CAPSTONE PROJECT
Shipping Logistic Analysis

(Asyrof Wajdi Bin muhamad nidzar)

1 Business Understanding

1.1 Background

Logistic companies are currently growing rapidly. To keep the company's performance constantly increasing. Customer satisfaction needs to be taken into account.

Major factor to ensure customer satisfaction is that each parcel needs to be delivered on time and there are several factors that affect on time delivery. If logistic company not able to deliver customer's parcel on time, then the customer will lost of interest and give a negative impact to the company such as financial loss and loss of trust. This issues is the main objective for the company to solved it and improve. According to research , accuracy is one of the criteria that customers were satisfied most. (Wan Ahmad, W. N. K., Shamsuddin, A., & Tham, J. H. , 2021).

In order to identify whether a shipment will reach on time or not, a classification model was built to predict whether a shipment will reach on time or not from several factor such as mode of transport and so on, so that the important result will be used by logistic department, customer services and higher management to improve.
Model performance will be assessed using accuracy. The goal is to achieve an accuracy of at least 70%

The project is appropriately scoped as it involves building a classification model using a dataset with 10,999 entries and a manageable number of features. This scope is neither too aggressive nor too easy.

To predict whether shipments will arrive on time or not is valuable for logistics companies to improve their service, customer satisfaction, and operational efficiency. The insights can help in identifying factors that contribute to delays and optimizing the shipping process.

The project will be completed within three weeks, including data preprocessing, model development, evaluation, and reporting.

Picture 1: The model will be guided based on CRISP-DM format

1.2 Data Mining Goals.

Model performance will be assessed using accuracy. The goal is to achieve an accuracy of at least 70%.

2 Data Understanding

On the Data Understanding phase, we will gather, describe and explore the data to make sure it fits the business goal.

2.1 Gathering and Describing Data

Data are collected from Kaggle.com in tabular format. The data consists of factors that interact with reached on time of the parcel. For target column which is Reached on time column, the number for 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time. It is our target variable. The collected data consists of 10,999 distinct customers with 11 variables. The description of each column/variable can be seen below:.

Here are some samples of the data:

ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
1	D	Flight	4	2	177	3	low	F	44	1233	1
2	F	Flight	4	5	216	2	low	M	59	3088	1
3	A	Flight	2	2	183	4	low	M	48	3374	1
4	B	Flight	3	3	176	4	medium	M	10	1177	1
5	C	Flight	2	2	184	3	medium	F	46	2484	1

Table 1: The data show sample for dataset

Columns names:

1. ID: It is a numerical column. Each product has its own unique id.
2. Warehouse_block:: It is a categorical column. The Company have big Warehouse which is divided in to block such as A,B,C,D,E.
3. Mode of shipment: It is a categorical column. The Company Ships the products in multiple way such as Ship, Flight and Road.
4. Customer care calls: It is a numerical column. The number of calls made from enquiry for enquiry of the shipment.
5. Customer rating: It is a numerical column. The company has rated from every customer. 1 is the lowest (Worst), 5 is the highest (Best).
6. Cost of the product: It is a numerical column. Cost of the Product in US Dollars.
7. Prior purchases: It is a numerical variable. The Number of Prior Purchase.
8. Product importance: It is a categorical column. The company has categorized the product in the various parameter such as low, medium, high.
9. Gender: It is a categorical column. Male and Female.
10. Discount offered: It is a numerical column. Discount offered on that specific product.
11. Weight in gms: It is a numerical column. It is the weight in grams.
12. Reached on time: It is a numerical column. It is the target variable, **where 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time. It is our target variable.**

2.2 Data Exploration

First before we start , we need to clean the data and since some of the column/variable is categorical, we can check the summary of the data and see the number of customer of each categories. By doing this, we can also check whether there are any data that need to be cleansed or to be transformed. For example, we can check if there is a missing/empty values.

```
: 1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   10999 non-null  int64
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64
 4   Customer_rating      10999 non-null  int64
 5   Cost_of_the_Product  10999 non-null  int64
 6   Prior_purchases      10999 non-null  int64
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64
 10  Weight_in_gms        10999 non-null  int64
 11  Reached.on.Time_Y.N  10999 non-null  int64
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

Figure 1: The info shows the total number of row and column also with type of variables

```
1 df.isnull().sum()
```

```
ID                   0
Warehouse_block      0
Mode_of_Shipment     0
Customer_care_calls  0
Customer_rating      0
Cost_of_the_Product  0
Prior_purchases      0
Product_importance   0
Gender               0
Discount_offered     0
Weight_in_gms        0
Reached.on.Time_Y.N  0
dtype: int64
```

Figure 2: The data shows no null value

To filter the data, we also check for total unique values in certain rows so that that values should look relevant for example “reached on time” column should have only 2 unique values as this column only represent ‘1’ and ‘0’ where number 1 for Indicates that the product has NOT reached on time and 0 indicates it has reached on time

