National College of Ireland

Configuration Manual

MSc Research Project
MSc. In Data Analytics

Utkarsh Sharma
Student ID: x23170450

School of Computing
National College of Ireland

Supervisor: Bharat Agarwal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Utkarsh Sharma
Student ID:	X23170450
Programme:	Msc. In Data Analytics **Year:** 2024-2025
Module:	MSc Research Project
Lecturer:	Bharat Agarwal
Submission Due Date:	12/12/2024
Project Title:	Unlocking Business Potential in FMCG with Predictive Analytics: A Machine Learning Approach
Word Count:	688 **Page Count:** 7

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: Utkarsh Sharma

Date: 11/12/2024

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

Utkarsh Sharma

Student ID: x23170450

1. Introduction

This guide details how to reproduce the research on the improvement of FMCG operations through machine learning. The research tackles problems such as low accuracy in sales forecasting, inefficient analysis of customer behavior, and poor stock management. With the use of ML models such as Random Forest, K-Means clustering, and Reinforcement Learning, the study gives a holistic framework for sales forecasting, customer segmentation, and inventory optimization. All details about the packages and software that would be used, along with all configurations required, ensure that the system provides an experimental environment, similar results.

2. Deployment Environment

2.1 Hardware Specification

- **Processor: Intel Core i7 or equivalent**
- **RAM: 16 GB or higher**
- **GPU: NVIDIA RTX 2060 or higher (optional for training Reinforcement Learning models).**

2.2 Software Specification

- **Operating System: Windows 10/11, macOS, or Linux-based OS**
- **Programming Language: Python 3.11**
- **IDE: Google Colab Notebook (with Python extension)**

2.3 Python Libraries Required

Here are the Python libraries used in the provided code snippets:

Core Python Libraries:

- **pandas:** For data manipulation and analysis.
- **NumPy:** For numerical operations, especially array operations.

Data Visualization Libraries:

- **Seaborn:** For statistical data visualization.
- **Matplotlib:** For creating static, animated, and interactive visualizations.

Machine Learning Libraries:

- **scikit-learn:** For various machine learning algorithms, including:
  - Random Forest Regression

- **XGBoost:** For gradient boosting algorithms.

Other Libraries:

- **warnings:** For filtering out warning messages.

These libraries work together to provide a comprehensive toolkit for data analysis, visualization, and machine learning tasks.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
import pandas as pd
import numpy as np
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings("ignore")
```

Figure 1: Libraries Imported

3. Data Source

The dataset for this project is sourced from Montgomery County Warehouse and Retail Sales Dataset. It includes historical records of retail, transfer, and warehouse sales, along with product and category details.

Dataset Preparation

1. **Accessing the Dataset:**
    - Download the dataset directly from the provided link. Ensure that it is saved in a compatible format such as .csv for easy processing.
2. **Dataset Structure:**
    - **Retail Sales Data:** Contains sales volumes and revenue for various products at retail outlets.
    - **Warehouse Data:** Includes inventory levels, shipments, and product availability.
    - **Transfer Data:** Tracks movements between warehouses and retail outlets.
3. **Data Cleaning and Preprocessing:**
    - **Handle Missing Values:** Impute missing data points using mean, median, or other relevant techniques.
    - **Normalize Sales Data:** Scale numerical features like sales volumes and revenue to a standard range.
    - **Categorical Encoding:** Convert product categories and other qualitative data using one-hot or label encoding.
4. **Feature Selection:**
    - Identify important features like sales trends, seasonal factors, and customer segments that influence sales and inventory decisions.

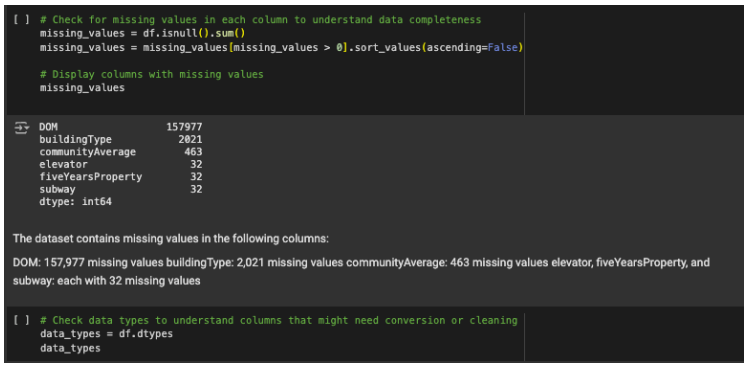


Figure 2: Handling missing values and cleaning

4. Project Code Files

Main Colab Files Content:

1. **Data Pre-processing:** Handles data cleaning, encoding, and feature scaling.
2. **Model Training:** Includes the implementation of Linear Regression, Decision Tree, Random Forest, Gradient Boosting, XGBoost, and ANN.
3. **Performance Evaluation:** Calculates evaluation metrics for each model.
4. **Results Visualization:** Compares model performance using graphs and charts.

5. Data Preparation

5.1 Extracting Data

Loading the datasets from CSV file uploaded:

