National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Rian Dwi Putra
Student ID: 22108637

School of Computing
National College of Ireland

Supervisor:
Rejwanul Haque
&
John Kelly

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rian Dwi Putra
Student ID:	22108637
Programme:	MSc in Data Analytics **Year:** 2023
Module:	Research Project
Lecturer:	Rejwanul Haque & John Kelly
Submission Due Date:	August 14 2023
Project Title:	Forecasting Sales and Inventory in Supply Chain using Machine Learning Methods
Word Count:	1149 **Page Count:** 9

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: Rian Dwi Putra

Date: August 14 2023

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

Rian Dwi Putra

22108637

1 Introduction

You will gain a better understanding of the project's system requirements from this configuration manual. The research project's hardware/software platforms, data sources, code, figures, and files are all described in this configuration manual. Implementing the research project “Forecasting Sales and Inventory in Supply Chain using Machine Learning Methods” will be made easier with this manual.

2 System Requirements

The following minimum system requirements are suggested for this data analytics projects:

1. Working Framework: Windows 10, macOS, or Linux (Ubuntu).
2. Processor: Intel Core i5 or AMD Ryzen 5 (or equivalent) for basic machine learning tasks.
3. RAM: 8 GB or greater RAM is beneficial for handling bigger datasets and complex models.
4. Storage: SSD (Solid State Drive) with 256 GB for quicker access to data and model training.
5. Graphics Processing Unit (GPU): A GPU (NVIDIA GTX or RTX series) can significantly accelerate model training for advanced projects, even though it is not necessary for basic projects.
6. Python and Anaconda: Install Python 3.7 or later which comes with essential libraries like NumPy, Pandas, and scikit-learn, needs to be installed alongside Python 3.7 or later.
7. Jupyter Notebook: For intelligent coding and visualization.
8. Integrated Development Environment (IDE): Visual Studio Code, PyCharm, or Jupyter Lab are IDEs that are recommended.

9. Internet connection: To get to online resources, datasets, and machine learning libraries.

3 Project Development

3.1 Importing Libraries

Importing libraries in Python means adding external code modules or packages to the current Python program to make it more useful. Python offers a vast ecosystem of libraries and bundles that give pre-composed code to different tasks, for example, data manipulation, visualization, machine learning, and more. At the point when importing a library, we will get access to its capabilities, classes, and variables, enable it to involve in the code. This empowers programmer to use existing code instead of composing all that from the scratch, saving time and effort.

Importing all required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import xgboost as xgb
import lightgbm as lgb
import datetime as dt
import calendar,warnings,itertools,matplotlib,keras,shutil
import tensorflow as tf
import statsmodels.api as sm
from datetime import datetime
from sklearn.model_selection import train_test_split,cross_val_score, cross_val_predict
from sklearn import svm,metrics,tree,preprocessing,linear_model
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import Ridge,LinearRegression,ElasticNet, Lasso
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import accuracy_score,mean_squared_error,recall_score,confusion_matrix,f1_score,roc_curve, auc, r2_score
from sklearn.datasets import load_iris,make_regression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.kernel_ridge import KernelRidge
from keras import Sequential
from keras.layers import Dense
from IPython.core import display as ICD
# from tensorflow_core.estimator import inputs

