A PROJECT REPORT ON

SUPERMARKET DATASET FOR PREDICTIVE MARKETING

Submitted to Osmania University

in partial fulfilment of the requirements for the award of

MASTER OF SCIENCEIN STATISTICS

DEPARTMENT OF STATISTICS

BANKATLAL BADRUKA COLLEGE FOR INFORMATION TECHNOLOGY

(AFFILIATED TO OSMANIA UNIVERSITY),KACHIGUDA HYDERABAD

BY

DEVARAKONDA SATYA ALEKHYA	**Roll No: 1066-21-507-001**
CHERUKU UMA DEVI	**Roll No: 1066-21-507-009**
AKKENA NITYA	**Roll No: 1066-21-507-017**
MALE NIKITHA	**Roll No: 1066-21-507-027**
SHAIK SANA	**Roll No: 1066-21-507-036**
GOUDAMPALLY ARCHANA	**Roll No: 1066-21-507-045**

Under the Supervision of

Dr. M. VENUGOPALA RAO

2023

A PROJECT REPORT ON

SUPERMARKET DATASET FOR PREDICTIVE MARKETING

Submitted to Osmania University

in partial fulfilment of the requirements for the award of

MASTER OF SCIENCE IN STATISTICS

DEPARTMENT OF STATISTICS

BANKATLAL BADRUKA COLLEGE FOR INFORMATION TECHNOLOGY (AFFILIATED TO OSMANIA UNIVERSITY),KACHIGUDA,HYDERABAD

By

DEVARAKONDA SATYA ALEKHYA	**Roll No: 1066-21-507-001**
CHERUKU UMA DEVI	**Roll No: 1066-21-507-009**
AKKENA NITYA	**Roll No: 1066-21-507-017**
MALE NIKITHA	**Roll No: 1066-21-507-027**
SHAIK SANA	**Roll No: 1066-21-507-036**
GOUDAMPALLY ARCHANA	**Roll No: 1066-21-507-045**

Under the supervision of

Mr.M.VENUGOPALA RAO

2023

CERTIFICATE

This is to certify that

DEVARAKONDA SATYA ALEKHYA	**Roll No: 1066-21-507-001**
CHERUKU UMA DEVI	**Roll No: 1066-21-507-009**
AKKENA NITYA	**Roll No: 1066-21-507-017**
MALE NIKITHA	**Roll No: 1066-21-507-027**
SHAIK SANA	**Roll No: 1066-21-507-036**
GOUDAMPALLY ARCHANA	**Roll No: 1066-21-507-045**

Have submitted the project titled **"SUPERMARKET DATASET FOR PREDICTIVE MARKETING"** in partial fulfilment for the degree of Master of Science in Statistics

Head

Department of Statistics

Internal Examiner

External Examiner

DECLARATION

The research presented in this project has been carried out in the **Department of Statistics, Bankatlal Badruka College for Information technology, Kachiguda, Hyderabad**. The work is original has not been submitted so far, in part or full, for any other degree of diploma of any university

DEVARAKONDA SATYA ALEKHYA

CHERUKU UMA DEVI

AKKENA NITYA

MALE NIKITHA

SHAIK SANA

GOUDAMPALLY ARCHANA

Department of Statistics

BBCIT

Hyderabad – 500027, T.S.

INDIA

ACKNOWLEDGEMENT

I deem it a great pleasure to express my deep sense of gratitude and indebtedness to my DMMLT faculty **Dr. M.VENUGOPALA RAO,** GM just smartmandate, Data science, for his valuable guidance, and enlightening discussions throughout the progress of my project work.

I also express my sincere and heartfelt thanks Co-Ordinator, KARANAM LAKSHMI, BBCIT for providing the necessary support and facilities in the department for completion of this work successfully.

It is indeed with great pleasure I record my thanks to Director **Dr.DVNS MURTHY**, Department of Statistics, BBCIT for havingprovided with all the facilities to carry out our work.

I thank all Non-Teaching members of the Department of Statistics, who helped me during my thesis work.

I am thankful to the BBCIT for permitting me to carry out this work.

CONTENTS

Chapter - I

INTRODUCTION AND
SCOPE OF THE PROBLEM

INTRODUCTION

1.1 Scope of the problem:

The hunters e-grocery brand is present in 10 countries and are looking for newways toimprove and anticipate their customer needs based on customer behaviour.

so we are using the data set propose business value for informative based descision making.

Source:

The website from which we extracted the data is given below:

https://www.kaggle.com/datasets/hunter0007/ecommerce-dataset-for-predictive-marketing-2023

1.2 Description of data:

The data is extracted fromKaggle.com, data has 2019501 columns and 12 columns.

A brief desription of the variables in the dataset:

order_id :
A unique number to identity the order.

user_id :
A unique number to identify the user.

order_number :
Number of the order.

order_dow :
Day of the Week the order was made.

order_hour_of_day :
Time of the order.

days_since_prior_order :
History of the order.

product_id :
Id of the product.

add_to_cart_order :
Number of items added to cart.

reordered :
If the reorder took place.

department_id :
Unique number allocated to each department.

department :
Names of the departments

product_name :
Name of the products

1.3 Review of chapters:

Chapter 2 gives the brief introduction about machine learning techniques like need of ML today, types of ML Algorithms and various models in each algorithm and what technique to use when and how to validate, Tune the ML algorithms and how to measure the performance of the ML model.

Chapter 3 describes the various results obtained for the problem. This section contains all the outputs generated through the ML algorithms applied on the data as well as validation and performance matrices.

Chapter 4 describes the summary and conclusions followed by Bibliography.

APPENDIX:

It describes the dataset and Machine Learning code used.

Chapter-2

Review of Machine Learning Process

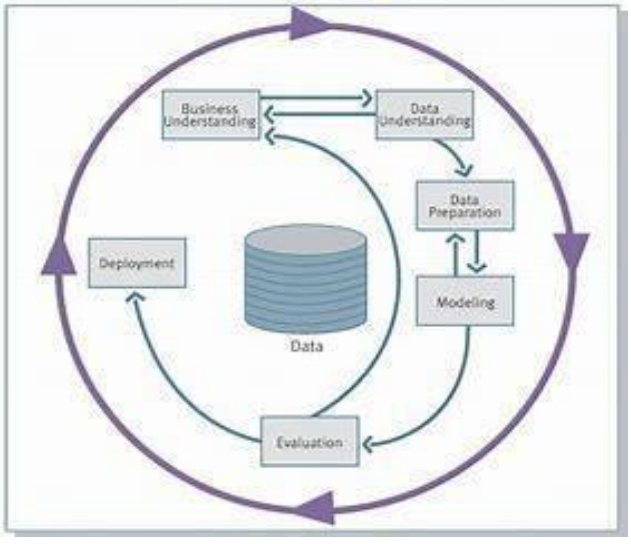
REVIEW OF MACHINE LEARNING PROCESS

2.0 Need of Machine Learning

In this age of modern technology, there is one resource that we have in abundance a large amount of structured and unstructured data. In the second half of the twentieth century, machine learning evolved as a subfield of artificial intelligence that involved the development of self-learning algorithms to gain knowledge from that data in order to make predictions. Instead of requiring humans to manually derive rules and build models from analyzing large amounts of data, machine learning offers a more efficient alternative for capturing the knowledge in data to gradually improve the performance of predictive models, and make data-Profiven decisions. Not only is machine learning becoming increasingly important in computer science research but it also plays an ever greater role in our everyday life.

2.1 Machine Learning Process

The CRISP-DM (Cross-Industry Standard Process for Data Mining) Process was designed specifically for the data mining. However, it is flexible and thorough enough that it can be applied to any analytical project whether it is predictive analytics, data science, or Machine learning. The Process has the following six phases :



- Business Understanding
- Data understanding
- Data preparation
    - Modelling
    - Evaluation
    - Deployment

2.1.1 Business Understanding

It is very important step of the process in achieving the success. The purpose of this step is to identify the requirements of the business so that you can translate them into analytical objectives. It has the following tasks:

1) Identify the Business objective
2) Assess the situation
3) Determine the Analytical goals
4) Produce a project plan

2.1.2 Data Understanding

After enduring the all-important pain of the first step, you can now get your hands on the data. The task in this process consist the following

1) Collect the data
2) Describe the data
3) Explore the data
4) Verify the data Quality

2.1.3 Data Preparation

This step is relatively self-explanatory and in this step the goal is to get the data ready to input in the algorithms. This includes merging, feature engineering, and transformations. If imputation for missing values / outliers is needed then, it happens in this step. The key five tasks under this step are as follows:

1) Select the data
2) Clean the data
3) Construct the data
4) Integrate the data

1) Format the data

2.1.4 Modelling

Oddly, this process step includes the consideration that you already thought of and prepared for. In this, one will need at least a modicum of an idea about how they will be modelling. Remember, that this is flexible, iterative process and some strict linear flow chart such as an aircrew checklist.

Below are the tasks in this step:

1) Select a modelling technique
2) Generate a test design
3) Build a model
4) Assess a Model

Both cross validation of the model (using train/test or K fold validation) and model assessment which involves comparing the models with the chosen criterion (RMSE, Accuracy, ROC) will be performed under this phase.

