COMPUTER AND INFORMATION SCIENCES

MAY 2024

TEB3113

BIG DATA ANALYTICS

GROUP ASSIGNMENT: ALIBABA SHIPMENT FORECAST (DATASET 6)

NO.	NAME	STUDENT ID	PROGRAMME
1.	NURSYAKIRAH SONIA BINTI MONOSARI	20001086	COMPUTER SCIENCE
2.	AININ SOFIYA BINTI MOHD EDYAMIN	20001438	INFORMATION SYSTEM
3.	AMALIN LIYANA BINTI MOHD IZNI ZUHDI	20001473	COMPUTER SCIENCE
4.	SYARIFAH NABILAH BINTI SYED ABDUL RAHMAN	20001090	COMPUTER SCIENCE

ABOUT DATASET

The dataset is designed to analyze and predict the timeliness of shipments, focusing on various features that capture both customer-specific and order-specific details. The 'Cust_ID' uniquely identifies each customer, ensuring that the data can be accurately tracked and analyzed at the individual level. 'Block' refers to a categorical attribute that may represent geographic or logistic zones within the shipment process, potentially influencing delivery times. 'Ship_mode' indicates the method of shipment, which could range from standard ground shipping to expedited air freight, directly impacting timeliness. The 'Discount' feature quantifies any price reductions applied to the shipment, while 'Weight_gm' specifies the weight of the package in grams, both of which could affect shipping logistics and speed.

Additional features include 'Cust_calls', capturing the number of times a customer has contacted customer service, which might correlate with shipment issues or delays. 'Prior_purchases' denotes the customer's purchase history, providing insight into their shopping behavior and potential priority in shipping. 'Priority' is a measure of the shipment's urgency, potentially influencing its handling and delivery speed. 'Gender' and 'Cust_rating' offer demographic and satisfaction data, respectively, which could indirectly relate to shipment timeliness. Lastly, 'Cost' represents the total cost of the shipment, which might be associated with service level and speed. The target variable, 'Timeliness', indicates whether shipments were delivered on time, serving as the primary outcome for prediction and analysis in this dataset.

1.0 BACKGROUND OF STUDY

1.1 Understanding Dataset Nature

i. **Field of the Data:** E-Commerce

The dataset is related to the e-commerce sector, specifically focusing on shipment records. E-commerce involves the buying and selling of goods and services. In this particular context, shipment records offer information about the flow of goods from the seller to the buyer, which is an essential aspect of the e-commerce supply chain.

ii. **Data Source:** Alibaba shipment records

The dataset is sourced from Alibaba, one of the world's largest e-commerce platforms. Since Alibaba promotes global trade, details on the logistic and delivery of the products sold through the platform are usually included in the shipment records. This data is valuable for analyzing patterns in trade, logistics efficiency, shipping times, and more.

iii. **Nature of the Data:** Structured dataset

The dataset is structured, meaning that it is organised in a predefined format that is easy to analyze and manipulate. This data is in tabular format and is organized with 10,999 rows and 12 columns that has headers. This makes the data easy to understand and work as each column has a consistent data type as the headers functions as a description of the type of data contained within the column. All of this is stored in a CSV (Comma-Separated Values) file, a common format for storing tabular data in plain text. This makes it easier to import into various data analysis tools and software.

1.2 Objectives of the Analytics Study

i. **To investigate the correlation between different parameters pertaining to shipping**
   For instance, how different shipping modes, such as ‘Ship’, ‘Flight’, ‘Road’, affect timeliness of deliveries, how customer calls may derive from shipping issues, as well as how their impact influences customer level of satisfaction.

ii. **To develop predictive models using different classifiers to further understand and discover ways shipping operations can be leveraged**
   Such that, how cost can be efficiently managed, and how the level of customer satisfaction can be enhanced through the use of discounts as a form of incentive and improved delivery timeliness.

iii. **To obtain insights and strategies aimed in improving operational efficiencies and level of customer satisfaction**

1.3 CRISP-DM Methodology

This section will elaborate on how Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology can be effectively implemented in the analysis of the Alibaba Shipment Forecast dataset. The methodology is divided into 6 consecutive phases as shown in ***Figure 1***. Phase 1 until phase 3 accounts for almost 85% of the total project time.

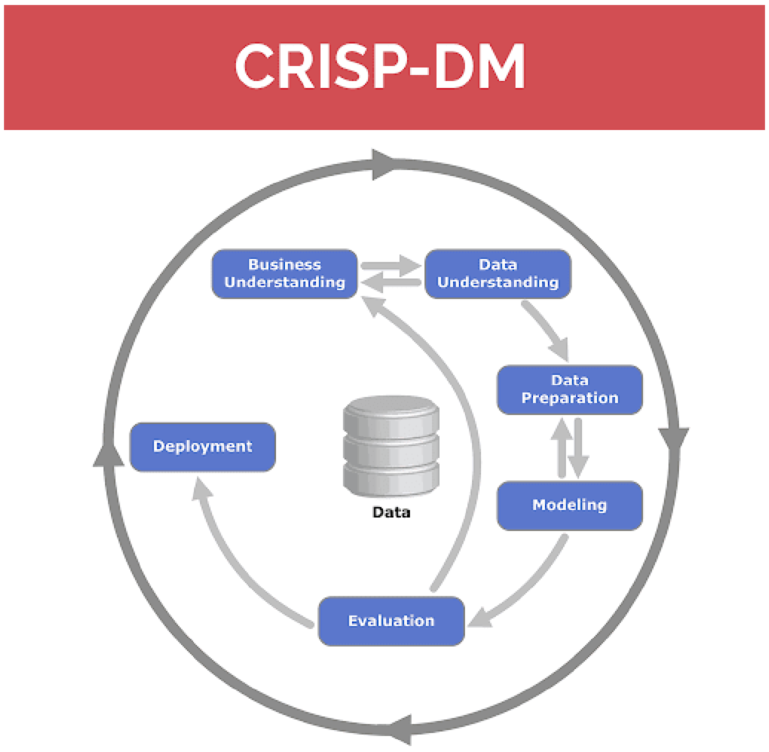


Figure 1 CRISP-DM Methodology

1.3.1 PHASE 1: BUSINESS UNDERSTANDING

This phase focuses on defining the project objectives from a business perspective. It involves converting these objectives into data mining or machine learning problem definitions. A plan of action is formulated, considering data requirements, input from business stakeholders, and the design of performance evaluation metrics. Therefore, below are the required information for this phase:

i. Determine Business Objectives

The primary objective of this project is to predict the timeliness of future Alibaba shipments using the provided dataset. This is due to the timeliness being a crucial for customer satisfaction and operational efficiency in the logistics sector. Accurate predictions can help Alibaba in optimizing delivery routes, managing inventory, and improving customer service. Below are the Business Objectives for this future shipment timeliness prediction model:

1. To forecast the timeliness of Alibaba shipments based on historical data in identifying potential risk of delays in the future shipments
2. To identify key factors influencing shipment delays for logistics performance improvement actions to be taken
3. To develop data visualizations to get predictive insights for better strategic planning and decision-making in logistics

ii. Assess Situation

The dataset includes 12 features consisting of Cust_ID, Block, Ship_mode, Discount, Weight_gm, Cust_calls, Prior_purchases, Priority, Gender, Cust_rating, Cost, and Timeliness. To assess the situation, it is vital to understand the influence of these features on shipment timeliness. Thus, the business can take necessary actions based on the targeted features which has the highest impact on the timeliness. For instance, the Ship_mode (Road, Ship, Flight) could significantly impact delivery time, while customer-related features like Cust_calls and Prior_purchases may reflect the customer's engagement level, potentially affecting the service they receive.

Moreover, it is also essential to know the acceptable range of accuracy for the prediction model. Therefore, we need to understand the industry benchmarks and expectations. Generally, an accuracy rate of 70% to 80% is considered good for many predictive models in logistics and shipping. This is because achieving perfect accuracy is often unrealistic due to the variability and complexity of shipping logistics, which can include factors such as weather, traffic conditions, and customs delays.

This is also proven by several research studies where a hybrid ensemble learning-based prediction model with bagging and stacking effectively predicts air cargo delays with at least 70% precision (Sahoo et. al., 2021). Another research study also reveals that they developed a machine learning model to predict delays in airfreight shipment delivery with accuracy to be over 75% (Shahid, 2020). Therefore, from here, we can use above 70% accuracy level as the benchmark for our prediction model in forecasting the timeliness of future Alibaba shipments.

iii. Determine Data Mining Goals

The data mining goal is to build a best-fit predictive model to forecast the Timeliness of the future shipments. Below are the goals:

1. **To identify the most significant variables that influence shipment timeliness**

The purpose of having this goal is to enable Alibaba to have targeted interventions once the key factors are identified. Therefore, we need to analyse and understand the correlation between the 12 features of the Alibaba Shipment Dataset.

2. **To train a model that can accurately predict the timeliness of the future shipment**

This can potentially be done using binary classification, 1 represents timely while 0 represents not timely. The model that is expected to achieve an accuracy level of at least 70% in predicting shipment timeliness.

1.3.2 PHASE 2: DATA UNDERSTANDING

In this phase, initial data collection is conducted to become familiar with the data. Activities include identifying data quality issues, discovering initial insights, and detecting interesting subsets to form hypotheses for hidden information. Thus, below are the tasks that we need to complete for this phase:

i. Collect Initial Data

   The initial data collection involves gathering the dataset necessary for predicting the timeliness of Alibaba shipments. The data source for this project is the provided dataset, which contains various attributes relevant to shipment details. These attributes include Cust_ID, Block, Ship_mode, Discount, Weight_gm, Cust_calls, Prior_purchases, Priority, Gender, Cust_rating, Cost, and Timeliness. For harvesting process, in reality, we may extract these data points from the company's databases or operational systems where shipment records are stored. This extraction can be performed using SQL queries, API calls, or data export tools, ensuring that the data is up-to-date and comprehensive. However, for this project purposes, we will be using only the provided dataset.

ii. Describe Data

   The structural description of the dataset includes an overview of its schema and an example of the data it contains. In terms of the size of the dataset, it is a structured table with 12 columns representing different features of the shipment records and 10,999 rows, each corresponding to a unique shipment. Knowing the data size can help us assess the scale and potential computational requirements for Alibaba Shipment Forecast. Additionally, understanding the shape of the dataset, such as 2D tabular data or 3D arrays for images, clarifies how the data is organized and how it can be processed. The data model is a relational table where each row is a shipment record, and each column is a feature of the shipment.

Furthermore, understanding different datatypes may aid us in identifying the different required preprocessing steps and modelling techniques. Lastly, the data distribution helps up to understand the central tendency and variability of numerical data. The datatypes and distribution of the Alibaba Shipment dataset will be further analysed in the next section, ***2.0 DATA ANALYSIS & VISUALIZATION***.

iii. Explore Data

Exploring the data involves generating descriptive statistics to understand the distribution and central tendencies of the dataset. This helps identify any anomalies or patterns that may be present. This will be further elaborated in the next section, ***2.0 DATA ANALYSIS & VISUALIZATION***.

iv. Verify Data Quality

The data quality can be measured using the six core dimensions of data quality as shown in ***Figure 2***. The purpose of verifying the data quality is to identify any issues existed in within the dataset. Further explanation on the Alibaba Shipment dataset quality will be elaborated in the next section, ***2.0 DATA ANALYSIS & VISUALIZATION***.

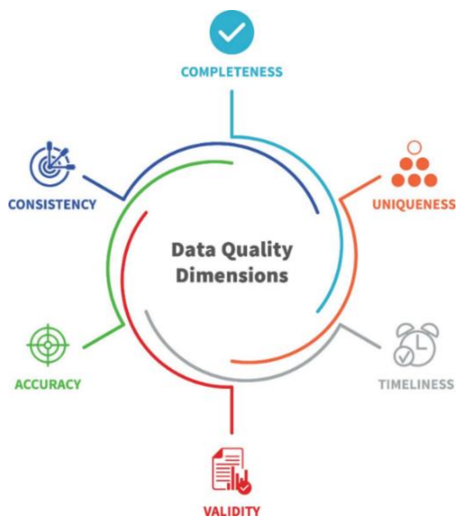


Figure 2 Data Quality Dimensions

1. Completeness

The completeness of the dataset refers to the extent to which all required data is present. Missing values in features such as Discount, Weight_gm, Cust_calls, Prior_purchases, and Timeliness can pose significant risks. Missing data can lead to biased models, as the absence of certain values might skew the analysis and result in inaccurate predictions. To address this, we need to identify any missing values by checking the sum of NaN or NA values. Then, we can handle the missing values either using statistical methods such as imputation to fill in missing values or exclude records with missing critical information.

2. Uniqueness

Uniqueness ensures that each record is distinct and not duplicated within the dataset. Thus, in cases where we encounter duplicated data, it needs to be removed to ensure uniqueness. Duplicates in the dataset, particularly in the Cust_ID, can lead to inflated data. Implementing uniqueness constraints on primary keys such as Cust_ID can mitigate this risk. Duplicates might arise from repeated entries of the same shipment data due to manual entry errors or system glitches that record multiple instances of the same event.

3. Timeliness

Timeliness refers to the availability of up-to-date data. Outdated information can adversely affect the model's predictions, especially in dynamic environments where shipment conditions and business operations frequently change. For timeliness of the data, since we do not have the information on dataset version history, we assume that the dataset provided is recent.

4. Validity

Validity ensures that data values conform to the expected formats and ranges. Invalid data entries in features such as Discount, Weight_gm, and Cust_calls can lead to incorrect model assumptions and predictions. Thus, we need to ensure that each feature conforms to the syntax of its definition. For instance, Block, Ship_mode, Priority, and Gender initial datatype of object needs to be encoded into categorical variables or numerical variables. Additionally, discounts should be within 0-100%, weights should be non-negative, and customer calls should be within a reasonable range. Invalid data might result from manual entry errors, incorrect data format conversions, or faulty sensors and systems.

5. Accuracy

The dataset should also represent the real-world values to ensure its accuracy. Therefore, we need to identify the outliers within the dataset and validate whether the value is acceptable generally. Inaccuracies might occur due to human errors during data entry, malfunctioning measurement devices, or incorrect data aggregation methods. For example, if the weight of a shipment is incorrectly measured as 5000 grams instead of 500 grams, it can affect the prediction model's accuracy.

6. Consistency

In terms of the data consistency, it ensures that data values are uniform across the dataset. Inconsistent data entries in features such as Ship_mode, Priority, and Gender can lead to confusion and affect the model's ability to learn patterns correctly. We need to ensure the data formats is correct and standardized. For instance, weights might be recorded in different units such as grams and kilograms, and Ship_mode entries might be recorded using shortforms ‘R’, ‘S’, and ‘F’, instead of "Road," "Ship," and "Flight" due to different entry practices by different clerks.

1.3.3 PHASE 3: DATA PREPARATION

This phase involves constructing the final dataset from raw data. Tasks include selecting tables, records, and attributes, cleaning data, creating new attributes, and transforming data for modelling tools. These tasks may be performed multiple times and in various orders. Thus, below are the necessary tasks to be completed for Alibaba Shipment Forecast dataset under this phase:

i. Data Selection

   The first task of this phase is to perform selection within the dataset as it is vital for building an effective model, in parallel with showing helpful insights to our clients. The method involves identifying the most relevant features that significantly impact the target variable, in our case, the Timeliness. Feature selection techniques like correlation analysis, mutual information, and domain knowledge are among the things we can employ to select relevant attributes. For instance, features such as Ship_mode, Weight_gm, and Priority are likely to have a strong influence on shipment timeliness. This step ensures that the model is built on the most pertinent data, enhancing its predictive accuracy. Moreover, identifier data or any data that is not helpful for our model can be removed, such as the Cust_ID.

ii. Data Cleaning

   Data cleaning addresses issues such as missing values, outliers, and inconsistencies within the dataset to ensure the data quality is maintained prior to building a model. These issues can be described using descriptive analytics. Then, methods like imputation are used to handle missing values, by replacing them with mean, median, or mode values depending on the attribute type. Moreover, we can use z-score or IQR to detect outliers and decide whether to treat or remove them based on their impact on the analysis. The data cleaning processes for our data will be further explained in the next section,

iii. Data Construction

Data construction involves creating new attributes that may provide better insights into the data. This can include feature engineering techniques such as creating interaction terms, polynomial features, or aggregating existing features. For instance, constructing a new feature such as Cost_per_gram by dividing Cost by Weight_gm could provide insights into cost efficiency and its impact on timeliness. Such derived attributes can enhance the model's ability to capture complex relationships in the data. However, for this project, we will not create any new features and only use the existing features within the dataset.

iv. Data Integration

Data integration combines data from multiple sources to create a comprehensive dataset. Although the provided dataset appears to be singular, if additional data sources are available, such as external data on traffic conditions and weather data along the shipment route, they can be integrated to enrich the dataset. Integration methods include merging datasets on common keys, performing joins, and ensuring consistency across integrated data. This holistic dataset may offer more powerful prediction model. However, in our case, we will only be using the single dataset which is provided.

v. Data Format

Data formatting ensures that the data is in a suitable structure and format for analysis. This includes converting categorical variables into numerical formats using techniques like one-hot encoding or label encoding. Continuous variables may be normalized or standardized to ensure they are on a comparable scale. Proper formatting ensures that the data is compatible with the modelling algorithms and helps improve the model's performance.

1.3.4 PHASE 4: MODEL BUILDING

In this phase, we need to select modelling technique, building test case and the model. Various data mining and modelling techniques can be selected and applied, with parameters optimized to achieve the best results. Furthermore, the more crucial element is explaining the choice. Set of tasks under this phase:

i. Selection of Modelling Technique

   In Phase 4, the first step is to select appropriate modelling techniques that align with the project's objectives and the nature of the dataset. Given the dataset, it is likely that we can do classification or regression models, thus, different approaches can be used to compare the model’s performances and select the best-fit model for our forecasting problem. The approaches that we use will be further elaborated in the 3.2 Model Training section.

ii. Test Design Generation

   We need to identify the split set of train and test data by dividing the provided dataset into a specific ratio. In general, there is no specific rule that we must use to split the data as long as the model have sufficient datapoints to train the model. However, the generally accepted ratio of train:test split data is 70:30, 80:20, and 72:25 ratios. As a benchmark, there is a study showing that the best split set for a shipment timeliness prediction model is train:test split data of 77:23 ratio (James et. al., 2021). In this case, we will be using split set of 80% train and 20% test. Training on 80% of the dataset allows models to learn patterns, while testing on the remaining 20% evaluates how well the models generalize to unseen data. This split helps in detecting overfitting and provides a realistic estimate of model performance on new data.

iii. Model Building

   This task requires us to build the models using the dataset train:test split. Here, we can proceed to building the models that we have selected

1.3.5 PHASE 5: TESTING AND EVALUATION

At this stage in the project, we have built one or more models that appear to have high quality, from a data analysis perspective. Before proceeding to final deployment of the model, it is important to more thoroughly evaluate the model, and review the steps executed to construct the model, to be certain it properly achieves the business objectives. A key objective is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached. In this phase, we will cover these tasks:

i. Result Evaluation

   Once the model has been built, the result will need to be assessed based on several key metrics, which includes accuracy, precision, recall and F1 score as shown in ***Table 1***.

Table 1 Model Assessment Metrics

METRIC	DESCRIPTION
Accuracy	Overall correctness of the model's predictions
Precision	Proportion of true positive predictions among all positive predictions made
Recall	Proportion of true positive predictions among all actual positive instances
F1 Score	Harmonic mean of precision and recall, providing a balanced measure between the two

ii. Review Process

   The review process involves critically analysing the results and ensuring they align with the business objectives as we have stated in Phase 1.

iii. Determine Next Steps

Once the models have been reviewed, we can determine the subsequent actions to be taken. The next actions include the model refinement, further data collection, documentation, and reporting.

