging Dataframes on basis of Date
merged = allcount.merge(dfg,on=['Date'],how='outer')
merged.drop(merged[merged['trend'].isna() == True].index, inplace = True)
merged['Date'] = merged['Date'].values.astype(float)
merged.reset_index(drop=True)
```

7 Exploratory Data Analysis (EDA):

Various steps were carried out for EDA of datasets, code snippets for the same are given below:

```
# Time series plots for closing values of DJIA, Gold and Crude oil
plt.figure(figsize = (6,6))
plt.plot(df.Close)
plt.title('Closing price of DJIA stocks')
plt.ylabel('Closing price ($)')
plt.xlabel('Trading Day')
plt.grid(False)
plt.show()

plt.figure(figsize = (6,6))
plt.plot(dfCrude.Close)
plt.title('Closing price of Crude oil')
plt.ylabel('Closing price ($)')
plt.xlabel('Trading Day')
plt.grid(False)
plt.show()

plt.figure(figsize = (6,6))
plt.plot(dfGold.Close)
plt.title('Closing price of Gold')
plt.ylabel('Closing price ($)')
plt.xlabel('Trading Day')
plt.grid(False)
plt.show()
```

```
# Checking for outliers in DJIA closing data
df['Close'].plot.box()
```

```
# Checking for outliers in Crudeoil closing prices
dfCrude['Close'].plot.box()
```

```
# Checking for outliers in Gold closing prices
dfGold['Close'].plot.box()
```

8 Code snippets for Granger Causality test

Calculating Granger Causality of Twitter

```
#importing stock market and Twitter positive ratio datasets
dfPosRatio = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/stocks_Posratio.xlsx', index_col = 0)

#Calculating Causality
grangercausalitytests(dfPosRatio[['Twitter_pos_ratio','Close']], maxlag=3)
```

Calculating Granger Causality of News Sentiments

```
# Granger Causality test on news sentiments data
dfGranger = pd.DataFrame()
dfGranger['StockPrice'] = df.Close
dfGranger['Sentiments'] = dfnews.Label
dfGranger['Sentiments'] = dfGranger['Sentiments'].ffill()
grangercausalitytests(dfGranger[['Sentiments','StockPrice']], maxlag=3)
```

Calculating Granger Causality of Crude oil

```
# Granger Causality test on crude oil
dfgranger1 = pd.DataFrame()
dfgranger1['StockPrice'] = df.Close
dfgranger1['CrudeOil'] = dfCrude.Close
dfgranger1['CrudeOil'] = dfgranger1['CrudeOil'].ffill()
grangercausalitytests(dfgranger1[['CrudeOil','StockPrice']], maxlag=3)
```

Calculating Granger Causality of Gold prices

```
# Granger Causality test on gold
dfgranger2 = pd.DataFrame()
dfgranger2['StockPrice'] = df.Close
dfgranger2['Gold'] = dfGold.Close
dfgranger2['Gold'] = dfgranger2['Gold'].ffill()
grangercausalitytests(dfgranger2[['Gold','StockPrice']], maxlag=3)
```

9 Machine Learning Models

Code snippets for machine learning models implemented in this research are given below:

1. Vector Auto Regression (VAR):

```
# Fitting VAR model
nobs = 600
df_train, df_test = data_csv[0:-nobs], data_csv[-nobs:]
model = VAR(data_csv)
for i in [1,2,3,4,5,6,7,8,9]:
    result = model.fit(i)
    print('Lag Order =', i)
    print('AIC : ', result.aic)
    print('BIC : ', result.bic)
    print('FPE : ', result.fpe)
    print('HQIC: ', result.hqic, '\n')
x = model.select_order(maxlags=12)
x.summary()
model_fitted = model.fit(7)
model_fitted.summary()
forecast_input = df_train.values
fc = model_fitted.forecast(y=forecast_input, steps=nobs)
df_forecast = pd.DataFrame(fc, index=data_csv.index[-nobs:], columns=data_csv.columns + '_2d')
df_forecast

# Calculating MAPE
MAPE = np.mean(np.abs((df_test['Close']) - (df_forecast['Close_2d'])) / np.abs(df_test['Close']))*100
print('The Mean Absolute Percentage Error is {:.2f}%'.format(MAPE))
```

```
# Plotting Actual vs Forecast plot
df_test['predicted_close'] = df_forecast['Close_2d']
plt.plot(df_test['Close'], label='Actual Close')
plt.plot(df_test['predicted_close'], label='Predicted Close')
plt.legend()
```

2. K-Nearest Neighbour (kNN):

```
# Splitting the data into train and test sets
X = merged[['StockPrice','sentiment']]
Y = merged['trend']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)

# import KNeighborsClassifier
neighbors = np.arange(1,9)

for i,k in enumerate(neighbors):
    # Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=2)