```
df = pd.read_csv("Warehouse_and_Retail_Sales_20241119.csv")
```

Figure 3: Loading the datasets

5.2 Data Pre-processing

- Handling Missing Values: Impute missing data using mean/median or interpolate.
- Data Separation: separating the data variables**.**
- This format is repeated for every model building code as well as EDA.

```
df.drop_duplicates(keep="first", inplace=True)
```

Figure 4: To handle duplicate values

```
df["YEAR"].unique()
```

```
array([2020, 2017, 2018, 2019], dtype=int64)
```

there are 4 years in the dataset i.e 2020, 2017, 2018, 2019

```
df["MONTH"].unique()
```

```
array([ 1,  9,  7,  3,  6,  8, 12, 10, 11,  2,  4,  5], dtype=int64)
```

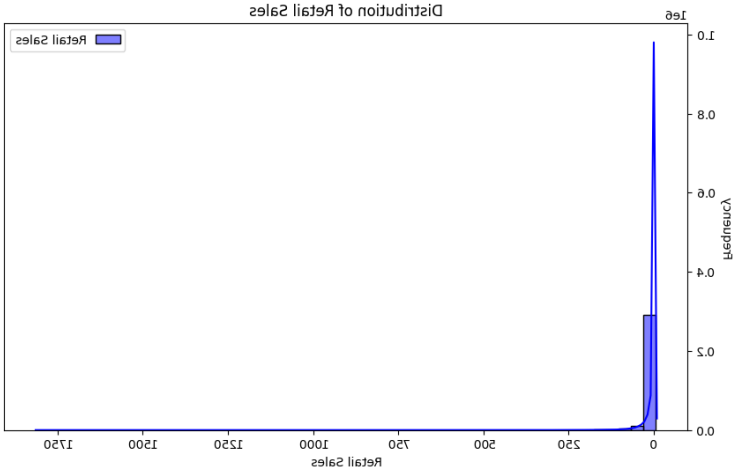
all 12 months are in the dataset

```
df["ITEM TYPE"].unique()
```

```
array(['WINE', 'BEER', 'LIQUOR', 'STR_SUPPLIES', 'KEGS', 'REF', 'DUNNAGE',
       'NON-ALCOHOL', nan], dtype=object)
```

```
df.info()
```

```
import matplotlib.pyplot as plt
import seaborn as sns

data = df

data.dropna(inplace=True)