#Hiding the warnings
warnings.filterwarnings('ignore')
```

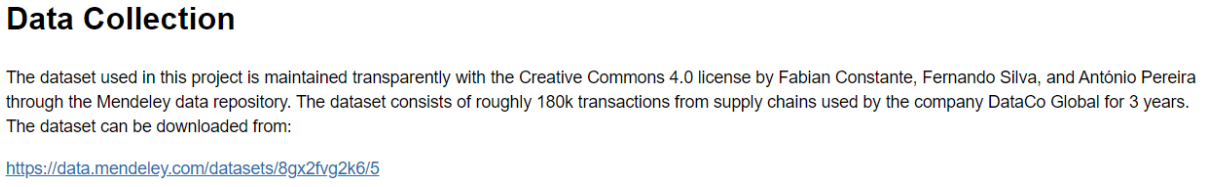
Figure 1. Importing required libraries

Generally, bringing in libraries in Python upgrades the programming capacities to access and use existing code, making Python a flexible and strong language for different tasks and domains.

3.2 Dataset

The method for gathering the project's data will be discussed in this phase. Describe the process by which the data will be gathered, transformed, cleaned, and prepared for analysis.

This study will utilize DataCo Smart Supply Chain dataset, which can be downloaded from https://data.mendeley.com/datasets/8gx2fvg2k6/5.



Data Collection

The dataset used in this project is maintained transparently with the Creative Commons 4.0 license by Fabian Constante, Fernando Silva, and António Pereira through the Mendeley data repository. The dataset consists of roughly 180k transactions from supply chains used by the company DataCo Global for 3 years. The dataset can be downloaded from:

https://data.mendeley.com/datasets/8gx2fvg2k6/5

Figure 2. Dataset definition

3.2.1 Importing Dataset

An organized collection of data that captures relevant information about the flows of goods, services, and materials throughout the supply chain network is captured in this Supply Chain dataset. Typically, data on pricing, customer-related metrics, product details, sales and demand data, inventory levels, supplier data, transportation and logistics data, and pricing data are included.

```
#Importing Dataset using pandas
dataset=pd.read_csv("DataCoSupplyChainDataset.csv",header= 0,encoding= 'unicode_escape')
dataset.head(5)# Checking 5 rows in dataset
```

	Type	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	Category Id	Category Name	Customer City	...	Order Zipcode	Product Card Id	Product Category Id	Des
0	DEBIT	3	4	91.250000	314.640015	Advance shipping	0	73	Sporting Goods	Caguas	...	NaN	1360	73	
1	TRANSFER	5	4	-249.089996	311.359985	Late delivery	1	73	Sporting Goods	Caguas	...	NaN	1360	73	
2	CASH	4	4	-247.779999	309.720001	Shipping on time	0	73	Sporting Goods	San Jose	...	NaN	1360	73	
3	DEBIT	3	4	22.860001	304.809998	Advance shipping	0	73	Sporting Goods	Los Angeles	...	NaN	1360	73	
4	PAYMENT	2	4	134.210007	298.250000	Advance shipping	0	73	Sporting Goods	Caguas	...	NaN	1360	73	

5 rows × 53 columns

Figure 3. Importing dataset

3.2.2 Data Cleaning & Preprocessing

- Data cleaning in Python refers to the most common way of distinguishing and correcting mistakes, irregularities, and errors in a dataset to guarantee its quality and reliability for analysis and modelling. Data cleaning is a significant stage in the data preprocessing pipeline, as raw data frequently contains missing data, outliers, duplicate entries, and different inconsistencies that can prompt bias or wrong outcomes whenever left neglected.

Data Cleaning

```
dataset.shape
```

```
(180519, 53)
```

The total data set comprises of 180519 records and 53 columns

```
dataset.apply(lambda x: sum(x.isnull())) #Checking missing values
```

Figure 4. Data definition

- Taking care of missing data in Python refers to the most common way of recognizing and overseeing data that are not accessible or incomplete in a dataset. Missing data can happen because of different reasons,such as data collection errors, system issues, or participant non-response. Managing missing data is fundamental to guarantee the exactness and reliable quality of data analysis.

The data comprises of a few missing values from Customer Lname, Product Description, Order Zipcode and, Customer Zipcode which should be taken out or removed prior to continuing with the analysis. And furthermore, since there is an opportunity various customers could have a similar first name or same last name another column with 'Customer Full Name' is created to avoid from any ambiguities.

```
# Adding first name and last name together to create new column
dataset['Customer Full Name'] = dataset['Customer Fname'].astype(str)+dataset['Customer Lname'].astype(str)
```

To make it easier for analysis some unimportant columns will be dropped

```
data=dataset.drop(['Customer Email','Product Status','Customer Password','Customer Street','Customer Fname','Customer Lname',
           'Latitude','Longitude','Product Description','Product Image','Order Zipcode','shipping date (DateOrders)'],axis=1)
data.shape
```

```
(180519, 42)
```

There are 3 missing values in Customer Zipcode column. Before proceeding with the analysis of the data, the values that are missing are simply zip codes, which are not very important. These values are replaced with zero.

