National College of Ireland

Impact of Crude Oil on Indian Economy Configuration Manual

MSc Research Project
Data Analytics

Ronit Kumar Pareta
Student ID: x21223645

School of Computing
National College of Ireland

Supervisor: Dr Ahmed Makki

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ronit Kumar Pareta		
Student ID:	X21223645		
Programme:	Data Analytics	**Year:**	2022-2023
Module:	MSc Research Project		
Supervisor:	Dr Ahmed Makki		
Submission Due Date:	14-Aug-23		
Project Title:	Impact of Crude Oil Price on Indian Economy Configuration Manual		
Word Count:	**7 Page 731**		

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: Ronit Kumar Pareta

Date: 18-Aug-23

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):	

Impact of Crude Oil Price on Indian Economy Configuration Manual

Ronit Kumar Pareta
X21223645

1 Introduction

This configuration manual provides detailed guidelines for replicating the experimental framework of the research study. It outlines the system requirements, software packages, and machine configuration needed to reconstruct the predictive modeling pipeline. The instructions cover the minimum essential setup to run the models as well as the optional tools to extend the analysis. The steps aim to equip readers with the technical knowledge to reproduce the modeling approach, evaluation techniques, and results analysis. The manual serves as a practical supplement to the research methodology and implementation. Following these configurations will facilitate seamless rebuilding of the computational environment underpinning the study. It enables technical verification and extension of the data science experiments performed.
In summary, this manual aims to specify the technical toolbox and system specifications required to reproduce the machine learning experiments and results described in the thesis. The step-by-step instructions empower readers to recreate the modeling environment and evaluation processes.

2 Project Files

For my thesis, I used jupyter notebook as IDE to carry out the detailed analysis.

Experiment 1:

- I have uploaded the dataset related to crude oil and used yahoo finance api to fetch the data for BSE stock index. Then Data cleaning, Data transformation, EDA, Modelling, Evaluation and Result are done.

Experiment 2:

- I have uploaded the dataset related to crude oil and GDP. Then Data cleaning, Data transformation, EDA, Modelling, Evaluation and Result are done.

Then same process is followed to carry out the experiment 3 and 4.

3 System Specification

The below image of device depicted is a system running Windows 10 Home Single Language operating system on an Intel Core i5-8265U CPU with a clock speed of 1.60GHz and 1.80

GHz turbo frequency. The system has 8GB of installed RAM with 7.88GB usable. To carry out the research this is optimal requirement, and it works well on this configuration.