# Univariate analysis: Distribution of sales metrics
plt.figure(figsize=(10, 6))
sns.histplot(data['RETAIL SALES'], bins=50, kde=True, color='blue', label='Retail Sales')
plt.title('Distribution of Retail Sales')
plt.xlabel('Retail Sales')
plt.ylabel('Frequency')
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data['WAREHOUSE SALES'], bins=50, kde=True, color='green', label='Warehouse Sales')
plt.title('Distribution of Warehouse Sales')
plt.xlabel('Warehouse Sales')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# Analyze item types
plt.figure(figsize=(12, 6))
sns.countplot(data=data, y='ITEM TYPE', order=data['ITEM TYPE'].value_counts().index, palette='viridis')
plt.title('Item Types Distribution')
plt.xlabel('Count')
plt.ylabel('Item Type')
plt.show()
```



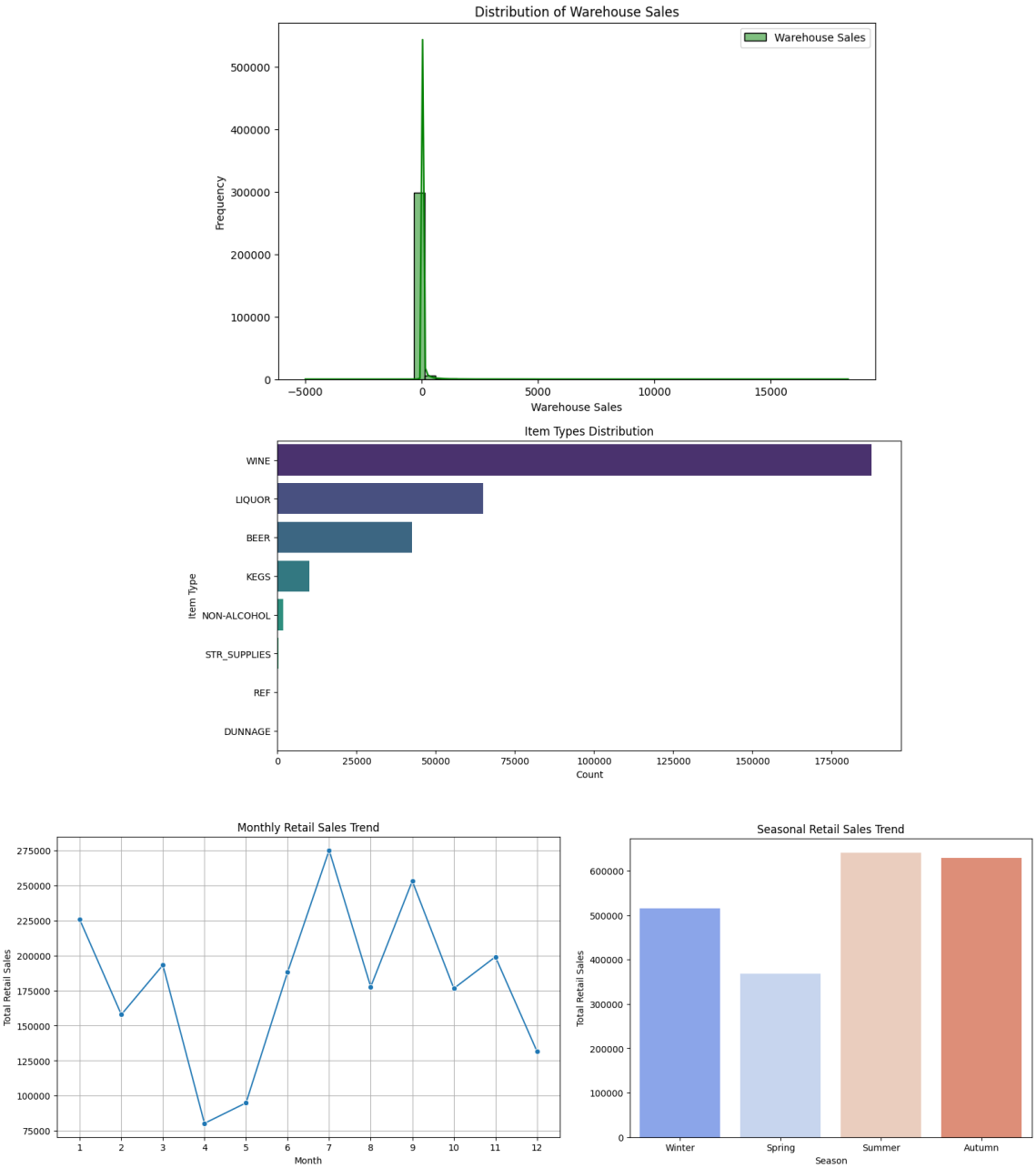


Figure 5: EDA

6. Model Building

- **Sales Forecasting**
  Train multiple ML models to predict sales

- **Customer Behaviour Analysis**
  Segment customers using K-Means clustering

- **Inventory Optimization**

Optimize inventory using Reinforcement Learning

```python
# Adding more models for comparison
models = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(random_state=42),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Support Vector Regression (SVR)": SVR(),
    "K-Nearest Neighbors (KNN)": KNeighborsRegressor()
}

