# Configuration Manual

MSc Research Project
MSC Data Analytics


Muddassir Ahmed
Student ID: x23138688

School of Computing
National College of Ireland


Supervisor: Arjun Chikkankod

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | …….Muddassir Ahmed…………………………………………………………………………… |
| **Student ID:** | ………x23138688…………………………………………………………………………..…… |
| **Programme:** | …… MSC Data Analytics …………………………… **Year:** …2024………………….. |
| **Module:** | ……… MSC Research Project ………………………………………………………….……… |
| **Lecturer:** | ……… Arjun Chikkankod ……………………………………………………………….……… |
| **Submission Due Date:** | …………12 August 2024…………………………………………………………………….……… |
| **Project Title:** | ………… Role of data analytics to overcome challenges in supply chain management …………………………….……… |
| **Word Count:** | ………919……………………… **Page Count:** ………10……………..…….……… |

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:** ……… Muddassir Ahmed……………………………………………

**Date:** …………12-08-2024……………………………………………………

**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

Muddassir Ahmed
Student ID: x23138688

# 1 Introduction

This document is explaining overall requirement for running this project and explaining its steps which is used in this project. I provide the information of the hardware and software which is required to run this project and provides its configuration manual. Steps by step will explain which is including, dataset preparation, exploratory data analysis, model building and results evaluations.

# 2 Hardware and Software Requirements

## 2.1 Hardware Configuration

I have conducted this research on my personal computer. Hardware details of the computer is given below. It's a core i5 processor with 2.0Ghz, 2.60Ghz CPU power and RAM is 16 GB with 64 bit operating systems. Windows 10 pro edition is installed on this computer which has 22H2 version.

Device specifications

| | |
|---|---|
| Device name | DESKTOP-VEQMU5V |
| Processor | Intel(R) Core(TM) i5-4310U CPU @ 2.00GHz 2.60 GHz |
| Installed RAM | 16.0 GB (15.9 GB usable) |
| Device ID | 2ADE9E9B-27B5-4F0B-ADF4-74AAA8370FDD |
| Product ID | 00330-50000-00000-AAOEM |
| System type | 64-bit operating system, x64-based processor |
| Pen and touch | Touch support with 2 touch points |

**Figure 1: Device Specifications**

Windows specifications

| | |
|---|---|
| Edition | Windows 10 Pro |
| Version | 22H2 |
| Installed on | 09/08/2023 |
| OS build | 19045.4651 |
| Experience | Windows Feature Experience Pack 1000.19060.1000.0 |

**Figure 2: Windows Specifications**

## 2.2 Software Configuration

This section tells which software's are required to run this project and what versions are installed in my computer. This software's must be installed before running this project. Code is written in python language which has 3.11.4 version and its package by anaconda. It is run on jupyter notebook that has 6.5.4 version on my computer.

```
!jupyter notebook --version
```

```
6.5.4
```

```
import sys
print(sys.version)
```

```
3.11.4 | packaged by Anaconda, Inc. | (main, Jul  5 2023,
```

**Figure 3: Software Specifications**

# 3 Methodology and Implementation

## 3.1 Dataset Collection and Preparation

Dataset for this research is downloaded from public repository Mendeley data which is showing in below figure 4.

Mendeley Data

DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS

Published: 12 March 2019 | Version 5 | DOI: 10.17632/8gx2fvg2k6.5
Contributors: Fabian Constante, Fernando Silva, António Pereira

Description

A DataSet of Supply Chains used by the company DataCo Global was used for the analysis. Dataset of Supply Chain , which allows the use of Machine Learning Algorithms and R Software.
Areas of important registered activities : Provisioning , Production , Sales , Commercial Distribution.It also allows the correlation of Structured Data with Unstructured Data for knowledge generation.

**Figure 4: Mendeley Dataset**

This dataset consists of three files; those are showing in below figure 5.