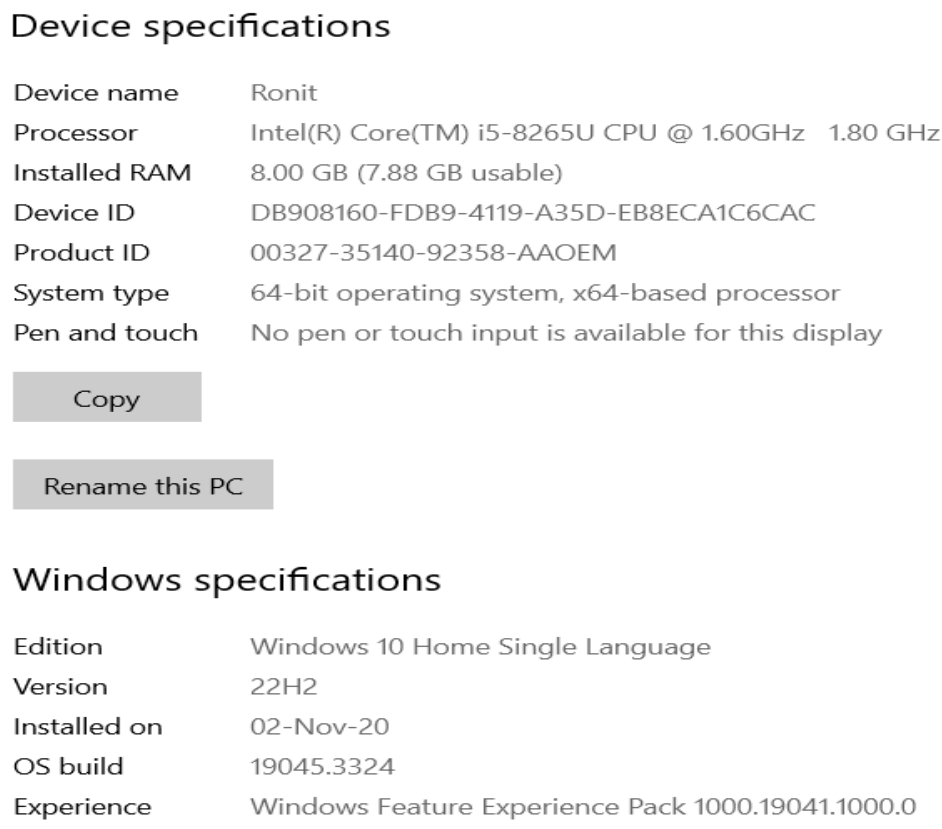

Figure 1 System configuration

4 Software and Tools

To develop the outputs and the outcomes, the following prerequisites for the software and libraries have been used:

• Programming Language - Python
• IDE – Jupyter Notebook
• Python Libraries/Modules:

Pre-processing- Numpy, Pandas
Feature Engineering - matplotlib, Plotly
Modelling and Evaluation - Sklearn, tensorflow, GPY

5 Download and Install

Jupyter Notebook, bundled with the Anaconda Python distribution, provides a common user-friendly platform for development. The Anaconda dashboard features pre-installed packages including Jupyter Notebook, as shown in Figure 2. To initiate Python code development in Jupyter Notebook, first launch it from the Anaconda dashboard, then create a new Python file. This streamlined process leverages the tools provided in the Anaconda distribution to facilitate seamless Jupyter Notebook-based coding.