2.1.5 Evaluation

In the evaluation process, the main goal is to confirm that the work that has been done and the model selected at this point meets the business objective. Ask yourself and others, have we achieved the definition of success? And, here are the tasks in this step:

1) Evaluate the results
2) Review the process
3) Determine the next steps

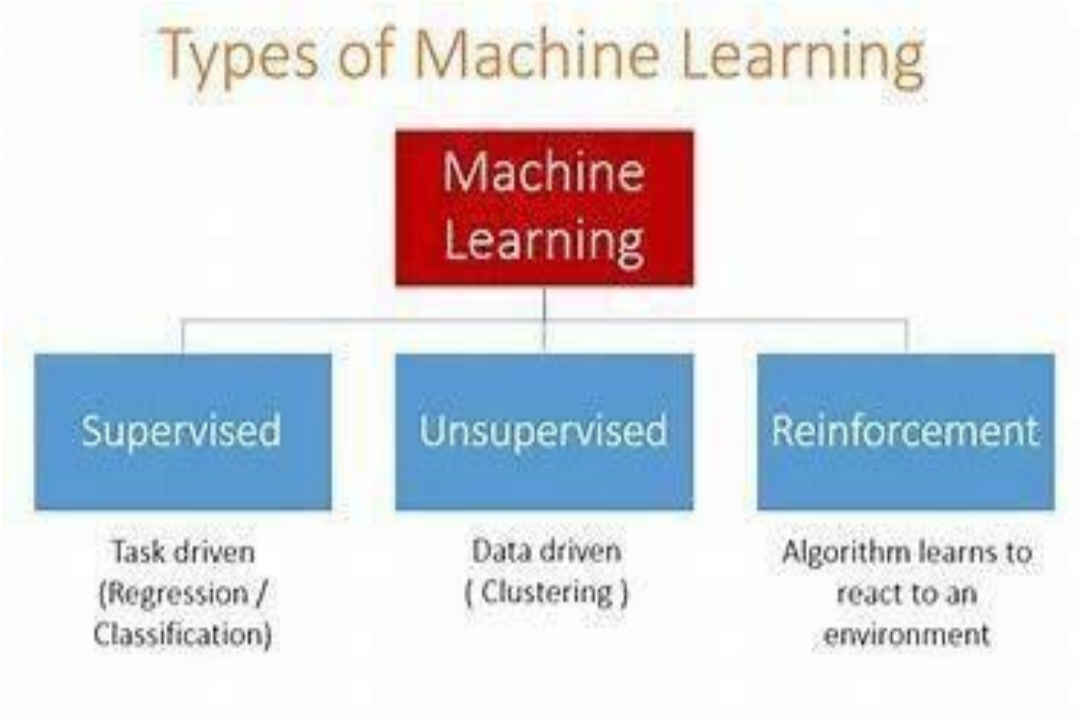
2.1.6 Deployment

If everything is done according to the plan up to this point, it might come down to flipping a switch and your model goes live. Here are the tasks in this step:

- Deploying the Plan.
- Monitoring and maintenance of the plan.
- Producing the final report.

2.2 Types of Machine Learning

Broadly, the Machine Learning Algorithms are classified into 3 types:



2.2.1 Supervised Learning

This algorithm consists of a target / outcome / dependent variable which is to be predicted from a given set of predictors / independent variables. Using these set of variables, we generate a function that maps inputs to desired output. The training process continues until the model achieves a desired level of accuracy on the training data.

The process of Supervised Learning model is illustrated i then below picture:

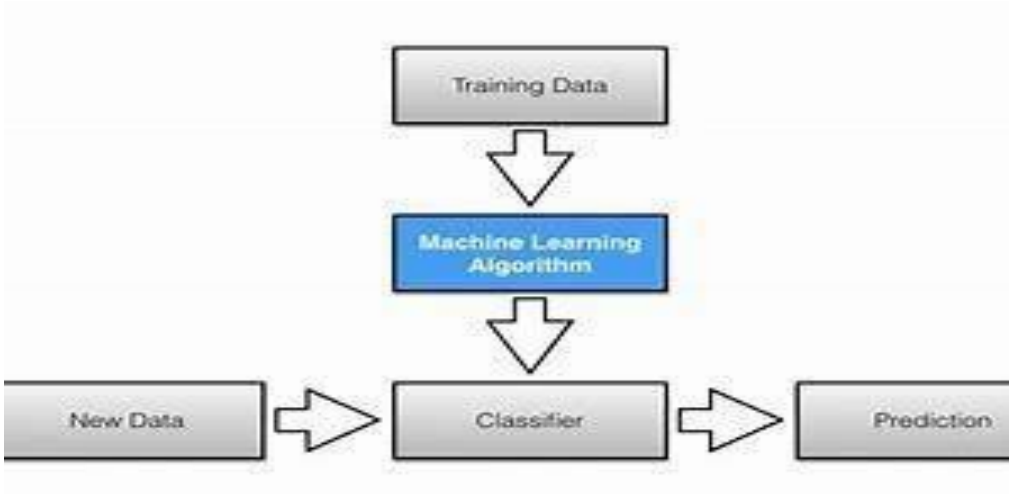
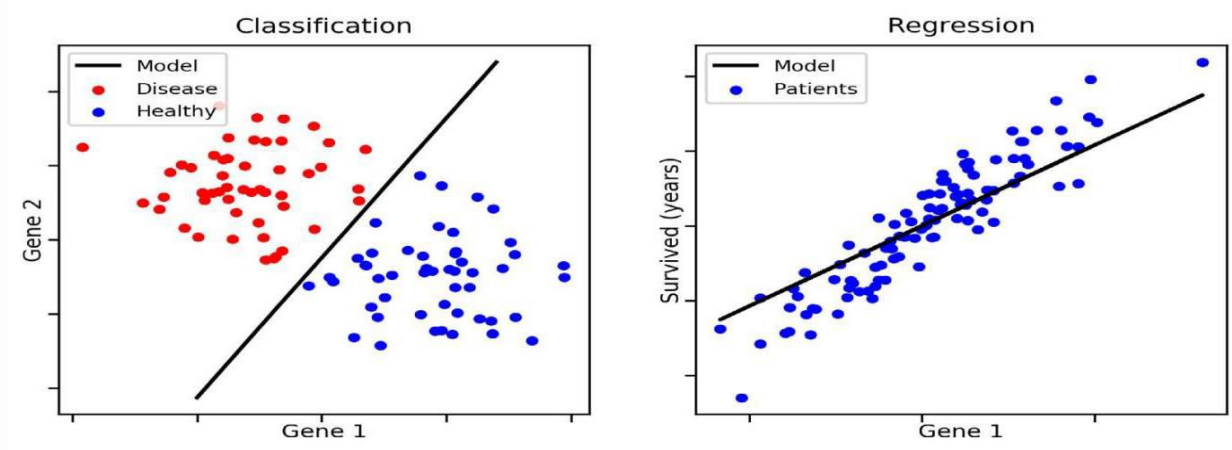


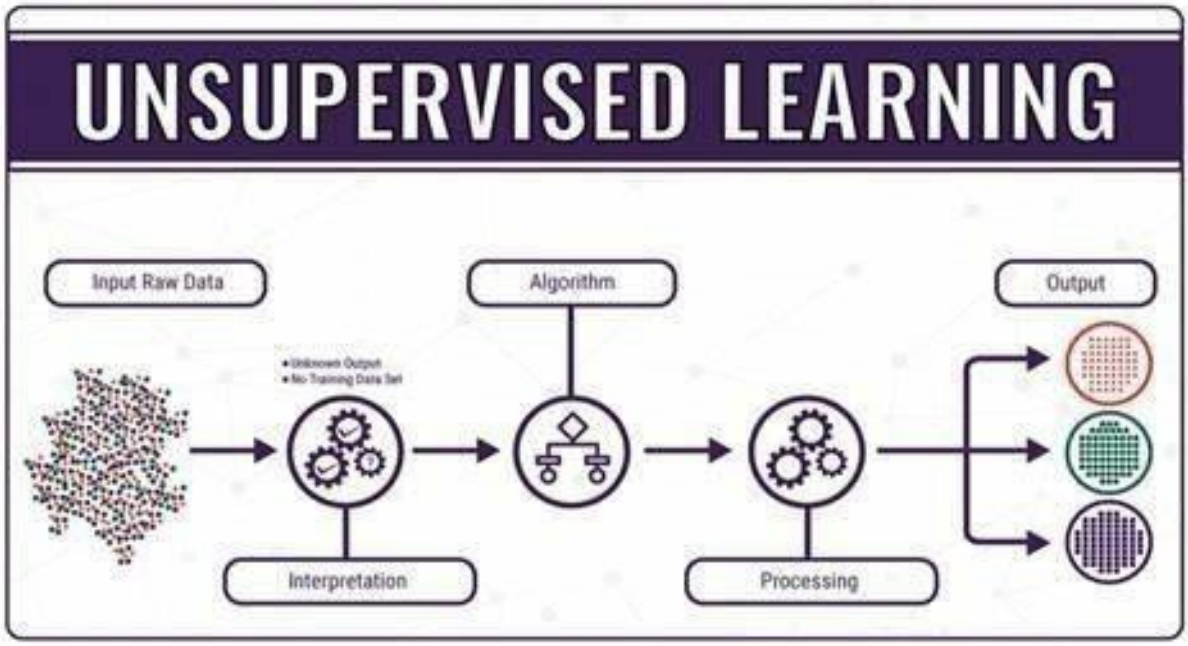
Fig. 2.2.1 Supervised Learning

Examples of Supervised Learning: Regression, Decision Tree, Random Forest, KNN, Logistic Regression,…etc

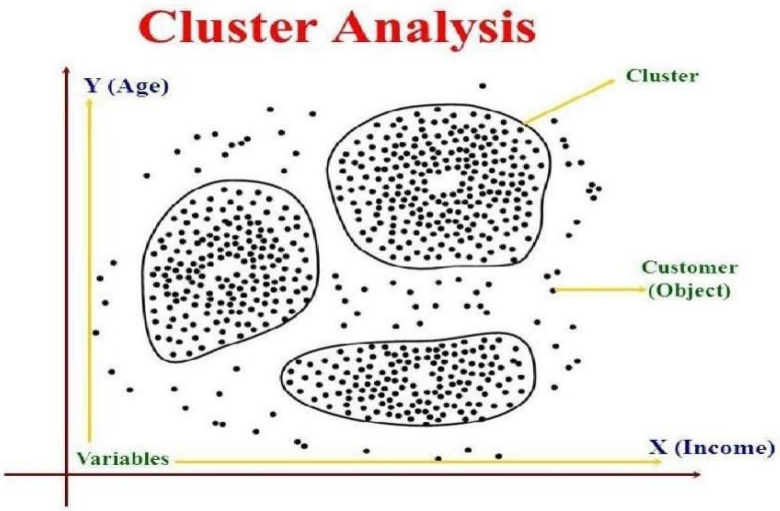


2.2.2 Unsupervised Learning

In this algorithm, we will not have any target or outcome variable to predict / estimate. It is used for clustering population into different groups, which is widely used for segmenting customers in different groups for specific intervention. (More of Exploratory Analysis

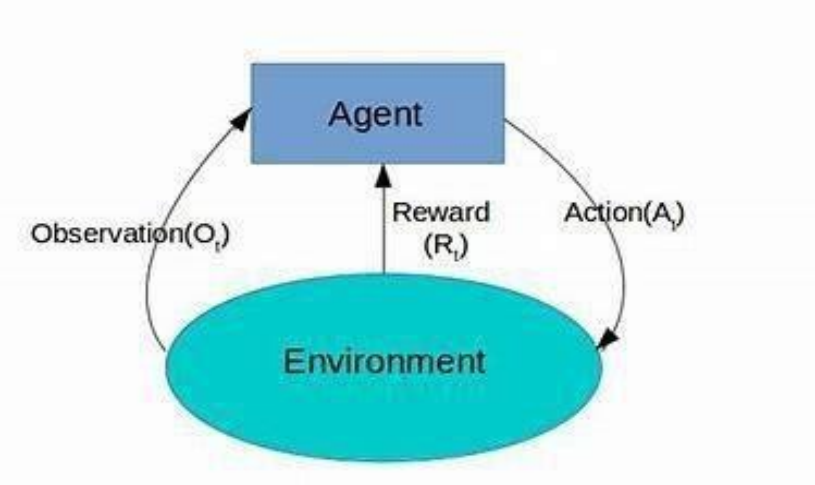


Examples of Unsupervised Learning: Data reduction techniques, Cluster Analysis, Market Basket Analysis, etc.