```
# Unique values in each columns
for i in df.columns:
    print(i,':',df[i].nunique())
```

```
ID : 10999
Warehouse_block : 5
Mode_of_Shipment : 3
Customer_care_calls : 6
Customer_rating : 5
Cost_of_the_Product : 215
Prior_purchases : 8
Product_importance : 3
Gender : 2
Discount_offered : 65
Weight_in_gms : 4034
Reached.on.Time_Y.N : 2
```

Figure 3: The picture show unique values in each column

```
# check the number of rows and column in dataframe
df.shape
```

```
(10999, 12)
```

Figure 4: The picture show the row and column in data frame

3 Data Preparation

On the Data Understanding phase, the data will be prepared and cleanse so the data are fit for analysis and making prediction.

3.1 Data Visualization

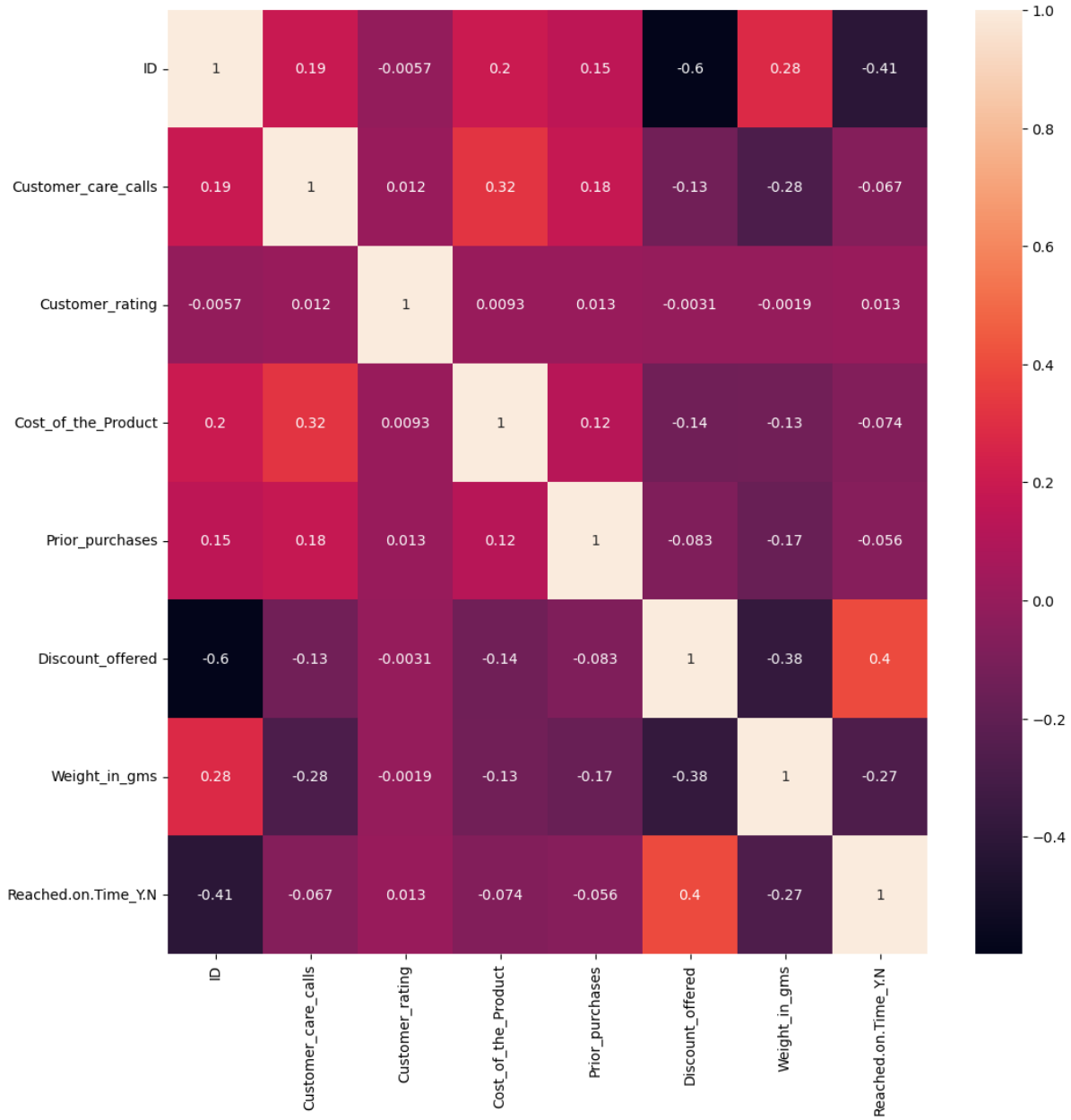


Table 2: The picture show the correlation heatmap for dataset.

