Configuration Manual

MSc Research Project
Msc Fintech

SAFA MASOOD
Student ID: x22186506

School of Computing
National College of Ireland

Supervisor: FAITHFUL ONWUEGBUCHE

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	SAFA MASOOD……
Student ID:	X22186506
Programme:	MSc in Fintech **Year:** 2024… ……
Module:	MSc Research Configuration Manual
Lecturer:	FAITHFUL ONWUEGBUCHE ……
Submission Due Date:	16/09/2024……..……
Project Title:	Predicting Sales and Analysing Customer Lifetime Value (CLV) in the E-Commerce Industry Using Machine Learning Methods
Word Count:	**Page Count: 08**

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: SAFA MASOOD……

Date: ……16 SEP 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

SAFA MASOOD
Student ID: x22186506

1 Introduction

In the context of e-commerce, the research focuses on evaluating the applicability of several regression algorithms for sales prediction and customer lifetime value (CLV) analysis. Utilising a variety of regression-based machine algorithms to anticipate sales using the Brazilian e-commerce dataset is the research goals. Optimising customer acquisition, retention, and overall profitability for the business is made possible by the research's precise sales prediction and CLV analysis. This configuration manual covers every step required for replication, from setting up the environment to assessing the model.

2 Configuring the System

This research uses the Jupyter Notebook IDE and the Windows operating system with 8GB RAM to analyse Customer Lifetime Value (CLV) and estimate sales value. The system configuration includes the CPU Intel Core with i7 (Octacore) having the RAM of 8GB, storage of 512 SSD with windows 10 operating system. Python 3.10 is used for coding.

3 Data Collection

The study is conducted by gathering the data from the Kaggle dataset repository. The dataset used in this study is derived from publicly available Brazilian datasets of orders placed at the Olist store (Brazilian Public E-commerce Dataset). The dataset includes records of 100,000 orders placed at several Brazilian marketplaces between the years of 2016 and 2018. The method entails converting many columns to datetime using distinct formats, such as datetime from pandas, and then separating the resulting strings into dates, times, and months. Features are chosen that are more pertinent to forecasting sales values. Using the train test split method from Scikit-learn, the dataset is divided into training and testing sets after the pertinent features have been chosen. This ensures that the model is robust and can be applied to new data. Three performance measures are used to evaluate the models: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). These metrics are used to determine how effective the models are.

4 Environment Setup

The project was completed using Jupyter notebook. To utilise the dataset on the Jupyter Notebook, download and unzip the dataset from Kaggle. It was then uploaded into the Jupyter notebook from the desktop as illustrated in the figure (fig.1).