2.2.3 Reinforcement Learning

Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions. The process of reinforcement learning is illustrated in the below picture:



2.3 Choosing the algorithm

Choosing the right algorithm will depend on the type of the problem we are solving and also depend on the scale of the dependent variable. In case of continuous target variable, we will use regression algorithms and in case of categorical target, we will use classification algorithms and for the model which doesn't have target variable, we will use either cluster analysis / data reduction techniques.

2.3.1 Types of Regression Algorithms

There are many Regression algorithms in machine learning, which will be used in different regression applications. Some of the main regression algorithms are as follows:

a) Simple Linear Regression:-In simple linear regression, we predict scores of the variable from the data of second variable. The variable we are forecasting is called the criterion variable and referred to as Y. The variable we are basing our predictions on is called the predictor variable and denoted as X.

b) Multiple Linear Regression:-Multiple linear regression is one of the algorithms of regression technique, and is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one dependent variable with two or more independent variables. The independent variables can be either continuous or categorical.

c) Polynomial Regression:-Polynomial regression is another form of regression in which the maximum power of the independent variable is more than 1.

In this regression technique, the best fit line is not a straight line instead it is in the form of a curve

a) Support Vector Machines: - Support Vector Machines can be applied to regression problems as well as Classification. It contains all the features that characterizes maximum margin algorithm. Linear learning machine maps a nonlinear function into high dimensional kernel-induced feature space. The system capacity will be controlled by parameters that do not depend on the dimensionality of feature space.

b) Decision Tree Regression: - Decision tree builds regression models in the form of a tree structure. It breaks down the data into smaller subsets and while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

c) Random Forest Regression: - Random Forest is also one of the algorithms used in regression technique. It is very a flexible, easy to use machine learning algorithm that produces, even without hyper -parameter tuning, a great result most of the time. It is also one of the most widely used algorithms because of its simplicity and the fact that it can used for both regression and classification tasks. The forest it builds is an ensemble of Decision Trees, most of the time trained with the "bagging" method.

d) Random Forest Regression: - Random Forest is also one of the algorithms used in regression technique. It is very a flexible, easy to use machine learning algorithm that produces, even without hyper -parameter tuning, a great result most of the time. It is also one of the most widely used algorithms because of its simplicity and the fact that it can used for both regression and classification tasks. The forest it builds is an ensemble of Decision Trees, most of the time trained with the "bagging" method.

Other than these we have regularized regression models like Ridge, LASSO and Elastic Net regression which is used to select the key parameters and these is also Bayesian regression which works with the Bayes theorem.

2.3.2 Types of Classification Algorithms

There are many Classification algorithms in machine Learning, which can be used for different classification applications. Some of the main classification algorithms are as follows:

a) Logistic Regression/Classification:- Logistic regression falls under the category of supervised learning; it measures the relationship between the dependent variable which is categorical with one or more than one independent variables by estimating probabilities using a logistic/sigmoid function. Logistic regression can generally be used when the dependent variable is Binary or Dichotomous. It means that the dependent variable can take only two possible values like "Yes or No", "Living or dead".

a) K-Nearest Neighbours:- K-NN algorithm is one of the most straight forward algorithms in classification, and it is one of the most used ML algorithms. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours. It can also use for regression- output is the value of the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbours.

b) Naive Bayes:-Naive Bayes is a type of Classification technique based on Bayes theorem, with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a Particular feature in a class is unrelated to the presence of any other function. Naive Bayes model is accessible to build and particularly useful for extensive datasets.

c) Decision Tree Classification:-Decision tree builds classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The first decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

d) Support Vector Machines:-A Support Vector Machine is a type of Classifier, in which a discriminative classifier is formally defined by a separating hyper plane. The algorithm outputs an optimal hyper plane which categorizes new examples. In two dimensional space, this hyper plane is a line dividing a plane in two parts where in each class lay in either side. f) Random Forest Classification:-Random Forest is a supervised learning algorithm. It creates a forest and makes it somehow random. The forest it builds is an ensemble of Decision Trees, most of the times the decision tree algorithm trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. And Random Forest is also very powerful to find the variable importance in classification/ Regression problems.

2.3.3 Types of Unsupervised Learning

Clustering is the type of unsupervised learning in which an unlabelled data is used to draw inferences. It is the process of grouping similar entities together. The goal of this unsupervised machine learning technique is to find similarities in the data points and group similar data points together and also to figure out which cluster should a new data point belong to.

Types of Clustering Algorithms: - There are many Clustering algorithms in machine learning, which can be used for different clustering applications. Some of the main clustering algorithms are as follows:

a) Hierarchical Clustering: - Hierarchical clustering is one of the algorithms of clustering technique, in which similar data is grouped in a cluster.

It is an algorithm that builds the hierarchy of clusters. This algorithm starts with all the data points assigned to a bunch of their own. Then, two nearest groups are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.

It starts by assigning each data point to its bunch. Finds the closest pair using Euclidean distance and merges them into one cluster. This process is continued until all data points are clustered into a single cluster.

b) K–Means Clustering: - K-Means clustering is one of the algorithms of clustering technique, in which similar data is grouped into a cluster. K-means is an iterative algorithm that aims to find local maxima in each iteration. It starts with K as the input which is the desired number of clusters. Input k centroids in random locations in your space. Now, with the use of the Euclidean distance method, calculates the distance between data points and centroids, and assign data point to the cluster which is close to its centroid. Re calculate the cluster centroids as a mean of data points attached to it. Repeat until no further changes occur.

Types of Dimensionality Reduction Algorithms: - There are many dimensionality reduction algorithms in machine learning, which are applied for different dimensionality reduction applications. One of the main dimensionality reduction techniques is Principal Component Analysis (PCA) / Factor Analysis.

Principal Component Analysis (Factor Analysis): - Principal Component Analysis is one of the algorithms of Dimensionality reduction. In this technique, it transforms data into a new set of variables from input variables, which are the linear combination of real variables. These Specific new set of variables are known as principal components. As a result of the transformation, the first primary component will have the most significant possible variance, and each following component in has the highest possible variance under the constraint that it is orthogonal to the above components. Keeping only the best $m < n$ components, reduces the data dimensionality while retaining most of the data information.

2.4 Choosing and comparing models through Pipelines

When you work on machine learning project, you often end up with multiple good models to choose from. Each model will have different performance characteristics. Using re-sampling methods like k-fold cross validation; you can get an estimate of how accurate each model may be on unseen data. You need to be able to use these estimates to choose one or two best models from the suite of models that you have created.

2.4.1 Model Validation

When you are building a predictive model, you need to evaluate the capability or generalization power of the model on unseen data. This is typically done by estimating accuracy using data that was not used to train the model, often referred as cross validation.

A few common methods used for Cross Validation:

1) The Validation set Approach (Holdout Cross validation)

In this approach, we reserve large portion of dataset for training and rest remaining portion of the data for model validation. Ideally people will use 70-30 or 80-20 percentages for training and validation purpose respectively.

A major disadvantage of this approach is that, since we are training a model on a randomly chosen portion of the dataset, there is a huge possibility that we might miss-out on some interesting information about the data which, will lead to a higher bias.

2) K-fold Cross validation

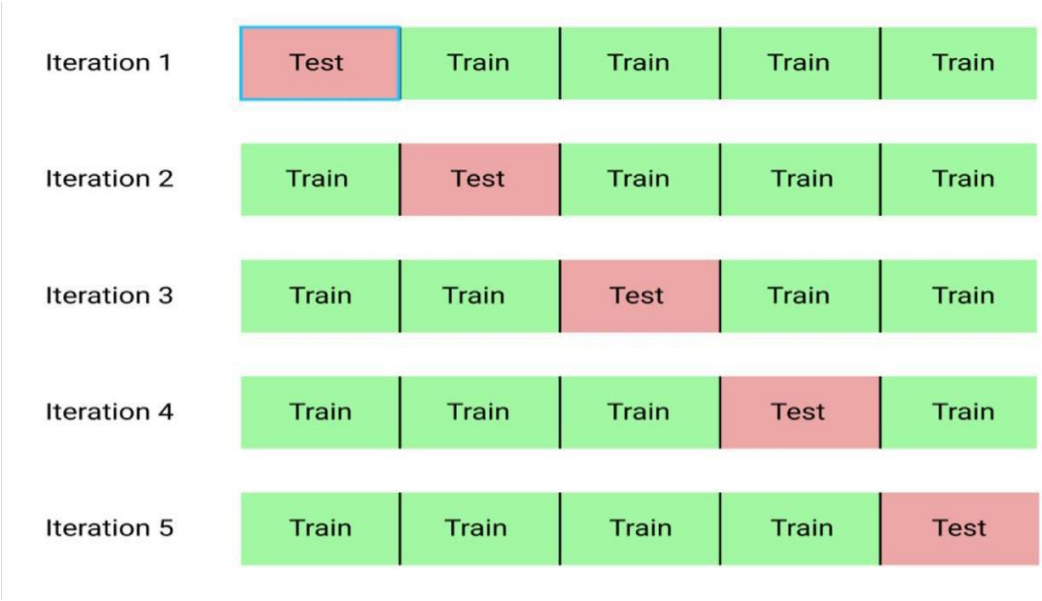
As there is never enough data to train your model, removing a part of it for validation may lead to a problem of under fitting. By reducing the training data, we risk losing important patterns/ trends in data set, which in turn increases error induced by bias. So, what we require is a method that provides ample data for training the model and also leaves ample data for validation. K Fold cross validation does exactly that.

