Cover page for Individual Assignment 1

SUBMISSION DETAILS	
Student Number: 5 3 0 7 2 0 3	**Student Name:** **Ho, Sy Anh Quoc**

DECLARATION

I declare that this assessment item is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere, and acknowledge that the assessor of this item may, for the purpose of assessing this item:

- Reproduce this assessment item and provide a copy to another member of the University; and/or,
- Communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the assessment item on its database for the purpose of future plagiarism checking).

By submitting this assessment, I certify that I have read and understood the University Rules in respect of Student Academic Misconduct.

INSTRUCTIONS

Proceed to next page for your Individual Assignment 1 Analytics Report.

Make sure you follow the file naming instructions to save and name this file before you submit this file via course Moodle Individual Assessment 1 Submission Link.

Make sure you submit your assignment with this cover sheet. Failure to do so will result in 10% penalty of the marks available for this assignment.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This report will aim to use two ML models to predict the likelihood that customer will return the order and give recommendations on how to minimise order return. Moreover, VGG 16 and clustering ML model will be used to classify waste, and e-waste items that is promised to minimise the company cost and improve operational efficiency.

ANALYSIS OF STRUCTURED DATA: PREDICTING ITEM RETURN LIKELIHOOD

RATIONALE OF THE CHOICE OF ML ALGORITHMS

Logistic regression and random forest will be employed to predict the possibilities of merchandised returned.

Logistic regression is an algorithm to use for classification problems by modelling the probability of a binary outcome given a set of input features. Logistic regression can provide insight on the most relevant coefficients that have impact on order return. The features with high coefficient will be addressed with solutions from global. Moreover, logistic regression is preferred over other model such as linear regression because it is easy to train, and interpret, especially when it comes to a large dataset. Using logistic regression can minimise the overfitting problems that can lead to low model's accuracy.

However, the main disadvantage of this model is that it assumes all the features are independent, which might limit the ability to model complex relationship in the data. For example, quantity might depend on discount because higher discount will result in higher number of orders placed.

The second machine learning algorithm that will be used is random forest. Random forest is the model that can create an ensemble of decision trees to improve the predictive performance of individual decision trees by decreasing overfitting and increasing generalisation. Random forest has strong predictive power because it trains the model by using multiple subsets of data, which can reduce potential variance, and bias. Moreover, this model can handle large feature sets and provide insights into feature importance. The global's dataset consists of multiple features with enormous amount of data. Hence, it is suitable to use random forest for analysis

DATA PRE-PROCESSING AND SELECTION OF KEY FEATURES

Data cleaning is the first step of data pre-processing.

Firstly, missing values are checked to ensure that there is no missing value in the dataset.

Secondly, drop all the irrelevant columns and entities that may create many possible dummy variables. Precisely, there are several columns that were removed because they are identifiers and cannot aid in predicting the likelihood of customer return. They are order_id, order_date, shipping_date, customer_id, product_id, and product_name. Moreover, columns that potentially create large dummy variable such as category, sub_category, country, city, state, market, and region were also dropped out because they do not give insight into order return, nor potentially have predictive power.

Thirdly, all the object data features will be converted to category. These features include returned order, order_priority, shipping_mode, and segment.