From the correlation matrix we can tell that weight of the product is the most affected the reached on time delivery

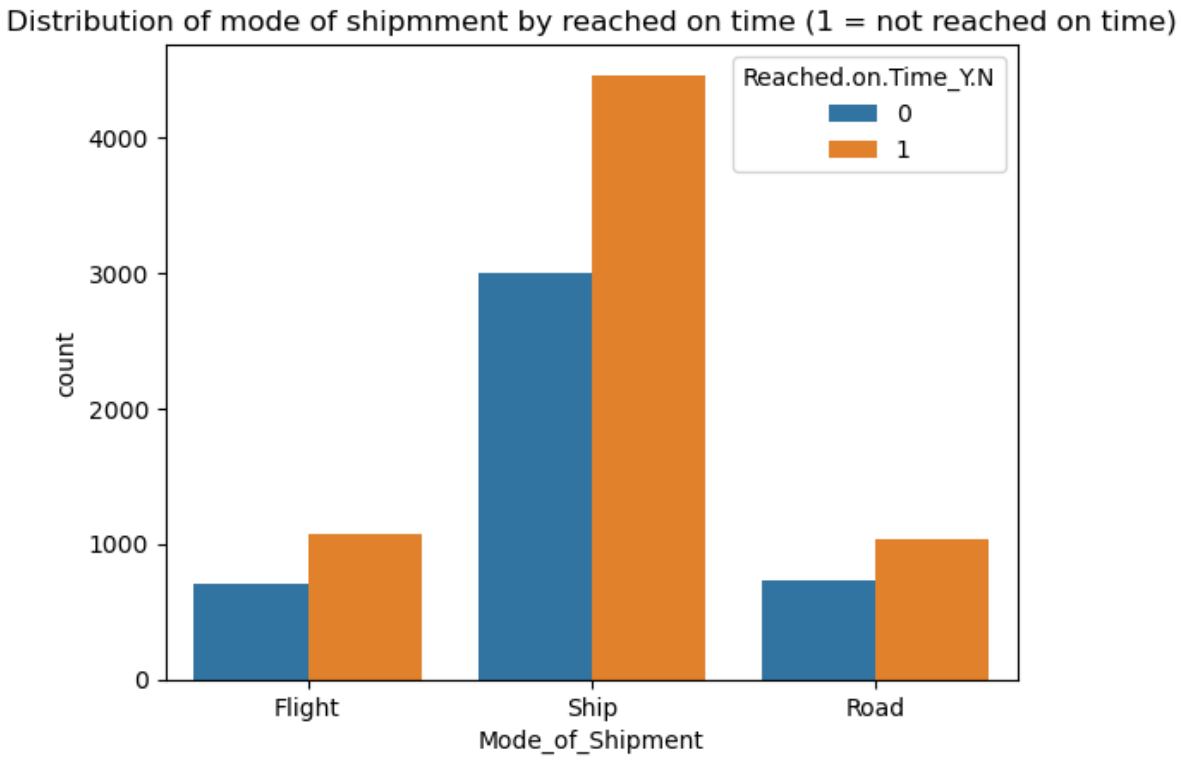


Figure 5: The picture show the graph for mode of shipment.

The chart show that shipping by sea is the highest count for delivery not reached on time compare to using flight and road

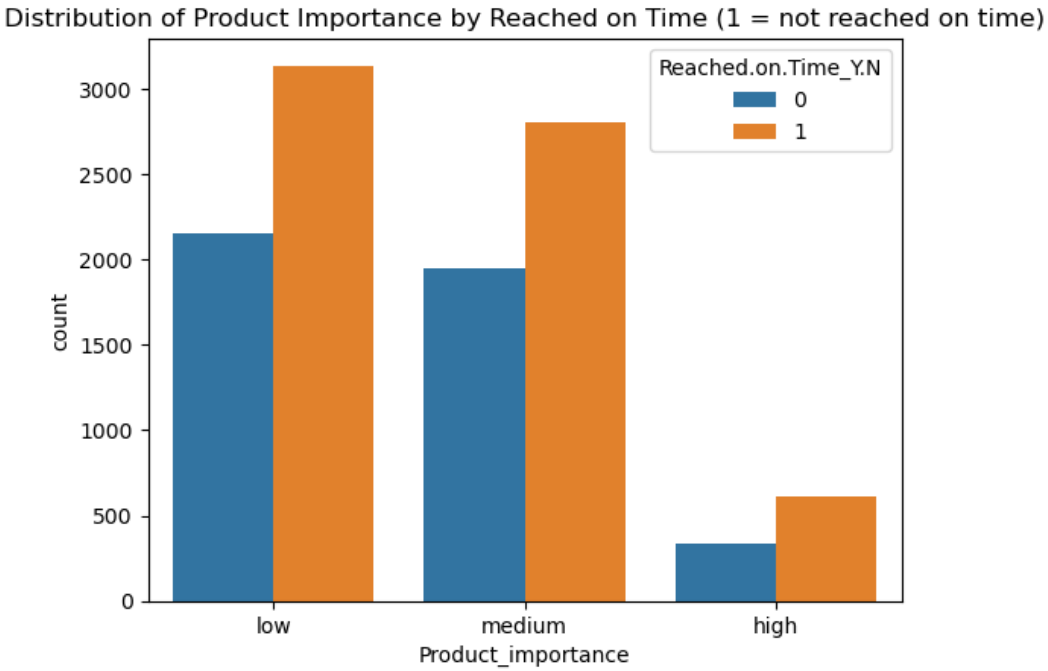


Figure 6: The picture show the graph for product important.

The chart show that low product important is the highest count for delivery not reached on time compare to high important product

Figure 7: The picture show the graph for Customer Rating.

The chart show that 3 is the highest rating given by customer for not reached on time delivery

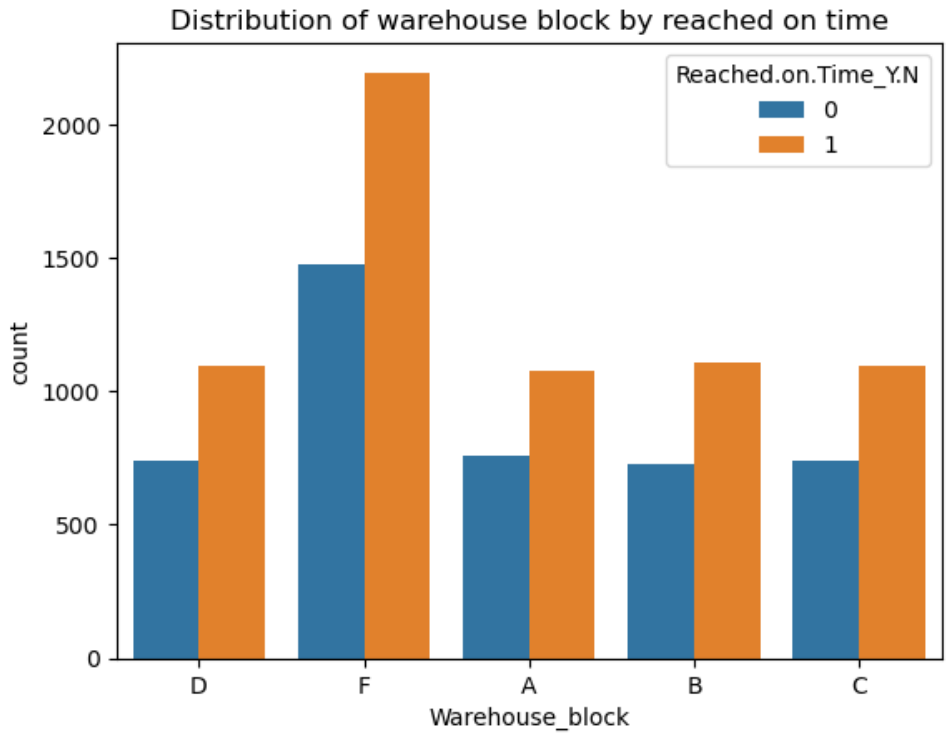


Figure 8: The picture show the graph for warehouse block.