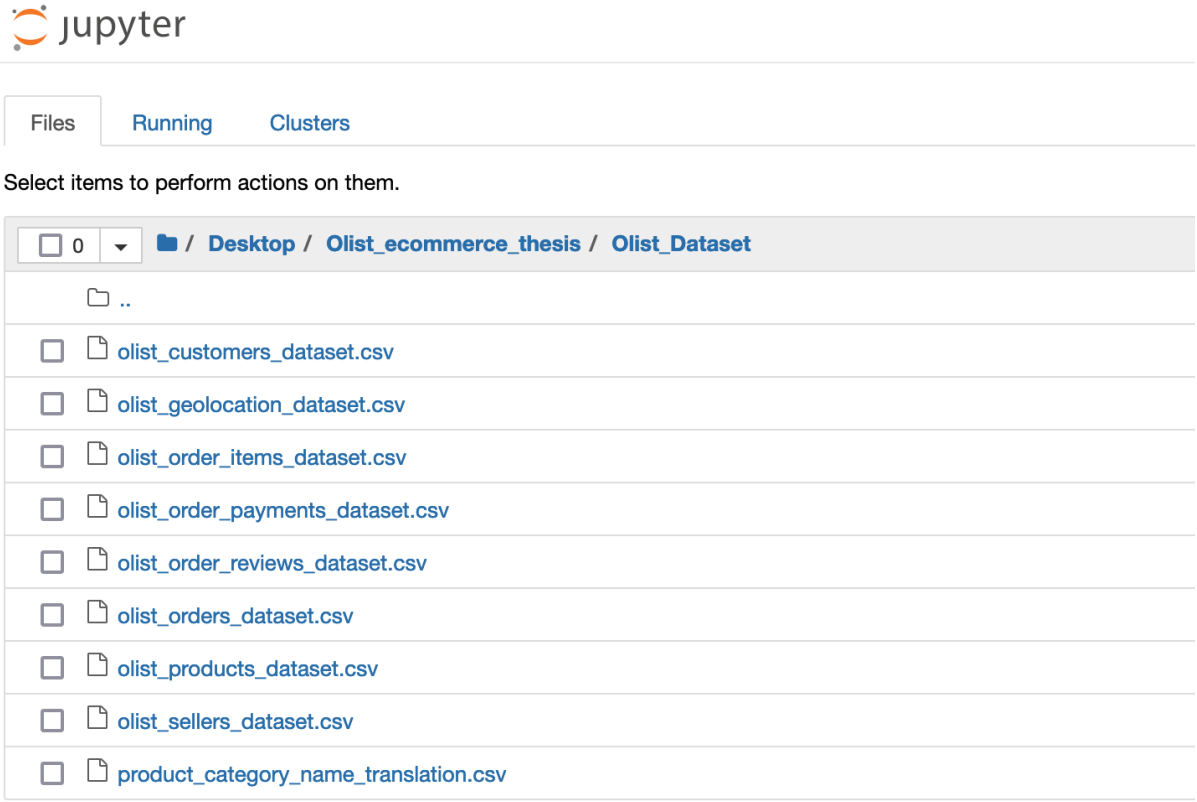

Fig.1

To read the dataset from different raw files the below code is given as in the figure (fig.2).

```
customers = pd.read_csv('Olist_Dataset/olist_customers_dataset.csv')      # reading raw customers data file
sellers = pd.read_csv('Olist_Dataset/olist_sellers_dataset.csv')         # reading raw sellers data file
products = pd.read_csv('Olist_Dataset/olist_products_dataset.csv')          # reading raw products data file
orders = pd.read_csv('Olist_Dataset/olist_orders_dataset.csv')           # reading raw orders data file
order_items = pd.read_csv('Olist_Dataset/olist_order_items_dataset.csv')   # reading raw order_items data file
order_payments = pd.read_csv('Olist_Dataset/olist_order_payments_dataset.csv')     # reading raw order_payments data
order_reviews = pd.read_csv('Olist_Dataset/olist_order_reviews_dataset.csv')        # reading raw order_reviews data
geolocation = pd.read_csv('Olist_Dataset/olist_geolocation_dataset.csv')       # reading raw geolocation data file
products_translation = pd.read_csv('Olist_Dataset/product_category_name_translation.csv') # reading raw products_tra
```

Fig.2

5 Data Exportation

5.1 Libraries Imports

Key libraries utilised in the study include Pandas for data translation and manipulation, Matplotlib and Seaborn for visualising data distributions and patterns, NumPy for numerical operations and array handling, and Plotly for generating interactive plots and visualisations as in the figure (fig.3).

```
import numpy as np     # importing numpy for arrays and numerical computations
import pandas as pd       # importing pandas for data manipualtion
import calendar
import seaborn as sns               # importing some visaulization libraries
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
from matplotlib import ticker
from matplotlib.ticker import FuncFormatter
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.offline as pyo
import matplotlib as mpl
import matplotlib.patches as patches
from matplotlib.patches import ConnectionPatch
from matplotlib.gridspec import GridSpec
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
from collections import OrderedDict   # importing ordered dictionary
from collections import Counter
import warnings                    #importing warnings
warnings.filterwarnings('ignore')
import random        # setting displaying setting of notebook
plt.style.use('fivethirtyeight')
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 200)
plt.rcParams["figure.figsize"] = (15,7)
%matplotlib inline
```

Fig.3

6 Data Pre-processing

For the data pre-processing firstly the dataset need to be summarized to check the number of columns, rows and summary of Boolean, categorial and numerical columns in each dataset. For that, one of the codes are given below in figures.

```
#checking number of columns , column_names and no_of_rows in each dataset
all_datasets = [customers,sellers,products, orders, order_items, order_payments,order_reviews,geolocation,products_t
data_titles = ['customers','sellers','products','orders', 'order_items', 'order_payments','order_reviews','geolocati

data_info = pd.DataFrame({},)
data_info['all_datasets']= data_titles

data_info['Columns']= [len(df.columns) for df in all_datasets ]
data_info['Column_name']= [', '.join(list(df.columns)) for df in all_datasets]
data_info['Rows'] = [len(df) for df in all_datasets]

data_info.style.background_gradient(cmap='RdBu')
```