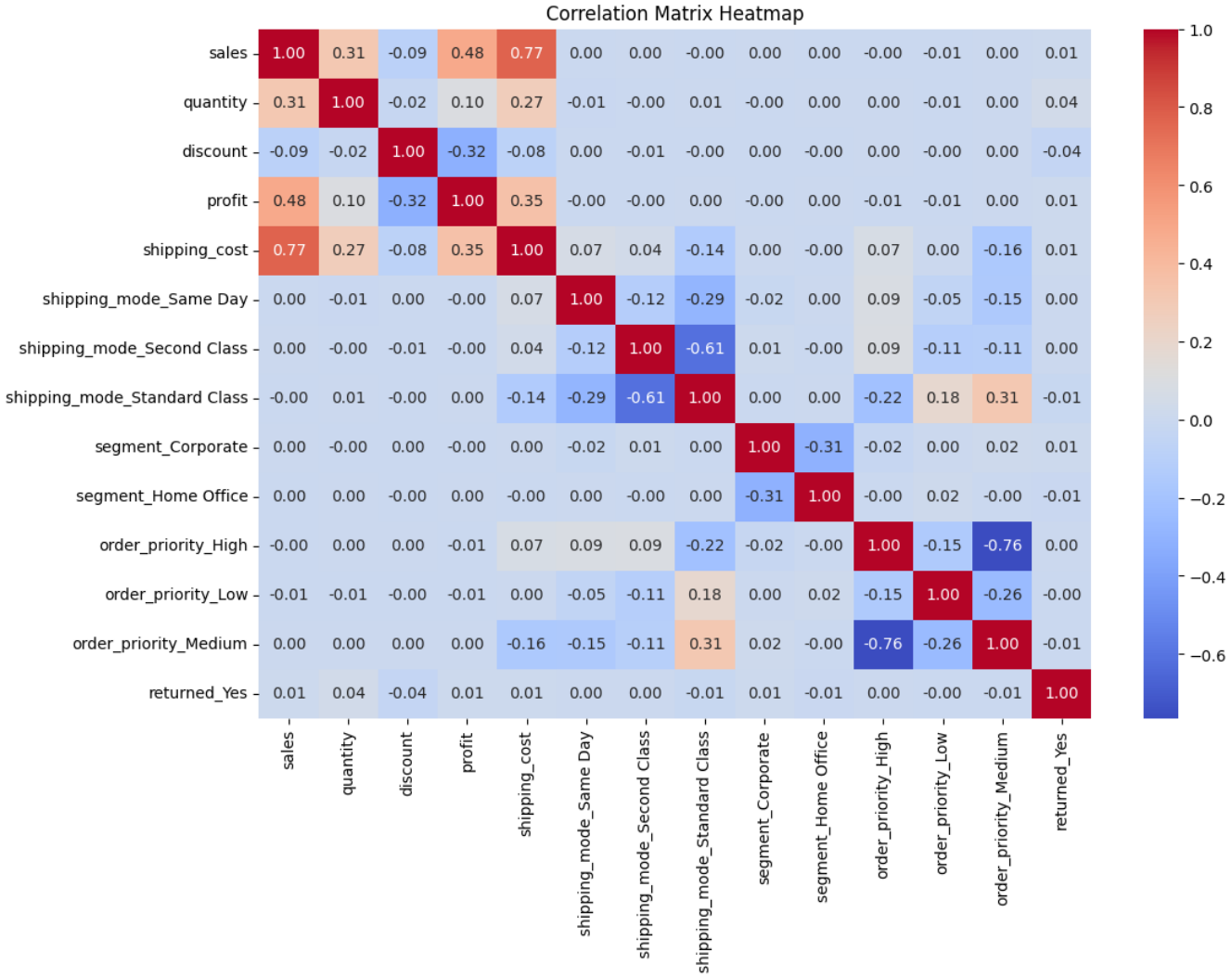
Fourthly, all the data will be converted to dummy variable which would aid in the features elimination process that can enhance model accuracy.

Fifthly, all the data will be spitted into X and Y columns ,of which X will consist of all relevant features, and Y will consist of target variable (Order returned)

The final step of data pre-processing will involves standardize data by using mean and standard deviation. It will ensure that all data are transformed to a more comparable unit.

Feature selections and Justification

Heat map



This heat map will be used to analyse the correlation coefficients between different multiples features to the target variable. All of the features with zero coefficients, or multicollinearity will be removed from the model.

Features	**Justification**
Sales (include)	The correlation coefficient of 0.01 indicates a weak relationship between sale and target variable. It demonstrates a slightly upward trend.
Quantity (include)	The correlation coefficient of 0.04 suggest a week positive relationship. It suggests a slight upward trend.

Discount (include)	Discount suggest a weak negative correlation of (-0.04). It indicates a slight downward trend
Profit (include)	The correlation coefficient of 0.01 indicates a week positive relationship. It suggests a slight upward trend
Shipping cost (exclude)	The shipping cost features will be eliminated from the model because it has a strong correlation coefficient to sale (0.77). Hence, it will be excluded from the model due to multicollinearity.
Shipping mode_same_day (exclude)	Zero coefficient indicate it has no relationship to order return.
Shipping mode_second_ class (exclude)	Zero coefficient indicate it has no relationship to order return.
Shipping mode_standard_ class (include)	The shipping mode indicates a week negative relationship to order return (-0.01). It suggests a slight downward trend
Segment_corporate(include)	The correlation coefficient of 0.01 suggest a week positive relationship. It suggests a slight upward trend.
Segment_home_office(include)	The shipping mode indicates a week negative relationship to order return (-0.01). It suggests a slight downward trend
Order_priority_High(exclude)	Zero coefficient indicate it has no relationship to order return.
Order_priority_Low(exclude)	Zero coefficient indicate it has no relationship to order return.
Order_priority_Medium	The shipping mode indicates a week negative relationship to order return (-0.01). It suggests a slight downward trend

The approach in creating training and testing dataset

Firstly, all the columns that have been proved have no relationship to the target variable will be removed. This will also consist of removing multicollinearity. All of the columns that will be removed are: Shipping_cost, Shipping mode_same_day, Shipping mode_second_ class, Order_priority_High, Order_priority_Low

Secondly, all of the data will be divided testing and training dataset with 20% of the data for testing and 80% for training, with the random state of 42 to help machine learning control the randomness and ensure that the results are reproducible.