1.3.6 PHASE 6: DEPLOYMENT

This phase covers the report generation and implementation of repeatable data mining process. In many cases it will be the user, not the data analyst, who will carry out the deployment steps. In any case, it is important to understand up front what actions will need to be carried out to make use of the created models. Among the tasks under this phase includes the deployment plan, monitoring and maintenance plan, final report and presentation preparation, and project review. However, for this project, we will only cover these tasks:

i. Final Report and Presentation Preparation

   The final report will provide a comprehensive documentation of the final model, including its architecture and performance metrics. The report will also clarify the data preprocessing steps and any assumptions made. For the presentation preparation, we will create a slide deck which summarizes our key points and data insights from the Alibaba Shipment Forecast. Therefore, the model visualizations and model forecast graphs will be demonstrated. The presentation will be held during the lab sessions and assessed by the panel.

ii. Project Review

   Upon completion of the final report and presentation, we will go through the project review process based on the project's outcomes. Stakeholders' feedback will be received during this session. Therefore, any concerns or issues raised will be addressed immediately and taken into consideration for future work in improving the model. The project review also covers the reflection on the challenges faced throughout the development and identify the best practices discovered during the project which will be helpful for future projects within Alibaba or any e-commerce companies.

2.0 DATA ANALYSIS & VISUALIZATION

2.1 Dataset Properties

i. **Cust_ID:** Unique identifier for each customer.

ii. **Block:** Geographic area where the customer resides or where the shipment is being sent.
- **F:** Represents the geographic area F
- **D:** Represents the geographic area D
- **B:** Represents the geographic area B
- **A:** Represents the geographic area A
- **C:** Represents the geographic area C

iii. **Ship_mode:** Mode of shipment used by Alibaba for delivering the products to the customers.
- Road: Shipment delivered via road transportation
- Ship: Shipment delivered via sea transportation
- Flight: Shipment delivered via air transport

iv. **Discount:** Discount applied to the customer’s order.

v. **Weight_gm:** Weight of the shipment or order in grams.

vi. **Cust_calls:** Number of calls made by the customer to the customer service regarding the shipment.

vii. **Prior_purchases:** Number of prior purchases made by the customer.

viii. **Priority:** Priority level assigned to the order made by the customer.
  - low: The order has a low priority and is less urgent
  - medium: The order has a medium priority with moderate urgency
  - high: The order has a high priority which is very urgent

ix. **Gender**: Gender of the customer.
  - F: Female
  - M: Male

x. **Cust_rating**: Customer's rating of the shipment process.

xi. **Cost**: Total cost of the order's shipment.

xii. **Timeliness**: Measure of how timely the delivery was.
  - 0: The delivery was late or not on time.
  - 1: The delivery was made on time.

1.1.2.1. Numerical data

i. Cust_ID, integer (int64)
ii. Discount, integer (int64)
iii. Weight_gm, integer (int64)
iv. Cust_calls, integer (int64)
v. Prior_purchases, integer (int64)
vi. Cust_rating, integer (int64)
vii. Cost, integer (int64)
viii. Timeliness, integer (int64)

1.1.2.2 Categorical data

i. Block, string (object)
ii. Ship_mode, string (object)
iii. Priority, string (object)
iv. Gender, string (object)

2.2 Dataset Issues

i. Existence of categorical data

Categorical data refers to variables that contain label values (string datatype), rather than numerical values. In our dataset, *Block, Ship_mode, Priority* and *Gender* are the categorical in nature. Thus, these needs to be properly encoded or labelled to be used effectively in the machine learning algorithms as all the algorithms used in our study requires numerical input. To solve this issue, we decided to transform these categorical data into numerical format by labelling them manually.

ii. Class imbalance

Class imbalance occurs when the distribution of classes in a dataset is uneven, where some classes are significantly underrepresented compared to others. The scenario is that the number of instances in one class is significantly higher or lower than those in other classes. This is common in real-world datasets such as ours. This imbalance happened after splitting the data for training and testing. No matter how small the imbalance was, it still could lead to challenges in training the model, as it can becomes more biased towards the majority class and performs poorly on the minority class.

iii. Irrelevant data

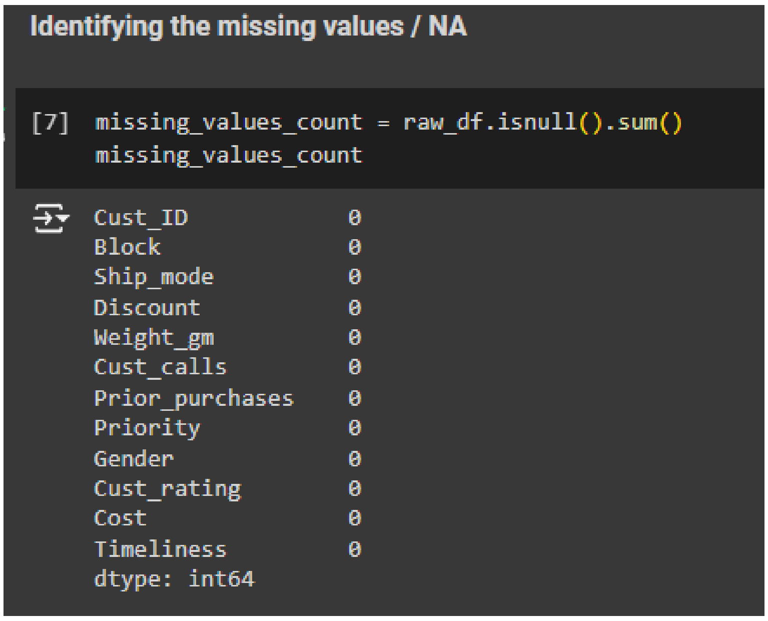
Irrelevant data is features or variables in a dataset that does not contribute any meaningful information to the predictive modelling task. In this dataset, *Cust_ID* is the irrelevant data at is only used as a unique identifier for the shipment. If included into the model, these data will only introduce noise and redundancy that may increase the model's complexity, overfitting and reduce the model's overall performance. As such, it is better to drop this to avoid causing the model to learn from random fluctuations rather than the underlying or hidden patterns.

2.2 Data Quality Dimensions

This section will elaborate the data quality based on the six core dimensions of data quality. This section would also serve as the initial data exploratory of the dataset.

i. Completeness

The data completeness can be assessed by calculating the sum of missing values. Based on this dataset, it has no missing values. This proves that the data achieved a 100% completeness.



Identifying the missing values / NA

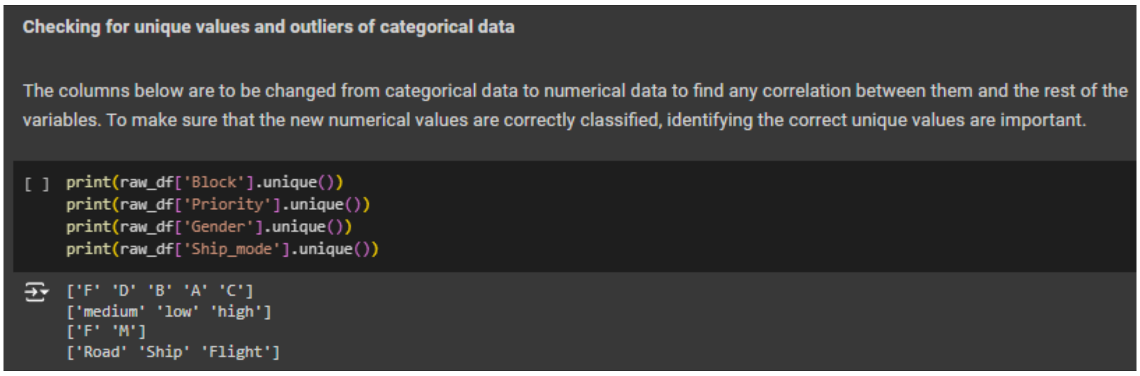
```
[7] missing_values_count = raw_df.isnull().sum()
    missing_values_count
```

```
Cust_ID            0
Block              0
Ship_mode          0
Discount           0
Weight_gm          0
Cust_calls         0
Prior_purchases    0
Priority           0
Gender             0
Cust_rating        0
Cost               0
Timeliness         0
dtype: int64
```

Figure 3 Snippet of codes and output for identifying missing values (NAs)

ii. Uniqueness

The data uniqueness can be assessed by searching for any outliers or unique values, especially in the categorical variables. Based on the dataset, we see that the dataset has no outliers in the categorical variables:


Checking for unique values and outliers of categorical data

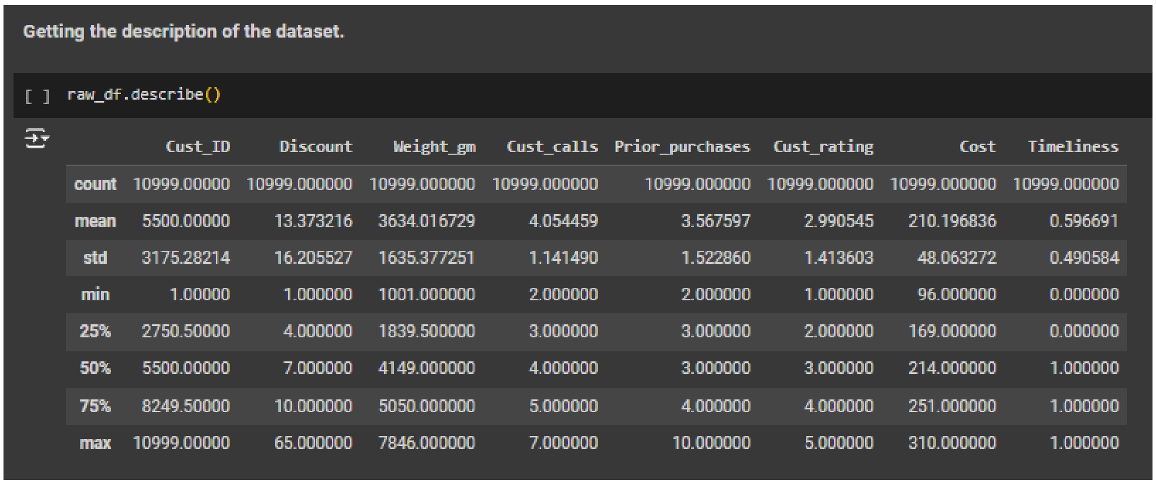
The columns below are to be changed from categorical data to numerical data to find any correlation between them and the rest of the variables. To make sure that the new numerical values are correctly classified, identifying the correct unique values are important.

```
print(raw_df['Block'].unique())
print(raw_df['Priority'].unique())
print(raw_df['Gender'].unique())
print(raw_df['Ship_mode'].unique())
```

```
['F' 'D' 'B' 'A' 'C']
['medium' 'low' 'high']
['F' 'M']
['Road' 'Ship' 'Flight']
```

Figure 4 Snippet of codes and output for checking unique values of categorical data

Looking into the other features, we found a huge range for Discount feature:


Getting the description of the dataset.

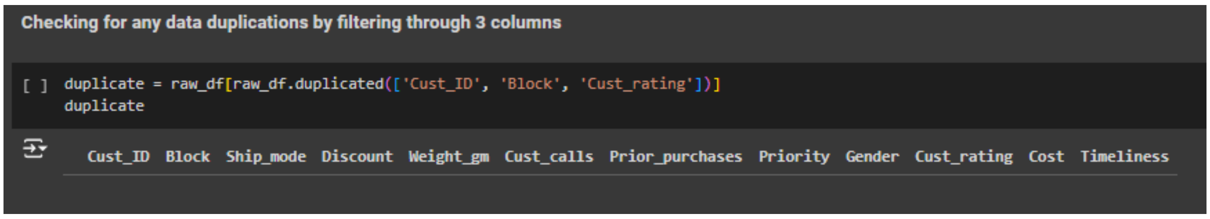
```
raw_df.describe()
```

	Cust_ID	Discount	Weight_gm	Cust_calls	Prior_purchases	Cust_rating	Cost	Timeliness
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	5500.00000	13.373216	3634.016729	4.054459	3.567597	2.990545	210.196836	0.596691
std	3175.28214	16.205527	1635.377251	1.141490	1.522860	1.413603	48.063272	0.490584
min	1.00000	1.000000	1001.000000	2.000000	2.000000	1.000000	96.000000	0.000000
25%	2750.50000	4.000000	1839.500000	3.000000	3.000000	2.000000	169.000000	0.000000
50%	5500.00000	7.000000	4149.000000	4.000000	3.000000	3.000000	214.000000	1.000000
75%	8249.50000	10.000000	5050.000000	5.000000	4.000000	4.000000	251.000000	1.000000
max	10999.00000	65.000000	7846.000000	7.000000	10.000000	5.000000	310.000000	1.000000

Figure 5 Snippet of codes and output for getting the description of the dataset

However, in this case, since discount is a percentage value from 0% to 100%. By logic, it is still an acceptable value as long as the value is not negative or lesser than zero and does not exceed 100. Thus, we can keep the data.

Looking deeper into the dataset, we can conclude that there's no duplications in the dataset as there are no recorded duplications:



Checking for any data duplications by filtering through 3 columns

```
duplicate = raw_df[raw_df.duplicated(['Cust_ID', 'Block', 'Cust_rating'])]
duplicate
```

```
Cust_ID  Block  Ship_mode  Discount  Weight_gm  Cust_calls  Prior_purchases  Priority  Gender  Cust_rating  Cost  Timeliness
```

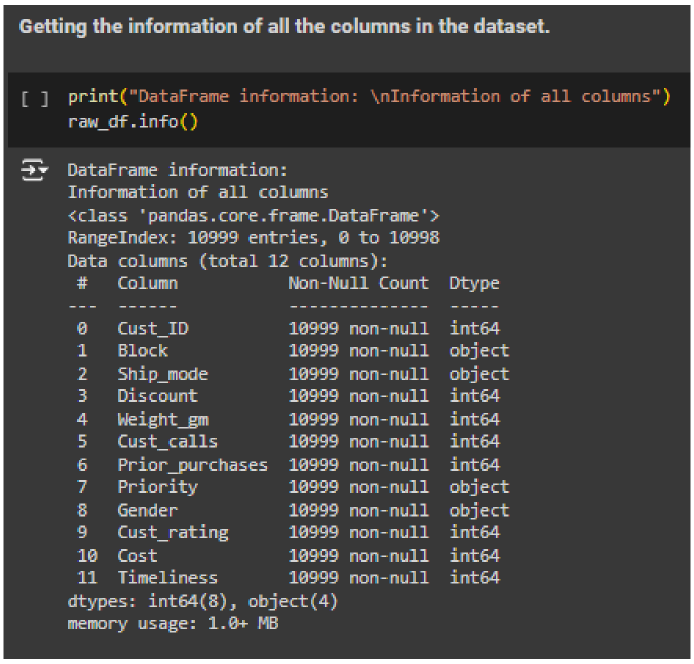
Figure 6 Snippet of codes and output for checking for any duplications

iii. Timeliness

Since the dataset is readily provided, we will assume that the data is recent and up to date.

iv. Validity

The data validity can be accessed by looking into the datatypes of each feature:



Getting the information of all the columns in the dataset.

```
print("DataFrame information: \nInformation of all columns")
raw_df.info()
```

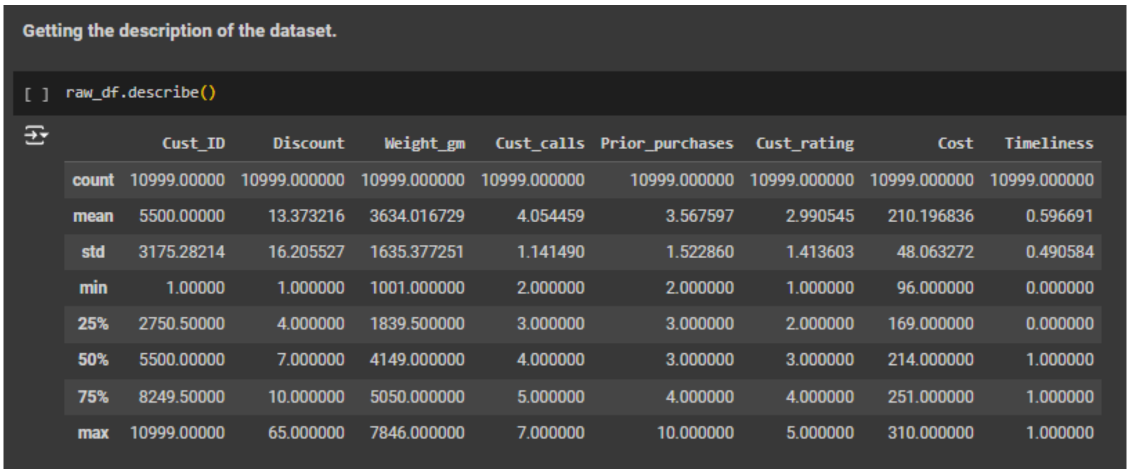
```
DataFrame information:
Information of all columns
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Cust_ID          10999 non-null  int64 
 1   Block            10999 non-null  object
 2   Ship_mode        10999 non-null  object
 3   Discount         10999 non-null  int64 
 4   Weight_gm        10999 non-null  int64 
 5   Cust_calls       10999 non-null  int64 
 6   Prior_purchases  10999 non-null  int64 
 7   Priority         10999 non-null  object
 8   Gender           10999 non-null  object
 9   Cust_rating      10999 non-null  int64 
 10  Cost             10999 non-null  int64 
 11  Timeliness       10999 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

Figure 7 Snippet of codes and output for getting the information of the dataset

The data shows that the categorical variables have incorrect datatypes as it is supposed to be encoded to numerical variables in order for it to be able to be used for model visualization and building.

v. Accuracy and Consistency

The data accuracy and consistency can be verified by referring to the description of the dataset.

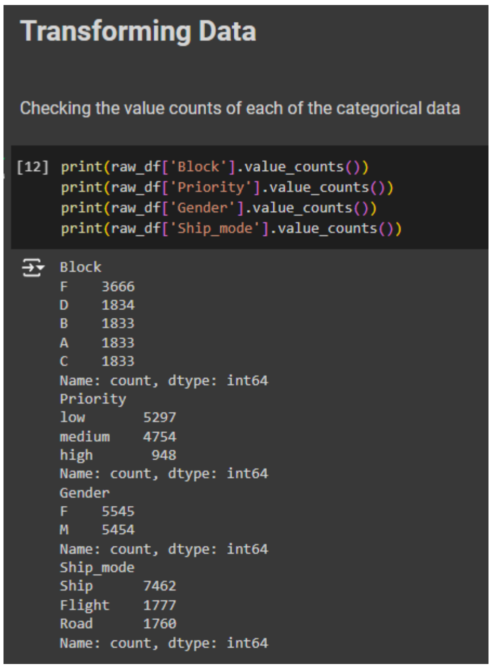

Getting the description of the dataset.