The chart show that block F is the higest for not reached on time delivery and reached on time delivery count compare to other blocks

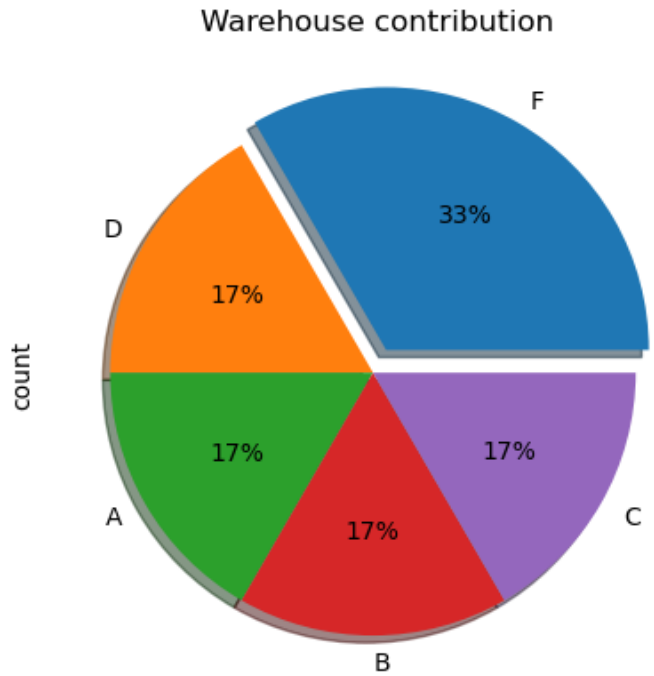


Figure 9: The picture show the pie chart for warehouse contribution

From the chart above , it shows that warehouse block F has the highest contribution for delivery count.

3.2 Dropping Column

The ID column dropped as this column is just a list of number and not important

Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
D	Flight	4	2	177	3	low	F	44	1233	1
F	Flight	4	5	216	2	low	M	59	3088	1
A	Flight	2	2	183	4	low	M	48	3374	1
B	Flight	3	3	176	4	medium	M	10	1177	1
C	Flight	2	2	184	3	medium	F	46	2484	1

Table 3: ID column removed

3.3 Interaction Term

Using interaction terms can result in a better fit to the data and better predictive performance. We can add interaction terms as a multiplication of the original features.