In K Fold cross validation, the data is divided into k subsets. Now the holdout method is repeated k times, such that each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get total effectiveness of our model. As can be seen, every data point gets to be in a validation set exactly once, and gets to be in a training set k-1 times. This significantly reduces the bias as we are using most of the data for fitting and also significantly reduces variance as most of the data is also being used in validation set. Interchanging the training and test sets also adds to the effectiveness of this method. As a general rule and empirical evidence, K = 5 or 10 is preferred, but nothing's fixed and it can take any value.

Below are the steps for it:

- Randomly split your entire dataset into k folds"
- For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold.
- Record the error you see on each of the predictions.
- Repeat this until each of the k-folds has served as the test set.
- The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model.

Below is the visualization of a k-fold validation when k=5.



How to choose K:

- Smaller dataset: 10-fold cross validation is better
- Moderate dataset: 5 or 6 fold cross validation works mostly Big dataset: Train – Val split for validation

Other than this, we have Leave one out cross validation (LOOCV), in which each record will be left over from the training and then, the same will be used for testing purpose. This process will be repeated across all the respondents.

2.5 Model Diagnosis with over fitting and under fitting

2.5.1 Bias and Variance

A fundamental problem with supervised learning is the bias variance trade-off.

Ideally, a model should have two key characteristics

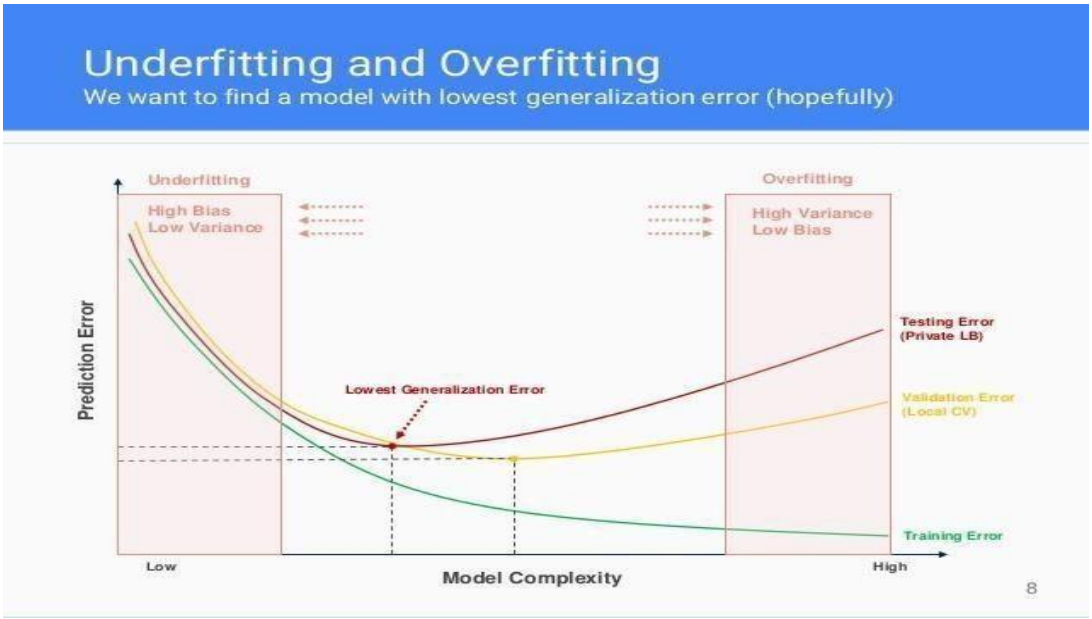
- ❖ Sensitive enough to accurately capture the key patterns in the training dataset.
- ❖ Generalized enough to work well on any unseen dataset.

Unfortunately, while trying to achieve the above-mentioned first point, there is an ample chance of over-fitting to noisy or unrepresentative training data points leading to a failure of generalizing the model. On the other hand, trying to generalize a model may result in failing to capture important regularities.

If model accuracy is low on a training dataset as well as test dataset, the model is said to be under-fitting or that the model has high bias. The Bias refers to the simplifying assumptions made by the algorithm to make the problem easier to solve. To solve an under-fitting issue or to reduce bias, try including more meaningful features and try to increase the model complexity by trying higher order interactions

The Variance refers to sensitivity of a model changes to the training data. A model is giving high accuracy on a training dataset, however on a test dataset the accuracy proofs prophetically then, the model is said to be over-fitting or a model that has high variance.

To solve the over-fitting issue try to reduce the number of features, that is, keep only the meaningful features or try regularization methods that will keep all the features. Ideal model will be the trade-off between Underfitting and over fitting like mentioned in the below picture.



And, the Hyper parameters will be tuned in the below mentioned ways to reach the optimal solution:

1) Grid Search
2) Random Search
3) Manual Tuning

2.5.2 Model Performance Matrix

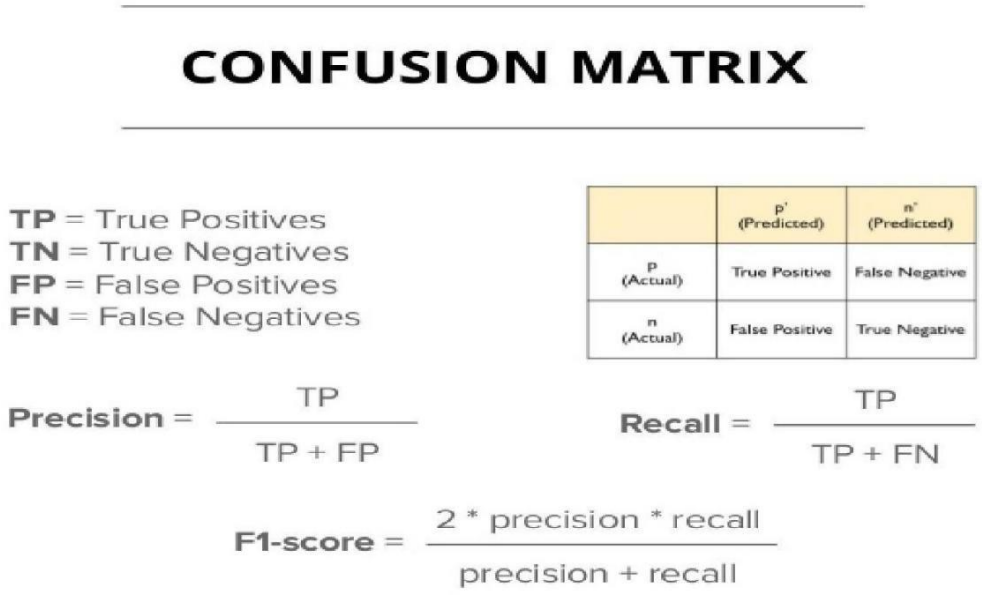
Model evaluation is an integral part of the model development. Based on model evaluation and subsequent comparisons, we can take a call whether to continue our efforts in model enhancement or cease them and select the final model that should be used / deployed.

1. Evaluating Classification Models:

- **Confusion Matrix**

Confusion matrix is one of the most popular ways to evaluate a classification model. A confusion matrix can be created for a binary classification as well as a multi-class classification model.

A confusion matrix is created by comparing the predicted class label of a data point with its actual class label. This comparison is repeated for the whole dataset and the results of this comparison are compiled in a matrix or tabular format.



2. Regression Model Evaluation:

A regression line predicts the y values for a given x value. Note that the values are around the average. The prediction error (called as root-mean-square error or RSME) is given by the following formula:

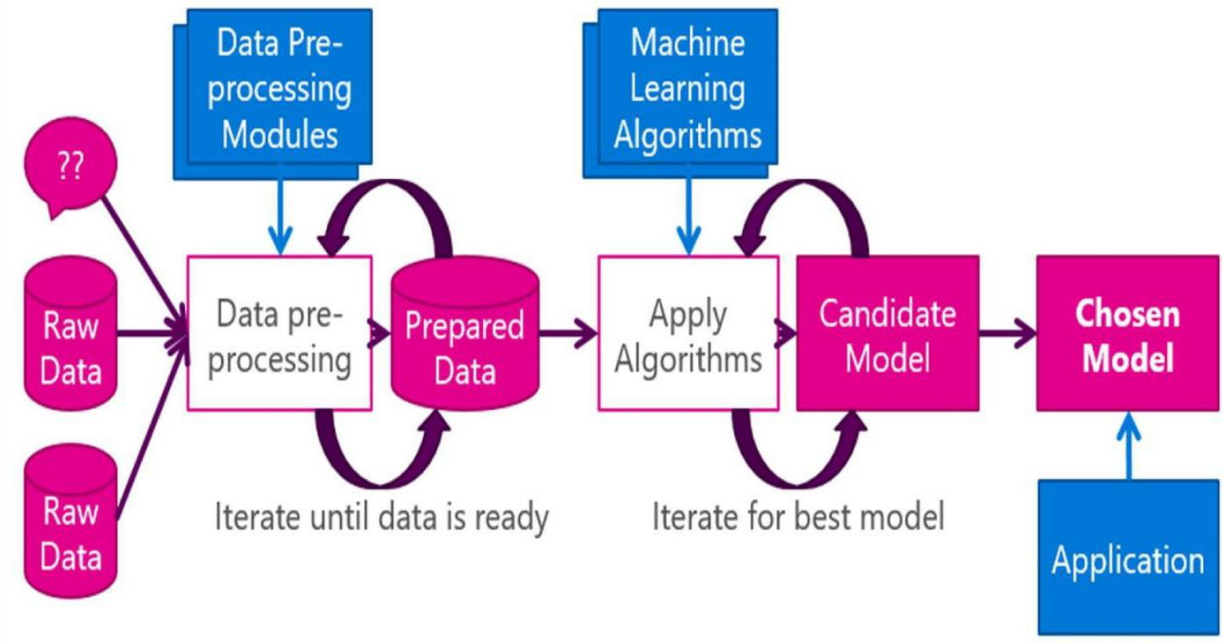
$$\mathbf{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

3. Evaluating Unsupervised Models

The Unsupervised algorithms will be assessed by the profile of the factors/ clusters which were derived through the models.