Fig.4 (checking number of columns, column names and rows)

Below figure shows the code to check the null values in each dataset so that can be dropped.

```
all_datasets = [customers,sellers,products, orders, order_items, order_payments,order_reviews,geolocation,products_t
titles = ['customers','sellers','products','orders', 'order_items', 'order_payments','order_reviews','geolocation','
data_desc = pd.DataFrame({},)

data_desc['all_datasets']= titles

#  name of columns in the all_datasets
data_desc['cols'] = [', '.join([col for col, null in df.isnull().sum().items() ]) for df in all_datasets]

# total number of columns in the all_datasets
data_desc['cols_no']= [df.shape[1] for df in all_datasets]

#counting total null values
data_desc['null_no']= [df.isnull().sum().sum() for df in all_datasets]

# total number of columns in the all_datasets with null-values
data_desc['null_cols_no']= [len([col for col, null in df.isnull().sum().items() if null > 0]) for df in all_datasets
data_desc['null_cols'] = [', '.join([col for col, null in df.isnull().sum().items() if null > 0]) for df in all_data
data_desc.style.background_gradient(cmap='Pastel1_r')
```

Fig.5

After determining the null values in the dataset, it must be dropped and rename the column for easier and creating the target variable in the dataset as illustrated in the below figure.

```
# Dropping columns having high Null values
order_reviews = order_reviews.drop(['review_comment_title','review_creation_date','review_id','review_answer_timesta
```

```
order_items.rename(columns={'order_item_id':'quantity'}, inplace=True)     # renaming column name for more easy
```

```
order_items = order_items.assign(sale= lambda x: x['quantity']*x['price'])      # Creating the target variable in th
```

Fig.6

For the ease of interpretation dataset is merged based on the data schema (fig.7) as seen in the picture and the merging code is given in figure (fig.8).

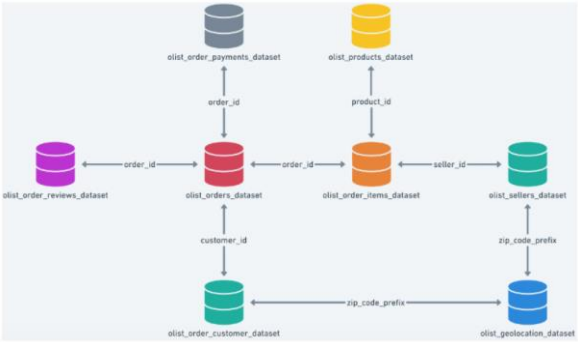

Fig.7

```
df = pd.merge(orders,order_payments, on="order_id")       # merging orders and orders payment dataset
df = df.merge(customers, on="customer_id")               # merging customers dataset
df = df.merge(order_items, on="order_id")                # merging order_items dataset
df = df.merge(products, on="product_id")                 # merging products data
df = df.merge(products_translation, on="product_category_name")     # merging products_translation dataset
df = df.merge(order_reviews, on="order_id")                         # merging order reviews dataset
df.head()          # visualising first five rows of merged dataset
```

Fig.8

After merging the dataset, the null values are checked, since merging may cause certain null values to appear. (fig.9)

```
df.isnull().sum()                    # checking null values in merged dataset
```