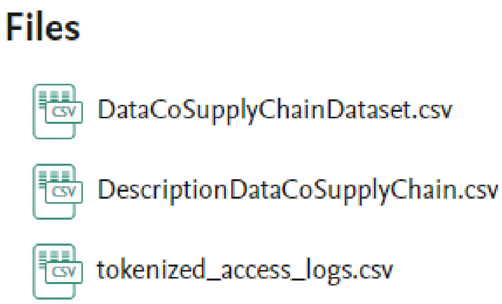


**Figure 5: Dataset Files**

## 3.2 Importing Libraries

We need to installed some important libraries run this dataset. Which is including data importing, data pre-processing, exploratory data analysis, splitting dataset for training and testing purpose, modelling of the data and then evaluations of the result. These libraries are showing in below figure 6.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import datetime as dt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,QuadraticDiscriminantAnalysis
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
from sklearn import model_selection
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import roc_auc_score,r2_score,mean_absolute_error,mean_squared_error,accuracy_score,classification
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.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.ensemble import RandomForestRegressor,RandomForestClassifier, GradientBoostingRegressor
from sklearn.metrics import accuracy_score,mean_squared_error,recall_score,confusion_matrix,f1_score,roc_curve, auc
from plotly.offline import iplot, init_notebook_mode
import pickle
import warnings
warnings.filterwarnings("ignore")
import datetime as dt
from datetime import datetime
import plotly.express as px
```

**Figure 6: Import Libraries**

## 3.3 Handling Missing Values

```
Department Id                     0
Department Name                   0
Latitude                          0
Longitude                         0
Market                            0
Order City                        0
Order Country                     0
Order Customer Id                 0
order date (DateOrders)           0
Order Id                          0
Order Item Cardprod Id            0
Order Item Discount               0
Order Item Discount Rate          0
Order Item Id                     0
Order Item Product Price          0
Order Item Profit Ratio           0
Order Item Quantity               0
Sales                             0
Order Item Total                  0
Order Profit Per Order            0
Order Region                      0
Order State                       0
Order Status                      0
Order Zipcode                155679
Product Card Id                   0
Product Category Id               0
```

```
Department Id                     0
Department Name                   0
Latitude                          0
Longitude                         0
Market                            0
Order City                        0
Order Country                     0
Order Customer Id                 0
order date (DateOrders)           0
Order Id                          0
Order Item Cardprod Id            0
Order Item Discount               0
Order Item Discount Rate          0
Order Item Id                     0
Order Item Product Price          0
Order Item Profit Ratio           0
Order Item Quantity               0
Sales                             0
Order Item Total                  0
Order Profit Per Order            0
Order Region                      0
Order State                       0
Order Status                      0
Order Zipcode                155679
Product Card Id                   0
Product Category Id               0
```

```
Order Profit Per Order            0
Order Region                      0
Order State                       0
Order Status                      0
Order Zipcode                155679
Product Card Id                   0
Product Category Id               0
Product Description          180519
Product Image                     0
Product Name                      0
Product Price                     0
Product Status                    0
shipping date (DateOrders)        0
Shipping Mode                     0
dtype: int64
```

**Figure 7: Handling Missing Values**

Below figure 8 showing identified missing values filled with appropriate naming and drop some columns those are irrelevant and some columns are merged each other.

```python
# Replace missing values in 'Customer Lname' with 'NotDetermined'
DataCo['Customer Lname'].fillna('NotDetermined', inplace=True)

# Replace missing values in 'Customer Zipcode' with 0
DataCo['Customer Zipcode'].fillna(0, inplace=True)

# Replace missing values in 'Order Zipcode' with 0
DataCo['Order Zipcode'].fillna(0, inplace=True)

# Drop the column 'Product Description' containing missing values
DataCo.drop(columns=['Product Description'], inplace=True)

# Combine customer first and last names into a new column 'Customer Full Name'
DataCo['Customer Full Name'] = DataCo['Customer Fname'] + ' ' + DataCo['Customer Lname']

# Verify that there are no more missing values
print(DataCo.isnull().sum())

# Display the first few rows of the updated dataset
DataCo.head()
```

**Figure 8: Replacing missing values**

# 3.4 Data Pre-processing

**EDA**

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
# Selecting relevant features
features = ['Customer City', 'Customer Country', 'Customer Segment', 'Customer State', 'Customer Zipcode']
data_selected = DataCo[features]

# Encoding categorical variables
data_encoded = pd.get_dummies(data_selected, drop_first=True)

# Scaling the features
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_encoded)

# Finding the optimal number of clusters using the Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)

# Plotting the results
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method')
plt.show()

# Applying KMeans with the optimal number of clusters (e.g., 4)
kmeans = KMeans(n_clusters=4, random_state=42)
DataCo['Cluster'] = kmeans.fit_predict(data_scaled)
```

**Figure 9: Exploratory Data Analysis**

```
# Grouping the data by Customer Segment and calculating total sales
grouped_data = DataCo.groupby('Market')['Sales'].sum().astype(int).reset_index()

# Display the grouped data
print(grouped_data)
from matplotlib.ticker import ScalarFormatter
# Plotting the histogram
plt.figure(figsize=(8, 6))
sns.barplot(x='Market', y='Sales', data=grouped_data, palette='viridis')
formatter = ScalarFormatter(useOffset=False)
formatter.set_scientific(False)
plt.gca().yaxis.set_major_formatter(formatter)
# Adding labels and title
plt.xlabel('Market')
plt.ylabel('Sales')
plt.title('Sales by Market ')

# Display the plot
plt.show()
```

**Figure 10: Customer Segment and calculating total sales**

# 3.5 Feature Scaling

Some features are selected for further processing. They are sowing in below figure of correlation heatmap.