2.6 Overall Process of Machine Learning

To put overall process together, below is the picture that describes the road map for building ML Systems



Segmentation

The technique of splitting customers into separate groups depending on their attributes or behavior, makes this possible.

Customer

Customer segmentation in machine learning can help you save money on marketing initiatives by reducing waste.

There are different methodologies for customer segmentation, and they depend on four types of parameters :

Geographic
Demographic
Behavioral
Psychological

Behavioral Segmentation :

It is the process of sorting and grouping customers based on the behviors they exhibit. These behaviors include the types of products and content they consume, andthe cadence of their interaction with an app, website, or business.

The benefits of behavioral segmentation include :

Personalization :

Behavioral segmentation doesn't just tell you what the product or service a certain group of customers likes. It helps you understand what channels they frequent and whattype of messaging they respond to, so you can boost your conversions.

Budget allocation :

Since you know which segements spends the most and how they spend , you can better allocate your efforts to target them.

Forecasting:

Looking at each segment's patterns, you can identify trends and more effectivelyplan for the future.

Apriori Algorithm: It is an algorithm for frequent item set mining and the association rule learning over relational databases.

Association Rule Mining :

Association rules can be thought of as an IF-THEN relationship. Suppose item **A** is being bought by the customer, then the chances of item **B** being picked by the customer too under the same **Transaction ID** is found out.

There are two elements of these rules:

Antecedent (IF): This is an item/group of items that are typically found in the Itemsets or Datasets.

Consequent (THEN): This comes along as an item with an Antecedent/group of Antecedents.

But here comes a constraint. Suppose you made a rule about an item, you still have around 9999 items to consider for rule-making. This is where the Apriori Algorithm comes into play. So before we understand the Apriori Algorithm, let's understand the math behind it. There are 3 ways to measure association:

- Support
- Confidence
- Lift

Support: It gives the fraction of transactions which contains item A and B. Basically Support tells us about the frequently bought items or the combination of items bought frequently.

$$Support = \frac{freq(A,B)}{N}$$

So with this, we can **filter out** the items that have a **low frequency**.

Confidence: It tells us how often the items A and B occur together, given the number times A occurs.

$$Confidence = \frac{freq(A,B)}{freq(A)}$$

Lift: Lift indicates the strength of a rule over the random occurrence of A and B. It basically tells us the strength of any rule.

$$Lift = \frac{Support}{Supp(A)\ x\ Supp(B)}$$

Chapter- 3

Machine Learning - At Work

3.1: An Approach to the Problem:

Order to carry out the analysis, we have extracted 2019501 from the kaggle.com and theinformation of the same is mentioned in Chapter 1.

In this Chapter, we are going to discuss about the results of different Machine Learning methods used in order to obtain the solution for the problem mentioned in Chapter1.

As mentioned in Chapter 2, the first step of a ML Algorithm is Data cleaning and preparing data for the modelling. As a first step, we have to check whether the data was read properly and all the scale types are as per the data.

Reading the Data:

	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id
0	2425083	49125	1	2	18	NaN	17	1	0	13
1	2425083	49125	1	2	18	NaN	91	2	0	16
2	2425083	49125	1	2	18	NaN	36	3	0	16
3	2425083	49125	1	2	18	NaN	83	4	0	4
4	2425083	49125	1	2	18	NaN	83	5	0	4
...	...	...	...	...	...	...	...	...	...	...
2019496	3390742	199430	16	3	18	5.0	83	8	0	4
2019497	458285	128787	42	2	19	3.0	115	1	1	7
2019498	458285	128787	42	2	19	3.0	32	2	1	4
2019499	458285	128787	42	2	19	3.0	32	3	1	4
2019500	458285	128787	42	2	19	3.0	123	4	1	4

2019501 rows × 12 columns

The data has 2019501 rows and 12 columns.

Now by using isnull() we are finding null values ,here day_since_prior _order_has null values which is 124342

```
order_id                      0
user_id                       0
order_number                  0
order_dow                     0
order_hour_of_day             0
days_since_prior_order    124342
product_id                    0
add_to_cart_order             0
reordered                     0
department_id                 0
department                    0
product_name                  0
dtype: int64
```