```
[ ] raw_df.describe()
```

	Cust_ID	Discount	Weight_gm	Cust_calls	Prior_purchases	Cust_rating	Cost	Timeliness
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	5500.00000	13.373216	3634.016729	4.054459	3.567597	2.990545	210.196836	0.596691
std	3175.28214	16.205527	1635.377251	1.141490	1.522860	1.413603	48.063272	0.490584
min	1.00000	1.000000	1001.000000	2.000000	2.000000	1.000000	96.000000	0.000000
25%	2750.50000	4.000000	1839.500000	3.000000	3.000000	2.000000	169.000000	0.000000
50%	5500.00000	7.000000	4149.000000	4.000000	3.000000	3.000000	214.000000	1.000000
75%	8249.50000	10.000000	5050.000000	5.000000	4.000000	4.000000	251.000000	1.000000
max	10999.00000	65.000000	7846.000000	7.000000	10.000000	5.000000	310.000000	1.000000

Figure 8 Snippet of codes and output for getting the dataset description

This description only states the numerical data that is in the dataset. From here we can confirm that the data is accurate and consistent as it does not have any negative values for all of them, while for Cust_ID all of the values are unique as the min is 1 and max is 10,999 indicating that the Cust_ID works as an identifier or index for the dataset. This column can be removed as when the dataset is converted into data frame format, it would automatically be indexed by default.


Transforming Data

Checking the value counts of each of the categorical data

```
[12] print(raw_df['Block'].value_counts())
     print(raw_df['Priority'].value_counts())
     print(raw_df['Gender'].value_counts())
     print(raw_df['Ship_mode'].value_counts())
```

```
Block
F    3666
D    1834
B    1833
A    1833
C    1833
Name: count, dtype: int64
Priority
low       5297
medium    4754
high       948
Name: count, dtype: int64
Gender
F    5545
M    5454
Name: count, dtype: int64
Ship_mode
Ship      7462
Flight    1777
Road      1760
Name: count, dtype: int64
```

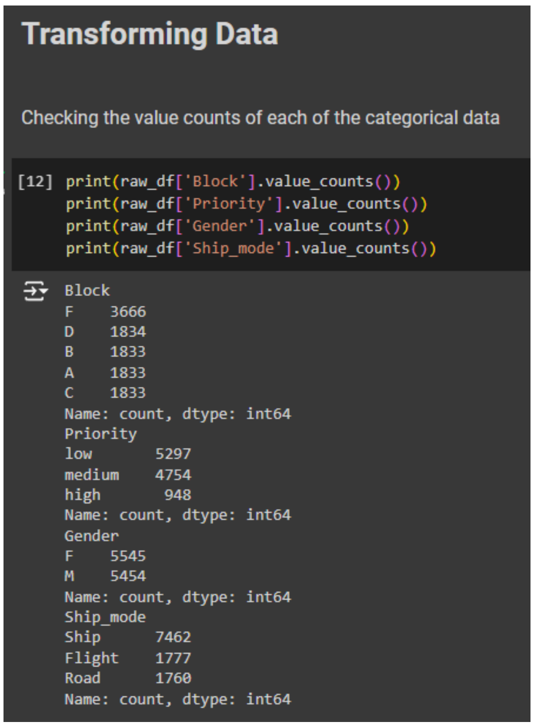
Figure 9 Snippet of codes and output for checking the value counts of categorical data

As for the categorical data, all of them have the expected unique values, implying that the dataset has neither inaccurate nor inconsistent data within the dataset.

2.3 Data Cleaning

Before continuing to do the further data exploratory, the identified categorical data needed to be encoded or changed into a numerical format. Without changing them, our model will not perform well, especially as all of the algorithms used only works with numerical values.

Checking for unique values



Transforming Data

Checking the value counts of each of the categorical data

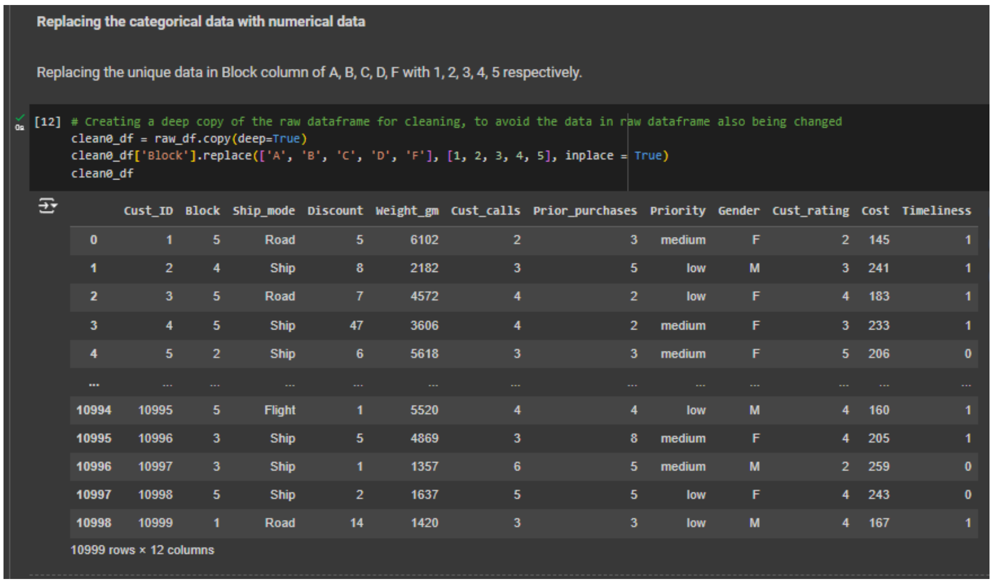
```
[12] print(raw_df['Block'].value_counts())
     print(raw_df['Priority'].value_counts())
     print(raw_df['Gender'].value_counts())
     print(raw_df['Ship_mode'].value_counts())
```

```
Block
F    3666
D    1834
B    1833
A    1833
C    1833
Name: count, dtype: int64
Priority
low       5297
medium    4754
high       948
Name: count, dtype: int64
Gender
F    5545
M    5454
Name: count, dtype: int64
Ship_mode
Ship      7462
Flight    1777
Road      1760
Name: count, dtype: int64
```

Figure 10 Snippet of codes and output for checking the value counts of categorical data

By checking for the unique values, we can identify which values are in the each of the categorical data. From here we can associate the string value with a numerical value by using the replace function in pandas.

Replacing the categorical data with numerical data



Replacing the categorical data with numerical data

Replacing the unique data in Block column of A, B, C, D, F with 1, 2, 3, 4, 5 respectively.

```
[12] # Creating a deep copy of the raw dataframe for cleaning, to avoid the data in raw dataframe also being changed
     clean0_df = raw_df.copy(deep=True)
     clean0_df['Block'].replace(['A', 'B', 'C', 'D', 'F'], [1, 2, 3, 4, 5], inplace = True)
     clean0_df
```

```
       Cust_ID  Block  Ship_mode  Discount  Weight_gm  Cust_calls  Prior_purchases  Priority  Gender  Cust_rating  Cost  Timeliness
0            1      5       Road         5       6102           2                3    medium       F            2   145           1
1            2      4       Ship         8       2182           3                5       low       M            3   241           1
2            3      5       Road         7       4572           4                2       low       F            4   183           1
3            4      5       Ship        47       3606           4                2    medium       F            3   233           1
4            5      2       Ship         6       5618           3                3    medium       F            5   206           0
...        ...    ...        ...       ...        ...         ...              ...       ...     ...          ...   ...         ...
10994    10995      5     Flight         1       5520           4                4       low       M            4   160           1
10995    10996      3       Ship         5       4869           3                8    medium       F            4   205           1
10996    10997      3       Ship         1       1357           6                5    medium       M            2   259           0
10997    10998      5       Ship         2       1637           5                5       low       F            4   243           0
10998    10999      1       Road        14       1420           3                3       low       M            4   167           1
10999 rows × 12 columns
```

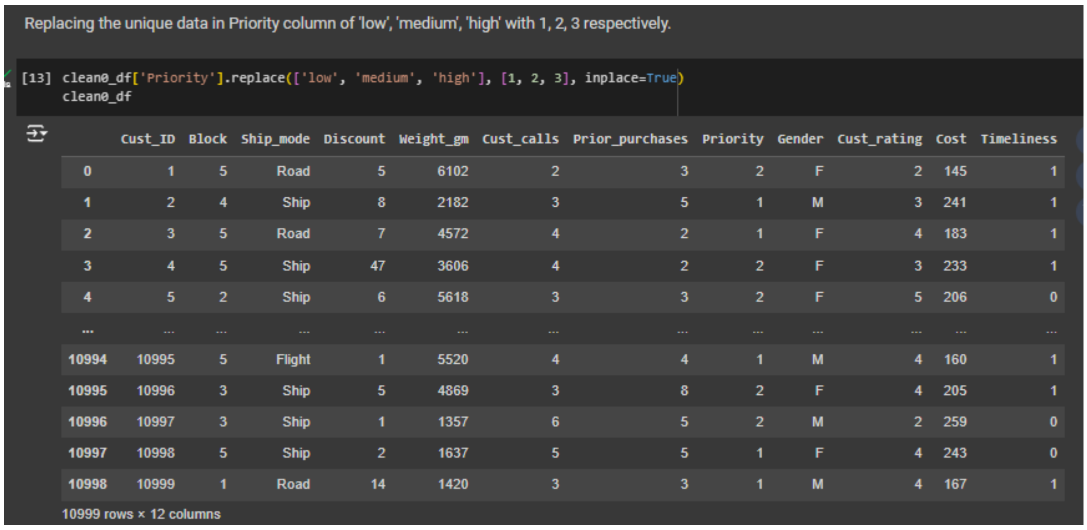
Figure 11 Snippet of codes and output for replacing the unique data in Block

Before we start transform the data, a deep copy of the raw_df named clean0_df is made to ensure that the previous instances of the data in raw_df will not be changed as we use inplace = True to make sure that the new data is saved completely. Hence, the need of a new data frame to store them in.

For the unique data in Block column, we replaced them with numerical values as such:

Table 2 Mapping of original and new values for Block

Original	New
A	1
B	2
C	3
D	4
F	5


Replacing the unique data in Priority column of 'low', 'medium', 'high' with 1, 2, 3 respectively.

```
[13] clean0_df['Priority'].replace(['low', 'medium', 'high'], [1, 2, 3], inplace=True)
     clean0_df
```

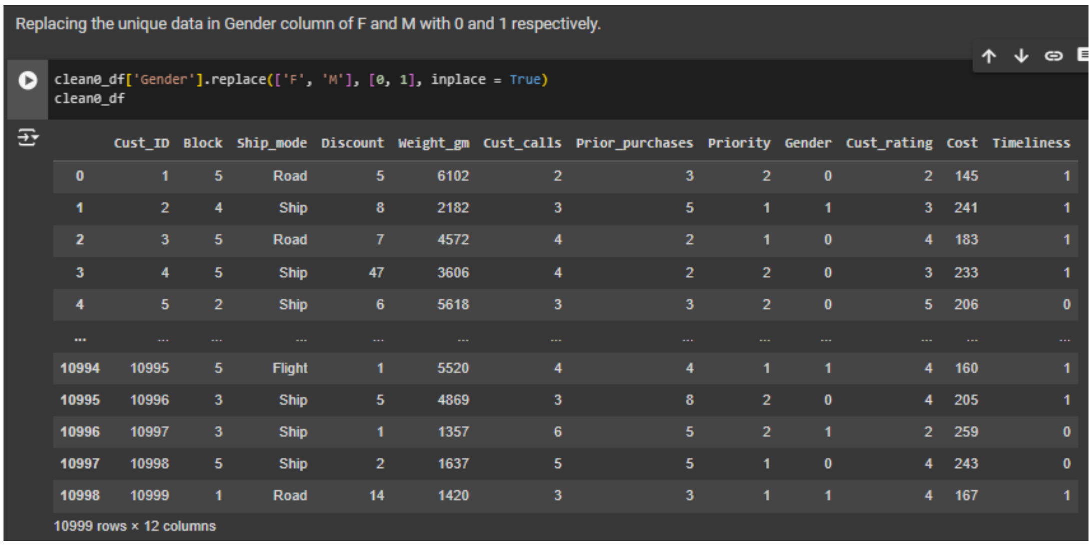
```
       Cust_ID Block Ship_mode Discount Weight_gm Cust_calls Prior_purchases Priority Gender Cust_rating Cost Timeliness
0            1     5      Road        5      6102          2               3        2      F           2  145          1
1            2     4      Ship        8      2182          3               5        1      M           3  241          1
2            3     5      Road        7      4572          4               2        1      F           4  183          1
3            4     5      Ship       47      3606          4               2        2      F           3  233          1
4            5     2      Ship        6      5618          3               3        2      F           5  206          0
...        ...   ...       ...      ...       ...        ...             ...      ...    ...         ...  ...        ...
10994    10995     5    Flight        1      5520          4               4        1      M           4  160          1
10995    10996     3      Ship        5      4869          3               8        2      F           4  205          1
10996    10997     3      Ship        1      1357          6               5        2      M           2  259          0
10997    10998     5      Ship        2      1637          5               5        1      F           4  243          0
10998    10999     1      Road       14      1420          3               3        1      M           4  167          1
10999 rows × 12 columns
```

Figure 12 Snippet of codes and output for replacing the unique data in Priority

For the unique data in Priority column, we replaced them with numerical values as such:

Table 3 Mapping of original and new data for Priority

Original	New
low	1
medium	2
high	3


Replacing the unique data in Gender column of F and M with 0 and 1 respectively.

```
clean0_df['Gender'].replace(['F', 'M'], [0, 1], inplace = True)
clean0_df
```

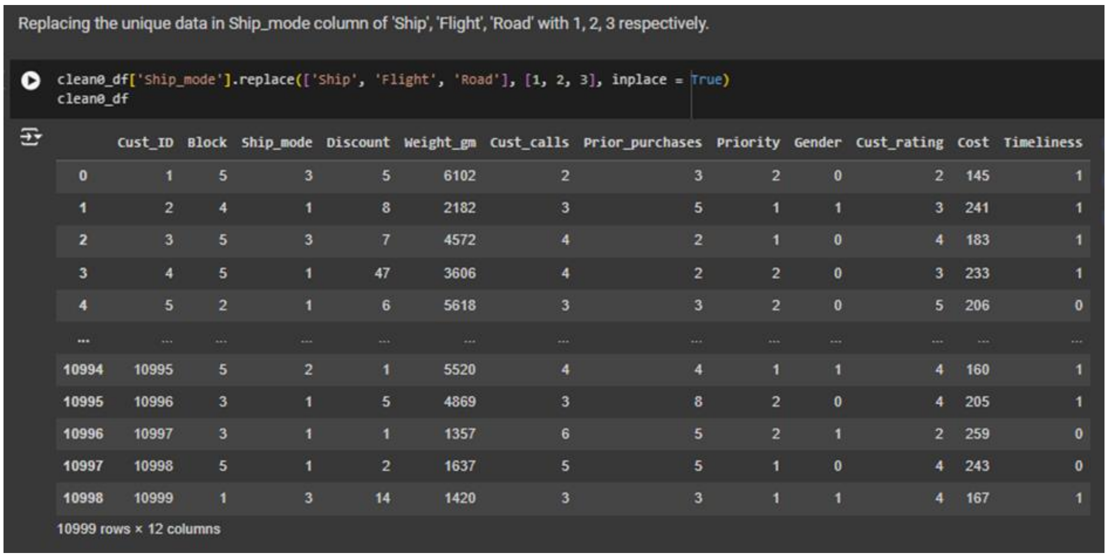
```
       Cust_ID Block Ship_mode Discount Weight_gm Cust_calls Prior_purchases Priority Gender Cust_rating Cost Timeliness
0            1     5      Road        5      6102          2               3        2      0           2  145          1
1            2     4      Ship        8      2182          3               5        1      1           3  241          1
2            3     5      Road        7      4572          4               2        1      0           4  183          1
3            4     5      Ship       47      3606          4               2        2      0           3  233          1
4            5     2      Ship        6      5618          3               3        2      0           5  206          0
...        ...   ...       ...      ...       ...        ...             ...      ...    ...         ...  ...        ...
10994    10995     5    Flight        1      5520          4               4        1      1           4  160          1
10995    10996     3      Ship        5      4869          3               8        2      0           4  205          1
10996    10997     3      Ship        1      1357          6               5        2      1           2  259          0
10997    10998     5      Ship        2      1637          5               5        1      0           4  243          0
10998    10999     1      Road       14      1420          3               3        1      1           4  167          1
10999 rows × 12 columns
```

Figure 13 Snippet of codes and output for replacing the unique data in Gender

For the unique data in Gender column, we replaced them with numerical values as such:

Table 4 Mapping of original and new data for Gender

Original	New
F	0
M	1



Replacing the unique data in Ship_mode column of 'Ship', 'Flight', 'Road' with 1, 2, 3 respectively.

```
clean0_df['Ship_mode'].replace(['Ship', 'Flight', 'Road'], [1, 2, 3], inplace = True)
clean0_df
```

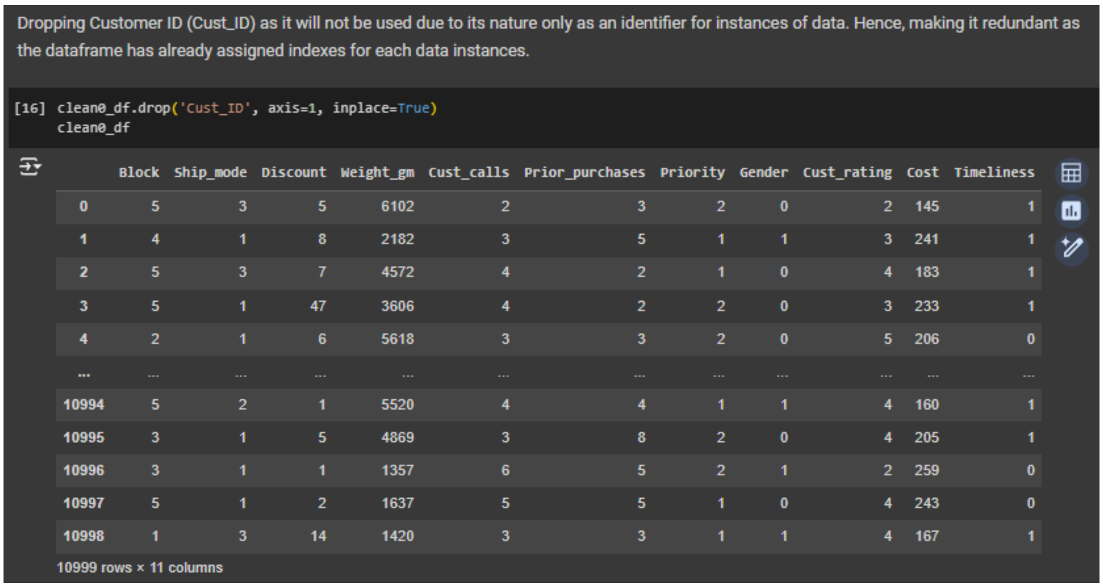
```
       Cust_ID  Block  Ship_mode  Discount  Weight_gm  Cust_calls  Prior_purchases  Priority  Gender  Cust_rating  Cost  Timeliness
0            1      5          3         5       6102           2                3         2       0            2   145           1
1            2      4          1         8       2182           3                5         1       1            3   241           1
2            3      5          3         7       4572           4                2         1       0            4   183           1
3            4      5          1        47       3606           4                2         2       0            3   233           1
4            5      2          1         6       5618           3                3         2       0            5   206           0
...        ...    ...        ...       ...        ...         ...              ...       ...     ...          ...   ...         ...
10994    10995      5          2         1       5520           4                4         1       1            4   160           1
10995    10996      3          1         5       4869           3                8         2       0            4   205           1
10996    10997      3          1         1       1357           6                5         2       1            2   259           0
10997    10998      5          1         2       1637           5                5         1       0            4   243           0
10998    10999      1          3        14       1420           3                3         1       1            4   167           1
10999 rows × 12 columns
```

Figure 14 Snippet of codes and output for replacing the unique data in Ship_mode

For the unique data in Block column, we replaced them with numerical values as such:

Table 5 Mapping of original and new data for Ship_mode

Original	New
Ship	1
Flight	2
Road	3

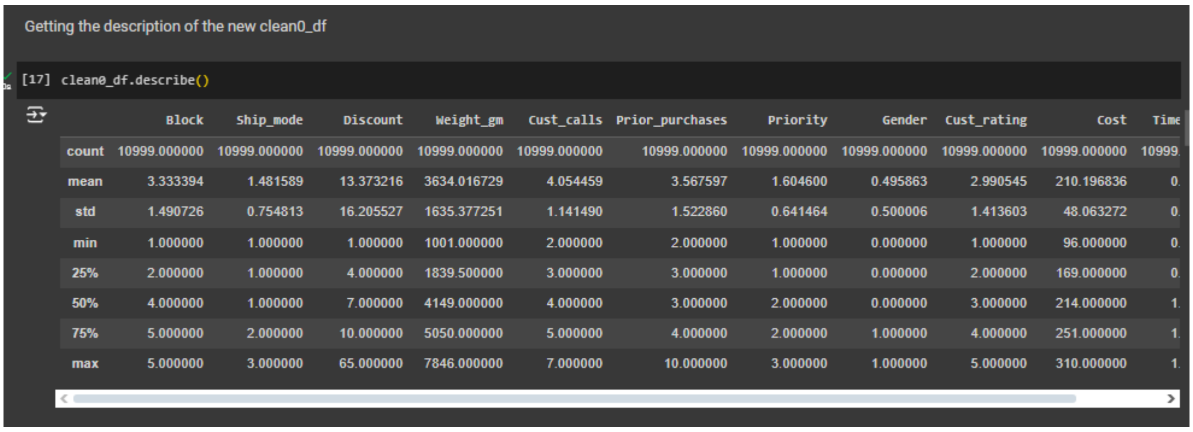

Dropping Customer ID (Cust_ID) as it will not be used due to its nature only as an identifier for instances of data. Hence, making it redundant as the dataframe has already assigned indexes for each data instances.