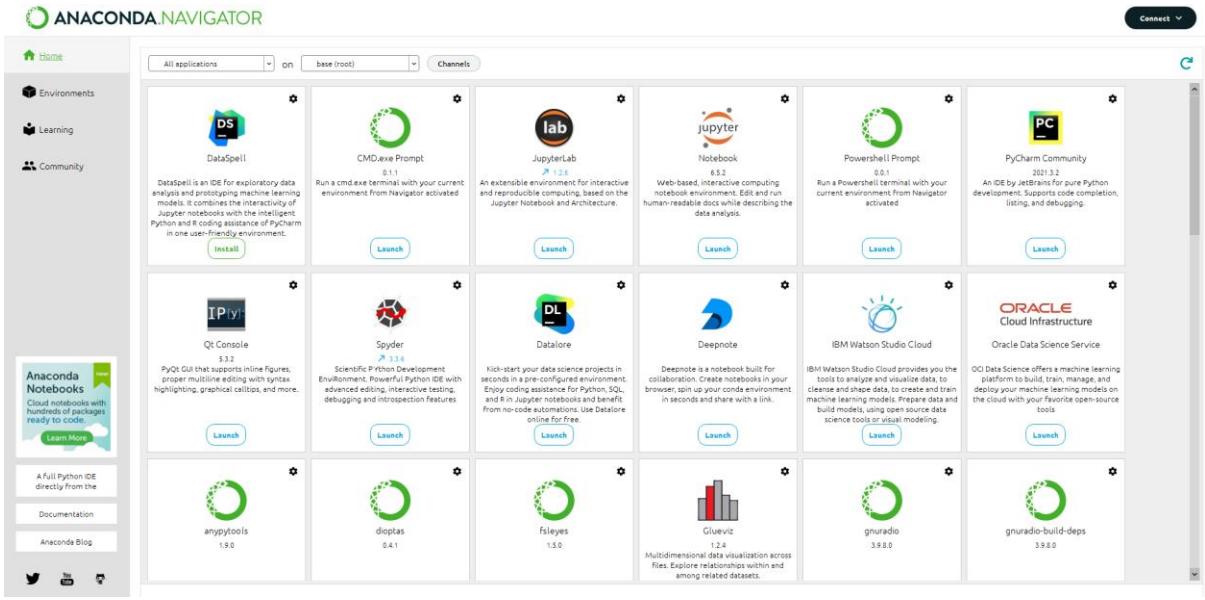

Figure 2 Anaconda Dashboard

6 Project Setup

Once Jupyter Notebook is launched from Anaconda, click new to open a Python script file. The code cells can be run together or individually via runtime options. To install any required packages, use the pip install package-name command. This allows loading the provided code files in Jupyter and executing the cells in a flexible manner after configuring the environment with necessary packages. The streamlined process enables running the implemented scripts in Jupyter Notebook for replication and extension.

7 Importing Libraries

I have carried out the experiment 1,2, and 3 with same libraries and experiment 4 with different setup because it requires only one machine learning modules while others have 4 models

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import seaborn as sns
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import GPy
from sklearn.preprocessing import PolynomialFeatures
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
```

Figure 3 Experiment 4 imports.

```
import pandas as pd
import matplotlib.pyplot as plt
# Import train_test_split
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.pipeline import Pipeline
import plotly.graph_objects as go
import seaborn as sns
from tabulate import tabulate
```

Figure 4 Experiment 1,2, and 3 imports.

8 Importing Files

```
df_inflation = pd.read_csv('India_Inflation_Rate.csv')
```

```
df_inflation.head()
```

	DATE	India_Inflation_Rate
0	01-01-00	2.619048
1	01-02-00	3.614458
2	01-03-00	4.830918
3	01-04-00	5.542169
4	01-05-00	5.011933

Figure 5 Inflation data

```
df_crude = pd.read_csv('DubaiCrudeOil.csv')
```

Figure 6 Crude oil file

```
import yfinance as yf

bse_sensex = yf.Ticker("^BSESN")
data = bse_sensex.history(start="2000-01-01", end="2021-12-31")
print(data)
```

```
                                 Open          High           Low  \
Date
2000-01-03 00:00:00+05:30   5209.540039   5384.660156   5209.540039
2000-01-04 00:00:00+05:30   5533.979980   5533.979980   5376.430176
2000-01-05 00:00:00+05:30   5265.089844   5464.350098   5184.479980
2000-01-06 00:00:00+05:30   5424.209961   5489.859863   5391.330078
2000-01-07 00:00:00+05:30   5358.279785   5463.250000   5330.580078
```

Figure 7 BSE Index data

```
df_gdp = pd.read_csv('Gdp_monthly.csv')
```

```
df_gdp.head()
```

	DATE	India_Gdp
0	01-01-2000	101.507990
1	01-02-2000	101.477515
2	01-03-2000	101.421633
3	01-04-2000	101.342571
4	01-05-2000	101.244041

Figure 8 GDP

```
df_unemp = pd.read_csv('india-unemployment-rate.csv')
df_unemp.head()
```

	Date	Unemployment Rate (%)	Annual Change	
0	31-12-2000	5.561	-0.18	NaN
1	31-12-2001	5.576	0.01	NaN
2	31-12-2002	5.530	-0.05	NaN
3	31-12-2003	5.643	0.11	NaN
4	31-12-2004	5.629	-0.01	NaN

Figure 9 Unemployment

After fetching the data, further analysis is carried out.

9 Implementation Process

- Null values and missing values are check in the data sets and then treated accordingly.
- Renaming the columns for better understanding.
- Checking outliers and removing them if they are not justifiable.
- Normalising the data for better training of machine learning models.
- Feature selection, selecting the only important variables.
- Merging the relevant datasets.

10 Modeling

First, we split the data set into training and testing data. Then the model is trained on training data and evaluated on test data.