```
# perform iteration term
df['Cost_of_the_Product*Discount_offered'] = df['Cost_of_the_Product']*df['Discount_offered']
```

```
df['Cost_of_the_Product*Weight_in_gms'] = df['Cost_of_the_Product']*df['Weight_in_gms']
```

```
df['Discount_offered*Weight_in_gms'] = df['Discount_offered']*df['Weight_in_gms']
```

```
df.shape
```

```
(10999, 14)
```

```
df.describe()
```

scount_offered	Weight_in_gms	Reached.on.Time_Y.N	Cost_of_the_Product*Discount_offered	Cost_of_the_Product*Weight_in_gms	Discount_offered*Weight_in_gms
10999.000000	10999.000000	10999.000000	10999.000000	1.099900e+04	10999.000000
13.373216	3634.016729	0.596691	2703.287753	7.534369e+05	38632.819256
16.205527	1635.377251	0.490584	3299.980491	3.651938e+05	41283.062793
1.000000	1001.000000	0.000000	96.000000	1.035860e+05	1004.000000
4.000000	1839.500000	0.000000	768.000000	4.369080e+05	11368.000000
7.000000	4149.000000	1.000000	1442.000000	7.544370e+05	27205.000000
10.000000	5050.000000	1.000000	2521.000000	1.029308e+06	46359.000000
65.000000	7846.000000	1.000000	18590.000000	1.713964e+06	368832.000000

Figure 10 : The last 3 columns are the results of interaction term and the number of row increases

3.4 Dummification and Label Encoder

Dummification or one-hot encoding is a process that converts categorical variables into a series of binary columns. Each category is transformed into a new column, where the presence of the category is indicated by a 1, and the absence is indicated by a 0. This technique is useful when there is no ordinal relationship between categories.

Label encoding assigns a unique integer to each category of a categorical variable. It is useful for ordinal data where the categories have a natural order.

```
# Dummification
df=pd.get_dummies(df,columns=['Mode_of_Shipment','Product_importance','Gender'],drop_first=True, dtype=int)
```

```
df.head()
```

ght_in_gms	Discount_offered*Weight_in_gms	Mode_of_Shipment_Road	Mode_of_Shipment_Ship	Product_importance_low	Product_importance_medium	Gender_M
218241	54252	0	0	1	0	0
667008	182192	0	0	1	0	1
617442	161952	0	0	1	0	1
207152	11770	0	0	0	1	1
457056	114264	0	0	0	1	0

```
# LabelEncoder
le = LabelEncoder()
df['Warehouse_block']=le.fit_transform(df['Warehouse_block'])
```

```
df.head()
```

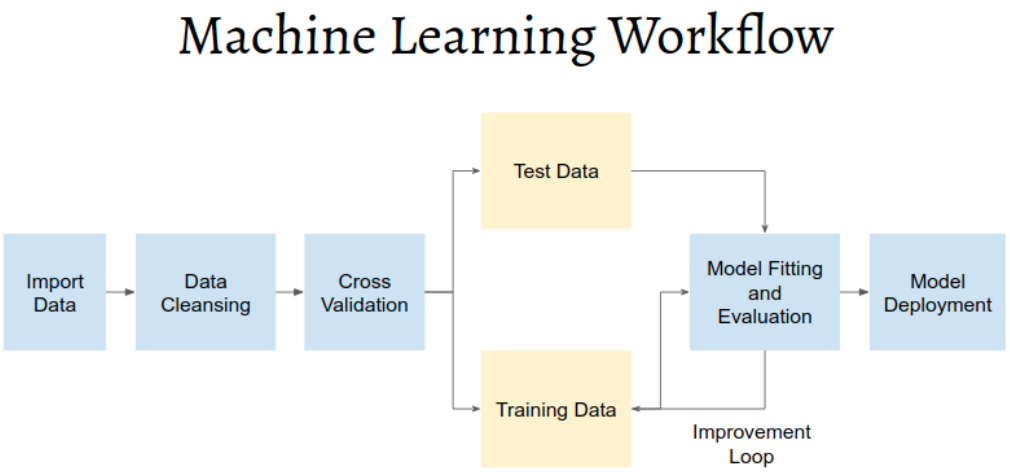
	Warehouse_block	ustomer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N	Cos
0	3	4	2	177	3	44	1233	1	
1	4	4	5	216	2	59	3088	1	
2	0	2	2	183	4	48	3374	1	
3	1	3	3	176	4	10	1177	1	
4	2	2	2	184	3	46	2484	1	

Figure 11: The picture show results for dummification and LabelEncoder

Finally, after careful and rigorous data cleansing, we acquire our final data that will be used for analysis and modeling.

4. Modeling

On the Modeling phase, we will start creating model to find pattern inside our data and to make future prediction for business purpose. Machine learning is a statistical model that is specifically designed to learn and find pattern from a data.



- Training Dataset: Dataset that will be used to train the machine learning model
- Testing Dataset: Dataset that will be used to evaluate the performance of the model.

Here we split the data with 80% of the data will be the training dataset and the rest will be the testing dataset. Each observation/row is randomly selected as either the training set or the testing set. The random selection is done to make sure we don't include any selection bias done by human.

4.1 Model and Machine Learning

There are 5 models that we are tested which are Logistic Regression, k-nearest neighbors (KNN), Decision Tree, Random Forest and lastly Support Vector Regression. Before start modelling or data processing. We must define X and y variables.