```
[16] clean0_df.drop('Cust_ID', axis=1, inplace=True)
     clean0_df
```

```
       Block  Ship_mode  Discount  Weight_gm  Cust_calls  Prior_purchases  Priority  Gender  Cust_rating  Cost  Timeliness
0          5          3         5       6102           2                3         2       0            2   145           1
1          4          1         8       2182           3                5         1       1            3   241           1
2          5          3         7       4572           4                2         1       0            4   183           1
3          5          1        47       3606           4                2         2       0            3   233           1
4          2          1         6       5618           3                3         2       0            5   206           0
...      ...        ...       ...        ...         ...              ...       ...     ...          ...   ...         ...
10994      5          2         1       5520           4                4         1       1            4   160           1
10995      3          1         5       4869           3                8         2       0            4   205           1
10996      3          1         1       1357           6                5         2       1            2   259           0
10997      5          1         2       1637           5                5         1       0            4   243           0
10998      1          3        14       1420           3                3         1       1            4   167           1
10999 rows × 11 columns
```

Figure 15 Snippet of codes and output for dropping Cust_ID

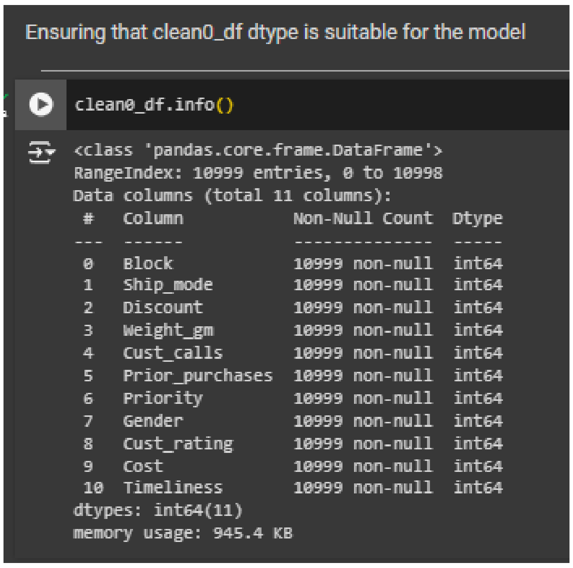
Next, Cust_ID will be dropped as it is only used as an identifier. This would introduce noise and redundancy to the model.


Getting the description of the new clean0_df

```
[17] clean0_df.describe()
```

```
              Block     Ship_mode      Discount     Weight_gm    Cust_calls  Prior_purchases      Priority        Gender   Cust_rating          Cost  Time
count  10999.000000  10999.000000  10999.000000  10999.000000  10999.000000     10999.000000  10999.000000  10999.000000  10999.000000  10999.000000  10999.
mean       3.333394      1.481589     13.373216   3634.016729      4.054459         3.567597      1.604600      0.495863      2.990545    210.196836  0.
std        1.490726      0.754813     16.205527   1635.377251      1.141490         1.522860      0.641464      0.500006      1.413603     48.063272  0.
min        1.000000      1.000000      1.000000   1001.000000      2.000000         2.000000      1.000000      0.000000      1.000000     96.000000  0.
25%        2.000000      1.000000      4.000000   1839.500000      3.000000         3.000000      1.000000      0.000000      2.000000    169.000000  0.
50%        4.000000      1.000000      7.000000   4149.000000      4.000000         3.000000      2.000000      0.000000      3.000000    214.000000  1.
75%        5.000000      2.000000     10.000000   5050.000000      5.000000         4.000000      2.000000      1.000000      4.000000    251.000000  1.
max        5.000000      3.000000     65.000000   7846.000000      7.000000        10.000000      3.000000      1.000000      5.000000    310.000000  1.
```

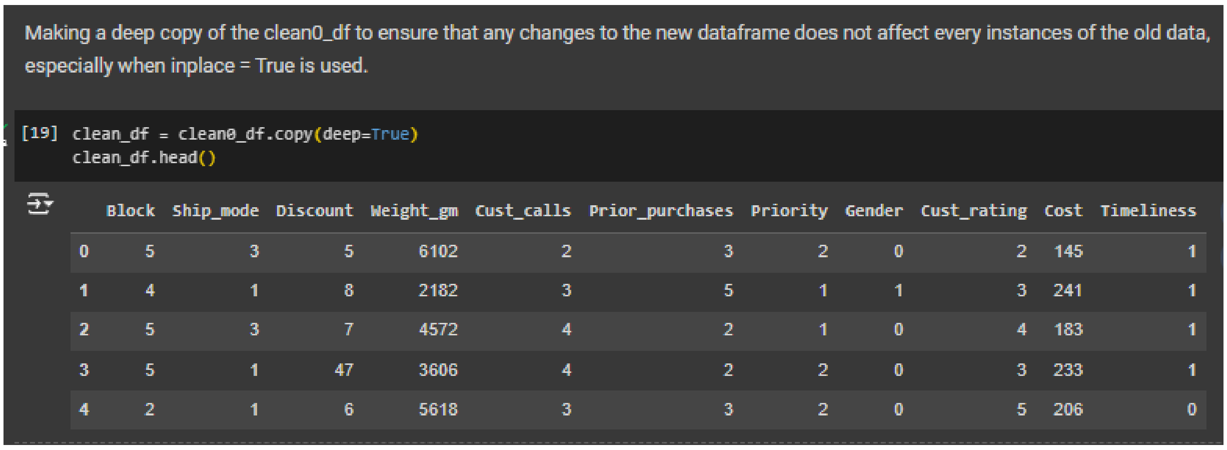
Figure 16 Snippet of codes and output for describing the clean data


Ensuring that clean0_df dtype is suitable for the model

```
clean0_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Block            10999 non-null  int64
 1   Ship_mode        10999 non-null  int64
 2   Discount         10999 non-null  int64
 3   Weight_gm        10999 non-null  int64
 4   Cust_calls       10999 non-null  int64
 5   Prior_purchases  10999 non-null  int64
 6   Priority         10999 non-null  int64
 7   Gender           10999 non-null  int64
 8   Cust_rating      10999 non-null  int64
 9   Cost             10999 non-null  int64
 10  Timeliness       10999 non-null  int64
dtypes: int64(11)
memory usage: 945.4 KB
```

To check whether the new data is in numerical format, both .describe and .info function are used. From there, we have verified that the categorical data columns are now in numerical format.


Making a deep copy of the clean0_df to ensure that any changes to the new dataframe does not affect every instances of the old data, especially when inplace = True is used.

```
[19] clean_df = clean0_df.copy(deep=True)
     clean_df.head()
```

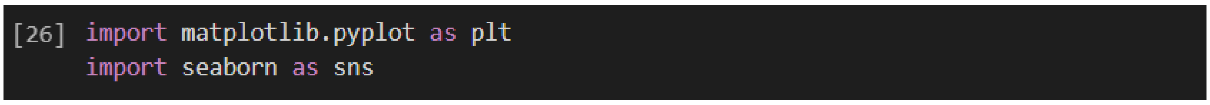
	Block	Ship_mode	Discount	Weight_gm	Cust_calls	Prior_purchases	Priority	Gender	Cust_rating	Cost	Timeliness
0	5	3	5	6102	2	3	2	0	2	145	1
1	4	1	8	2182	3	5	1	1	3	241	1
2	5	3	7	4572	4	2	1	0	4	183	1
3	5	1	47	3606	4	2	2	0	3	233	1
4	2	1	6	5618	3	3	2	0	5	206	0

Lastly, a new deep copy (clean_df) of the clean0_df is made for the model building and further data exploratory. This would serve as the main dataset that will be used throughout the rest of the project.

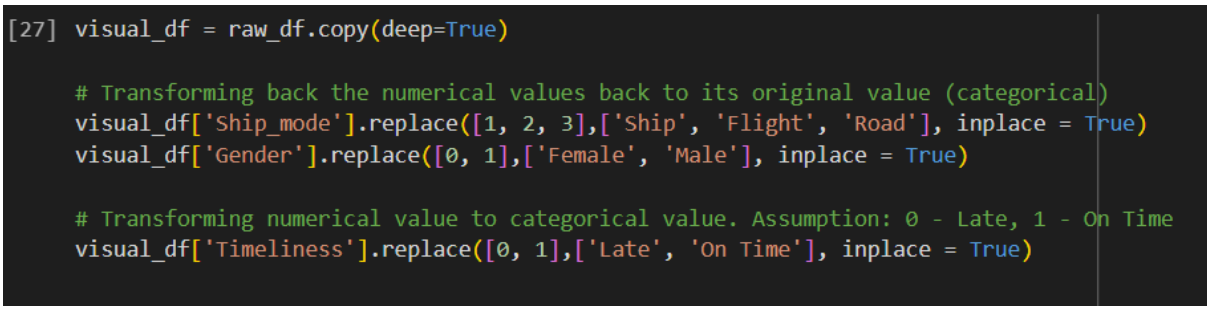
2.4 Visualization of Further Data Exploratory

In the context of Alibaba's shipping forecast, data visualization is crucial because it enables decision-makers to make more informed and effective decisions, thereby enhancing operational efficiency and improving customer satisfaction. This section will present various visualizations to explore and understand the correlations among different data within the dataset.

Re-import Libraries for Data Visualization



```
[26] import matplotlib.pyplot as plt
     import seaborn as sns
```



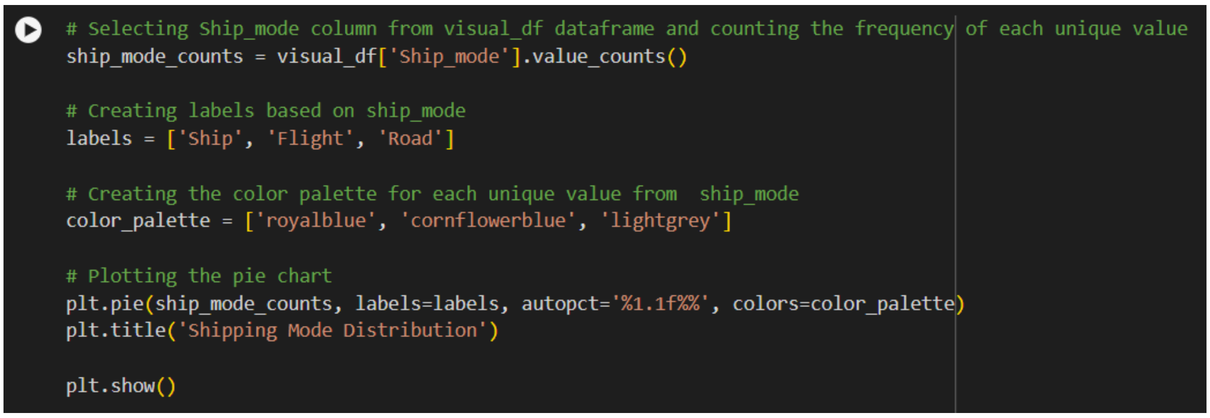
```
[27] visual_df = raw_df.copy(deep=True)

     # Transforming back the numerical values back to its original value (categorical)
     visual_df['Ship_mode'].replace([1, 2, 3],['Ship', 'Flight', 'Road'], inplace = True)
     visual_df['Gender'].replace([0, 1],['Female', 'Male'], inplace = True)

     # Transforming numerical value to categorical value. Assumption: 0 - Late, 1 - On Time
     visual_df['Timeliness'].replace([0, 1],['Late', 'On Time'], inplace = True)
```

Before starting off with the data visualisation, a deep copy of the cleaned DataFrame is created and named as visual_df. The purpose of this is to avoid the data in the cleaned DataFrame from being altered or modified. The '*DataFrame.copy(deep=True)*' method creates a completely independent DataFrame. Therefore, any modifications made to the copied DataFrame will not affect the original one that it has copied from.

Previously, the original state of the data was categorical. However, during data cleaning, these categorical data were transformed into numerical data as it is to be used for data modelling. Thus, to provide better visualization, the data that were transformed from categorical values to numerical values must be transformed back to its original state. This helps readers to have a better understanding when looking into the visualisations as it prevents them from getting confused of seeing labels and legends that are in numeric.

To add, as shown from the last line of code from the figure above, the original data from column *‘Timeliness’* was already in a numerical format, but no context was given to it as it only had shown two values, which are 0 and 1. Thus, an assumption is made with 0 being labelled as ‘Late’ and 1 labelled as 'On time’ indicating the timeliness of the delivery.

2.4.1 Pie Chart: Visualising Shipping Mode Distribution



```
# Selecting Ship_mode column from visual_df dataframe and counting the frequency of each unique value
ship_mode_counts = visual_df['Ship_mode'].value_counts()

# Creating labels based on ship_mode
labels = ['Ship', 'Flight', 'Road']

# Creating the color palette for each unique value from  ship_mode
color_palette = ['royalblue', 'cornflowerblue', 'lightgrey']

# Plotting the pie chart
plt.pie(ship_mode_counts, labels=labels, autopct='%1.1f%%', colors=color_palette)
plt.title('Shipping Mode Distribution')

plt.show()
```

Output:

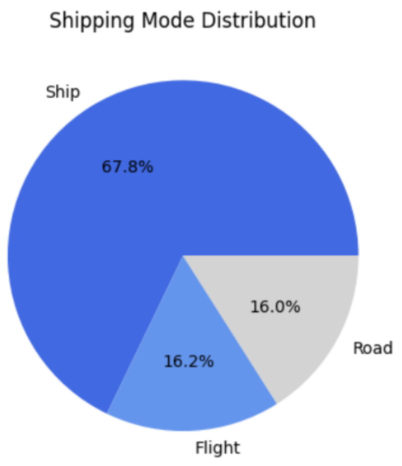
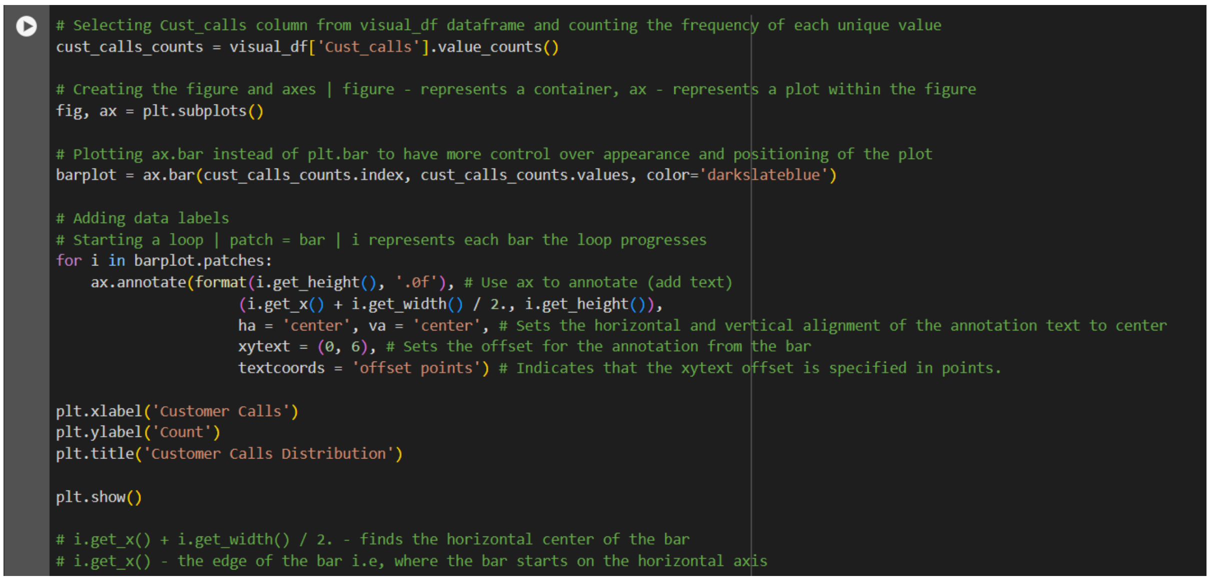


Figure 17 Shipping Mode Distribution Pie Chart

This pie chart visualises the shipping mode distribution for Alibaba. As shown from the pie chart, the highest mode of shipping is ‘Ship’, with its frequency being equivalent to 67.8%, which is then followed by Flight (16.2%) and Road (16%). Although this visualisation may seem simple, however it can provide valuable information for decision making.

The purpose of plotting this pie chart is to strategize for Alibaba’s operational efficiency. Understanding the distribution of the shipping modes allows Alibaba to know which method of shipping is primarily used and how it can help the company to make cost-effective decisions regarding their deliveries and enables them to reduce the overall transportation costs. Moreover, some shipping modes are faster than others. For instance, shipping deliveries by air is much faster than ocean shipping. Therefore, knowing the different shipping options allows Alibaba to plan for strategies that can meet delivery deadlines and customer expectations more effectively.

2.4.2 Bar Chart: Visualising Customer Calls Distribution



```python
# Selecting Cust_calls column from visual_df dataframe and counting the frequency of each unique value
cust_calls_counts = visual_df['Cust_calls'].value_counts()

# Creating the figure and axes | figure - represents a container, ax - represents a plot within the figure
fig, ax = plt.subplots()

# Plotting ax.bar instead of plt.bar to have more control over appearance and positioning of the plot
barplot = ax.bar(cust_calls_counts.index, cust_calls_counts.values, color='darkslateblue')

# Adding data labels
# Starting a loop | patch = bar | i represents each bar the loop progresses
for i in barplot.patches:
    ax.annotate(format(i.get_height(), '.0f'), # Use ax to annotate (add text)
                     (i.get_x() + i.get_width() / 2., i.get_height()),
                     ha = 'center', va = 'center', # Sets the horizontal and vertical alignment of the annotation text to center
                     xytext = (0, 6), # Sets the offset for the annotation from the bar
                     textcoords = 'offset points') # Indicates that the xytext offset is specified in points.

plt.xlabel('Customer Calls')
plt.ylabel('Count')
plt.title('Customer Calls Distribution')

plt.show()