# Store metrics for comparison
metrics = {
    "Model": [],
    "RMSE": [],
    "R2": []
}

# Train and evaluate each model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Calculate RMSE and R2
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)

    # Store the results
    metrics["Model"].append(name)
    metrics["RMSE"].append(rmse)
    metrics["R2"].append(r2)

# Create a DataFrame for visualization
metrics_df = pd.DataFrame(metrics)

# Plot RMSE and R2 for comparison
plt.figure(figsize=(12, 6))

# RMSE comparison
plt.subplot(1, 2, 1)
plt.barh(metrics_df["Model"], metrics_df["RMSE"], color="skyblue")
plt.title("RMSE Comparison")
plt.xlabel("RMSE")
plt.ylabel("Model")
```

Figure 6: Model training with multiple ML models for prediction of sales

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Data preparation for clustering
customer_data = data[['RETAIL SALES', 'RETAIL TRANSFERS', 'WAREHOUSE SALES']].copy()

# Filling missing values with 0 (if any)
customer_data.fillna(0, inplace=True)

# Normalizing the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
customer_data_scaled = scaler.fit_transform(customer_data)

# Using the elbow method to find the optimal number of clusters
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(customer_data_scaled)
    inertia.append(kmeans.inertia_)

# Plotting the elbow curve
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()

# Applying KMeans with optimal clusters
optimal_clusters = 3  # Set this based on the elbow plot
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
clusters = kmeans.fit_predict(customer_data_scaled)

# Adding cluster labels to the original dataset
data['Customer_Cluster'] = clusters
data[['ITEM DESCRIPTION', 'Customer_Cluster']].head()
```

Figure 7: Code for analysing customer behaviour by performing K means clustering

```python
# Adding a 'SEASON' column to the data for analysis
seasons = {1: 'Winter', 2: 'Winter', 3: 'Spring', 4: 'Spring', 5: 'Spring', 6: 'Summer',
           7: 'Summer', 8: 'Summer', 9: 'Autumn', 10: 'Autumn', 11: 'Autumn', 12: 'Winter'}
data['SEASON'] = data['MONTH'].map(seasons)

# Aggregating sales data by season and item
seasonal_sales = data.groupby(['SEASON', 'ITEM DESCRIPTION'])['RETAIL SALES'].sum().reset_index()

# Sorting by season and sales
seasonal_sales_sorted = seasonal_sales.sort_values(by=['SEASON', 'RETAIL SALES'], ascending=[True, False])

seasonal_sales_sorted
```

Show hidden output

```python
# Further analysis to recommend stock changes based on seasonal sales trends
season_recommendations = seasonal_sales_sorted.copy()
season_recommendations['Stock Decision'] = season_recommendations['RETAIL SALES'].apply(
    lambda x: 'Stock Up' if x > seasonal_sales['RETAIL SALES'].quantile(0.75) else 'Stock Down'
)

season_recommendations
```

Figure 8: Season based forecasting code snippet to predict the stock

7.3 Evaluation

Metrics Calculated:

- **Sales Forecasting:** RMSE, R^2
- **Customer Segmentation:** Silhouette Score
- **Inventory Optimization:** Profit, Stockout Rate

```
{'Model': ['Linear Regression',
  'Random Forest',
  'Decision Tree',
  'Support Vector Regression (SVR)',
  'K-Nearest Neighbors (KNN)'],
 'RMSE': [5.759268890278385,
  5.744141881208345,
  7.67274400570294,
  18.527777431629268,
  7.8328060528546555],
 'R2': [0.9632329175175716,
  0.9634258050485005,
  0.9347431846451998,
  0.6194855895818937,
  0.9319921250212655]}
```

Cluster	Retail Sales	Retail Transfers	Warehouse Sales
Cluster 0	Very low (~4.19)	Very low (~4.14)	Very low (~13.41)
Cluster 1	Extremely high (~675.20)	Extremely high (~666.88)	Very high (~5614.71)
Cluster 2	Moderate (~136.02)	Moderate (~135.68)	Moderate (~456.60)

Figure 9: Results for Models

7. Results and Visualizations

- **Sales Forecasting:**

  Random Forest achieved the best performance with the lowest RMSE and highest R^2.