Fig.9

7 Exploratory Data Analysis

Data analysis, entails examining, cleaning, transforming, and modelling the data to provide relevant information, draw conclusions, and facilitate decision-making. It is the essential part of the machine learning pipeline that makes it possible to extract valuable information from the data. One of the examples is the code as in the below figure (fig.10).

```python
def pareto_plot(df, x=None, y=None, title=None, show_pct_y=False, pct_format='{0:.0%}'):
    xlabel = x
    ylabel = y
    tmp = df.sort_values(y, ascending=False)
    x = tmp[x].values
    y = tmp[y].values
    weights = y / y.sum()
    cumsum = weights.cumsum()

    fig, ax1 = plt.subplots(figsize=(10,6))
    ax1.bar(x, y,color='#15F5BA',edgecolor='black',alpha=0.9)
    ax1.set_xlabel(xlabel)
    ax1.set_ylabel(ylabel)

    ax2 = ax1.twinx()
    ax2.plot(x, cumsum, '-ro', alpha=0.5,color='black')
    ax2.set_ylabel('', color='r')
    ax2.tick_params('y', colors='r')

    vals = ax2.get_yticks()
    ax2.set_yticklabels(['{:,.2%}'.format(x) for x in vals])

    # hide y-labels on right side
    if not show_pct_y:
        ax2.set_yticks([])

    formatted_weights = [pct_format.format(x) for x in cumsum]
    for i, txt in enumerate(formatted_weights):
        ax2.annotate(txt, (x[i], cumsum[i]),fontsize=15)

    if title:
        plt.title(title,color='black',fontsize=20)

    plt.tight_layout()
    plt.show()
```

Fig.10

The code gives output with bars representing individual values in descending order and a line displaying the cumulative proportion, this function generates a Pareto chart. Percentage labels are displayed on the secondary y-axis and can be hidden if required. With options for the title, axis labels, and percentage format, the function provides a high degree of customisation.

8 Feature Engineering

Feature engineering assists in the selection of features that are more pertinent to the training of the model and in the conversion of raw data into a format that is appropriate for algorithm training.

CLV (CUSTOMER LIFETIME VALUE) ANALYSIS

CLV is a metric used to measure the approximate value that a customer is expected to bring to a company over the course of a customer-provider relationship.

In the below figure the code gives output on calculating the entire revenue for each order, Calculating the Average Order Value (AOV) , total number of orders for each customer, When total revenue per customer is calculated, combining total orders, total revenue, and AOV, Finding the Purchase Frequency Customer Value, Customer Lifespan, and Customer Lifetime Value (CLV)

```
# Calculating total revenue for each order
df['total_price'] = df['price'] * df['quantity']

# Calculating Average Order Value (AOV) per customer
aov = df.groupby('customer_unique_id')['total_price'].mean().reset_index()
aov.columns = ['customer_unique_id', 'AOV']

# Calculating total number of orders per customer
total_orders = df.groupby('customer_unique_id')['order_id'].count().reset_index()
total_orders.columns = ['customer_unique_id', 'total_orders']

# Calculating total revenue per customer
total_revenue = df.groupby('customer_unique_id')['total_price'].sum().reset_index()
total_revenue.columns = ['customer_unique_id', 'total_revenue']

# Merging AOV, total_orders, and total_revenue
customer_data = pd.merge(aov, total_orders, on='customer_unique_id')
customer_data = pd.merge(customer_data, total_revenue, on='customer_unique_id')

# Calculating Purchase Frequency
purchase_frequency = total_orders['total_orders'].sum() / total_orders.shape[0]

# Calculating Customer Value
customer_data['customer_value'] = customer_data['AOV'] * purchase_frequency

# Calculating Customer Lifespan (assuming 2 years for simplicity)
customer_lifespan = 2

# Calculating Customer Lifetime Value (CLV)
customer_data['CLV'] = customer_data['customer_value'] * customer_lifespan

# Displaying the first few rows of the CLV calculation
customer_data.head()
```

Fig.11

Making a layout for the subplot, including a trace for CLV Including the Total Orders trace updating the combined plot's layout displaying

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Creating a subplot layout with 1 row and 2 columns
fig = make_subplots(rows=1, cols=2, subplot_titles=('Distribution of CLV', 'Distribution of Total Orders'))

# Adding trace for CLV in the first column
fig.add_trace(
    go.Histogram(x=customer_data['CLV'], nbinsx=50, name='CLV', opacity=0.75, marker=dict(color='purple')),
    row=1, col=1
)

# Adding trace for Total Orders in the second column
fig.add_trace(
    go.Histogram(x=total_orders['total_orders'], nbinsx=50, name='Total Orders', opacity=0.75, marker=dict(color='re
    row=1, col=2
)

# Updating layout for the combined plot
fig.update_layout(
    title_text='Combined Distributions of CLV and Total Orders',
    showlegend=False,
    xaxis_title_text='Value',
    xaxis2_title_text='Value',
    yaxis_title_text='Count',
    yaxis2_title_text='Count'
)

# Showing the plot
fig.show()
```

Fig.12

9 Training Model

For training and evaluation of the result three machine learning algorithms are used on the data. By dividing the dataset into an 80% training set and a 20% testing set, each machine learning algorithm is trained on the training set. The linear regression, random forest and XGBoost are the 3 Machine learning algorithms used in this dataset.
For the data preparation for machine learning some columns are selected which are relevant for machine learning (fig.13).