# i.get_x() + i.get_width() / 2. - finds the horizontal center of the bar
# i.get_x() - the edge of the bar i.e, where the bar starts on the horizontal axis
```

Output:

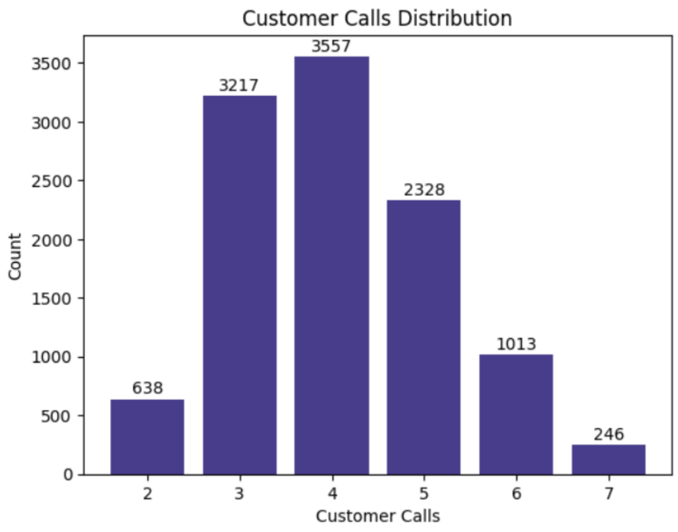
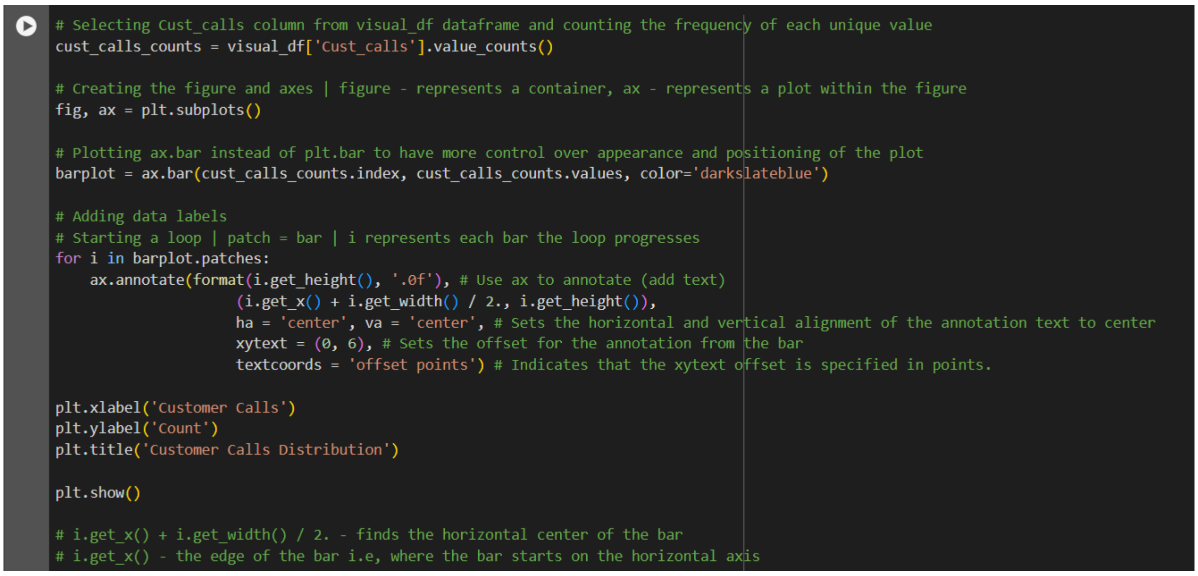


Figure 18 Customer Calls Distribution Bar Chart

This bar chart shows the distribution of customer calls. By analysing the distribution of customer calls, Alibaba can identify both positive and negative customers experience pertaining to their delivery and chosen modes of shipping. The distribution of calls can serve as a metric for customer level of satisfaction. A high volume of calls might indicate customer's dissatisfaction. On the other hand, lower volume of calls may indicate that the customer had a positive delivery experience. Or if a positive experience wasn't the case, it could also be that customers are just asking queries pertaining to their shipped goods.

It is plausible that customers who call multiple times are facing issues regarding their delivery. Problem such as delays, lost packages or goods in the delivery process can signify problems that Alibaba needs to address. In addition, frequent calls made by customers may also indicate that their initial complaints or inquiries were not resolved or handled in a way that satisfies them, thus leading to increased frustration and dissatisfaction. Therefore, this visualisation is crucial for Alibaba in aiding them to further understand their customers' needs and expectations. This visualisation aims to provide Alibaba with insights that can help them to address issues rooted from customers dissatisfaction with their deliveries.

2.4.3 Stacked Bar Chart: Visualising Customer Calls Distribution



```python
# Selecting Cust_calls column from visual_df dataframe and counting the frequency of each unique value
cust_calls_counts = visual_df['Cust_calls'].value_counts()

# Creating the figure and axes | figure - represents a container, ax - represents a plot within the figure
fig, ax = plt.subplots()

# Plotting ax.bar instead of plt.bar to have more control over appearance and positioning of the plot
barplot = ax.bar(cust_calls_counts.index, cust_calls_counts.values, color='darkslateblue')

# Adding data labels
# Starting a loop | patch = bar | i represents each bar the loop progresses
for i in barplot.patches:
    ax.annotate(format(i.get_height(), '.0f'), # Use ax to annotate (add text)
                     (i.get_x() + i.get_width() / 2., i.get_height()),
                     ha = 'center', va = 'center', # Sets the horizontal and vertical alignment of the annotation text to center
                     xytext = (0, 6), # Sets the offset for the annotation from the bar
                     textcoords = 'offset points') # Indicates that the xytext offset is specified in points.

plt.xlabel('Customer Calls')
plt.ylabel('Count')
plt.title('Customer Calls Distribution')

plt.show()

# i.get_x() + i.get_width() / 2. - finds the horizontal center of the bar
# i.get_x() - the edge of the bar i.e, where the bar starts on the horizontal axis
```

Output:

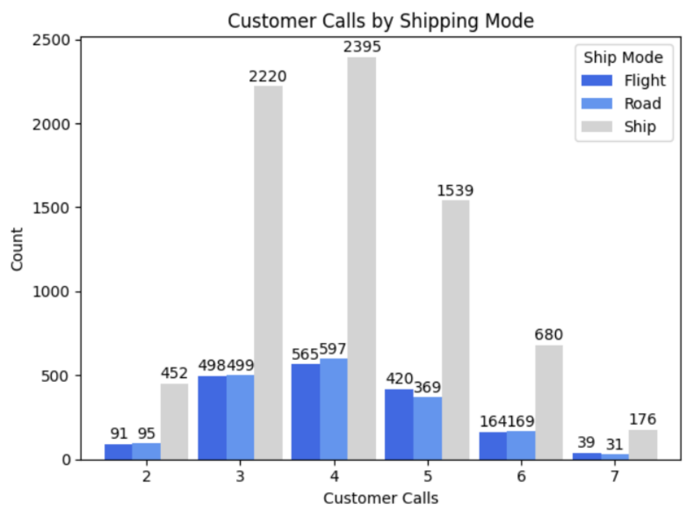
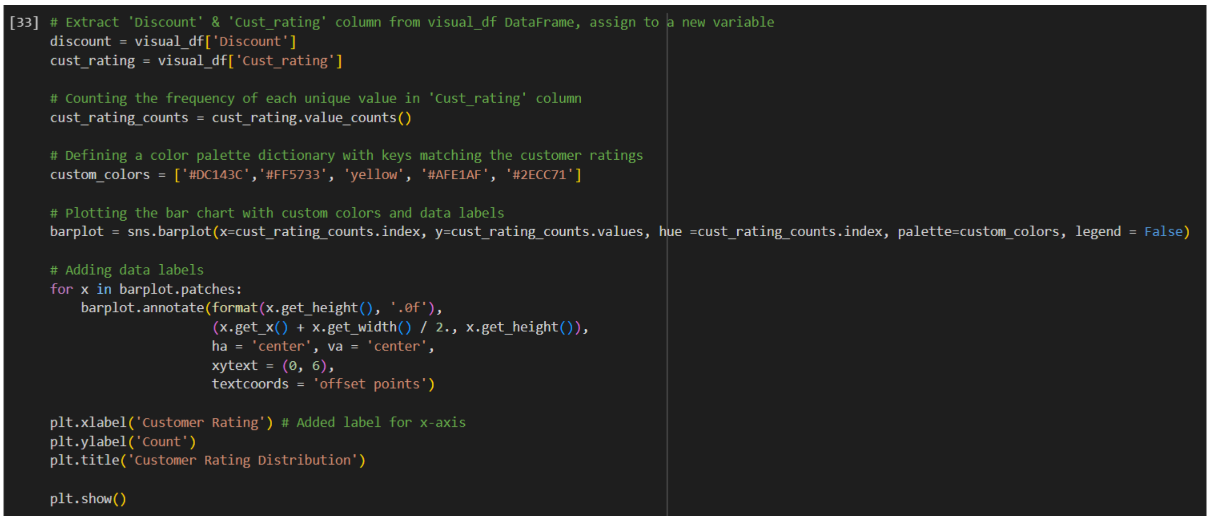


Figure 19 Customer Calls by Shipping Mode Chart

The figure above is a stacked bar chart. While it may look similar to the previous bar chart, the key difference is that this chart shows the distribution of customer calls based on the different shipping modes. The previous bar chart only provided the general insights about the call distribution; however, this stacked bar chart illustrates the distribution based on different shipping modes. From this stacked bar chart, we can conclude that the highest number of calls made is from the ‘Ship’ shipping method. We can assume that the reason being is due to the fact that ocean deliveries may involve factors such as unfavourable weather conditions that can affect the shipping by causing it to delay.

2.4.4 Bar Chart: Visualising Customer Rating Distribution



```
[33] # Extract 'Discount' & 'Cust_rating' column from visual_df DataFrame, assign to a new variable
     discount = visual_df['Discount']
     cust_rating = visual_df['Cust_rating']

     # Counting the frequency of each unique value in 'Cust_rating' column
     cust_rating_counts = cust_rating.value_counts()

     # Defining a color palette dictionary with keys matching the customer ratings
     custom_colors = ['#DC143C','#FF5733', 'yellow', '#AFE1AF', '#2ECC71']

     # Plotting the bar chart with custom colors and data labels
     barplot = sns.barplot(x=cust_rating_counts.index, y=cust_rating_counts.values, hue =cust_rating_counts.index, palette=custom_colors, legend = False)

     # Adding data labels
     for x in barplot.patches:
         barplot.annotate(format(x.get_height(), '.0f'),
                          (x.get_x() + x.get_width() / 2., x.get_height()),
                          ha = 'center', va = 'center',
                          xytext = (0, 6),
                          textcoords = 'offset points')

     plt.xlabel('Customer Rating') # Added label for x-axis
     plt.ylabel('Count')
     plt.title('Customer Rating Distribution')

     plt.show()
```

Output:

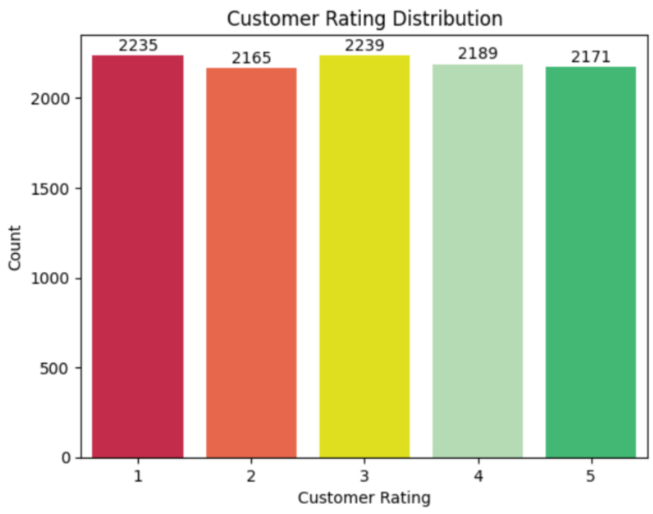


Figure 20 Customer Rating Distribution

Customer ratings provide direct feedback on how well the company is meeting customer expectations. Based on the customer ratings visualised from Alibaba's shipment forecast dataset, the customer's ratings towards the company are quite varied and evenly distributed across the scale. This means that customers have diverse experiences with the service. As shown, the customer ratings from 1 to 5 are closely similar in frequency, which indicates that there is a mixed opinion among the customers regarding the company's performance and their deliveries. To overcome the issue of mixed customer ratings, Alibaba can implement a comprehensive strategy focusing on improving service quality to improve their customer level of satisfaction

3.0 MODELLING

The model aims to predict the "Timeliness" of shipments using the provided features. The model would use the features to learn patterns and relationships that help predict whether a shipment will be timely or not. This involves training on historical data where the timeliness of shipments is already known, allowing the model to identify which features are most indicative of on-time or delayed deliveries.

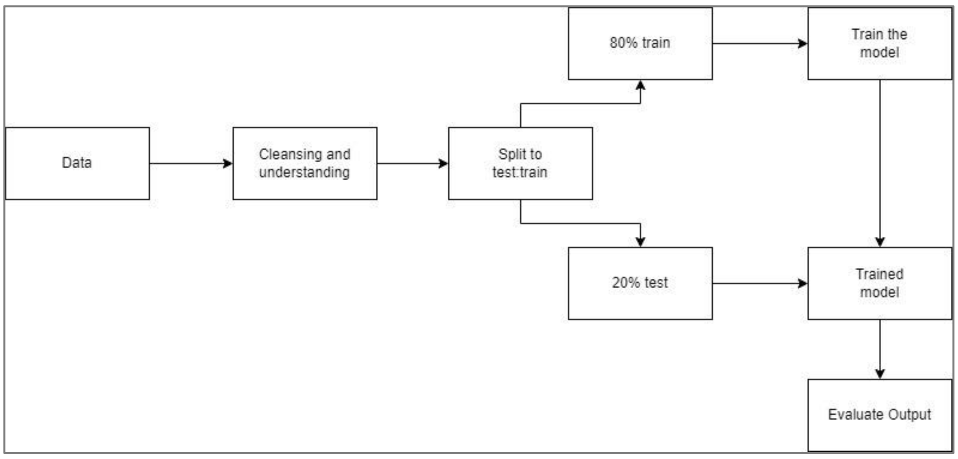


Figure 21 Machine Learning Flowchart

3.1 Machine Learning Algorithms

We utilize 10 algorithms for model training to compare and determine which one is best suited for predicting shipment timeliness. The algorithms include:

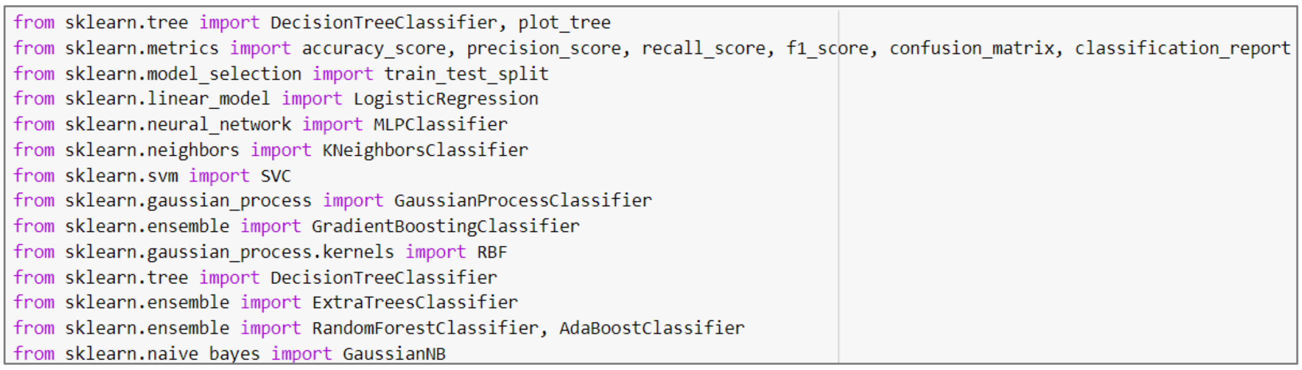
No	Machine Learning Algorithms	Types of Models
1.	Logistic Regression	Linear
2.	Neural Network	Neural Networks
3.	K-Nearest Neighbours	Instance-Based Learning
4.	Support Vector Machine	Support Vector Machine
5.	Gaussian Process	Probabilistic Models
6.	Decision Tree	Tree-Based Models
7.	Extra Trees	Tree-Based Models
8.	Random Forest	Tree-Based Models
9.	AdaBoost	Tree-Based Models
10.	Gaussian Naive Bayes	Probabilistic Models

All the algorithms under the category of **supervised learning**. In supervised learning, the model learns from labelled data where each data point has a corresponding label (target variable). In this case, the features (e.g., weight, priority, origin, destination) describe a shipment, and the label indicates whether it was timely (1) or non-timely (0). The model learns the relationship between features and timeliness to predict the outcome for unseen shipments.

3.2 Model Training

In this section, we outline the steps to train the machine learning models on the shipment timeliness dataset.

1. Import Necessary Libraries:
   The first step is to import all the necessary libraries that will be used throughout the model training process.

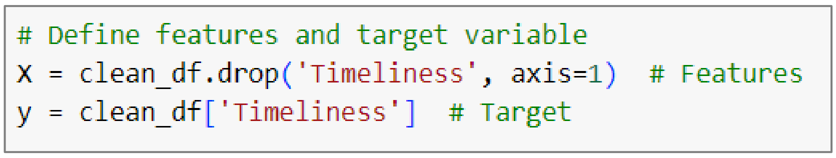


```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
```

Figure 22 Code to Import Libraries

2. Define Features and Target Variable:

   Next, we define the features and the target variable. The features (X) are the input variables that the model will use to make predictions, while the target (y) is the output variable that we want to predict.



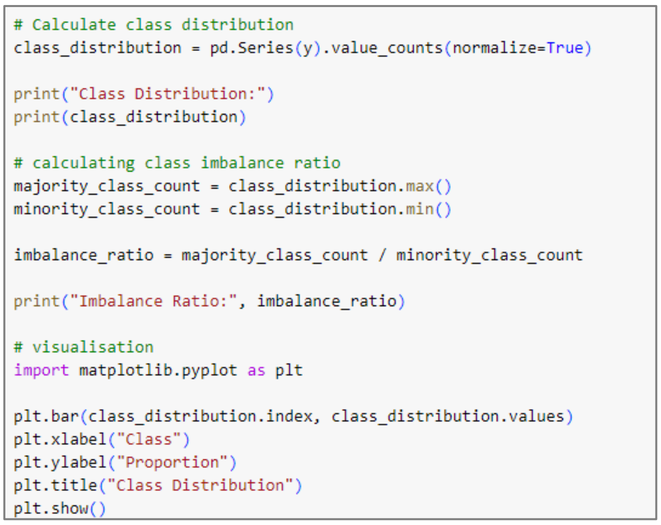
```python
# Define features and target variable
X = clean_df.drop('Timeliness', axis=1)  # Features
y = clean_df['Timeliness']  # Target
```

Figure 23 Code to define features and target variable

3. Balancing the classes

   To ensure that the model is performing well, it is important to check for class imbalances.

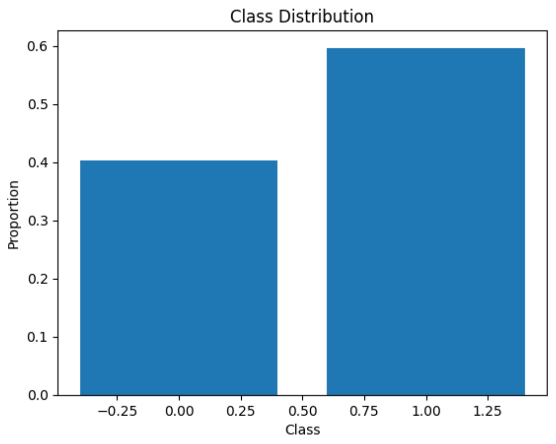
   i. Detecting class imbalance



```python
# Calculate class distribution
class_distribution = pd.Series(y).value_counts(normalize=True)

print("Class Distribution:")
print(class_distribution)

# calculating class imbalance ratio
majority_class_count = class_distribution.max()
minority_class_count = class_distribution.min()

imbalance_ratio = majority_class_count / minority_class_count

print("Imbalance Ratio:", imbalance_ratio)

# visualisation
import matplotlib.pyplot as plt

plt.bar(class_distribution.index, class_distribution.values)
plt.xlabel("Class")
plt.ylabel("Proportion")
plt.title("Class Distribution")
plt.show()
```

Figure 24 Code to Detect Class Imbalance by calculating Class Distribution

```
Class Distribution:
Timeliness
1    0.596691
0    0.403309
Name: proportion, dtype: float64
Imbalance Ratio: 1.4794860234445446
```

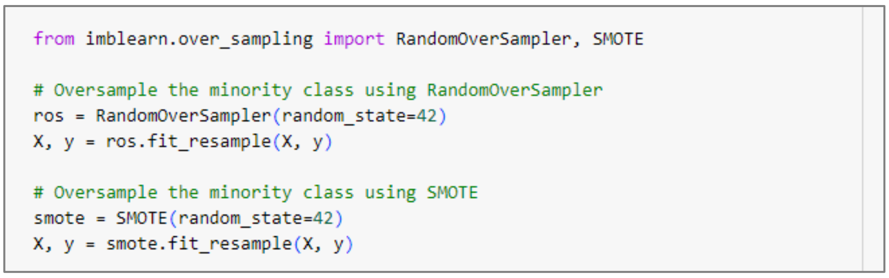


Class 1 has about 1.48 times more instances than Class 0, hence making it slightly imbalanced.

Figure 25 Class Distribution Output

To detect the class imbalance, we can calculate the proportion of each class in the target variable (y). And based on the output, Class 1 has about 1.48 times more instances than Class 0, hence making it slightly imbalanced.

ii. Addressing the class imbalance



```python
from imblearn.over_sampling import RandomOverSampler, SMOTE

# Oversample the minority class using RandomOverSampler
ros = RandomOverSampler(random_state=42)
X, y = ros.fit_resample(X, y)