```
data['Customer Zipcode']=data['Customer Zipcode'].fillna(0)#Filling NaN columns with zero
```

Figure 5. Handling missing data

- Data modeling in Python refers to the most common way of making numerical or statistical representations of a dataset to make predictions, gain insight, or solve of explicit issues. It involves building models that identify relationships between input features and target variables by utilizing a variety of statistical and machine learning methods.

Data Modelling

To measure the performance of different models the machine learning models are trained to predict sales, order quantity is predicted for regression type models.

A new dataset is created with the copy of original data for training the data and validation.

```
train_data=data.copy()
```

```
#Dropping columns with repeated values
train_data.drop(['Delivery Status','Late_delivery_risk','Order Status','order date (DateOrders)'], axis=1, inplace=True)
```

There are some columns with object type data which cannot be trained in machine learning models so all the object type data is converted to int type using preprocessing label encoder library.

```
# create the Labelencoder object
le = preprocessing.LabelEncoder()
#convert the categorical columns into numeric

train_data['Customer Country']  = le.fit_transform(train_data['Customer Country'])
train_data['Market']            = le.fit_transform(train_data['Market'])
train_data['Type']              = le.fit_transform(train_data['Type'])
train_data['Product Name']      = le.fit_transform(train_data['Product Name'])
train_data['Customer Segment']  = le.fit_transform(train_data['Customer Segment'])
train_data['Customer State']    = le.fit_transform(train_data['Customer State'])
train_data['Order Region']      = le.fit_transform(train_data['Order Region'])
train_data['Order City']        = le.fit_transform(train_data['Order City'])
train_data['Category Name']     = le.fit_transform(train_data['Category Name'])
train_data['Customer City']     = le.fit_transform(train_data['Customer City'])
train_data['Department Name']   = le.fit_transform(train_data['Department Name'])
train_data['Order State']       = le.fit_transform(train_data['Order State'])
train_data['Shipping Mode']     = le.fit_transform(train_data['Shipping Mode'])
train_data['Order Country']     = le.fit_transform(train_data['Order Country'])
train_data['Customer Full Name']= le.fit_transform(train_data['Customer Full Name'])

#display the initial records
train_data.head()
```

Figure 6. Data modelling

Presently every each of the data is changed into int type. The dataset is parted into train data and test data so model can be trained with train data and the performance of model can be assessed utilizing test data.

4 Model Building

Model building in Python refers to the most common way of making and preparing prescient or logical models utilizing machine learning, statistical, or other computational methods. Preparing the data, selecting an appropriate algorithm, and iteratively adjusting the model's parameters to improve performance are all part of it.

4.1 Splitting the data into training and test set

Splitting data into training and test sets in Python refers to the most common way of dividing a dataset into two separate subsets: one for training the machine learning model for use and the other for assessing how well it performs. This division is essential for determining the model's generalizability to unknown, new data and avoiding overfitting.

This study uses the split ratio of 70-30, with the larger portion going to training and the smaller portion going to testing. By fitting the model to the data in the training set, the algorithm is able to learn about data patterns and relationships. The test set, then again, is utilized to assess the model's performance by making forecasts on the unseen data and contrasting the predictions with the actual target values.

For comparison of regression models sales and order quantity are predicted

```
xs=train_data.loc[:, train_data.columns != 'Sales']
ys=train_data['Sales']
xs_train, xs_test,ys_train,ys_test = train_test_split(xs,ys,test_size = 0.3, random_state = 42)
xq=train_data.loc[:, train_data.columns != 'Order Item Quantity']
yq=train_data['Order Item Quantity']
xq_train, xq_test,yq_train,yq_test = train_test_split(xq,yq,test_size = 0.3, random_state = 42)
```

Figure 7. Splitting dataset

4.2 Building the regression model

The data is currently fit to be utilized in machine learning models. The result is regression type, so every each of the models are compared by Mean Absolute Error (MAE) and RMSE. The lower the value of MAE & RMSE, the better the model is performing.