TRAINING 2 ML MODELS

Coding

```
import pandas as pd

import numpy as np
```

```
import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

from sklearn.metrics import mean_absolute_error, mean_squared_error

from sklearn import preprocessing

from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

import matplotlib.pyplot as plt

# Load the data

data = pd.read_csv("/content/golabl_store.csv")

print(data)

# Check for missing values

missing_values = data.isna()

print(missing_values.any().any())

# Remove rows with missing values

data = data.dropna()

# Print the data type of all variables

print(data.dtypes)

# Features selection and drop columns that are not useful for prediction or are identifiers

data = data.drop(['order_id', 'order_date', 'shipping_date', 'customer_id', 'product_id',
'product_name','category','sub_category','country','city','state','market','region'], axis=1)

# Convert categorical columns to 'category' dtype

data['returned'] = data['returned'].astype('category')

data['order_priority'] = data['order_priority'].astype('category')

data['shipping_mode'] = data['shipping_mode'].astype('category')

data['segment'] = data['segment'].astype('category')

# Convert categorical variables to dummy/indicator variables

data = pd.get_dummies(data, drop_first=True)

print(data)

# Split the data into inputs (X) and outputs (y)
```

```
X = data.drop(columns=['returned_Yes'])

y = data['returned_Yes']


# Scale the numeric input data

# Identify numeric columns

numeric_columns = X.select_dtypes(include=['float64', 'int64']).columns


# Scale numeric columns

scaler = preprocessing.StandardScaler().fit(X[numeric_columns])

X[numeric_columns] = scaler.transform(X[numeric_columns])

# Compute the correlation matrix

corr_matrix = pd.concat([X, y], axis=1).corr()

# Plot the heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')

plt.title('Correlation Matrix Heatmap')

plt.show()

#Drop insignificant variables

X = X.drop(columns=['order_priority_Low','order_priority_High','shipping_mode_Second
Class','shipping_mode_Same Day','shipping_cost'])

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Logistic regression:

```
# Train a logistic regression model on the training data

model = LogisticRegression()

model.fit(X_train, y_train)

# Make predictions on the testing data

y_pred = model.predict(X_test)

# Calculate the accuracy of the model
```

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

print("Logistic Regression", classification_report(y_test, y_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

Random Forest

from sklearn.ensemble import RandomForestClassifier

 # Create the Random Forest model

Random_forest_model = RandomForestClassifier(random_state=42)

 # Train the model

Random_forest_model.fit(X_train, y_train)

Training machine learning is crucial for understanding the pattern in historical data and making accurate predictions. Succeeds in training a good model will help global to predict the likelihood of returns that can aid in optimizing inventory, enhancing customer satisfaction, and increasing operational efficiency. Logistic regression and random forest are chosen for making predictions due to its appropriateness to counter data variance, bias, and large dataset.

Steps involved in creating Logistic Regression

1) **Data preparation**
   Firstly, load the dataset and conduct data preprocessing task. These include remove missing value, and encode categorical variables.

   Secondly, features selection by using heat map would ensure that only relevant features are selected which will increase the likelihood of making accurate predictions.

   Lastly, identify numeric columns and use standardization methods to convert the number into one

   comparable unit

2) **Splitting X and Y for training Data**

   This step would separate the dataset into features (X) and the target variable (Y) which represents for returned order from client. Additionally, data will be separated 20% for testing and 80% for training to maintain same class distribution as the original dataset. Lastly, data will be fitted into the model for training.

3) **Model training**

Logistic regression:

Create a logistic regression model with the random state 42 for reproducibility. Training the model using the cleaned data.

Random Forest:

Create a random forest classifier and fit the cleaned data into model for training.

4) **Evaluation the output of the models**

In this step, classification report are generate to measure the accuracy of both models. The model accuracy can be assessed by using several key metrics such as precision, call, F1-score, and accuracy by using confusion matrix. A high precision for returned orders indicates the model's effectiveness in predicting returns order correctly.

TESTING AND EVALUATING 2 ML MODELS

Coding

1) **Logistic regression model**

```
# Make predictions on the testing data
y_pred = model.predict(X_test)
# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Logistic Regression", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