Seeing datatypes present in the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2019501 entries, 0 to 2019500
Data columns (total 12 columns):
 #   Column                  Dtype
---  ------                  -----
 0   order_id                int64
 1   user_id                 int64
 2   order_number            int64
 3   order_dow               int64
 4   order_hour_of_day       int64
 5   days_since_prior_order  float64
 6   product_id              int64
 7   add_to_cart_order       int64
 8   reordered               int64
 9   department_id           int64
 10  department              object
 11  product_name            object
dtypes: float64(1), int64(9), object(2)
memory usage: 184.9+ MB
```

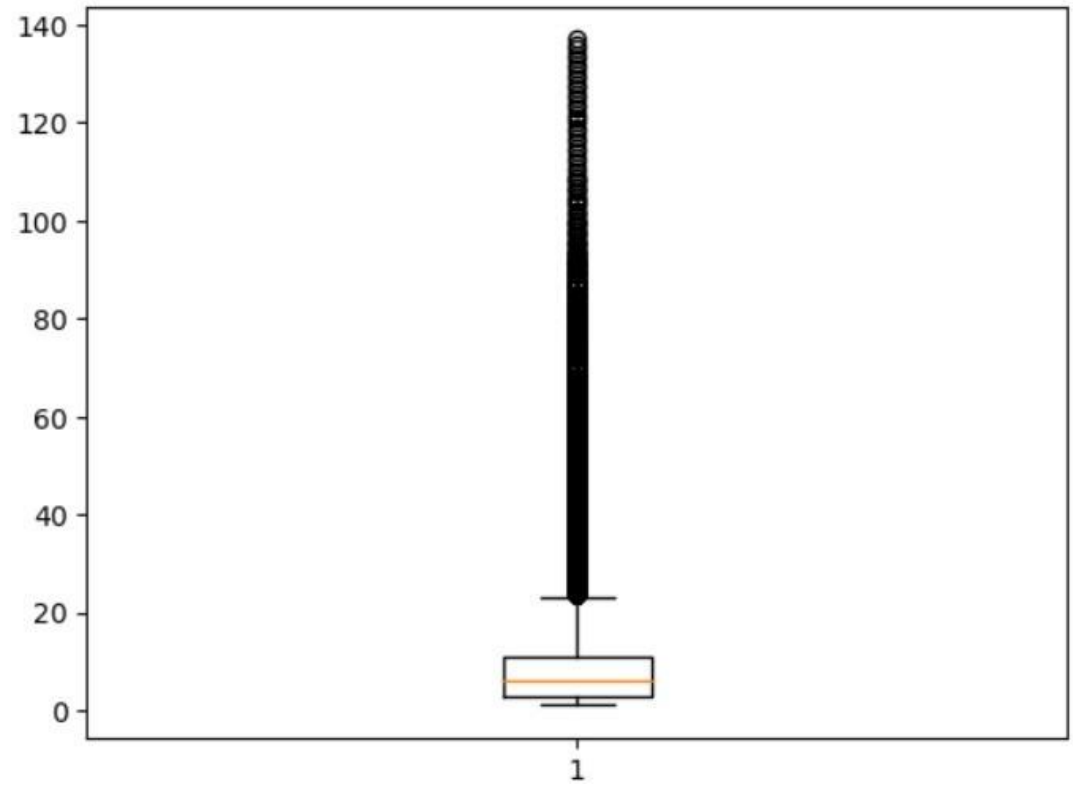
changing null values present in the dataset by using fillna(-1)

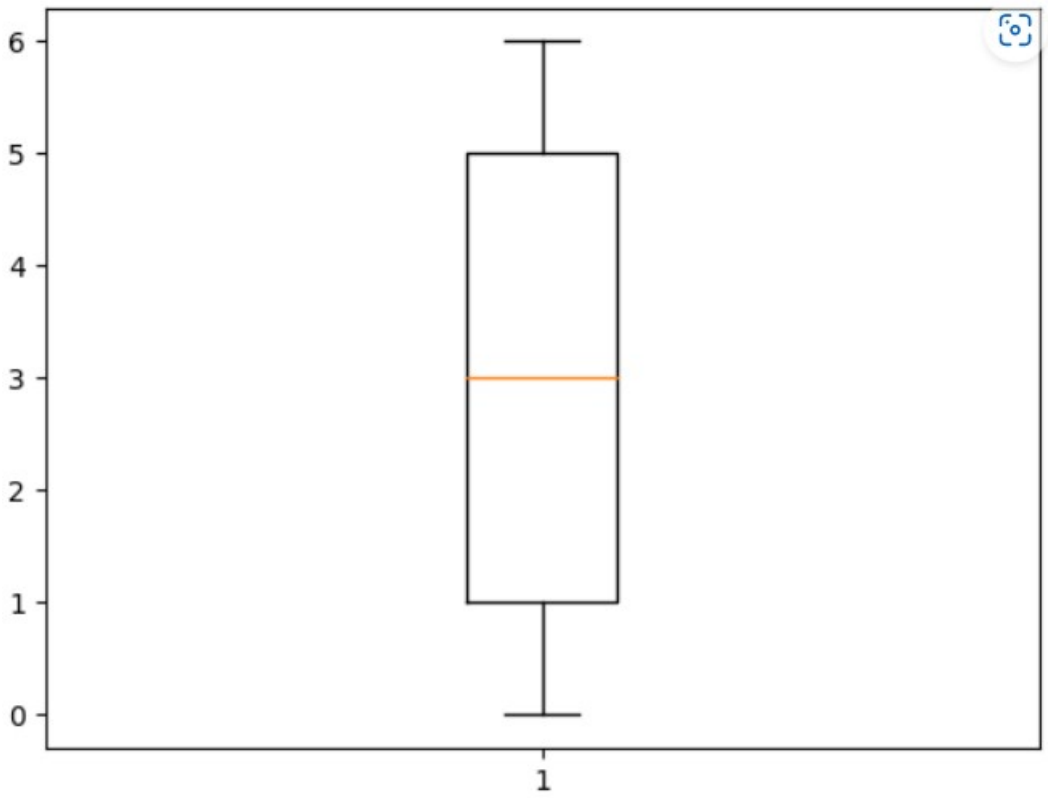
```
order_id                    0
user_id                     0
order_number                0
order_dow                   0
order_hour_of_day           0
days_since_prior_order      0
product_id                  0
add_to_cart_order           0
reordered                   0
department_id               0
department                  0
product_name                0
```

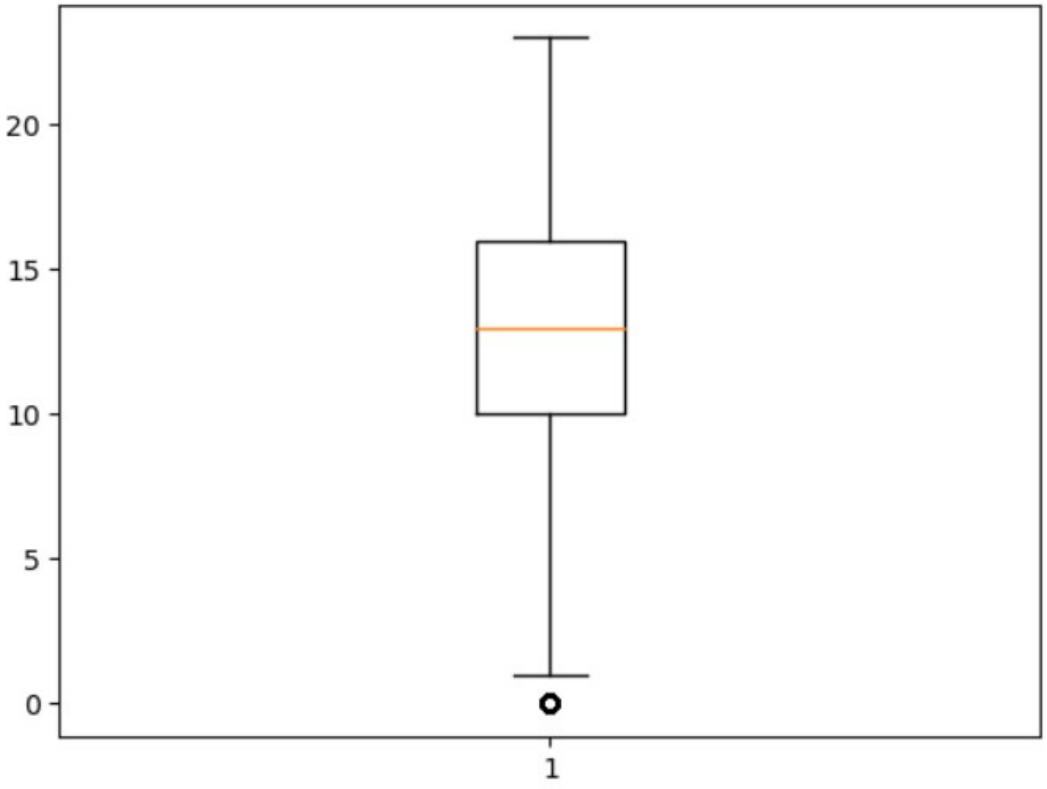
Changing float datatypes to int datatype

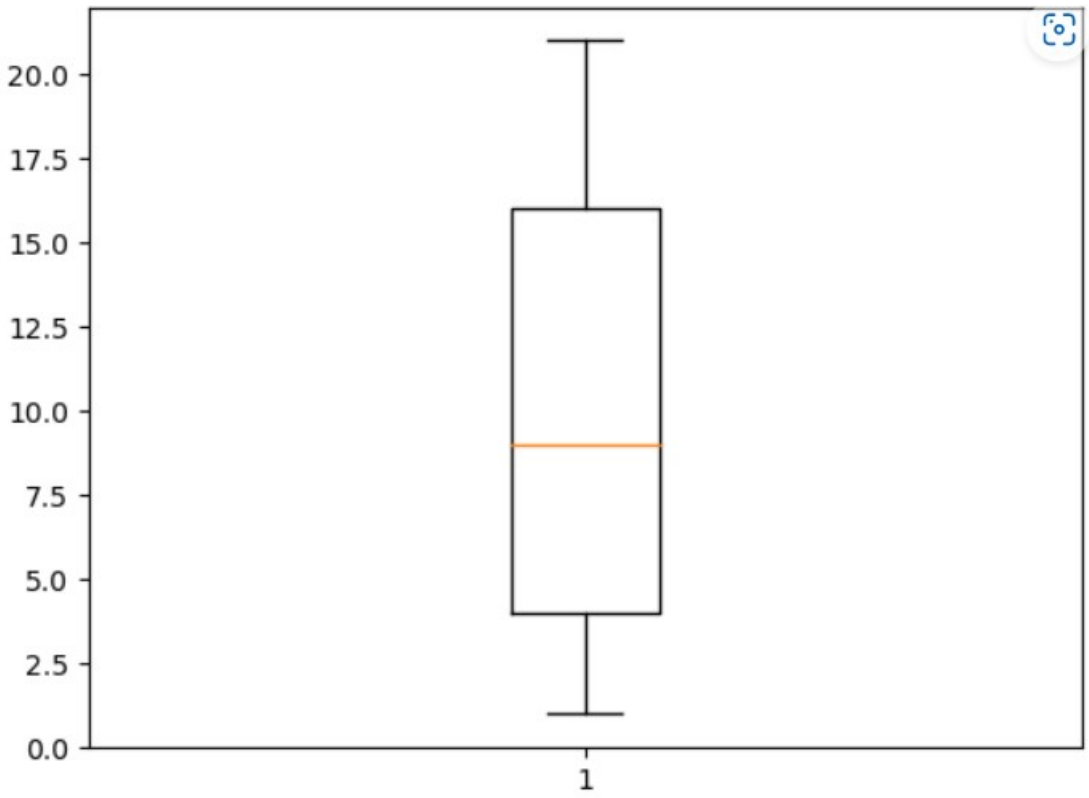
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2019501 entries, 0 to 2019500
Data columns (total 12 columns):
 #   Column                  Dtype
---  ------                  -----
 0   order_id                int64
 1   user_id                 int64
 2   order_number            int64
 3   order_dow               int64
 4   order_hour_of_day       int64
 5   days_since_prior_order  int64
 6   product_id              int64
 7   add_to_cart_order       int64
 8   reordered               int64
 9   department_id           int64
 10  department              object
 11  product_name            object
dtypes: int64(10), object(2)
memory usage: 184.9+ MB
```

Now plotting a box plot for add_to_cart_order, order_dow, order_hour_of _the day, department-Id




6
5
4
3
2
1
0
1


20
15
10
5
0
1



Finding outlier customers based on add_to_cart_order

user_id	order_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id
10	8.339500e+04	4.000000	3.000000	15.000000	14.00000	60.000000	15.500000	0.433333	10.400000
133	2.415920e+05	11.000000	3.000000	13.000000	5.00000	68.133333	15.500000	0.400000	9.033333
197	2.357070e+05	7.000000	6.000000	14.000000	6.00000	74.206897	15.000000	0.793103	11.551724
216	1.500908e+06	5.000000	5.000000	15.000000	24.00000	68.448276	15.000000	0.172414	13.655172
264	3.022479e+06	29.000000	2.000000	9.000000	9.00000	79.551724	15.000000	0.827586	11.103448
...	...	...	...	...	...	...	...	...	...
205789	3.418993e+06	13.000000	6.000000	12.000000	7.00000	63.900000	15.500000	0.600000	9.466667
205926	2.385091e+06	11.000000	1.000000	15.000000	6.00000	73.828571	18.000000	0.514286	10.171429
205956	4.562900e+05	3.000000	3.000000	14.000000	30.00000	69.151515	17.000000	0.303030	10.909091
206146	3.185446e+06	2.000000	2.000000	9.000000	30.00000	72.256410	20.000000	0.025641	11.230769
206165	1.847443e+06	10.439024	3.853659	19.073171	10.04878	77.024390	17.012195	0.292683	12.695122

Finding outlier customers in the purchases made by each customer

	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id
2538	677735	175561	2	3	12	16	32	1	0	4
8764	2018358	185514	22	4	17	17	128	1	0	3
8765	2018358	185514	22	4	17	17	54	2	1	17
15536	2545752	185279	9	5	9	12	84	1	0	16
16290	1365995	136421	1	3	10	-1	88	1	0	13
...	...	...	...	...	...	...	...	...	...	...
2010592	1097536	97028	5	0	16	7	24	1	1	4
2010593	1097536	97028	5	0	16	7	24	2	0	4
2015806	1581072	93695	5	4	17	30	97	1	0	13
2016905	1373400	139870	11	2	12	30	116	1	0	1
2016906	1373400	139870	11	2	12	30	32	2	0	4