```
:  1  # Define X and y variables
   2  X=df.drop(['Reached.on.Time_Y.N'],axis=1)
   3  y=df['Reached.on.Time_Y.N']
```

Figure 12 : Above picture shows X and y variables.

```
# Split our data into training and testing sets.

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size = 0.2,
                                                    random_state = 42
                                                   )
```

Figure 13: Data splitting

The data that is used to train the model is called the Training Dataset

- Training Dataset: Dataset that will be used to train the machine learning model
- Testing Dataset: Dataset that will be used to evaluate the performance of the model.

Here we split the data with 80% of the data will be the training dataset and the rest will be the testing dataset. Each observation/row is randomly selected as either the training set or the testing set. The random selection is done to make sure we don't include any selection bias done by human.

4.2 Models Parameters

```
# Define the parameter grids for each model

param_grid_logreg = {
    'logreg__C':[10],
    'logreg__penalty': ['l1'],
    'logreg__solver': ['liblinear']
}

param_grid_knn = {
    'knn__n_neighbors':range(1, 51, 10),
    'knn__metric': ['euclidean', 'manhattan']
}

param_grid_dt = {
    'dt__max_depth': [5, 10, 20],
    'dt__min_samples_leaf': [1, 2, 4],
    'dt__criterion': ['gini', 'entropy']
}

param_grid_svm = {
    'svc__C': [0.1, 1, 10, 100]
}

param_grid_rf = {
    'rf__n_estimators': [100, 200],
    'rf__max_depth': [None, 10, 20],
    'rf__min_samples_leaf': [1]
}
```

Figure 14: Define Parameter

These hyperparameters are used to improve the learning of the model, and their values are set before starting the learning process of the model.

4.3 Models Pipelines

```
# Define pipelines for each model
pipeline_logreg = Pipeline([('ss', MinMaxScaler()), ('logreg', LogisticRegression())])
pipeline_knn = Pipeline([('ss', MinMaxScaler()), ('knn', KNeighborsClassifier())])
pipeline_dt = Pipeline([('ss', MinMaxScaler()), ('dt', DecisionTreeClassifier())])
pipeline_svm = Pipeline([('ss', MinMaxScaler()), ('svc', SVC())])
pipeline_rf = Pipeline([('ss', MinMaxScaler()), ('rf', RandomForestClassifier())])

```

Figure 15: Define Pipeline

In machine learning, a pipeline refers to a sequence of data processing steps and machine learning models arranged in a structured way. The purpose of a pipeline is to streamline and automate the workflow involved in training and evaluating models, standardize and streamline the process of building, training, evaluating and deploying machine learning models. making the process more efficient, repeatable, and less prone to errors

Pipelines will allow us to do two things:

1. Chain many transformers together before ending in an estimator.
2. Allow us to GridSearch over a transformer's hyperparameters.

4.4 Models GridSearchCV

GridSearchCV is a method in machine learning for optimizing hyperparameters of a model. It systematically works through multiple combinations of parameter values, cross-validating as it goes to determine which combination provides the best performance. In this case we use gridsearch for each model with cross validation of 4

```
# Perform Grid Search with Cross-Validation
grid_search_logreg = GridSearchCV(pipeline_logreg, param_grid_logreg, cv=4)
grid_search_knn = GridSearchCV(pipeline_knn, param_grid_knn, cv=4)
grid_search_dt = GridSearchCV(pipeline_dt, param_grid_dt, cv=4)
grid_search_svm = GridSearchCV(pipeline_svm, param_grid_svm, cv=4)
grid_search_rf = GridSearchCV(pipeline_rf, param_grid_rf, cv=4)
```

Figure 16: Perform GridsearchCV

4.5 Model Fitting

Here, we fit or train the model using the data train.