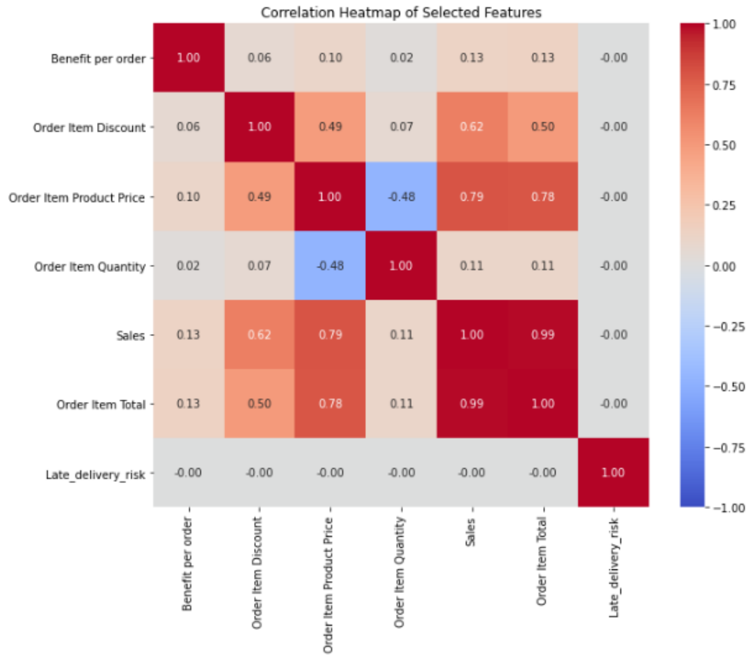


**Figure 11: Correlation matrix**

## 3.6 Model Building

OLS Regression model is used to check future demand of the goods based on the sales of the products also classification different algorithms are used to check late deliveries.

## 3.6.1 OLS Regression Modelling

Predictor and response variables were selected from dataset for calculating OLS regression. Apply label encoding and convert the categorical values into appropriate format.

```
le = preprocessing.LabelEncoder()# create the Labelencoder object
reg_SC['market']= le.fit_transform(reg_SC['market'])#convert the categorica
reg_SC['type']= le.fit_transform(reg_SC['type'])
reg_SC['product_name']= le.fit_transform(reg_SC['product_name'])
reg_SC['customer_segment']= le.fit_transform(reg_SC['customer_segment'])
reg_SC['order_region']= le.fit_transform(reg_SC['order_region'])
reg_SC['category_name']= le.fit_transform(reg_SC['category_name'])
reg_SC['shipping_mode']= le.fit_transform(reg_SC['shipping_mode'])
reg_SC['delivery_status']= le.fit_transform(reg_SC['delivery_status'])
reg_SC['customer_country']  = le.fit_transform(reg_SC['customer_country'])
reg_SC['customer_state']= le.fit_transform(reg_SC['customer_state'])
reg_SC['order_city'] = le.fit_transform(reg_SC['order_city'])
reg_SC['customer_city']= le.fit_transform(reg_SC['customer_city'])
reg_SC['department_name']= le.fit_transform(reg_SC['department_name'])
reg_SC['order_state'] = le.fit_transform(reg_SC['order_state'])
reg_SC['order_status'] = le.fit_transform(reg_SC['order_status'])
reg_SC['order_country']= le.fit_transform(reg_SC['order_country'])
```

```
reg_SC['Intercept'] = 1
```

```
independants=reg_SC[['Intercept',
       'order_item_product_price','order_country','order_item_discount',
      'order_profit_per_order','order_item_quantity','delivery_status','cus
'market','type','product_name','customer_segment','order_region','category_
ols_model = sm.OLS(reg_SC['order_item_total'], independants)
results = ols_model.fit()
results.summary()
```