2) **Random Forest model**

```
# Predict on the test data
y_pred = Random_forest_model.predict(X_test)
y_pred_prob = Random_forest_model.predict_proba(X_test)[:, 1]
# Evaluate the model
print("Random Forest", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
roc_auc = roc_auc_score(y_test, y_pred_prob)
accuracy = accuracy_score(y_test, y_pred)
```

print("Accuracy:", accuracy)

```
Accuracy: 0.9420939754338078
Logistic Regression              precision    recall  f1-score   support

       False       0.94      1.00      0.97      9664
        True       0.00      0.00      0.00       594

    accuracy                           0.94     10258
   macro avg       0.47      0.50      0.49     10258
weighted avg       0.89      0.94      0.91     10258

Confusion Matrix:
 [[9664    0]
 [ 594    0]]
ROC-AUC Score: 0.5042453369233171
```

```
Random Forest                precision    recall  f1-score   support

       False       0.94      0.99      0.97      9664
        True       0.10      0.01      0.02       594

    accuracy                           0.94     10258
   macro avg       0.52      0.50      0.49     10258
weighted avg       0.89      0.94      0.91     10258

Confusion Matrix:
 [[9604   60]
 [ 587    7]]
Accuracy: 0.9369272762721779
ROC-AUC Score: 0.5606787905266797
```

Testing Process

Logistic Regression and Random Forest are used to make predictions on the test data that was set aside during the data splitting phase. This evaluation on unseen data helps to assess how well the models generalize to new instances. Both logistic regression and random forest have the same steps from data preprocessing, splitting X and Y features, and standardize the features. The major differences in these two-model testing process are that logistic regression would use logistic regression function, and random forest would have their own command to predict the Y value. After all, data will be tested by using classification report, confusion matrix, ROC-AUC score, and accuracy to measure the precision of each model.

Evaluation and comparison between the two ML models

Logistic regression is a parametric model that use probabilities to predict value. Moreover, it involves finding the best-fitting coefficients that minimize the difference between the predicted and actual values. Additionally, it has high interpretability, good precision and recall for non-returns, fast training. The accuracy rate for logistic regression model is extremely high with up to 94,2%. The major weakness of this model is that it struggles with predicting returns due to low precision and recall for returns. Also, the ROC-AUC score of 0.504 indicates that the model's performance is close to random guessing. Logistic Regression has a very low recall and F1-score for the True class, making it unsuitable for scenarios where identifying positive cases is crucial.

On the other hand, random forest involves training multiple decision trees on different subsets of the data. Each tree is built differently by splitting the data based on feature values that maximize the separation between classes. The decision with most votes will be the result of the model. Random forest is very effective in eliminating potential bias. Moreover, it has a higher accuracy (93.7%) and performs slightly better in identifying positive cases compared to Logistic Regression, of which it can recognize 10% of the positive case.

Justifications of the selection of the best performing ML model

All in all, both models have similar accuracy (~94%), but they struggle with the minority class (True). This indicates that they are struggling to predict order return correctly. Logistic Regression has a very low recall and F1-score for the True class, making it unsuitable for scenarios where identifying positive cases is crucial. Random Forest, while also struggling, performs slightly better in identifying positive cases with a better ROC-AUC score of 0.561 compared to Logistic Regression 0.504. Hence, random forest is clearly a better performing ML model

DISCUSS THE FINDING AND DERIVE ACTIONABLE INSIGHT

Coefficient for logistic regression

	Feature	Coefficient
0	sales	-0.023044
1	quantity	0.156469
2	discount	-0.174310
3	profit	0.010080
4	shipping_mode_Standard Class	-0.085897
5	segment_Corporate	0.040336
6	segment_Home Office	-0.115051
7	order_priority_Medium	-0.040665

Firstly, each sale tends to decrease 0.008 the likelihood that order can be returned. Therefore, it is recommended that Global should closely monitor and manage inventories for all the best-seller items to ensure accuracy and quality, which can lead to decrease the likelihood of returns

Secondly, quantity shows a positive correlation of 0.156. This figure suggest that larger quantities are slightly associated with more returns. Customer may order defecting items and incorrect order. As a result, they would return the order due to dissatisfaction. Therefore, Global should aim to take extra precaution for high quantity order by confirming with customer before dispatching the order. Moreover, they should inspect the product with extra attention to minimise the product defect possibility.

Thirdly, discount shows strong negative correlations of up to -0.17. This indicates that each increase in the discount rate would result in -0.17 less order returned. Higher discounts potentially make customers feel they received a good deal. As a result, they wouldn't return the order because returning the order meaning that they would lose the discount. Hence, global can minimise order returned by using discount strategically, especially on high-return rate order, or defect items.

Fourthly, the profit coefficient of 0.0108 shows a minimal positive relationship with returns. It shows that increase in profit margin may lead to high product return. This may be due to Global have maximised its profit by cutting some of the essential expense which led to customer dissatisfaction. Hence, global should revise its policies to ensure that they can balance between profit and customer satisfaction which potentially reduce the customer return rate

Fifthly, the standard class shipping mode has the coefficient of -0.085897 indicates a negative relationship with order returns. This suggests that orders shipped via Standard Class are less likely to be returned. Standard Class shipping might meet customer expectations better, or customers choosing this mode may have lower expectations regarding speed, leading to fewer returns. Hence, global can minimise order return by focus on standard class shipping method

Sixthly, for customer segments, corporate customers show a slight positive relationship with returns (0.040336), indicating a higher likelihood of returns, potentially due to stricter quality requirements or policies. In contrast, the Home Office segment has a negative coefficient (-0.115051), suggesting fewer returns, likely due to higher satisfaction or flexible return processes.