    # Fit the model
    knn.fit(X_train, y_train)

    # Prediction
    pred_knn = knn.predict(X_test)


# Evaluating kNN model
evaluate_model(knn, X_test, y_test)

# Accuracy
accuracy = accuracy_score(y_test,pred_knn)
print("Accuracy is")
print(accuracy)
```

3. Support Vector Machine (SVM):

```
# Training and Testing SVM model on twitter sentiments
SVM = SVC(C=1.0, kernel='linear')
merged.drop(merged[merged['sentiment'].isna() == True].index, inplace = True)
merged.reset_index(drop=True)
X = merged[['StockPrice','sentiment']]
Y = merged['trend']


# Splitting the data into test and train
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)


# Fitting SVM model and making predictions
SVM.fit(X_train, y_train)
y_pred = SVM.predict(X_test)


# Function for classification matrix
def evaluate_model(model, testX, testY):
    temp = model.predict(testX)
    y_true = testY
    precision = precision_score(y_true, temp, average='weighted')
    recall = recall_score(y_true, temp, average='weighted')
    F1_score = f1_score(y_true, temp, average='weighted')
    print("Precision: " + str(precision) + "\n")
    print("Recall: " + str(recall) + "\n")
    print("f1 score: " + str(F1_score) + "\n")
    return precision, recall, f1_score


# Classification matrix for SVM model on Twitter data
evaluate_model(SVM, X_test, y_test)

# Accuracy
accuracy = accuracy_score(y_test,y_pred)
print("Accuracy is")
print(accuracy)
```

10 Transfer Learning

Transfer learning process was carried out under the same environment as above models. Code snippets for transfer learning process for all the three models are given below:

1. SVM model:

```
# importing news dataset into a dataframe
dfnews = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/News_sentiments.xlsx', index_col = 0)


# Creating a dataframe with stock price close column and news sentiments label column
dfGranger = pd.DataFrame()
dfGranger['StockPrice'] = df.Close
dfGranger['Sentiments'] = dfnews.Label
dfGranger['Sentiments'] = dfGranger['Sentiments'].ffill()
dfg1 = dfGranger.reset_index()


# Defining the trend (0 = uptrend, 1 = downtrend)
for i in range(0, len(dfg1)-1):
    value1 = dfg1.loc[i+1, 'StockPrice'] - dfg1.loc[i, 'StockPrice']
    if(value1 > 1):
        dfg1.loc[i+1,'trend'] = 1
    else:
        dfg1.loc[i+1,'trend'] = 0

dfg1['trend'][0] = 0


# Defining test variables
X_test1 = dfg1[['StockPrice','Sentiments']]
y_test1 = dfg1['trend']


# Making predictions using pre-trained SVM model
y_pred1 = SVM.predict(X_test1)


# Classification matrix
evaluate_model(SVM, X_test1, y_test1)


# Accuracy
accuracy = accuracy_score(y_test1,y_pred1)
print("Accuracy is")
print(accuracy)
```

2. KNN

```
#Fit the model
knn.fit(X_train, y_train)

# Prediction
pred_knn1 = knn.predict(X_test1)

# Evaluating pre-trained kNN model on new dataset
evaluate_model(knn, X_test1, y_test1)

# Accuracy
accuracy = accuracy_score(y_test1,pred_knn1)
print("Accuracy is")
print(accuracy)
```

3. VAR

Testing VAR model with News sentiments

```
# Reading the Tweets Positive ratio Dataset
data_csv1 = pd.read_excel(r'C:/Users/yash8/Desktop/Research project/Datasets/News_stocks.xlsx')
data_csv1 = data_csv1.set_index('Date')
data_csv1.info()
```

```
# Fitting VAR model
nobs = 600
df_train1, df_test1 = data_csv1[0:-nobs], data_csv1[-nobs:]

model1 = VAR(data_csv1)
for i in [1,2,3,4,5,6,7,8,9]:
    result = model.fit(i)
    print('Lag Order =', i)
    print('AIC : ', result.aic)
    print('BIC : ', result.bic)
    print('FPE : ', result.fpe)
    print('HQIC: ', result.hqic, '\n')
x1 = model1.select_order(maxlags=12)
x1.summary()
model_fitted1 = model1.fit(7)
model_fitted1.summary()
forecast_input1 = df_train1.values
fc1 = model_fitted1.forecast(y=forecast_input1, steps=nobs)
df_forecast1 = pd.DataFrame(fc1, index=data_csv1.index[-nobs:], columns=data_csv1.columns + '_2d')
df_forecast1

# Calculating MAPE
MAPE = np.mean(np.abs((df_test1['Close']) - (df_forecast1['Close_2d'])) / np.abs(df_test1['Close']))*100
print('The Mean Absolute Percentage Error is {:.2f}%'.format(MAPE))
```