**Figure 12: OLS Regression**

Dataset is splitted as training and testing purpose. Applied PCA to reduced dimensionality and then applied model.

```
train_SC = SCData.copy()
```

```
le = preprocessing.LabelEncoder()# create the Labelencoder object
train_SC['customer_country']  = le.fit_transform(train_SC['customer_country'])#
train_SC['market']= le.fit_transform(train_SC['market'])
train_SC['delivery_status']= le.fit_transform(train_SC['delivery_status'])
train_SC['type']= le.fit_transform(train_SC['type'])
train_SC['product_name']= le.fit_transform(train_SC['product_name'])
train_SC['customer_segment']= le.fit_transform(train_SC['customer_segment'])
train_SC['customer_state']= le.fit_transform(train_SC['customer_state'])
train_SC['order_region']= le.fit_transform(train_SC['order_region'])
train_SC['order_city'] = le.fit_transform(train_SC['order_city'])
train_SC['category_name']= le.fit_transform(train_SC['category_name'])
train_SC['customer_city']= le.fit_transform(train_SC['customer_city'])
train_SC['department_name']= le.fit_transform(train_SC['department_name'])
train_SC['order_state'] = le.fit_transform(train_SC['order_state'])
train_SC['order_status'] = le.fit_transform(train_SC['order_status'])
train_SC['shipping_mode']= le.fit_transform(train_SC['shipping_mode'])
train_SC['order_country']= le.fit_transform(train_SC['order_country'])


train_SCData=train_SC.drop(['shipping_date_dateorders','order_date_dateorders']

xorderitemquantity=train_SCData .loc[:, train_SCData .columns !='order_item_qua
yorderitemquantity=train_SCData['order_item_quantity']
xorderitemquantity_train, xorderitemquantity_test,yorderitemquantity_train,yord
```

**Figure 13: OLS training and testing**

Calculate MAE and RMSE value of linear regression and random forest repressor.

```
scaler=MinMaxScaler()
xorderitemquantity_train=scaler.fit_transform(xorderitemquantity_train)
xorderitemquantity_test=scaler.transform(xorderitemquantity_test)
```

```
Linear Regression
```

```
model_orderitemquantity=LinearRegression()
regressionmodel(model_orderitemquantity,xorderitemquantity_train, xorderit
```

```
Model parameter used are: LinearRegression()
MAE of Total amount per order is          : 0.33908849785509737
RMSE of Total amount per order is         : 0.5253875506211831
```

```
RandomForestRegressor
```

```
model_orderitemquantity = RandomForestRegressor(n_estimators=100,max_depth
regressionmodel(model_orderitemquantity,xorderitemquantity_train, xorderit
```

```
Model parameter used are: RandomForestRegressor(max_depth=10, random_state
MAE of Total amount per order is          : 7.201418125415458e-05
RMSE of Total amount per order is         : 0.005801260843366288
```

```
Regression model evaluation
```

**Figure 14: Regression model**

### 3.6.2 Classification modelling

```
Classification Model
```

```
train_DataSC = SCData.copy()
train_DataSC['late_delivery']=np.where(train_DataSC['delivery_status'] == 'Late deliver
train_dataSC=train_DataSC.drop(['delivery_status','late_delivery_risk'], axis=1)
le = preprocessing.LabelEncoder()# create the Labelencoder object
train_dataSC['customer_country']  = le.fit_transform(train_dataSC['customer_country'])#
train_dataSC['market']= le.fit_transform(train_dataSC['market'])
train_dataSC['type']= le.fit_transform(train_dataSC['type'])
train_dataSC['product_name']= le.fit_transform(train_dataSC['product_name'])
train_dataSC['customer_segment']= le.fit_transform(train_dataSC['customer_segment'])
train_dataSC['customer_state']= le.fit_transform(train_dataSC['customer_state'])
train_dataSC['order_region']= le.fit_transform(train_dataSC['order_region'])
train_dataSC['order_city'] = le.fit_transform(train_dataSC['order_city'])
train_dataSC['category_name']= le.fit_transform(train_dataSC['category_name'])
train_dataSC['customer_city']= le.fit_transform(train_dataSC['customer_city'])
train_dataSC['department_name']= le.fit_transform(train_dataSC['department_name'])
train_dataSC['order_state'] = le.fit_transform(train_dataSC['order_state'])
train_dataSC['order_status'] = le.fit_transform(train_dataSC['order_status'])
train_dataSC['shipping_mode']= le.fit_transform(train_dataSC['shipping_mode'])
train_dataSC['order_country']= le.fit_transform(train_dataSC['order_country'])
```

```
train_dataSCM=train_dataSC.drop(['shipping_date_dateorders','order_date_dateorders'],ax

xlatedelivery=train_dataSCM .loc[:, train_dataSCM .columns != 'late_delivery']
ylatedelivery=train_dataSCM['late_delivery']
xlatedelivery_train, xlatedelivery_test,ylatedelivery_train,ylatedelivery_test = train_
```