1282 rows × 14 columns

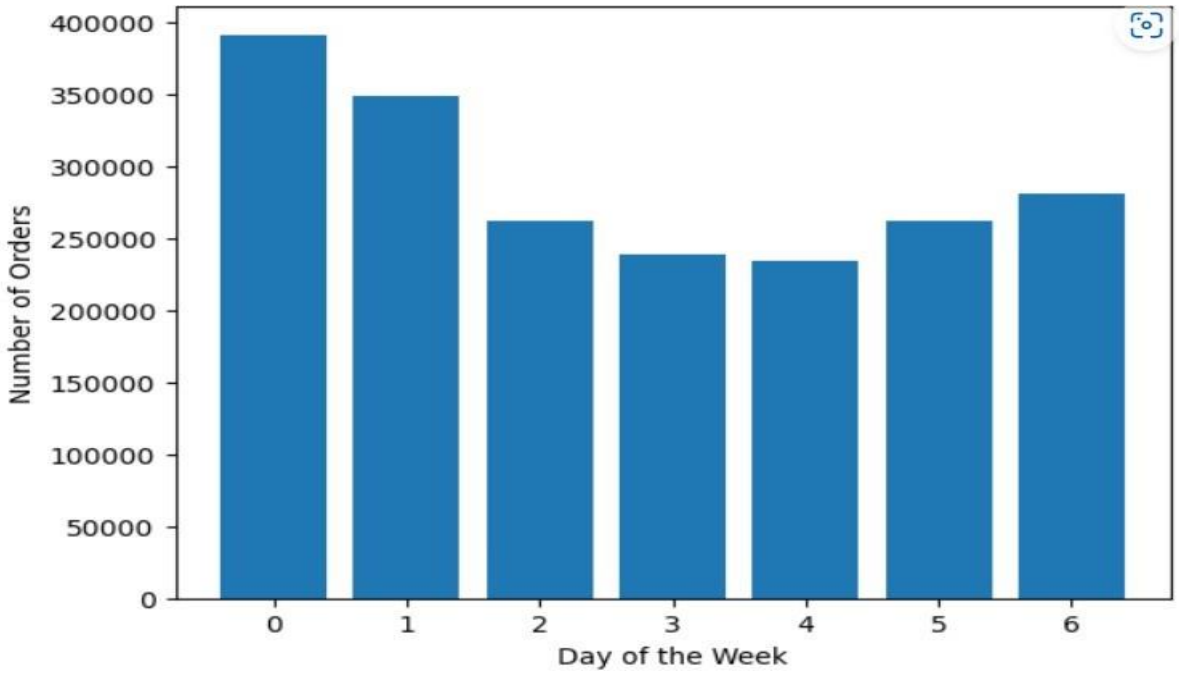
Now dropping the outliers present in the dataset

	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id
0	2425083	49125	1	2	18	-1	17	1	0	13
1	2425083	49125	1	2	18	-1	91	2	0	16
2	2425083	49125	1	2	18	-1	36	3	0	16
3	2425083	49125	1	2	18	-1	83	4	0	4
4	2425083	49125	1	2	18	-1	83	5	0	4
...	...	...	...	...	...	...	...	...	...	...
2019496	3390742	199430	16	3	18	5	83	8	0	4
2019497	458285	128787	42	2	19	3	115	1	1	7
2019498	458285	128787	42	2	19	3	32	2	1	4
2019499	458285	128787	42	2	19	3	32	3	1	4
2019500	458285	128787	42	2	19	3	123	4	1	4

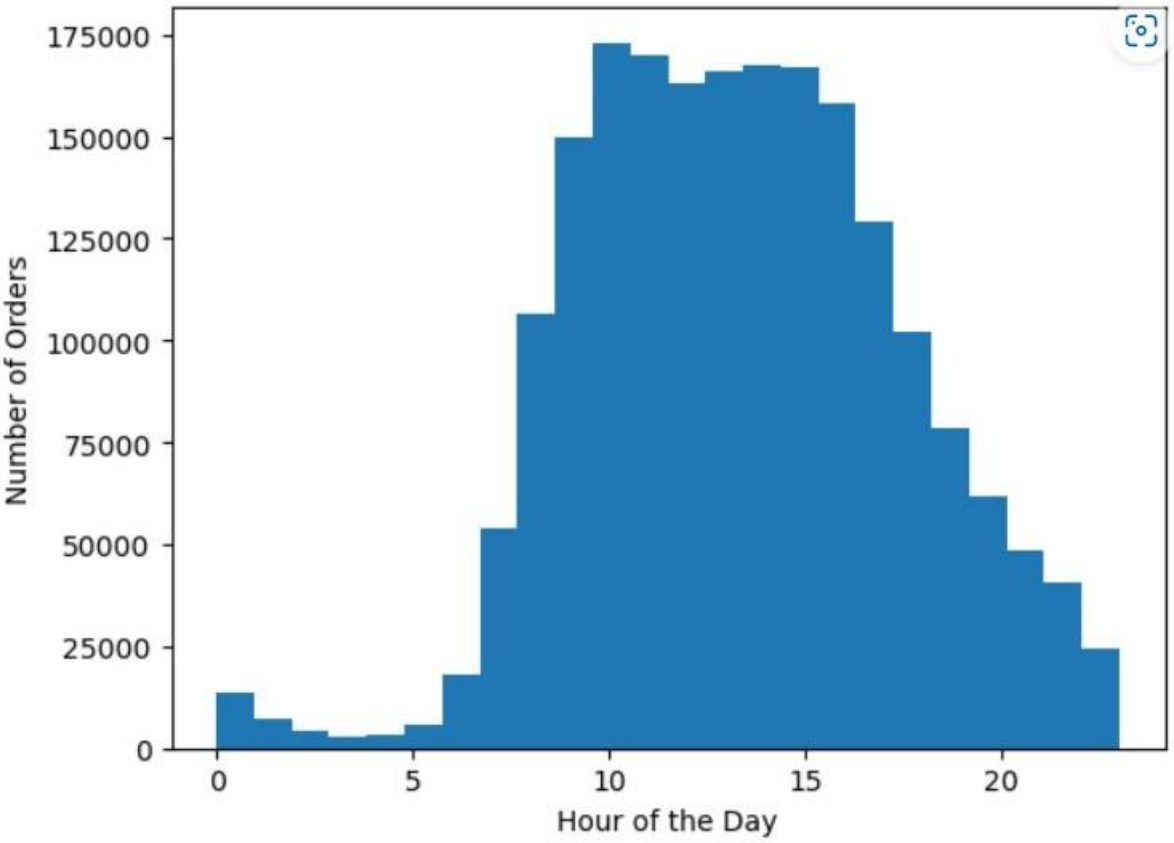
2016558 rows × 14 columns

Understanding data visually:

Plotting bar graph to the day of the week and number of orders:
The code uses the groupby() method to group the orders in the DataFrame df by their day of the week (order_dow) and calculates the size of each group using the size() method. This results in a pandas Series object containing the count of orders for each day of the week.

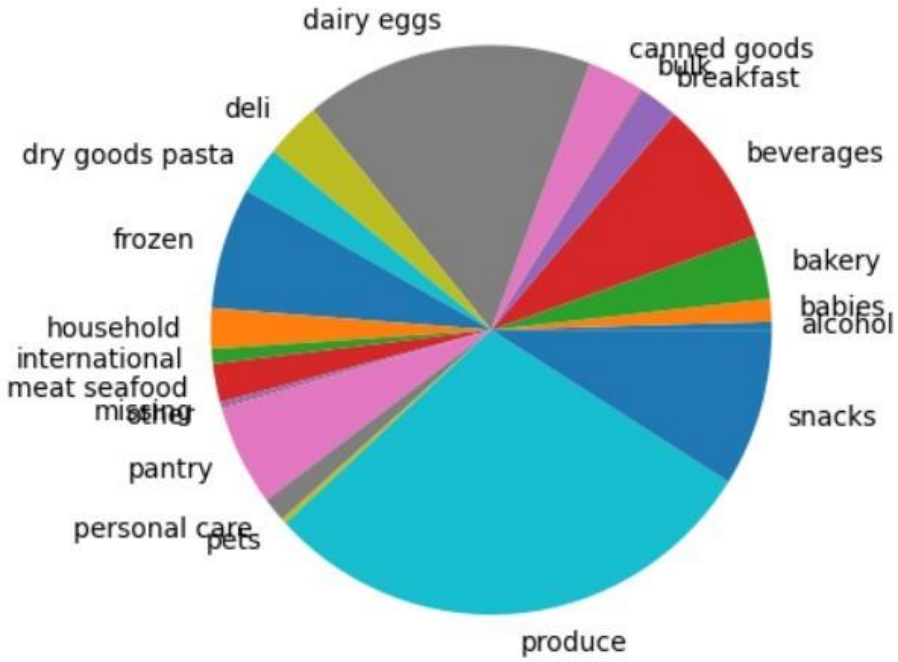


Now plotting histogram chart on orders by hour of the day:
The resulting plot shows the distribution of orders across the 24 hours of the day, allowing us to see when most orders are placed.



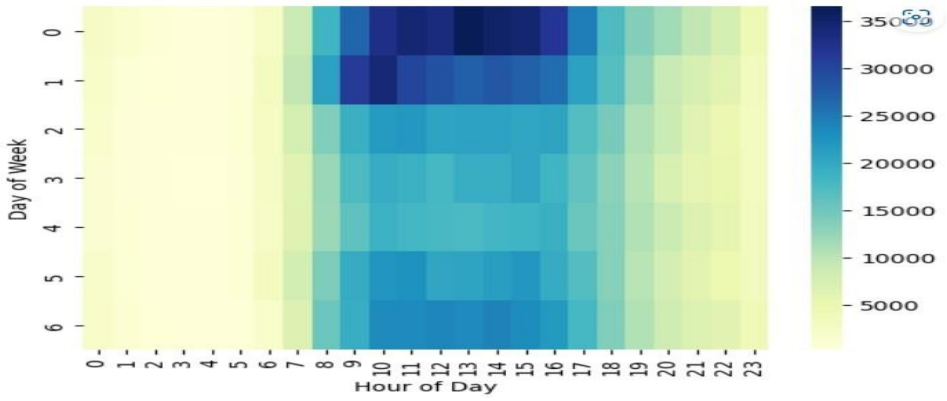
Plotting a Pie chart based on the order by Department:

The code is plotting a pie chart to visualize the distribution of orders across different departments. It first groups the orders by department and calculates the number of orders for each department using the groupby() and size() functions. Then, it uses plt.pie() function to plot the pie chart, where the labels parameter specifies the department names and the values parameter specifies the number of orders for each department. The resulting chart shows the proportion of orders that belong to each department.