```
# Train and evaluate models
grid_search_logreg.fit(X_train, y_train)
```

```
grid_search_knn.fit(X_train, y_train)
```

```
grid_search_dt.fit(X_train, y_train)
```

```
grid_search_svm.fit(X_train, y_train)
```

```
grid_search_rf.fit(X_train, y_train)
```

Figure 17: Train and Evaluate

```
# Print the best parameters and accuracies
print(f'logreg Best Parameters: {grid_search_logreg.best_params_}')
print(f'logreg Accuracy: {accuracy_logreg:.3f}')
print()
print(f'KNN Best Parameters: {grid_search_knn.best_params_}')
print(f'KNN Accuracy: {accuracy_knn:.3f}')
print()
print(f'Decision Tree Best Parameters: {grid_search_dt.best_params_}')
print(f'Decision Tree Accuracy: {accuracy_dt:.3f}')
print()
print(f'Random Forest Best Parameters: {grid_search_rf.best_params_}')
print(f'Random Forest Accuracy: {accuracy_rf:.3f}')
print()
print(f'SVM Best Parameters: {grid_search_svm.best_params_}')
print(f'SVM Accuracy: {accuracy_svm:.3f}')
print()

```

```
logreg Best Parameters: {'logreg__C': 10, 'logreg__penalty': 'l1', 'logreg__solver': 'liblinear'}
logreg Accuracy: 0.655

KNN Best Parameters: {'knn__metric': 'manhattan', 'knn__n_neighbors': 31}
KNN Accuracy: 0.657

Decision Tree Best Parameters: {'dt__criterion': 'entropy', 'dt__max_depth': 5, 'dt__min_samples_lea
f': 1}
Decision Tree Accuracy: 0.690

Random Forest Best Parameters: {'rf__max_depth': 10, 'rf__min_samples_leaf': 1, 'rf__n_estimators': 1
00}
Random Forest Accuracy: 0.682

SVM Best Parameters: {'svc__C': 0.1}
SVM Accuracy: 0.672
```

Figure 18: Best parameters and Accuracy

Models Accuracy Score:

1. Logistic Regression Model : 65.5%
2. KNN Model : 65.7%
3. Decision Tree Model : 69%
4. Random Forest Model : 68.2%
5. Support Vector Model: 67.2%

Based on the models that we choose, the best model for this prediction is Decision Tree which is 69% accuracy

5. Evaluation

Based on Decision Tree Model that we choose, we evaluate model by looking at how many of their predictions are correct. This can be plotted into something called Confusion Matrix.

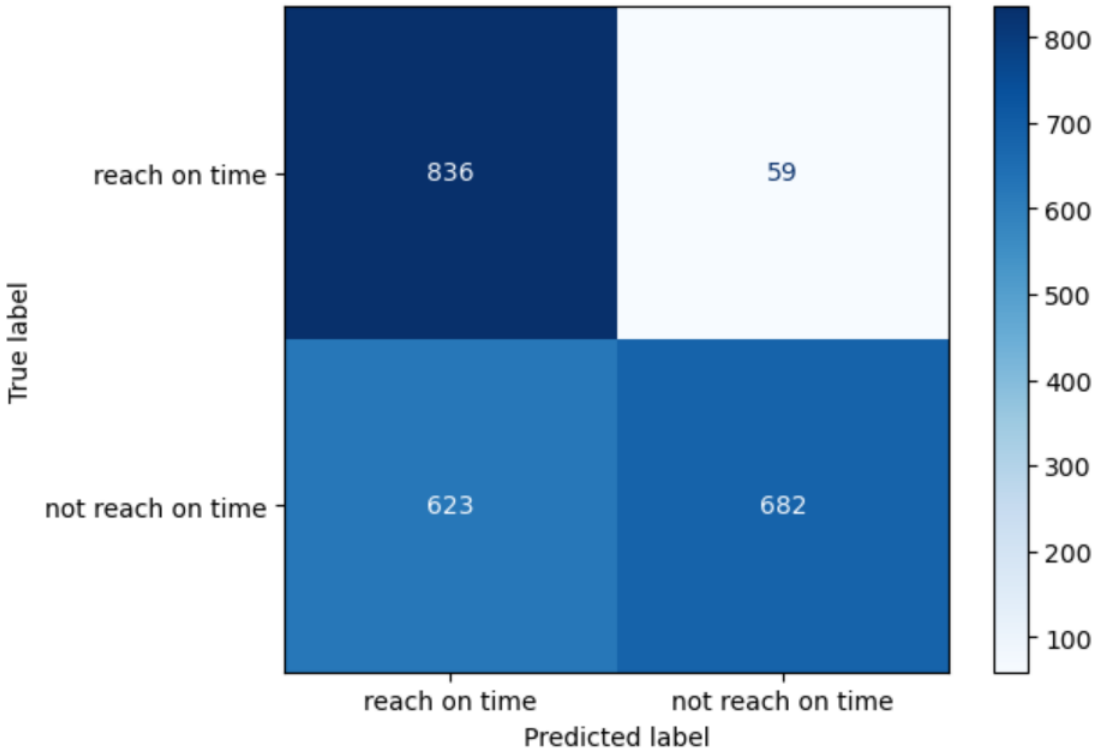


Figure 19: Confusion Matrix Table

The matrix is divided into four area:

- **True Positive (TP = 682):** The model predict delivery not reach on time and the prediction is correct (delivery not reach on time)
- **False Positive (FP = 59):** The model predict delivery not reach on time and the prediction is incorrect (delivery reach on time)
- **True Negative (TN = 836):** The model predict delivery reach on time and the prediction is correct (delivery reach on time)
- **False Negative (FN = 623):** The model predict delivery reach on time and the prediction is incorrect (delivery not reach on time)

Next , accuracy simply tell us how many prediction is true compared to the total dataset.