Thus, Global should pay extra attention to segment corporate to minimise order return. It can be done by implementing strict quality of control for all the order placed by corporate segments. Moreover, they should offer dedicated support and tailored solutions to corporate customers to reduce the likelihood of returns.

Lastly, Medium order priority often result in less customer return, with the negative coefficient of -0.04. This indicate that medium priority orders represent a balanced handling process, avoiding the rush of high priority and the delays of low priority, leading to fewer issues and returns. Hence, global should maintain efficient processes for handling medium priority orders, as they are associated with fewer returns.

ANALYSIS OF UNSTRUCTURED DATA: DETECTING E-WASTE

RATIONALE OF THE CHOICE OF ML ALGORITHM

Global should aim to use clustering ML algorithms to accurately identify damaged returned items as e-waste, which will improve its reverse logistics operations and ensure proper disposal of e-waste.

Firstly, clustering algorithms are particularly well-suited for image recognition and classification tasks due to their ability to automatically and adaptively learn spatial hierarchies of features from input images as well as learning pattern from these images. Identifying damaged items involves analysing images of returned products, so clustering is the most appropriate choice of ML models to recognize the pattern between waste and e-waste items.

Secondly, clustering algorithms excel in feature extraction, which is crucial for identifying specific types of damage in returned items. By using cluster all images with same characteristic, it can detect edges, textures, and more complex patterns that signify damage. Moreover, techniques like VGG16 and ResNet, which are based on CNN architectures, have demonstrated high accuracy in extracting relevant features from images and performing classification tasks.

Thirdly, clustering can handle large volumes of unstructured data efficiently, making them scalable for Global Store potentially high number of returned items. The current global's data set is comprised of various

unstructured data. Hence, using clustering will help global to identify the features between waste and e-waste without having to train model using label data.

Fourthly, the hierarchical nature of clustering enables them to recognize complex and subtle forms of damage that might be missed by simpler algorithms. This ability is crucial for ensuring that all types of e-waste are correctly identified, preventing damaged items from being mistakenly restocked.

DATA TRANSFORMATION AND THE SELECTION OF MODEL PARAMETERS

Feature extraction steps

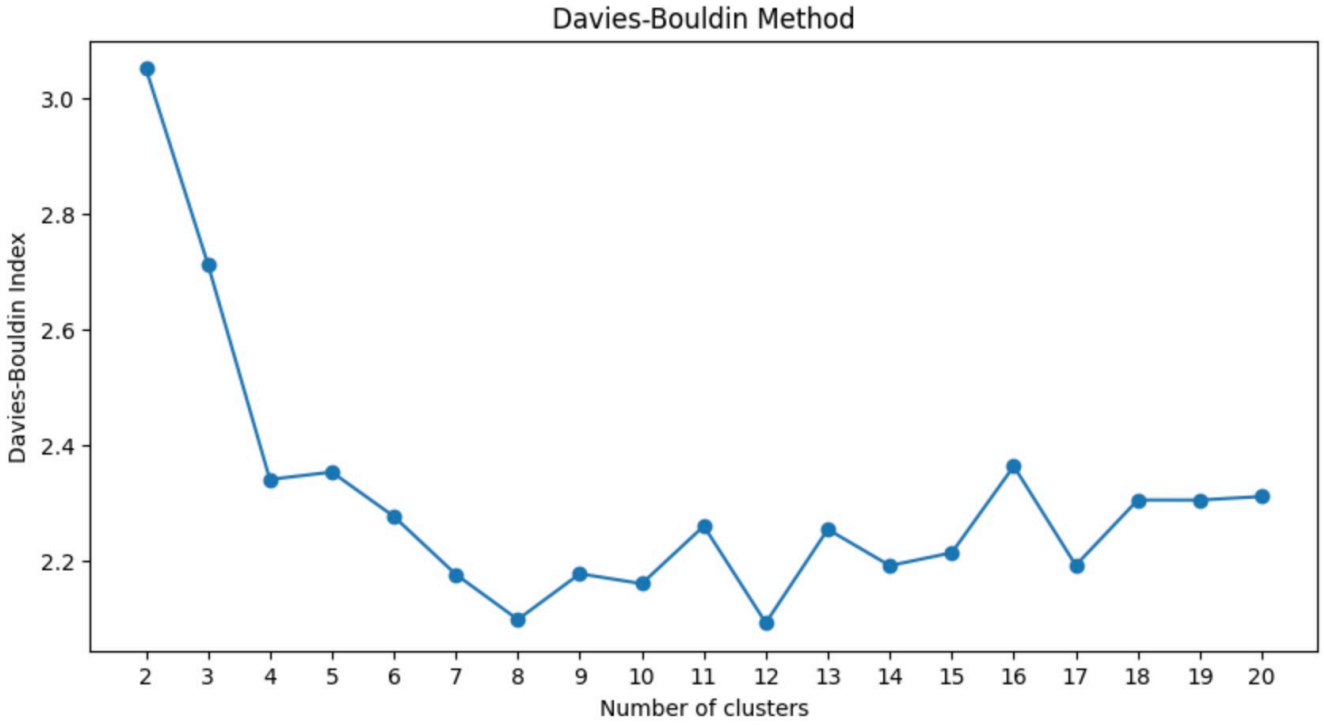
Firstly, loading the images into the platform for analysis. After successfully loading all the images, each image will be resized to 150x150 pixels for consistency. This process is handled by the load_images_from_folder function, which reads each image file, resizes it, and appends it to a list.