```
def regressionmodel(model_s,model_q,xs_train, xs_test,ys_train,ys_test,xq_train, xq_test,yq_train,yq_test):
    model_s=model_s.fit(xs_train,ys_train)#Fitting train data for sales
    model_q=model_q.fit(xq_train,yq_train)#Fitting train data for order quantity
    ys_pred=model_s.predict(xs_test)#predicting sales with test data
    yq_pred=model_q.predict(xq_test)#predicting order quantity with test data

    print('Model parameter used are:',model_s) #Printing the model to see which parameters are used
    #Printing mean absolute error for predicting sales
    print("MAE of sales is         :", metrics.mean_absolute_error(ys_test,ys_pred))
    #Printing Root mean squared error for predicting sales
    print("RMSE of sales is        :",np.sqrt(metrics.mean_squared_error(ys_test,ys_pred)))
    #Printing R2 for predicting sales
    print("R2 of sales is        :",metrics.r2_score(ys_test,ys_pred))

    #Printing mean absolute error for predicting order quantity
    print("MAE of order quantity   :", metrics.mean_absolute_error(yq_test,yq_pred))
    #Printing Root mean squared error for predicting order quantity
    print("RMSE of order quantity  :",np.sqrt(metrics.mean_squared_error(yq_test,yq_pred)))
    #Printing R2 for predicting order quantity
    print("R2 of order quantity   :", metrics.r2_score(yq_test,yq_pred))
```

Figure 8. Building regression model

- Linear Regression

Linear Regression

```
model_s=LinearRegression()
model_q=LinearRegression()
regressionmodel(model_s,model_q,xs_train, xs_test,ys_train,ys_test,xq_train, xq_test,yq_train,yq_test)
```

Figure 9. Linear regression

- Lasso Regression

Lasso Regression

```
model_s = linear_model.Lasso(alpha=0.1)
model_q = linear_model.Lasso(alpha=0.1)
regressionmodel(model_s,model_q,xs_train, xs_test,ys_train,ys_test,xq_train, xq_test,yq_train,yq_test)
```

Figure 10. Lasso Regression

- Ridge Regression

Ridge Regression

```
model_s = Ridge(alpha=1.0)
model_q = Ridge(alpha=1.0)
regressionmodel(model_s,model_q,xs_train, xs_test,ys_train,ys_test,xq_train, xq_test,yq_train,yq_test)
```

Figure 11. Ridge Regression

- Decision Tree Regression

Decision Tree Regression

```
model_s = tree.DecisionTreeRegressor()
model_q = tree.DecisionTreeRegressor()
regressionmodel(model_s,model_q,xs_train, xs_test,ys_train,ys_test,xq_train, xq_test,yq_train,yq_test)
```

Figure 12. Decision Tree Regression

- Random Forest Regression

Random Forest Regression

```
model_s = RandomForestRegressor(n_estimators=100,max_depth=10, random_state=40)
model_q = RandomForestRegressor(n_estimators=100,max_depth=10, random_state=40)
regressionmodel(model_s,model_q,xs_train, xs_test,ys_train,ys_test,xq_train, xq_test,yq_train,yq_test)
```

Figure 13. Random Forest Regression

- Light Gradient Boosting Regression

Light Gradient Boosting Regression

```
model_s = lgb.LGBMRegressor()
model_q = lgb.LGBMRegressor()
regressionmodel(model_s,model_q,xs_train, xs_test,ys_train,ys_test,xq_train, xq_test,yq_train,yq_test)
```

Figure 14. Light Gradient Boosting Regression

- eXtreme Gradient Boosting Regression

eXtreme Gradient Boosting Regression