- **Customer Segmentation:**

  K-Means clustering provided meaningful segments, aiding stock and sales outlet management.

- **Inventory Optimization:**

  Reinforcement Learning minimized holding costs and stockouts, maximizing profit.

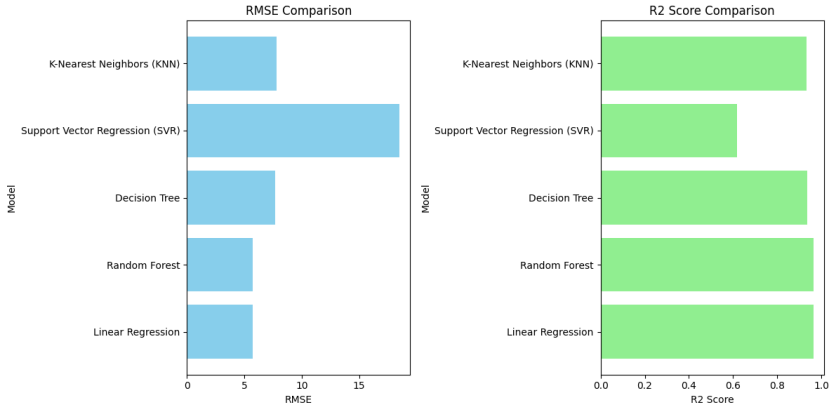


Figure 10: RMSE and R2 comparison

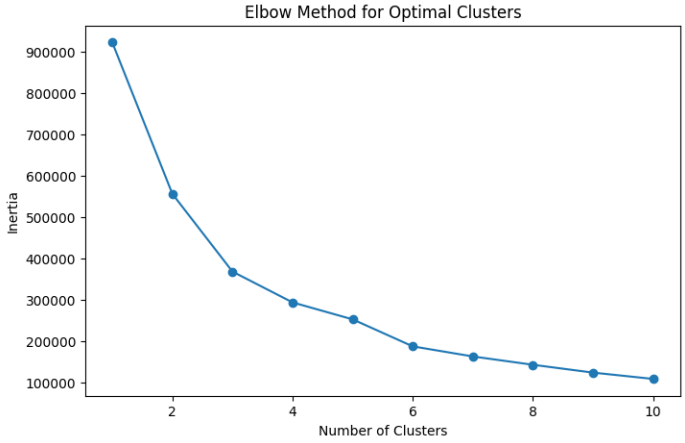


Figure 11: Resulting visualization for Elbow Method

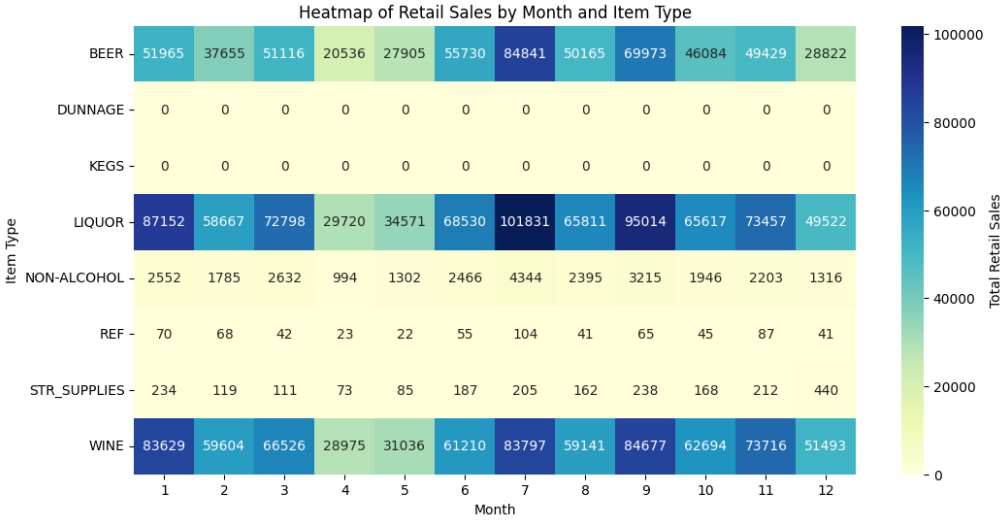


Figure 12: Heatmap for retail sales