Secondly, VGG 16 with the pre-train on the ImageNet dataset are used for features extraction. Precisely, this model is known for its deep convolutional architecture, which is effective in capturing high-level features from images. Moreover, VGG 16 is highly effective to obtain feature maps. These feature maps represent the high-level features extracted by the model. The deeper that it can go through each layer, the more likely that important features will be identified, which later be used to classify e-waste.

Thirdly, the feature maps obtained from VGG16 are multidimensional arrays. To make these features suitable for clustering, they need to be flattened into a 2D array where each row represents an image, and each column represents a feature.

Lastly, principal Component Analysis (PCA) is applied to reduce the dimensionality of the feature set. This step is highly effective when dealing with high-dimensional data, to reduce computational complexity and remove noise. The number of components is set to 50, which is a balance between retaining important information and reducing dimensionality.

Selection of model parameters



Firstly, Davies Bouldin index was used to determine the optimal number of clusters. DBI evaluates clustering quality based on the average similarity ratio of each cluster with its most similar cluster, where a lower DBI indicates better clustering. Hence, 8 cluster will be used to perform k-mean clustering.

Secondly, the DBI plot has revealed that the optimal number of clusters is identified as the one with the lowest DBI score. In this example, the optimal number of clusters will be 8 for the K-Means algorithm. K-Means clustering is then performed on the reduced feature set to assign each image to a cluster.

IMPLEMENTATION OF ML MODELS

Coding

 # Load the libraries needed

```
import os

import cv2

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.keras.applications import VGG16

from tensorflow.keras.applications.vgg16 import preprocess_input

from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.metrics import davies_bouldin_score
import shutil
# Load images from the folder
def load_images_from_folder(folder):
    images = []
    filenames = []
    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder, filename))
        if img is not None:
            img = cv2.resize(img, (150, 150))
            images.append(img)
            filenames.append(filename)
    return np.array(images), filenames
    # Load images
    folder_path = "/content/"
    images, filenames = load_images_from_folder(folder_path)
    # Feature extraction using pre-trained VGG16 model
    def extract_features(images):
        model = VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
        model.trainable = False
        features = model.predict(preprocess_input(images))
        return features
    # Flatten the extracted features
    def flatten_features(features):
        return features.reshape(features.shape[0], -1)
    # Extract features
    features = extract_features(images)
```

```
#davies boulding
def davies_bouldin_method(features, max_clusters=20):
    davies_bouldin_scores = []
    for n_clusters in range(2, max_clusters + 1)
        kmeans = KMeans(n_init=10, n_clusters=n_clusters, random_state=42)
        labels = kmeans.fit_predict(features)
        db_score = davies_bouldin_score(features, labels)
        davies_bouldin_scores.append(db_score)
    plt.figure(figsize=(10, 5))
    plt.plot(range(2, max_clusters + 1), davies_bouldin_scores, marker='o')
    plt.xlabel('Number of clusters')
    plt.ylabel('Davies-Bouldin Index')
    plt.title('Davies-Bouldin Method')
    plt.xticks(range(2, max_clusters + 1))
    plt.show()
#step3
# Flatten features
flattened_features = flatten_features(features)
#Step 4
# Apply PCA for dimensionality reduction
pca = PCA(n_components=50)
reduced_features = pca.fit_transform(flattened_features)
#step5
#  Identify the optimal number of clusters
davies_bouldin_method(reduced_features)
# Perform k-means clustering
n_clusters = 8
kmeans = KMeans(n_clusters, n_init=10, random_state=42)
kmeans.fit(reduced_features)
```

```
labels = kmeans.labels_

# Visualise the clusters

for cluster in range(n_clusters):

    cluster_images = images[labels == cluster]

    plt.figure(figsize=(20, 20))

    for i in range(min(len(cluster_images), 25)):

        plt.subplot(5, 5, i + 1)

        plt.imshow(cluster_images[i])

        plt.axis('off')

    plt.suptitle(f'Cluster {cluster + 1}')

    plt.show()
```

There are seven steps in implementing clustering ML models.