```
:  1 # Accuracy
   2 accuracy = (tn + tp) / (tn + fp + fn + tp)
   3 print(f'Accuracy is {accuracy:.3f} compared to 0.700 goal')
```

```
Accuracy is 0.690 compared to 0.700 goal
```

Figure 20: Accuracy based on confusion matrix

Accuracy evaluation based on Confusion matrix which is 69% where our goal is to get 70% accuracy which almost reach our target.

On the Evaluation phase, we will further evaluate the model into the context of the business problem.

6. Deployment

Deployment of this model is in term of dashboard , dashboard is a way of displaying various types of visual data in one place. Usually, a dashboard is intended to convey different, but related information in an easy-to-digest form. And oftentimes, this includes things like key performance indicators (KPI)s or other important business metrics that stakeholders need to see and understand at a glance. Dashboards also use visualizations like tables, graphs, and charts, others who aren't as close to the data can quickly and easily understand the story it tells or the insights it reveals.

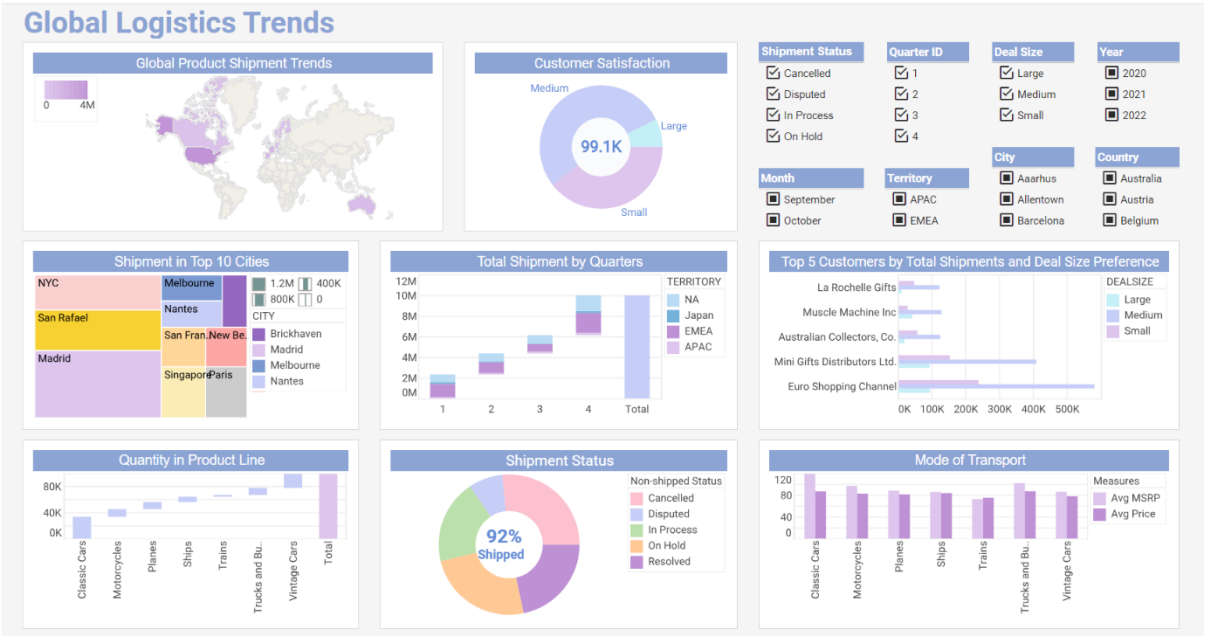


Table 4 : Propose dashboard