# Oversample the minority class using SMOTE
smote = SMOTE(random_state=42)
X, y = smote.fit_resample(X, y)
```

Figure 26 Code to Address Class Imbalance

Using the method of Oversampling the Minority Class, we'll create additional instances of the minority class using the two functions below.

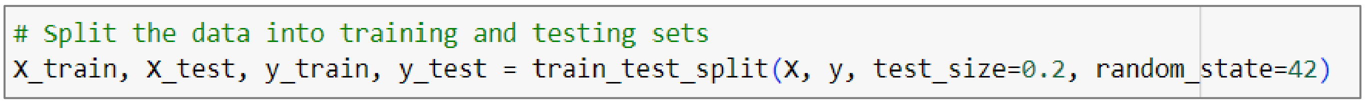
- Random Oversampling: Randomly duplicate instances from the minority class
- SMOTE (Synthetic Minority Over-sampling Technique): Create new instances by interpolating between existing minority class instances

4. Split the Data

Data splitting is a fundamental step in the machine learning workflow that ensures the models learn effectively and generalize well to unseen data. It involves dividing the dataset into distinct subsets used for different purposes:

i. Training Set:
This is the core dataset used to train your machine learning model. For this model, we used 80% of dataset for training. The model learns patterns and relationships within the training data, building its internal representation of the problem that need to be solved.

ii. Test Set:
This is the final hold-out set used for unbiased evaluation of the model's generalizability. We used 20% of the dataset for testing.

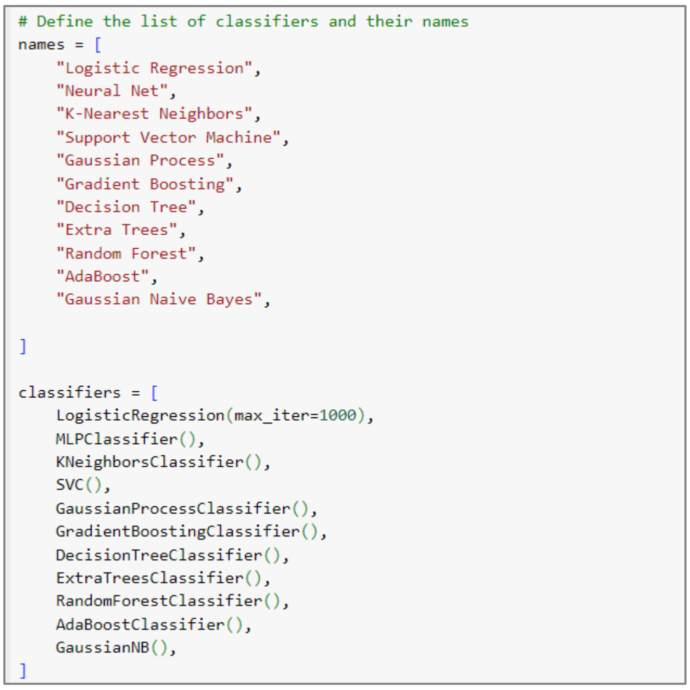


```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 27 Code for Splitting the Data

5. Define the Models

We define a list of machine learning classifiers and their names. Each model is trained using the training data, and its performance is evaluated on the test data.

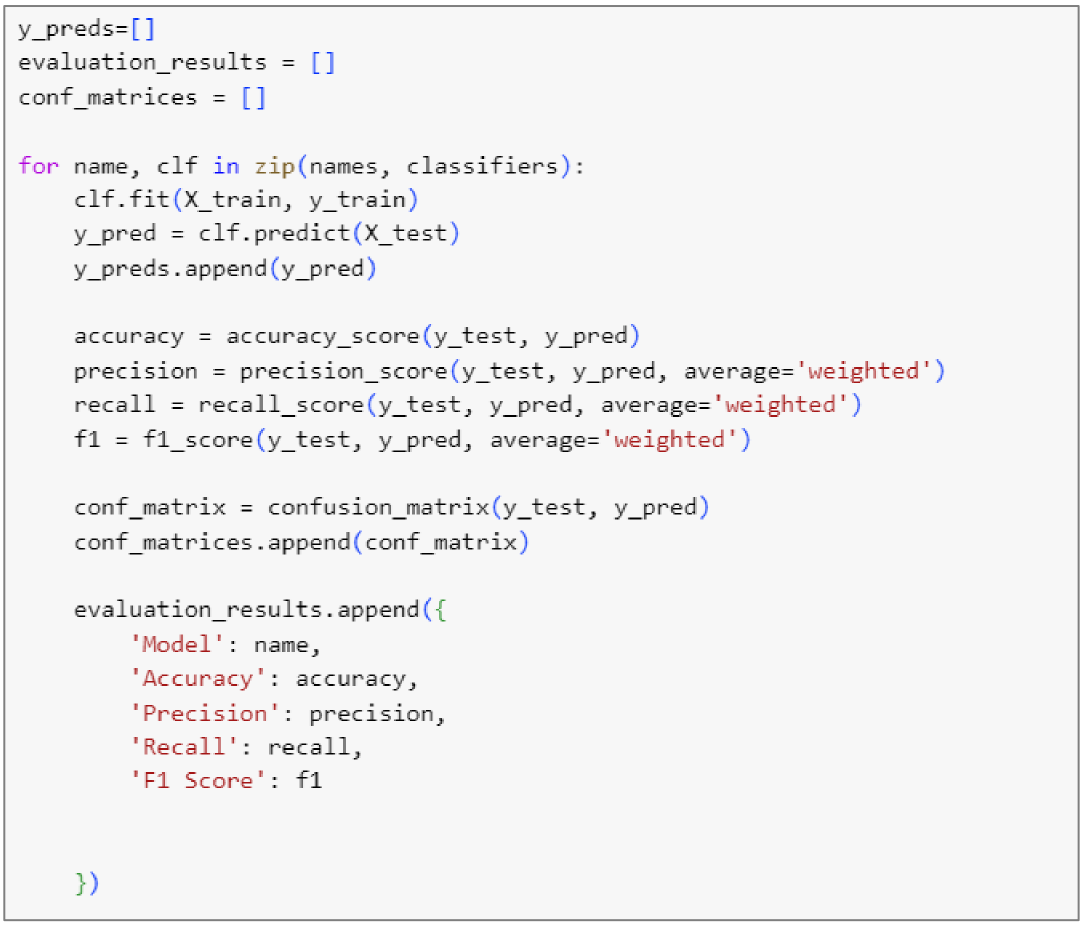


```
# Define the list of classifiers and their names
names = [
    "Logistic Regression",
    "Neural Net",
    "K-Nearest Neighbors",
    "Support Vector Machine",
    "Gaussian Process",
    "Gradient Boosting",
    "Decision Tree",
    "Extra Trees",
    "Random Forest",
    "AdaBoost",
    "Gaussian Naive Bayes",

]

classifiers = [
    LogisticRegression(max_iter=1000),
    MLPClassifier(),
    KNeighborsClassifier(),
    SVC(),
    GaussianProcessClassifier(),
    GradientBoostingClassifier(),
    DecisionTreeClassifier(),
    ExtraTreesClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GaussianNB(),
]
```

Figure 28 Code to define models' names and classifiers

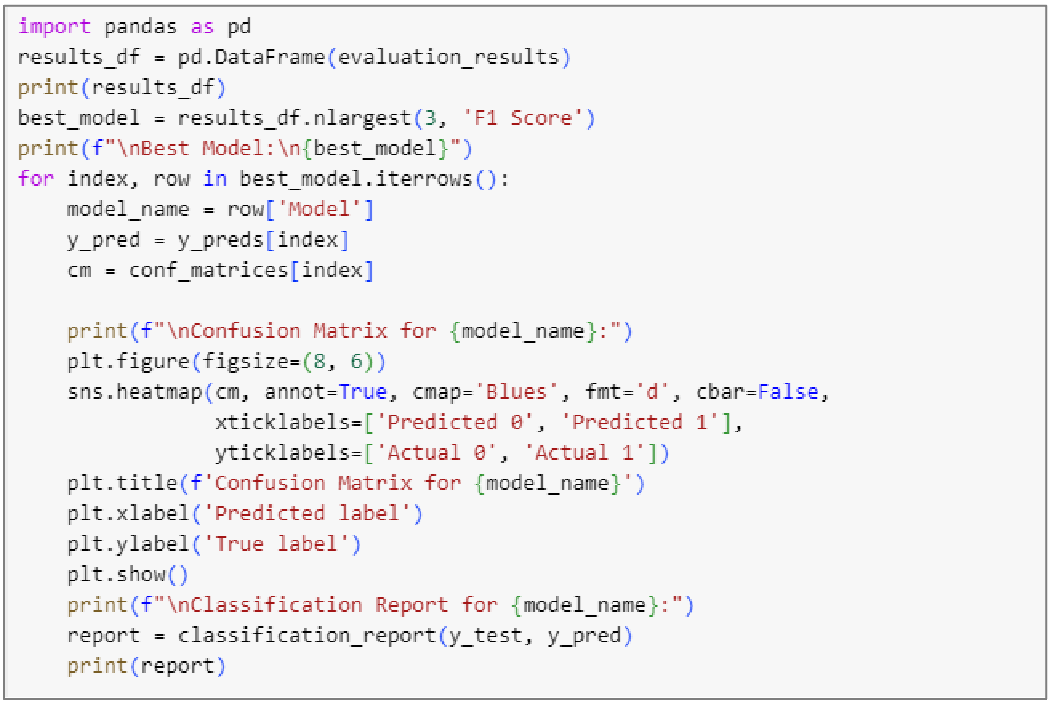
6. Train and evaluate the models

   Each model is trained using the training data, and its performance is evaluated on the test data.



```python
y_preds=[]
evaluation_results = []
conf_matrices = []

for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    y_preds.append(y_pred)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    conf_matrix = confusion_matrix(y_test, y_pred)
    conf_matrices.append(conf_matrix)

    evaluation_results.append({
        'Model': name,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1


    })
```

Figure 29 Code to train and evaluate the model



```python
import pandas as pd
results_df = pd.DataFrame(evaluation_results)
print(results_df)
best_model = results_df.nlargest(3, 'F1 Score')
print(f"\nBest Model:\n{best_model}")
for index, row in best_model.iterrows():
    model_name = row['Model']
    y_pred = y_preds[index]
    cm = conf_matrices[index]

    print(f"\nConfusion Matrix for {model_name}:")
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, cmap='Blues', fmt='d', cbar=False,
                xticklabels=['Predicted 0', 'Predicted 1'],
                yticklabels=['Actual 0', 'Actual 1'])
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.show()
    print(f"\nClassification Report for {model_name}:")
    report = classification_report(y_test, y_pred)
    print(report)
```

Figure 30 Code to show the output for the evaluation

3.3 Results and Evaluation

3.3.1 Analysis of Model Performance

No	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.688500	0.719038	0.688500	0.678076
1	Neural Net	0.643184	0.643776	0.643184	0.642562
2	K-Nearest Neighbors	0.705636	0.714031	0.705636	0.703122
3	Support Vector Machine	0.685072	0.689915	0.685072	0.683422
4	Gaussian Process	0.732673	0.735197	0.732673	0.732128
5	Gradient Boosting	0.730769	0.799626	0.730769	0.715129
6	Decision Tree	0.749429	0.751366	0.749429	0.749077
7	Extra Trees	0.749048	0.764316	0.749048	0.745737
8	Random Forest	0.746002	0.778861	0.746002	0.738816
9	AdaBoost	0.726200	0.800626	0.726200	0.708952
10	Gaussian Naive Bayes	0.702209	0.813084	0.702209	0.674356

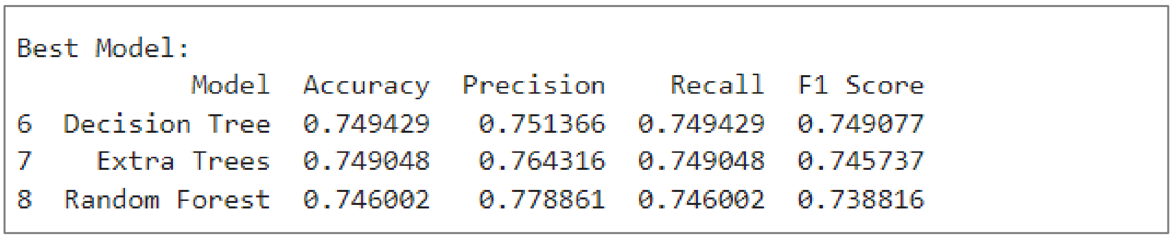

```
Best Model:
           Model  Accuracy  Precision    Recall  F1 Score
6  Decision Tree  0.749429   0.751366  0.749429  0.749077
7    Extra Trees  0.749048   0.764316  0.749048  0.745737
8  Random Forest  0.746002   0.778861  0.746002  0.738816
```

Figure 31 Top 3 Best Model

Model Performance Summary

This table summarizes the performance of various machine learning models that we have evaluated on the classification task. The models are listed in the first column ("Model") and the remaining columns display different evaluation metrics:

i) **Accuracy**: The proportion of correctly predicted examples (both positive and negative).

ii) **Precision**: The proportion of positive predictions that were actually correct (positive predictive value).

iii) **Recall**: The proportion of actual positive examples that were correctly identified (sensitivity).

iv) **F1 Score**: A harmonic mean of precision and recall, combining both measures.

Results Evaluation:

- Overall Performance: The models seem to perform reasonably well, with accuracy scores ranging from 0.64 to 0.73.
- Best Performing Models: Based on accuracy, the top three performers are:
  1) Decision Tree with an accuracy of 0.749429
  2) Extra Trees with an accuracy of 0.749048
  3) Random Forest with an accuracy of 0.746002

Observations on Other Metrics:

- Decision Tree, Extra Trees, and Random Forest generally achieve good precision, meaning a high proportion of their positive predictions are correct.
- Random Forest seems to have the highest precision, but slightly lower accuracy compared to Decision Tree and Extra Trees. This might indicate some trade-off between precision and recall.
- Gaussian Naive Bayes has high precision but lower recall and F1 score, suggesting it might miss some true positive examples.

Conclusion:

The **Decision Tree** classifier performed the best on the dataset, achieving a balanced performance across accuracy, precision, recall, and F1 score. This indicates that the model is effective at both identifying timely shipments and minimizing false positives.

3.3.2 Model Visualization

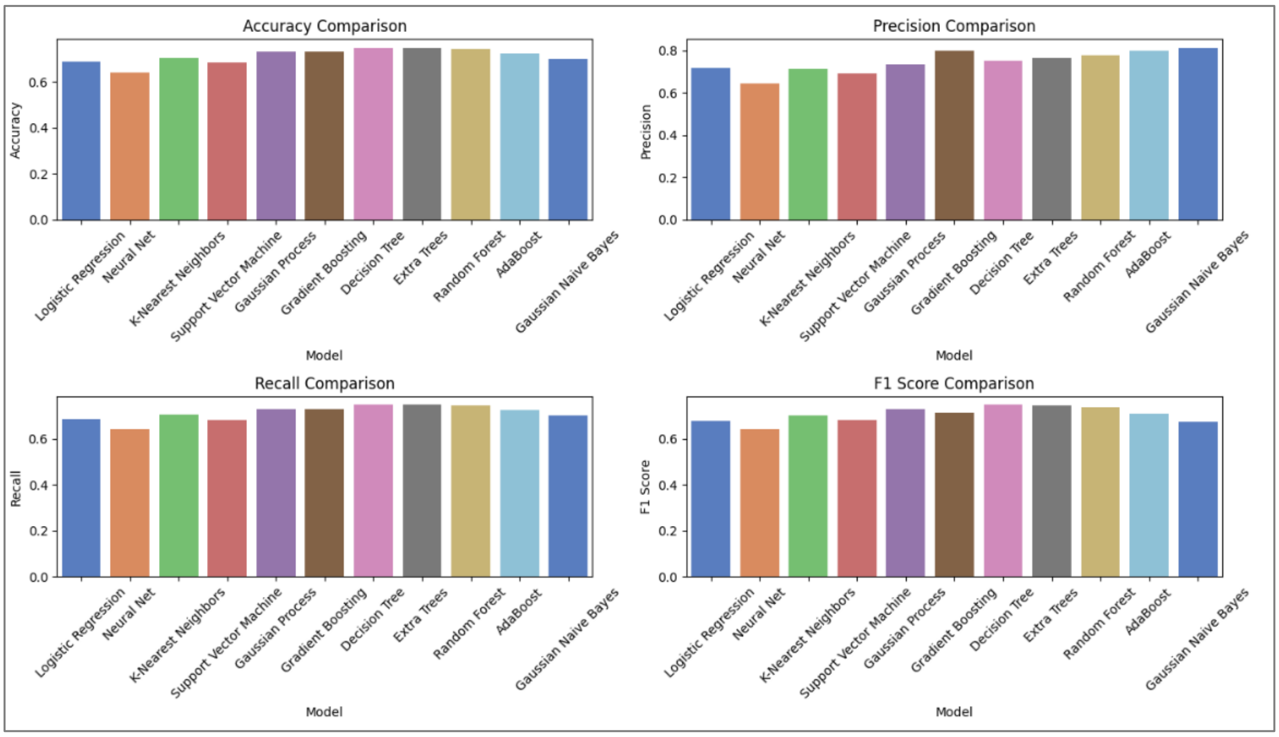
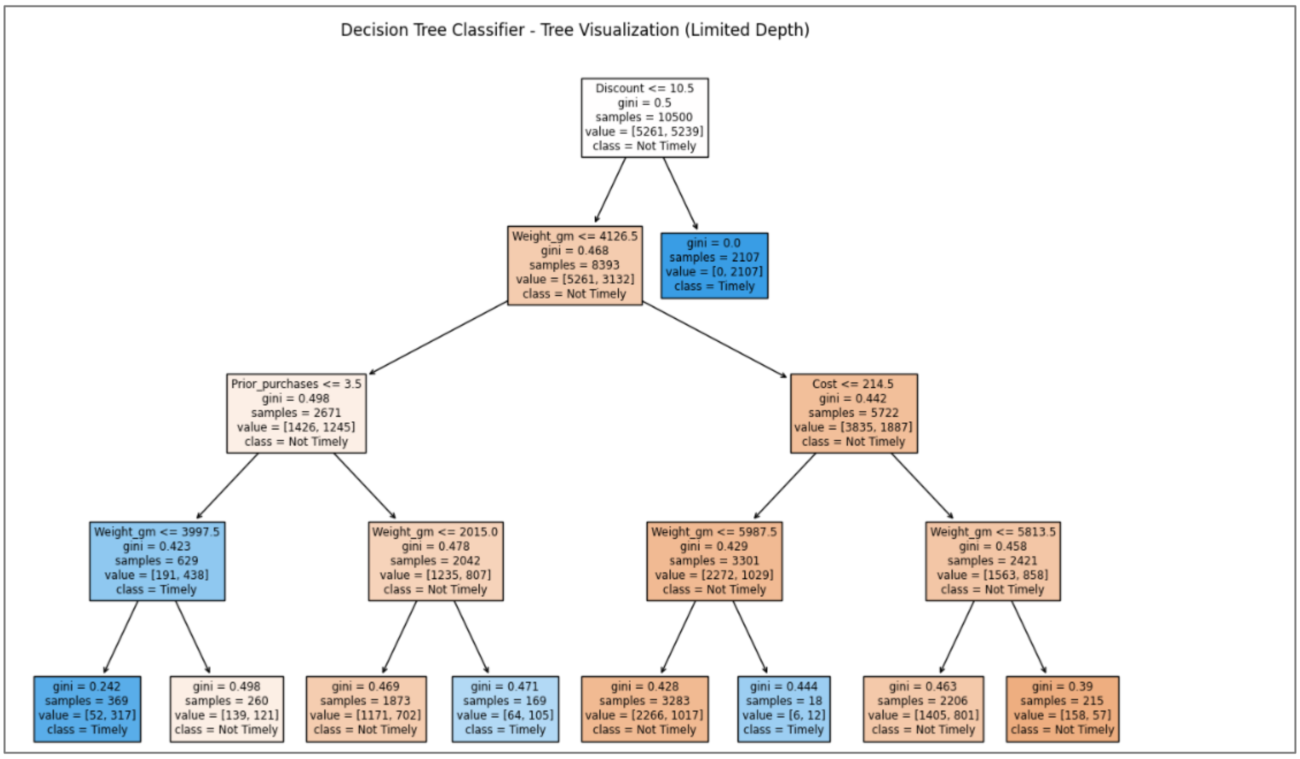