The first and second step is data preprocessing involves image loading and resize them to 150x150 pixels. Moreover, they must be converted to arrays and pre-processed to be compatible with the VGG16 model. VGG 16 with pre-trained on ImageNet will then be used to extract features from each image. After capturing all the essential features, the top fully connected layers are excluded to get the feature maps from the convolutional layers.

The third step involves flattening all the extracted features into a 2D array. This step is important because it is a pre-requisite to conducting K-means analysis, of which it expects the input data to be in a 2D array format where each row represents a data point, and each column represents a feature. By flattening the extracted features, we ensure that the input is in the correct format for these algorithms to process.

The fourth step involves applying PCA for dimensionality reduction. Precisely, the Principal Component Analysis (PCA) reduces the dimensionality of the feature vectors to 50 components. This helps in speeding up the clustering process and making the clusters more distinct.

The fifth step involves identifying the optimal number of clusters using Davis Bouldin method. A DBI plot were created to aid in finding optimal number of clusters by providing a measure of how well the clusters are separated. By calculating the Davies-Bouldin score for different numbers of clusters, 8 clusters are selected because it has lowest number of DBI index, indicating the best balance between cluster separation and compactness.

The sixth step involves performing K-mean clustering. Precisely, k mean is performed on the reduced feature vectors. For each number of clusters, a K-Means model is initialized and fitted to the data. Moreover, n_init=10 specifies the number of times the K-Means algorithm will be run with different centroid seeds. The random state 42 will ensure that all the result are reproducible.

Lastly, all the images were clustered into eight group and each cluster may consist of waste and e-waste items.

ANALYSIS FINDING AND ACTIONABLE INSIGHT

The clustering algorithm successfully categorized the images into distinct clusters based on visual features. The clusters predominantly separated different types of waste and e-waste items, such as electronics, furniture, and appliances.

From the clustering model, we can withdraw that cluster 1,3,5,7,8 will be e-waste while cluster 4, and 6 will be classified as general waste. However, cluster 2 has failed to correctly classify waste and e-waste items because the table images look identical to keyboard when capturing from a close distance that led to model error. Hence, Global should pay greater care when taking product image because recording product image incorrectly may lead to model incorrectly classify the products.

Overall, the model has proved its efficiency in classifying waste and e-waste with great accuracy. It is recommended that Global should replicate this ML models, with greater dataset for training. Precisely, they can enhance the model accuracy by incorporating additional pre-processing steps or using different pre-trained models. This could help in distinguishing subtle differences between categories that are visually similar. Moreover, this could help to avoid the mistake that cluster 2 made, of which tables look identical to keyboard when capturing from a short distance.

Furthermore, they can improve the data representation by enhancing the dataset with more labelled examples, particularly for items that are visually similar but belong to different categories. This can improve the model ability to learn distinguishing features. Global can improve the data quality by collecting more labelled images of waste and e-waste, especially focusing on items that were misclassified (table and keyboard in cluster 2) and use data augmentation techniques to increase the diversity of the training set. This will not only improve the model accuracy when classifying waste and e-waste in the future but help Global to save time, and cost.

CONCLUSION

In conclusion, this report has clearly analysed how different features affect order return and provided with recommendations on how to minimise order return from customer. Moreover, Clustering ML algorithms was recommended to Global to correctly classify waste, and e-waste items along with several methods on how to improve the accuracy of the model.