```
model_s = xgb.XGBRegressor()
model_q = xgb.XGBRegressor()
regressionmodel(model_s,model_q,xs_train, xs_test,ys_train,ys_test,xq_train, xq_test,yq_train,yq_test)
```

Figure 15. eXtreme Gradient Boosting Regression

4.3 Comparison table for Regression Model

In this table, the key performance metrics used to assess the models, for example, R-squared (coefficient of determination), RMSE (Root Mean Squared Error), and MAE (Mean Absolute Error), are recorded. These metrics show how well each model makes predictions and fits the data.

The comparison table guides in the selection of the most appropriate regression model for the requirements of the DataCo Smart Supply Chain project, considering factors like interpretability, performance, and handling of different data characteristics.

a. MAE & RMSE for Sales & Inventory Forecasting

Table 1. Comparison of Regression Model for MAE and RMSE

Comparison Table for Regression Model Scores

```
Regression_comparison #Printing dataframe
```

	Regression Model	MAE Value for Sales	RMSE Value for Sales	MAE Value for Quantity	RMSE Value for Quantity
0	Lasso	1.3300	2.090	1.2500	1.430
1	Ridge	0.2600	0.470	0.3400	0.520
2	Light Gradient Boosting	0.5400	4.360	0.0004	0.004
3	Random Forest	0.2000	1.770	6.6100	0.005
4	eXtreme gradient boosting	0.1500	3.040	0.0001	0.006
5	Decision tree	0.0110	0.799	0.0001	0.010
6	Linear Regression	0.0005	0.001	0.3300	0.520

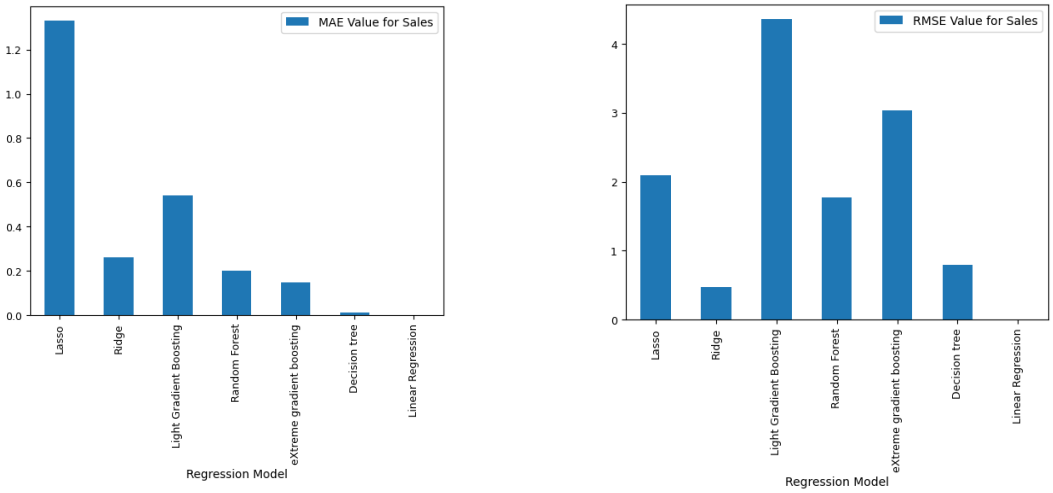


Figure 16. MAE and RMSE for Sales Forecasting

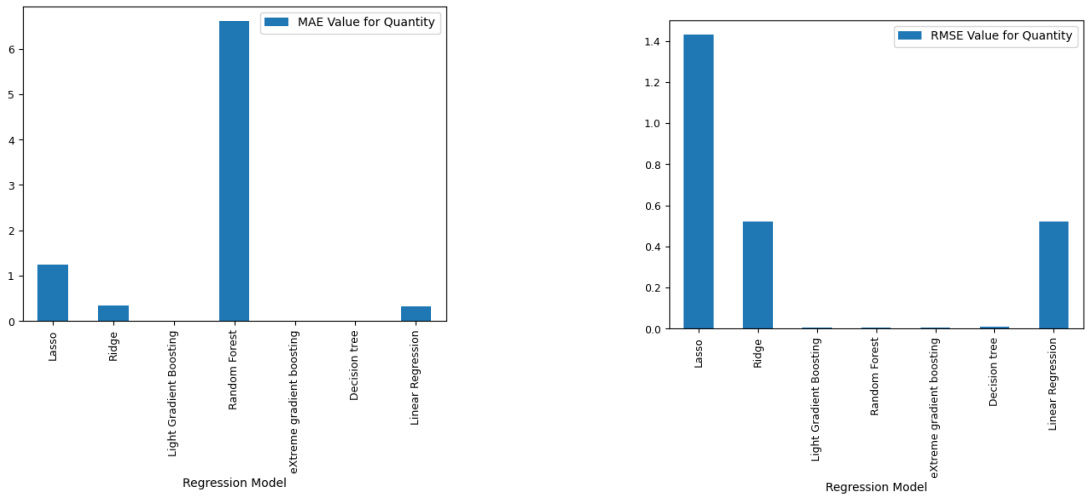


Figure 17. MAE and RMSE for Inventory Forecasting

b. R^2 for Sales & Inventory Forecasting

Table 2. Comparison of Regression Model for MAE and RMSE

	Regression Model	R2 for Sales	R2 for Quantity
0	Lasso	0.999	0.021
1	Ridge	0.999	0.868
2	Light Gradient Boosting	0.998	0.999
3	Random Forest	0.999	0.999
4	eXtreme gradient boosting	0.999	0.999
5	Decision tree	0.999	1.000
6	Linear Regression	0.999	0.869

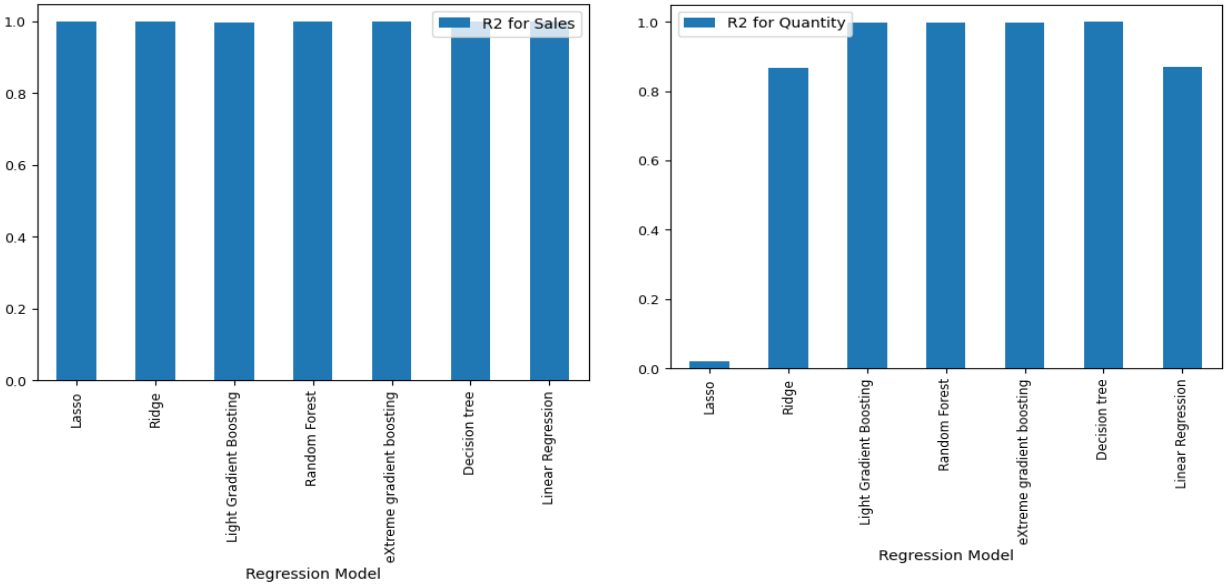


Figure 18. R^2 for Sales & Inventory Forecasting