Figure 32 Comparison on Different Model



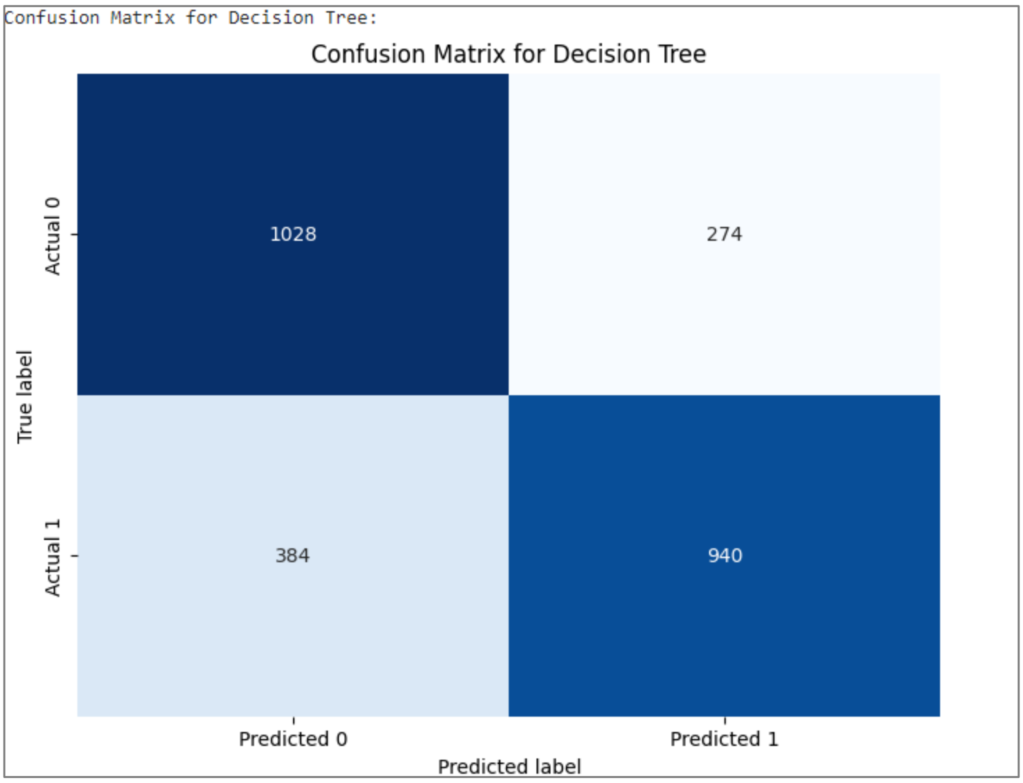
Decision Tree Classifier - Tree Visualization (Limited Depth)
Discount <= 10.5 gini = 0.5 samples = 10500 value = [5261, 5239] class = Not Timely
Weight_gm <= 4126.5 gini = 0.468 samples = 8393 value = [5261, 3132] class = Not Timely
gini = 0.0 samples = 2107 value = [0, 2107] class = Timely
Prior_purchases <= 3.5 gini = 0.498 samples = 2671 value = [1426, 1245] class = Not Timely
Cost <= 214.5 gini = 0.442 samples = 5722 value = [3835, 1887] class = Not Timely
Weight_gm <= 3997.5 gini = 0.423 samples = 629 value = [191, 438] class = Timely
Weight_gm <= 2015.0 gini = 0.478 samples = 2042 value = [1235, 807] class = Not Timely
Weight_gm <= 5987.5 gini = 0.429 samples = 3301 value = [2272, 1029] class = Not Timely
Weight_gm <= 5813.5 gini = 0.458 samples = 2421 value = [1563, 858] class = Not Timely
gini = 0.242 samples = 369 value = [52, 317] class = Timely
gini = 0.498 samples = 260 value = [139, 121] class = Not Timely
gini = 0.469 samples = 1873 value = [1171, 702] class = Not Timely
gini = 0.471 samples = 169 value = [64, 105] class = Timely
gini = 0.428 samples = 3283 value = [2266, 1017] class = Not Timely
gini = 0.444 samples = 18 value = [6, 12] class = Timely
gini = 0.463 samples = 2206 value = [1405, 801] class = Not Timely
gini = 0.39 samples = 215 value = [158, 57] class = Not Timely

Figure 33 Decision Tree

Confusion Matrix

A Confusion Matrix is a table used to evaluate the performance of a machine learning model on a test dataset. It provides a summary of the predictions against the actual true labels.

Confusion Matrix for the 3 best models:



```
Confusion Matrix for Decision Tree:
```

Figure 34 Confusion Matrix (Decision Tree)

```
Confusion Matrix for Extra Trees:
```

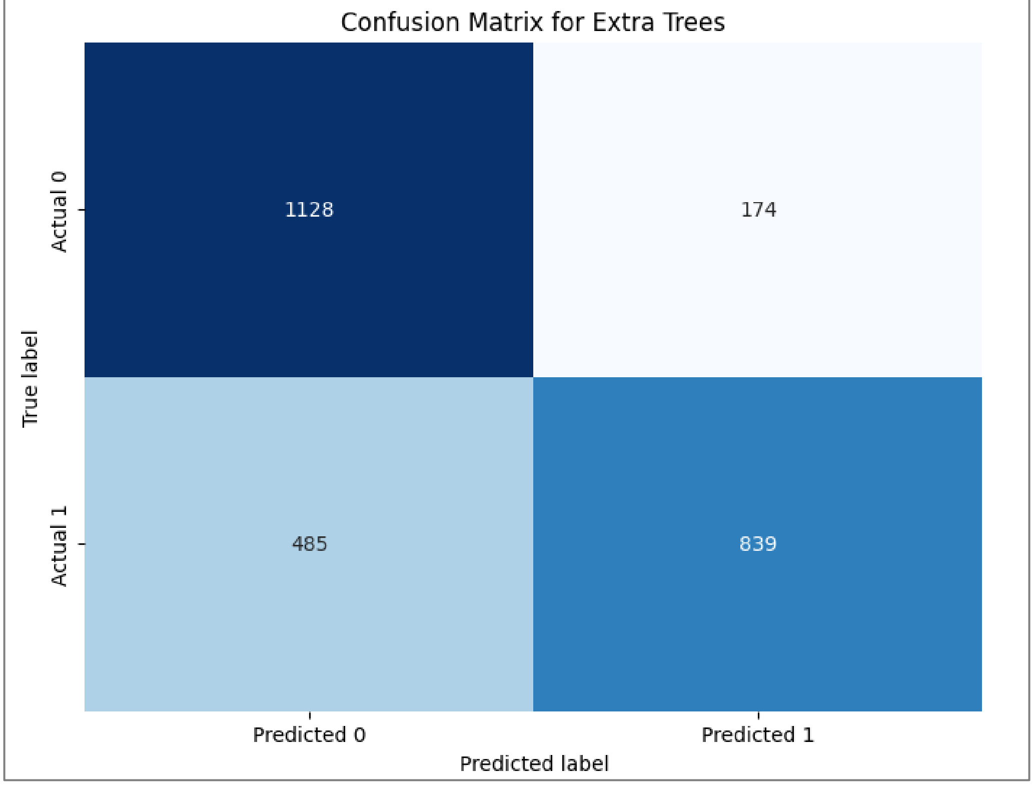


Figure 35 Confusion Matrix (Extras Trees)

```
Confusion Matrix for Random Forest:
```

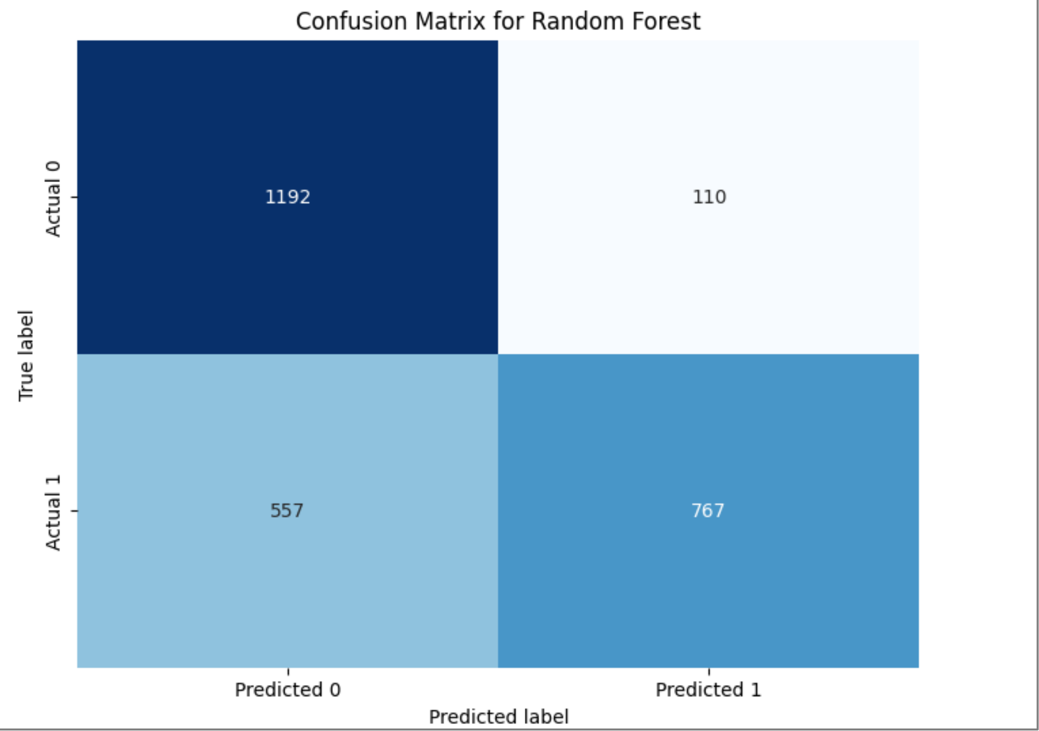


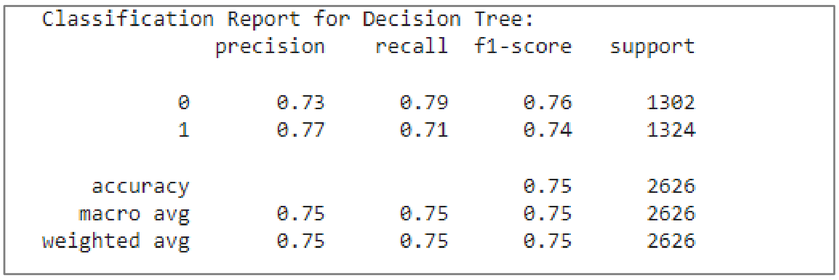
Figure 36 Confusion Matrix (Random Forest)

3.3.3 Classification Report

A Classification Report is a summary of the performance of a machine learning model on a classification problem. It provides a detailed breakdown of the model's performance on each class, including precision, recall, F1 score, and support.

Classification Report for 3 best models:

i) Decision Tree

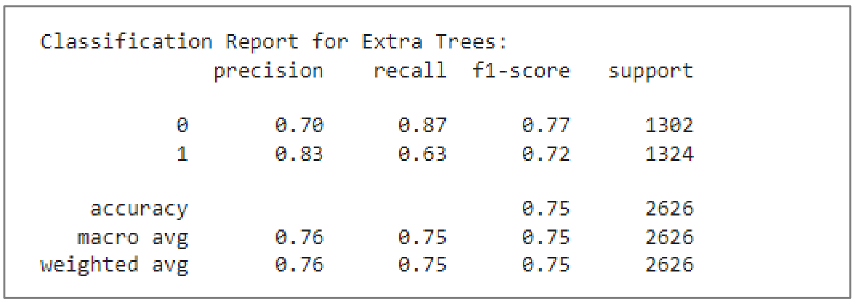


```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.73      0.79      0.76      1302
           1       0.77      0.71      0.74      1324

    accuracy                           0.75      2626
   macro avg       0.75      0.75      0.75      2626
weighted avg       0.75      0.75      0.75      2626
```

Figure 37 Classification Report (Decision Tree)

- Overall: Achieves an accuracy of 75%.
- Class 0: Performs reasonably well with a precision of 0.73 and recall of 0.79.
- Class 1: Slightly lower performance with a precision of 0.77 and recall of 0.71.

ii) Extras Tree



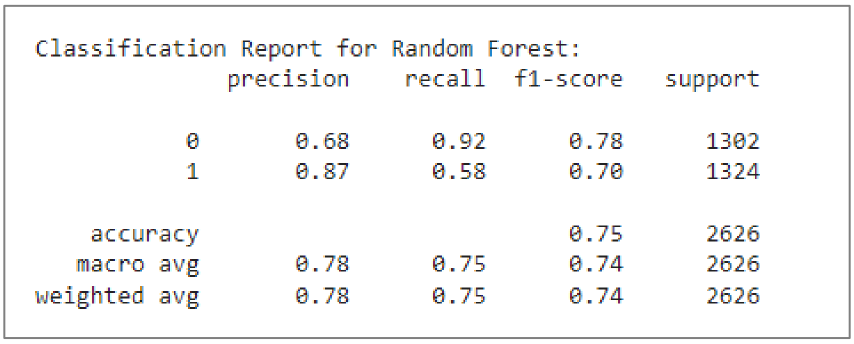
```
Classification Report for Extra Trees:
              precision    recall  f1-score   support

           0       0.70      0.87      0.77      1302
           1       0.83      0.63      0.72      1324

    accuracy                           0.75      2626
   macro avg       0.76      0.75      0.75      2626
weighted avg       0.76      0.75      0.75      2626
```

Figure 38 Classification Report (Extras Trees)

- Overall: Similar accuracy of 75% to Decision Tree.
- Class 0: Lower precision (0.70) compared to Decision Tree but higher recall (0.87). This suggests it might predict more positive examples (some might be false positives).

- Class 1: Lower performance in both precision (0.83) and recall (0.63) compared to Decision Tree.

iii) Random Forest



```
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.68      0.92      0.78      1302
           1       0.87      0.58      0.70      1324

    accuracy                           0.75      2626
   macro avg       0.78      0.75      0.74      2626
weighted avg       0.78      0.75      0.74      2626
```

Figure 39 Classification Report (Random Forest)

- Overall: Accuracy of 75%, similar to the other models.
- Class 0: Similar performance to Decision Tree but with slightly lower precision (0.68) and higher recall (0.92). Similar trend as Extra Trees, potentially predicting more positive examples (some might be incorrect).
- Class 1: Lower performance in both precision (0.87) and recall (0.58) compared to Decision Tree. Shows the most difficulty in correctly identifying Class 1 examples.

Summary:

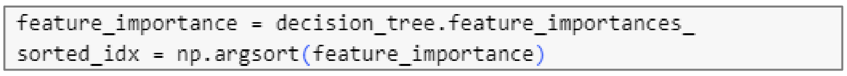
- All 3 best models achieve similar overall accuracy (around 75%).
- Decision Tree seems to have a more balanced performance across both classes.
- Extra Trees and Random Forest show a trade-off: They might identify more positive examples (potentially including some false positives) in Class 0 but struggle with Class 1.
- Random Forest has the lowest performance in identifying Class 1 examples (low recall of 0.58) among the three.

3.3.4 Feature Importances

Feature importance is a technique used in machine learning to determine the most significant variables or features that contribute the most to predicting the target variable in a model. It helps identify which features have the most influence on the model's predictions and can provide insights into the underlying relationships within the dataset. In this project, Decision Tree, Extras Trees and Random Forest models are identified as the 3 best models, feature importance analysis helps prioritize which attributes such as discount, weight, and cost play the most crucial roles in predicting shipment timeliness.

Feature Importances for the 3 best models:

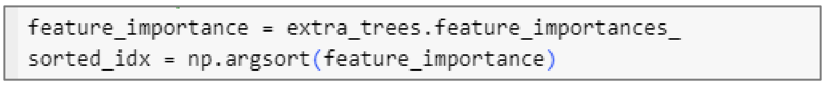
i) Decision Tree



```
feature_importance = decision_tree.feature_importances_
sorted_idx = np.argsort(feature_importance)
```

Figure 40 Code to define and sort feature importances (Decision Tree)

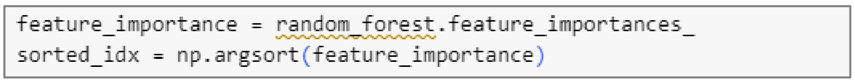
ii) Extras Trees



```
feature_importance = extra_trees.feature_importances_
sorted_idx = np.argsort(feature_importance)
```

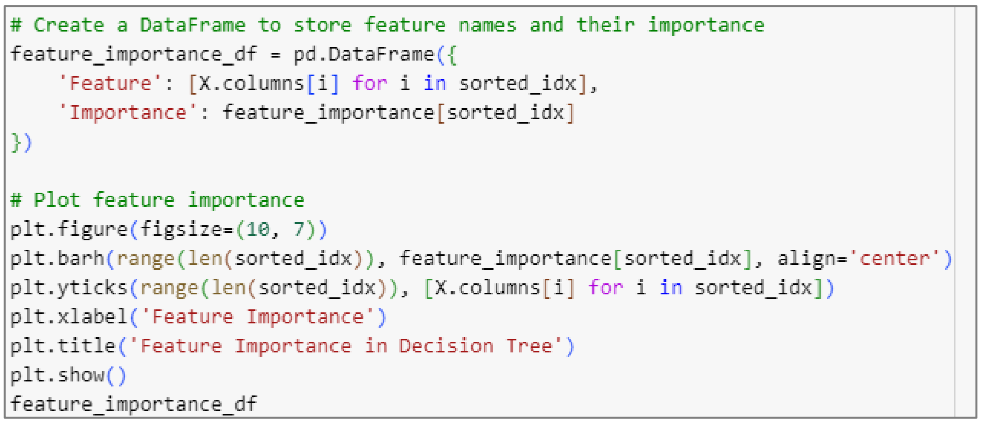
Figure 41 Code to define and sort feature importances (Extras Trees)

iii) Random Forest



```
feature_importance = random_forest.feature_importances_
sorted_idx = np.argsort(feature_importance)
```

Figure 42 Code to define and sort feature importances (Radom Forest)



```
# Create a DataFrame to store feature names and their importance
feature_importance_df = pd.DataFrame({
    'Feature': [X.columns[i] for i in sorted_idx],
    'Importance': feature_importance[sorted_idx]
})

# Plot feature importance
plt.figure(figsize=(10, 7))
plt.barh(range(len(sorted_idx)), feature_importance[sorted_idx], align='center')
plt.yticks(range(len(sorted_idx)), [X.columns[i] for i in sorted_idx])
plt.xlabel('Feature Importance')
plt.title('Feature Importance in Decision Tree')
plt.show()
feature_importance_df
```

Figure 43 Code to plot feature importance

No	Feature	Decision Tree	Random Forest	Extras Trees
0	Gender	0.014895	0.022905	0.023971
1	Ship_mode	0.030919	0.030792	0.037648
2	Priority	0.030053	0.032201	0.039173
3	Cust_calls	0.044488	0.049693	0.066444
4	Cust_rating	0.047477	0.052549	0.067039
5	Block	0.055252	0.055473	0.073843
6	Prior_purchases	0.053712	0.056831	0.077021
7	Cost	0.146266	0.169032	0.141421
8	Weight_gm	0.243563	0.248636	0.224413
9	Discount	0.333376	0.281887	0.249026