```
# Create MinMaxScaler objects
scaler_stock = MinMaxScaler()
scaler_crude_oil = MinMaxScaler()

# Normalize the columns and update the DataFrame
merged_df['India_Inflation_Rate'] = scaler_stock.fit_transform(merged_df['India_Inflation_Rate'].values.reshape(-1, 1))
merged_df['Crude_oil_price'] = scaler_crude_oil.fit_transform(merged_df['Crude_oil_price'].values.reshape(-1, 1))
```

```
merged_df.head()
```

	DATE	Crude_oil_price	India_Inflation_Rate
0	2000-01-01	0.052073	0.101500
1	2000-01-02	0.064157	0.167276
2	2000-01-03	0.067505	0.247660
3	2000-01-04	0.043093	0.294659
4	2000-01-05	0.072275	0.259622

```
x = merged_df['Crude_oil_price'] # Features
y = merged_df['India_Inflation_Rate']  # Target
```

```
x.count(), y.count()
```

```
(276, 276)
```

```
# Splitting the dataset into training and testing set (80/20)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 30)
```

Figure 10 Feature selection and Splitting Data Process

After this process, we have trained the model using this data with hyperparameter tuning.

```
# Expanded hyperparameter grids
rf_param_grid = {
    'n_estimators': [100, 500, 1000],
    'max_depth': [5, 10, 15, 20, 25],
    'max_features': ['auto', 'sqrt', 'log2'],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

gb_param_grid = {
    'n_estimators': [100, 500],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 5, 8, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'loss': ['squared_error', 'absolute_error', 'huber']
}

# Randomized grid search
rf_random = RandomizedSearchCV(estimator=RandomForestRegressor(),
                               param_distributions=rf_param_grid,
                               n_iter=10, cv=5, verbose=2, random_state=42)

gb_random = RandomizedSearchCV(estimator=GradientBoostingRegressor(),
                               param_distributions=gb_param_grid,
                               n_iter=10, cv=5, verbose=2, random_state=42)


rf_random.fit(X_train, y_train)
gb_random.fit(X_train, y_train)

# Best models
best_rf_model = rf_random.best_estimator_
best_gb_model = gb_random.best_estimator_
```

Figure 11 Random forest and Gradient boosting model

```
# Create a linear regression model
linear_model = LinearRegression()

# Create a pipeline with a StandardScaler and a linear model
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('linear_model', linear_model)
])

# Define the hyperparameters and their potential values
param_grid = {
    'linear_model__fit_intercept': [True, False]
}

# Create a GridSearchCV object
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit the grid search to your data
grid_search.fit(X_train, y_train)

# Get the best hyperparameters and the best model
best_params = grid_search.best_params_
best_model_lr = grid_search.best_estimator_

print("Best Hyperparameters:", best_params)
print("Best Model:", best_model_lr)
```

```
Best Hyperparameters: {'linear_model__fit_intercept': True}
Best Model: Pipeline(steps=[('scaler', StandardScaler()),
                ('linear_model', LinearRegression())])
```

Figure 12 Linear Regression

```
nn_model = Sequential([
    Dense(64, activation='relu', input_dim=X_train.shape[1]),
    Dense(32, activation='relu'),
    Dense(1)
])

nn_model.compile(optimizer='adam', loss='mean_squared_error')
nn_model.fit(X_train, y_train, epochs=100, batch_size=32)
```

Figure 13 Neural Network model

```
#Create and train the Gaussian Process Regression model
kernel = GPy.kern.RBF(input_dim=1)  # Radial basis function (RBF) kernel
model = GPy.models.GPRegression(X_crude, Y_unemp, kernel)
model.optimize()  # Optimize the model hyperparameters
```

```
<paramz.optimization.optimization.opt_lbfgsb at 0x20d0f3b2ef0>
```

Figure 14 Gaussian Process Regression model