APPENDIX

Full coding for part 1:

```
import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
from sklearn import preprocessing

from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

import matplotlib.pyplot as plt

# Load the data

data = pd.read_csv("/content/golabl_store.csv")

print(data)

# Check for missing values

missing_values = data.isna()

print(missing_values.any().any())

# Remove rows with missing values

data = data.dropna()

# Print the data type of all variables

print(data.dtypes)

# Features selection and drop columns that are not useful for prediction or are identifiers

data = data.drop(['order_id', 'order_date', 'shipping_date', 'customer_id', 'product_id',
'product_name','category','sub_category','country','city','state','market','region'], axis=1)

# Convert categorical columns to 'category' dtype

data['returned'] = data['returned'].astype('category')

data['order_priority'] = data['order_priority'].astype('category')

data['shipping_mode'] = data['shipping_mode'].astype('category')

data['segment'] = data['segment'].astype('category')

# Convert categorical variables to dummy/indicator variables

data = pd.get_dummies(data, drop_first=True)

print(data)

# Split the data into inputs (X) and outputs (y)

X = data.drop(columns=['returned_Yes'])

y = data['returned_Yes']


# Scale the numeric input data

# Identify numeric columns
```

```
numeric_columns = X.select_dtypes(include=['float64', 'int64']).columns


# Scale numeric columns

scaler = preprocessing.StandardScaler().fit(X[numeric_columns])

X[numeric_columns] = scaler.transform(X[numeric_columns])

# Compute the correlation matrix

corr_matrix = pd.concat([X, y], axis=1).corr()

# Plot the heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')

plt.title('Correlation Matrix Heatmap')

plt.show()

#Drop insignificant variables

X = X.drop(columns=['order_priority_Low','order_priority_High','shipping_mode_Second
Class','shipping_mode_Same Day','shipping_cost'])

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train a logistic regression model on the training data

model = LogisticRegression()

model.fit(X_train, y_train)

# Make predictions on the testing data

y_pred = model.predict(X_test)

# Calculate the accuracy of the model

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

print("Logistic Regression", classification_report(y_test, y_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

#Coefficients

coefficients = pd.DataFrame({

    'Feature': X.columns,
```

```
    'Coefficient': model.coef_[0]
})
print(coefficients)


from sklearn.ensemble import RandomForestClassifier
# Create the Random Forest model
Random_forest_model = RandomForestClassifier(random_state=42)
# Train the model
Random_forest_model.fit(X_train, y_train)
# Predict on the test data
y_pred = Random_forest_model.predict(X_test)
y_pred_prob = Random_forest_model.predict_proba(X_test)[:, 1]
# Evaluate the model
print("Random Forest", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
roc_auc = roc_auc_score(y_test, y_pred_prob)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("ROC-AUC Score:", roc_auc)
```

Full coding for part 2

```
# Load the libraries needed
import os
import cv2
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
```

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.metrics import davies_bouldin_score
import shutil
# Load images from the folder
def load_images_from_folder(folder):
    images = []
    filenames = []
    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder, filename))
        if img is not None:
            img = cv2.resize(img, (150, 150))
            images.append(img)
            filenames.append(filename)
    return np.array(images), filenames
    # Feature extraction using pre-trained VGG16 model
    def extract_features(images):
        model = VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
        model.trainable = False
        features = model.predict(preprocess_input(images))
        return features
    # Flatten the extracted features
    def flatten_features(features):
        return features.reshape(features.shape[0], -1)
    #step2
    # Extract features
    features = extract_features(images)
#Davies boulding
```

```
def davies_bouldin_method(features, max_clusters=20):
    davies_bouldin_scores = [] # Initialize an empty list to store Davies-Bouldin scores for different cluster numbers.
    for n_clusters in range(2, max_clusters + 1): # Loop through Cluster Numbers
        kmeans = KMeans(n_init=10, n_clusters=n_clusters, random_state=42)
        labels = kmeans.fit_predict(features)
        db_score = davies_bouldin_score(features, labels)
        davies_bouldin_scores.append(db_score)
    plt.figure(figsize=(10, 5))
    plt.plot(range(2, max_clusters + 1), davies_bouldin_scores, marker='o')
    plt.xlabel('Number of clusters')
    plt.ylabel('Davies-Bouldin Index')
    plt.title('Davies-Bouldin Method')
    plt.xticks(range(2, max_clusters + 1))  # Set x-axis to show only integers
    plt.show()
```

#step3

 # Flatten features

```
flattened_features = flatten_features(features)
```

#Step 4

 # Apply PCA for dimensionality reduction

```
pca = PCA(n_components=50)
reduced_features = pca.fit_transform(flattened_features)
```

#step5

 # Identify the optimal number of clusters

```
davies_bouldin_method(reduced_features)
```

#step5

 # Perform k-means clustering

```
n_clusters = 8
kmeans = KMeans(n_clusters, n_init=10, random_state=42)
kmeans.fit(reduced_features)
```

```
labels = kmeans.labels_

# Visualise the clusters

for cluster in range(n_clusters):
    cluster_images = images[labels == cluster]
    plt.figure(figsize=(20, 20))
    for i in range(min(len(cluster_images), 25)):
        plt.subplot(5, 5, i + 1)
        plt.imshow(cluster_images[i])
        plt.axis('off')
    plt.suptitle(f'Cluster {cluster + 1}')
    plt.show()
```

Cluster image

Cluster 1:

Cluster 1

Cluster 2

Cluster 2

Cluster 3:

Cluster 3

Cluster 4:

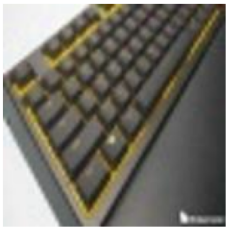
Cluster 5:

Cluster 6:

Cluster 6

Cluster 7:

















SANYO





Cluster 8:

Cluster 8