Visualization on Feature Importance

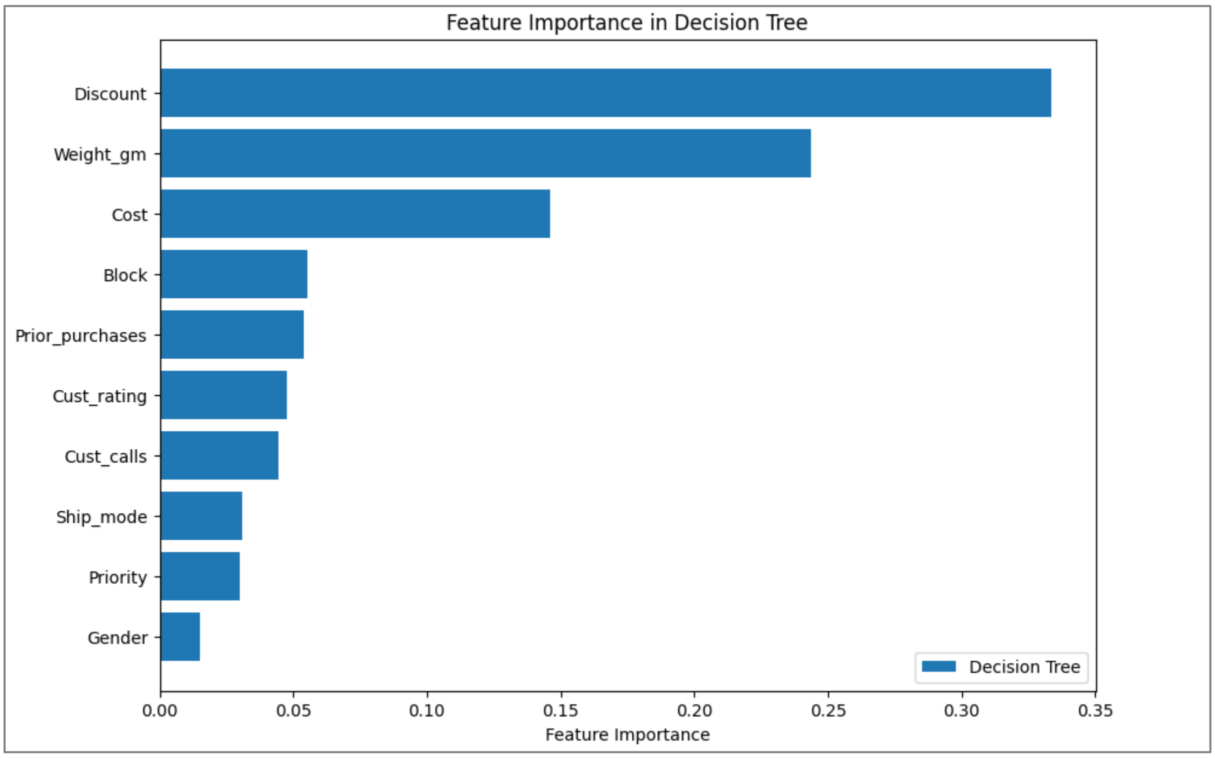


Figure 44 Feature Importance (Decision Tree)

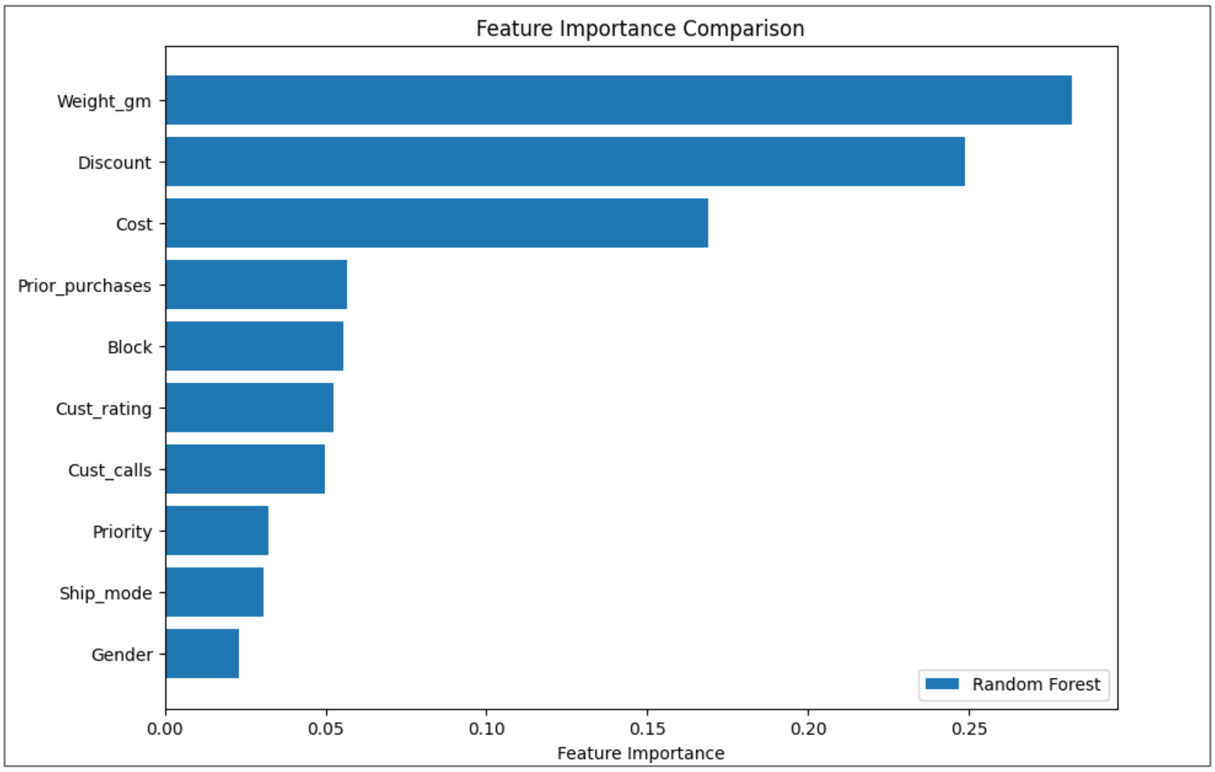


Figure 45 Feature Importance (Random Forest)

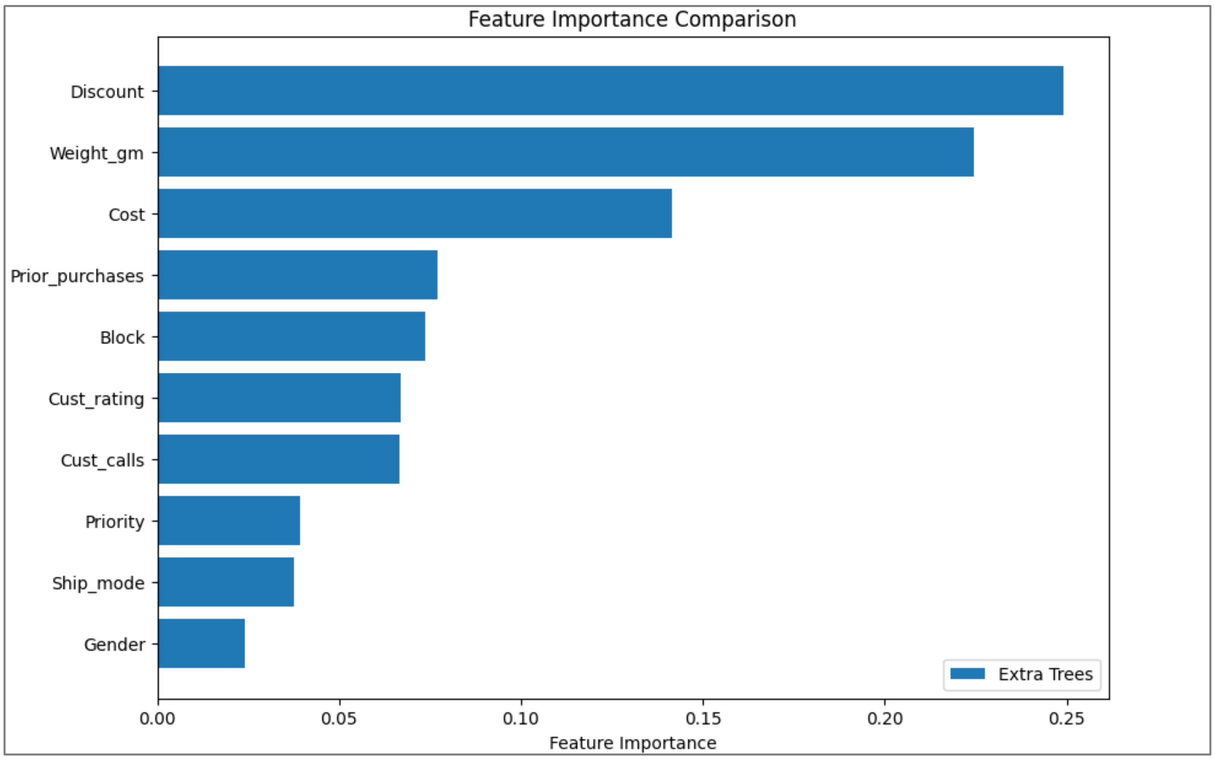


Figure 46 Feature Importance (Extras Trees)

From the table and figures above, we can observe the following trends:

- Weight_gm and Discount are consistently among the top 2 most important features across all three models.
- Cost is also consistently important, ranking 3rd or 4th in all three models.
- Block, Prior_purchases, and Cust_rating have moderate importance, with similar rankings across the three models.
- Cust_calls has varying importance, ranking 3rd in Decision Tree, 4th in Random Forest, and 5th in Extra Trees.
- Priority and Ship_mode have relatively low importance, with varying rankings across the three models.
- Gender has the lowest importance, with similar rankings across the three models.

From the table, we can also observe the following model-specific insights:

- Decision Tree: Discount is the most important feature, followed closely by Weight_gm. Cost is also relatively important.
- Random Forest: Weight_gm is the most important feature, followed closely by Discount. Cost is also relatively important.
- Extra Trees: Discount is the most important feature, followed closely by Weight_gm. Cost is also relatively important.

Comparison of Model Performance

The table suggests that the three models have similar performance in terms of feature importance. However, there are some differences in the ranking of features across the models. This could be due to the different algorithms and hyperparameters used in each model.

3.4 Prediction using Trained Model on New Data

The provided code demonstrates how to use a trained machine learning model to predict the timeliness of a new shipment on unseen data.

1) Chose a trained machine learning model
   i) DecisionTreeClassifier

```
chosen_model = classifiers[6]
```

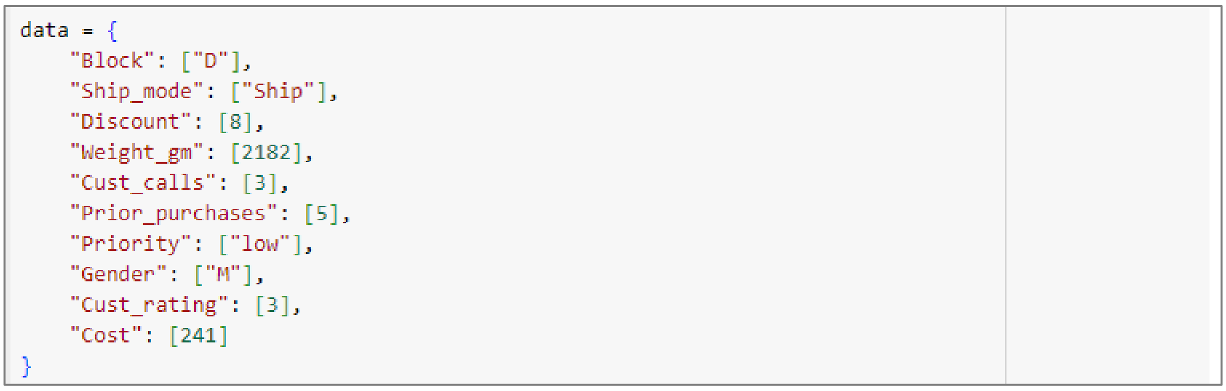
   ii) ExtraTreesClassifier,

```
chosen_model = classifiers[7]
```

   iii) RandomForestClassifier

```
chosen_model = classifiers[8]
```

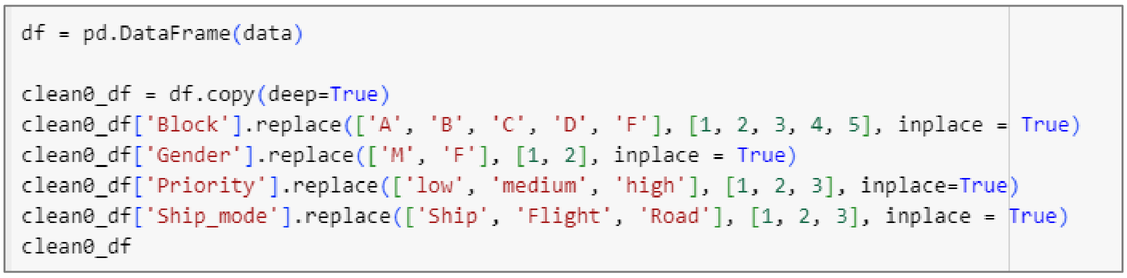
2) Creating New Data (Shipment Information):



```
data = {
    "Block": ["D"],
    "Ship_mode": ["Ship"],
    "Discount": [8],
    "Weight_gm": [2182],
    "Cust_calls": [3],
    "Prior_purchases": [5],
    "Priority": ["low"],
    "Gender": ["M"],
    "Cust_rating": [3],
    "Cost": [241]
}
```

Figure 47 Code to Create New Data

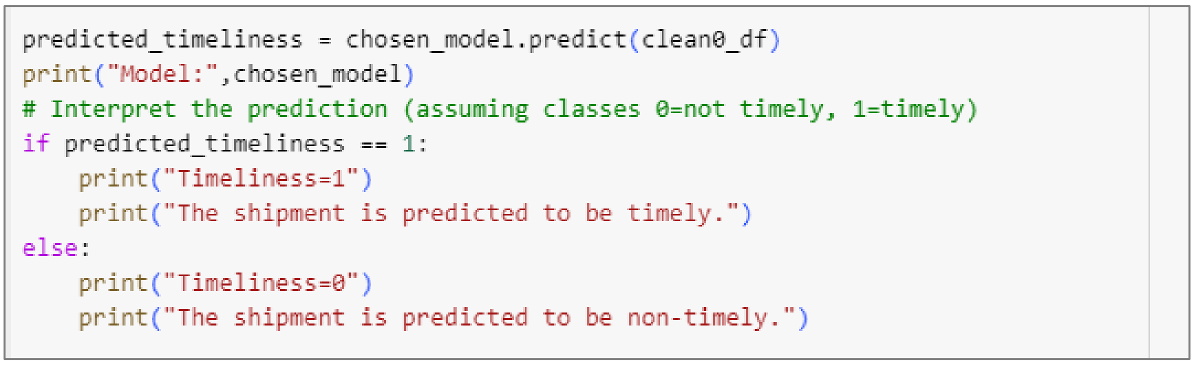
3) Data Preprocessing



```
df = pd.DataFrame(data)

clean0_df = df.copy(deep=True)
clean0_df['Block'].replace(['A', 'B', 'C', 'D', 'F'], [1, 2, 3, 4, 5], inplace = True)
clean0_df['Gender'].replace(['M', 'F'], [1, 2], inplace = True)
clean0_df['Priority'].replace(['low', 'medium', 'high'], [1, 2, 3], inplace=True)
clean0_df['Ship_mode'].replace(['Ship', 'Flight', 'Road'], [1, 2, 3], inplace = True)
clean0_df
```

Figure 48 Code to clean the new data

4) Prediction on New Data

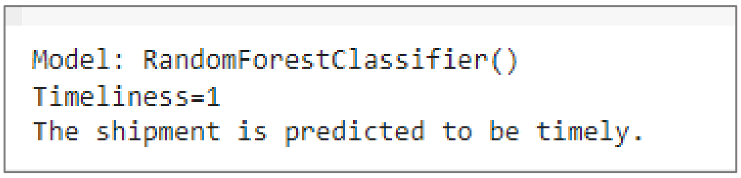


```python
predicted_timeliness = chosen_model.predict(clean0_df)
print("Model:",chosen_model)
# Interpret the prediction (assuming classes 0=not timely, 1=timely)
if predicted_timeliness == 1:
    print("Timeliness=1")
    print("The shipment is predicted to be timely.")
else:
    print("Timeliness=0")
    print("The shipment is predicted to be non-timely.")
```

Figure 49 Code to do the prediction on the new data

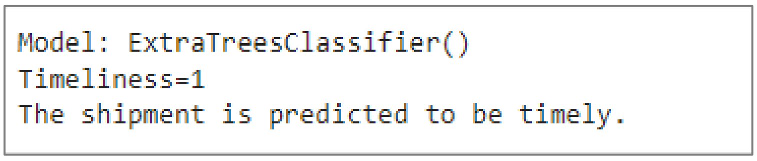
Output:

1) Random Forest



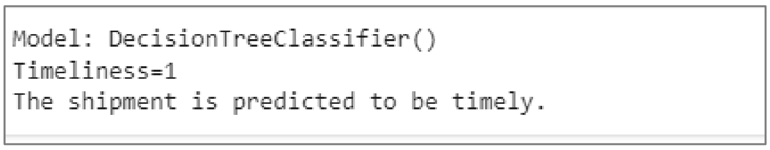
```
Model: RandomForestClassifier()
Timeliness=1
The shipment is predicted to be timely.
```

2) Extras Trees



```
Model: ExtraTreesClassifier()
Timeliness=1
The shipment is predicted to be timely.
```

3) Decision Tree



```
Model: DecisionTreeClassifier()
Timeliness=1
The shipment is predicted to be timely.
```

Explanation:

- All three models utilize the same features and aim to achieve the same goal: predict timeliness based on historical data.
- Consequently, they're likely to learn similar decision rules that differentiate timely and non-timely shipments.
- If a new data point (a customer order) exhibit features that historically correlated with timely deliveries according to the training data, all three models are likely to predict "Timeliness=1".

SUMMARY

Data visualization transforms complex data into visual insights that can be easily understood by decision-makers. Several visualizations have been presented to explore and understand the correlations among different data within the dataset. Some of the visualizations that have been made include plotting pie charts, bar charts, and stacked charts to understand the distributions of shipping modes, customer calls, and customer ratings. Through these data visualizations, various insights were uncovered, aiding Alibaba in making cost-effective decisions, strategizing for operational efficiencies, as well as enhancing customers' level of satisfaction by addressing their issues.

The data processing for predicting shipment timeliness involved several key steps to prepare and evaluate multiple machine learning models. Initially, the dataset was loaded and preprocessed, handling missing values and encoding categorical variables as needed. Then, we will define the features (shipment characteristics) and the target variable (timely or non-timely). As the data has an imbalanced class distribution, we use method of Oversampling the Minority Class to create additional instances of the minority class to address this imbalance.

The dataset was split into training and testing sets, with 80% used for training to allow models to learn patterns and relationships, and 20% reserved for testing to evaluate model generalization. Ten different algorithms were trained, including Logistic Regression, Neural Network, K-Nearest Neighbors, Support Vector Machine, Gaussian Process, Decision Tree, Extra Trees, Random Forest, AdaBoost and Gaussian Naive Bayes.

Each model's performance was evaluated using metrics such as accuracy, precision, recall, and F1 score, with particular emphasis on the F1 score to account for class imbalance in the target variable. The top three models based on their F1 scores were identified and further analyzed with confusion matrices and detailed classification reports to understand their predictive capabilities. Overall, this process aimed to select the best-performing model for predicting shipment timeliness, providing insights into model strengths and areas for improvement in future iterations of the analysis.

REFERENCES

GeeksforGeeks. (2024, June 11). *What is Data Visualization and Why is It Important?* GeeksforGeeks. https://www.geeksforgeeks.org/data-visualization-and-its-importance/

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). Tree-Based Methods. Springer Texts in Statistics. https://doi.org/10.1017/cbo9780511842061.012.

Sahoo, R., Pasayat, A. K., Bhowmick, B., Fernandes, K., & Tiwari, M. K. (2021). A hybrid ensemble learning-based prediction model to minimise delay in air cargo transport using bagging and stacking. *International Journal of Production Research*, *60*(2), 644–660. https://doi.org/10.1080/00207543.2021.2013563

Shahid, S. (2020). *Predicting delays in delivery process using Machine Learning-Based approach*. https://doi.org/10.25394/pgs.13350764.v1