```
# selecting some columns which are more relevent for predicition purpose
selected_cols = ['order_status','payment_sequential','payment_type','payment_installments','payment_value','customer
                 'product_name_lenght','product_description_lenght', 'product_photos_qty', 'product_weight_g','produc
                  'year','month','day','day_of_week','hour','actual_delivery_time','estimated_delivery_time','deliver
final_data = data[selected_cols]
final_data.shape
final_data.to_csv('final_merged_dataset.csv')
```

Fig.13

9.1 Implementing Machine Learning Model

For implementing Machine learning model, the ML model and important libraries are imported as in the figure (fig.14)

```
from sklearn.ensemble import RandomForestRegressor       #importing ML models and some important Libaries
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_validate
import xgboost as xgb
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve,confusion_matrix,precision_score,recall_score,roc_auc_score,accuracy_score,cla
import tensorflow as tf
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

Fig.14

To assess the effectiveness of ML model the dataset is separated into training set and testing set. For this, make use of the train_test_split function from the scikit-learn module. (fig.15).

```
X_train, X_test, y_train, y_test = train_test_split(pc_data,target, test_size =0.2,random_state = 0,shuffle=True)
```

Fig.15

To initialise the linear regression model and fit training data into the model (fig.16) and for predicting the model (fig.17).

```
lr= LinearRegression()                    # initialisng linear regression model
lr.fit(X_train,y_train)                 # fitting data into model for training
```

Fig.16

```
y_pred1 = lr.predict(X_test)                  #making prediction on test data
mse_lr = mean_squared_error(y_pred1,y_test)
rmse_lr = mean_squared_error(y_pred1,y_test,squared=False)
mae_lr = mean_absolute_error(y_pred1,y_test)

print('MSE: ',mse_lr)
print('RMSE: ',rmse_lr)
print('MAE: ',mae_lr)
```

Fig.17

To initialise the Random Forest and fit training data into the model (fig.18) and for predicting the model (fig.19).

```
rf=RandomForestRegressor()            #initialisng random forest model
rf.fit(X_train,y_train)         # fitting data into model for training
```

Fig.18

```
y_pred2 = rf.predict(X_test)          #predicting test data
mse_rf = mean_squared_error(y_pred2,y_test)
rmse_rf = mean_squared_error(y_pred2,y_test,squared=False)
mae_rf = mean_absolute_error(y_pred2,y_test)

print('MSE: ',mse_rf)
print('RMSE: ',rmse_rf)
print('MAE: ',mae_rf)
```

Fig.19

To initialise the XGBoost and fit training data into the model (fig.20) and for predicting the model (fig.21).

```
xgb_reg = xgb.XGBRegressor()          #initialisng Xgboost model
xgb_reg.fit(X_train, y_train)            # fitting data into model for training
```

Fig.20

```
y_pred3 = xgb_reg.predict(X_test)          #predicting test data
mse_xgb = mean_squared_error(y_pred3,y_test)
rmse_xgb = mean_squared_error(y_pred3,y_test,squared=False)
mae_xgb = mean_absolute_error(y_pred3,y_test)

print('MSE: ',mse_xgb)
print('RMSE: ',rmse_xgb)
print('MAE: ',mae_xgb)
```

Fig.21

10 Evaluation

The algorithms performance is assessed using three performance measures. To assess the model's generalisability on the test data, three metrics are used. It is used the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) metrics to assess the effectiveness of the e-commerce sales forecast model. MSE highlighted significant errors by providing a measure of the average squared differences between the actual and projected values. The corresponding measurement of error magnitude in the same units as the data was provided by RMSE, which is the square root of MSE. MAE, on the other hand, provided a clear indicator of prediction accuracy by measuring the average absolute errors. When taken as a whole, these indicators provide a thorough assessment of the predictive performance of the model.