**Figure 15: Classification model**

```
def classifiermodel(model_latedelivery,xlatedelivery_train, xlatedelivery_test,ylatedeliver
    model_latedelivery=model_latedelivery.fit(xlatedelivery_train,ylatedelivery_train) # Fi
    ylatedelivery_pred=model_latedelivery.predict(xlatedelivery_test)
    accuracy_latedelivery=accuracy_score(ylatedelivery_pred, ylatedelivery_test) #Accuracy
    recall_latedelivery=recall_score(ylatedelivery_pred, ylatedelivery_test)# Recall score
    conf_latedelivery=confusion_matrix(ylatedelivery_test, ylatedelivery_pred)#predection o
    f1_latedelivery=f1_score(ylatedelivery_test, ylatedelivery_pred)#predection of late del
    print('Model paramters used are :',model_latedelivery)
    print('Accuracy of late delivery status is:', (accuracy_latedelivery)*100,'%')
    print('Recall score of late delivery status is:', (recall_latedelivery)*100,'%')
    print('F1 score of late delivery status is:', (f1_latedelivery)*100,'%')
    print('Conf Matrix of late delivery status is: \n',(conf_latedelivery))
```

**Figure 16: model training and testing**

## 3.7 Model Evaluation

OLS regression model coefficient, standard error calculated with MAE and RMSE. Few columns are showing in figure 17.

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | -35.4732 | 0.637 | -55.714 | 0.000 | -36.721 | -34.225 |
| order_item_product_price | 0.9688 | 0.001 | 1070.831 | 0.000 | 0.967 | 0.971 |
| order_country | -0.0022 | 0.002 | -1.075 | 0.282 | -0.006 | 0.002 |
| order_item_discount | -0.6025 | 0.005 | -132.234 | 0.000 | -0.611 | -0.594 |
| order_profit_per_order | 0.0120 | 0.001 | 15.571 | 0.000 | 0.010 | 0.013 |
| order_item_quantity | 53.7354 | 0.071 | 751.562 | 0.000 | 53.595 | 53.876 |
| delivery_status | 0.0821 | 0.082 | 1.001 | 0.317 | -0.079 | 0.243 |
| customer_country | 0.0980 | 0.231 | 0.423 | 0.672 | -0.356 | 0.552 |
| customer_state | 0.0065 | 0.008 | 0.846 | 0.397 | -0.009 | 0.021 |
| order_city | 4.667e-05 | 8.22e-05 | 0.568 | 0.570 | -0.000 | 0.000 |
| customer_city | 0.0009 | 0.001 | 1.541 | 0.123 | -0.000 | 0.002 |

**Figure 17: OLS coefficient and standard error**

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | order_item_total | **R-squared:** | 0.920 |
| **Model:** | OLS | **Adj. R-squared:** | 0.920 |
| **Method:** | Least Squares | **F-statistic:** | 1.043e+05 |
| **Date:** | Mon, 12 Aug 2024 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 07:42:01 | **Log-Likelihood:** | -8.9208e+05 |
| **No. Observations:** | 180519 | **AIC:** | 1.784e+06 |
| **Df Residuals:** | 180498 | **BIC:** | 1.784e+06 |
| **Df Model:** | 20 | | |
| **Covariance Type:** | nonrobust | | |

**Figure 18: OLS Regression**

| | Regression Model | MAE | RMSE |
|---|---|---|---|
| **0** | Linear Regression | 0.339088 | 0.525388 |
| **1** | Random Forest | 7.201418 | 0.005801 |

**Figure 19: OLS Regression Model Comparison**

Confusion matrix is calculated by using different classification models and also checked accuracy, recall and F1 scores.

| | Classification Model | Accuracy | Recall | F1 | TN | FP | FN | TP |
|---|---|---|---|---|---|---|---|---|
| 0 | Random Forest Classification | 99.868897 | 99.761873 | 99.880794 | 24340 | 71 | 0 | 29745 |
| 1 | Support Vector Machines | 98.255041 | 96.933055 | 98.436130 | 23470 | 941 | 4 | 29741 |
| 2 | Logistic Classification Model | 98.253194 | 96.932955 | 98.434449 | 23470 | 941 | 5 | 29740 |
| 3 | Linear Discriminant Analysis | 96.185095 | 96.262744 | 96.537043 | 23293 | 1118 | 948 | 28797 |
| 4 | Gaussian Naive Bayes Model | 85.032129 | 88.033888 | 86.070250 | 21007 | 3404 | 4702 | 25043 |

**Figure 20: Classification model results**

# References

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