Plotting heatmap to find the relation between day of the week and hour of the day orders:

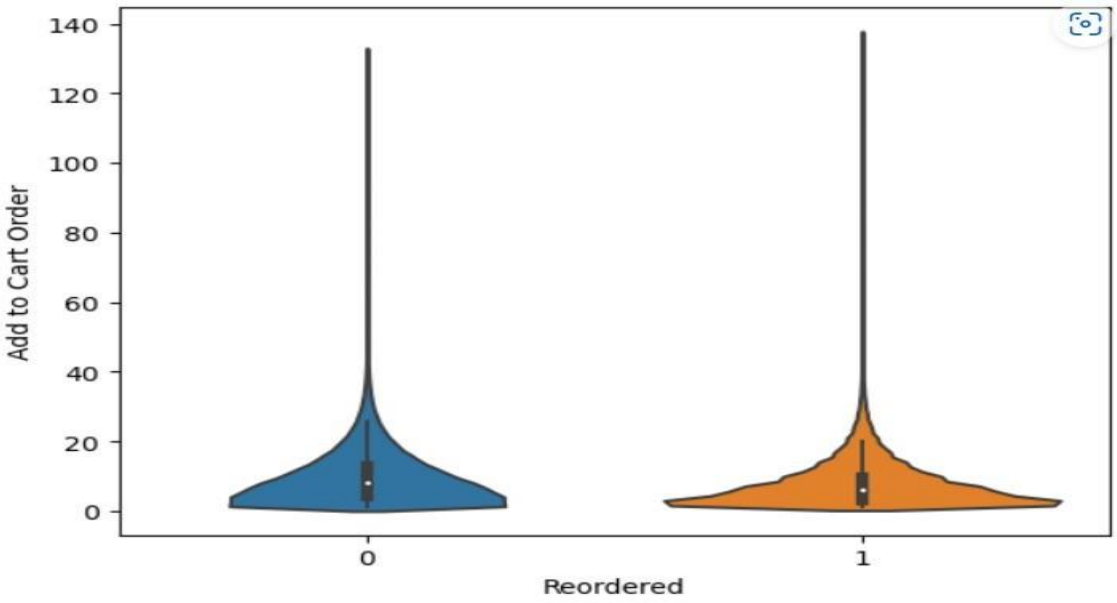
It uses the seaborn heatmap function to plot the pivot table as a color-coded matrix, where darker colors indicate higher order frequencies. The cmap parameter sets the color map to use, and the xlabel and ylabel parameters set the labels for the x-axis and y-axis, respectively. Overall, this code provides a useful visualization to analyze trends in order frequency based on the day of the week and hour of the day.



Now we are Plotting a violinplot to find relation between number of items added to cart and

reorder status:

This code generates a violin plot using Seaborn library to show the relationship between the reorder status and the number of items added to cart. The x-axis represents the reorder status (0 for not reordered and 1 for reordered) and the y-axis represents the add to cart order (the order in which the item was added to the cart). The width of each "violin" represents the number of data points within that category, and the height represents the density of data points at that particular value. The plot shows the distribution of the add to cart order for both reordered and not reordered items, and allows us to compare the two distributions visually. We can see from the plot whether there is a significant difference in the distribution of add to cart order between the two reorder statuses.

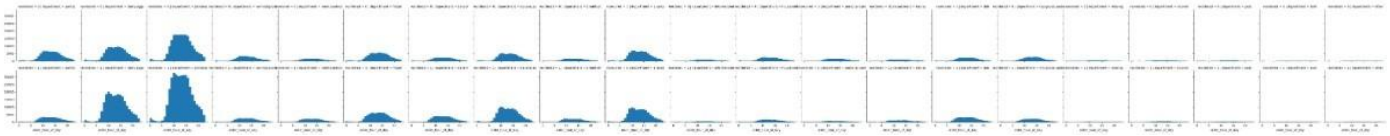


Plotting Kernel Distribution Estimation Plot for Days_since_prior_order:

This code creates a kernel density estimate plot using the sns.kdeplot() function to show the distribution of days since prior order. The shade=True parameter adds shading under the curve. The x-axis represents the number of days since the customer's last order and the y-axis represents the density (i.e., the relative frequency) of those values. The resulting plot can give insights into the typical time between orders for customers in the dataset.

Plotting a Facegrid to reorder of the department with hour of the day:
The FacetGrid function from the Seaborn library is used to create a grid of plots with the rows representing the reordered status (either reordered or not reordered) and the columns representing each department. Then, the map function is used to map a histogram onto each plot, with the x-axis representing the hour of the day and the y-axis representing the number of orders.

This visualization allows us to see how the distribution of orders varies by department and reordered status, and if there are any patterns in the times of day when orders are more likely to be reordered for a particular department.



Product reorder prediction:

The data is first grouped by the user_id and product_id and then the mean of the 'reordered' column is calculated for each group. This gives the probability of a particular product being reordered by a specific user. The resulting dataframe is then assigned to the variable 'reorder_prob' and the column 'reordered' is renamed to 'reorder_prob'. Finally, the 'reorder_prob' dataframe is returned.

	user_id	product_id	reorder_prob
0	2	1	0.000000
1	2	21	0.000000
2	2	23	1.000000
3	2	24	0.666667
4	2	67	0.000000
...	...	...	...
1108531	206209	112	0.500000
1108532	206209	116	0.000000
1108533	206209	120	0.000000
1108534	206209	121	0.000000
1108535	206209	123	1.000000

1108536 rows × 3 columns

Predicts

Department Popularity:

The output is a DataFrame that shows the number of products ordered per department, sorted in descending order by product count. This is useful for understanding the popularity of different product departments based on the number of orders.

	department	product_count
19	produce	588125
7	dairy eggs	336440
20	snacks	180450
3	beverages	167909
10	frozen	139316
16	pantry	116101
2	bakery	72881
6	canned goods	65951
8	deli	65059
9	dry goods pasta	53956
11	household	46380
4	breakfast	44543
13	meat seafood	44206
17	personal care	28093
1	babies	25898
12	international	16711
0	alcohol	9428
18	pets	6000
14	missing	4743
15	other	2235
5	bulk	2133

Product Affinity:(Apriori Algorithm)

product affinity analysis using the Apriori algorithm. The dataset is first prepared by grouping by order and product and then converting the data into a binary format to indicate whether a product was purchased in an order or not. A sample of the dataset is then taken to reduce the computation time, and the Apriori algorithm is run to find frequent itemsets with a minimum support of 0.005. Finally, association rules are generated using a minimum lift of 1.

	support	itemsets
0	0.0085	(air fresheners candles)
1	0.0425	(asian foods)
2	0.0470	(baby food formula)
3	0.0130	(bakery desserts)
4	0.0795	(baking ingredients)
...	...	...
13454	0.0070	(packaged vegetables fruits, packaged cheese, ...
13455	0.0050	(packaged vegetables fruits, packaged cheese, ...
13456	0.0050	(packaged vegetables fruits, packaged cheese, ...
13457	0.0055	(packaged vegetables fruits, soup broth bouill...
13458	0.0065	(packaged vegetables fruits, water seltzer spa...

13459 rows × 2 columns

Rules

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(asian foods)	(baking ingredients)	0.0425	0.0795	0.0060	0.141176	1.775805	0.002621	1.071815
1	(baking ingredients)	(asian foods)	0.0795	0.0425	0.0060	0.075472	1.775805	0.002621	1.035663
2	(asian foods)	(bread)	0.0425	0.1640	0.0110	0.258824	1.578192	0.004030	1.127937
3	(bread)	(asian foods)	0.1640	0.0425	0.0110	0.067073	1.578192	0.004030	1.026340
4	(asian foods)	(canned jarred vegetables)	0.0425	0.0670	0.0060	0.141176	2.107112	0.003152	1.086370
...	...	...	...	...	...	...	...	...	...
172991	(packaged cheese)	(packaged vegetables fruits, water seltzer spa...	0.2290	0.0105	0.0065	0.028384	2.703265	0.004095	1.018407
172992	(fresh vegetables)	(packaged vegetables fruits, water seltzer spa...	0.4405	0.0075	0.0065	0.014756	1.967461	0.003196	1.007365
172993	(milk)	(packaged vegetables fruits, water seltzer spa...	0.2440	0.0085	0.0065	0.026639	3.134041	0.004426	1.018636
172994	(yogurt)	(packaged vegetables fruits, water seltzer spa...	0.2665	0.0075	0.0065	0.024390	3.252033	0.004501	1.017313
172995	(fresh fruits)	(packaged vegetables fruits, water seltzer spa...	0.5740	0.0065	0.0065	0.011324	1.742160	0.002769	1.004879

172996 rows × 9 columns

Next product recommendation:(KNN)

In the code example provided, the Surprise library is used to create a recommender system that makes product recommendations based on user behavior data. Here's what each line of code does:

Define a Reader object with a rating scale of (1, 1). This indicates that the ratings in the data set are binary, meaning that each user either has or has not reordered a particular product.
Load the data from the df DataFrame into a Dataset object, using the load_from_df() method. The Dataset object is a container for the data that is used to train and test the recommendation model.
Split the data into training and test sets using the train_test_split() method from the surprise.model_selection module. The test_size parameter is set to 0.25, indicating that 25% of the data should be reserved for testing the model's performance.
Train a KNNWithMeans model using the fit() method. This model is based on the k-nearest neighbors algorithm and calculates the similarity between items (in this case, products) based on user ratings. The sim_options parameter specifies that the Pearson correlation coefficient should be used to measure similarity, and that the model should be item-based (as opposed to user-based).
Make product recommendations for a specific user (user_id) by calling the get_neighbors() method of the KNNWithMeans model. The method returns a list of the n_recommendations most similar products to those that the user has already reordered, based on the similarity scores calculated during training.
The output of predictions is a list of tuples, where each tuple contains the ID of a recommended product and its similarity score (i.e., how similar the product is to the products that the user has already reordered).

```
Estimating biases using als...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.

[104, 132, 26, 84, 124, 54, 110, 103, 75, 17]
```

Recommendation

Churn Prediction:(Random forest classifier algorithm)

The data is prepared by grouping the user data by the maximum order number and the average number of days since the last order, then a churn column is created based on whether or not a user's last order was within the bottom 10 percentile of order numbers. The data is split into training and test sets, and a random forest classifier is trained on the training set using the number of days since the last order as the predictor variable and the churn column as the target variable. Finally, the model makes predictions on the test set, and the predicted churn values are outputted as an array.

```
array([0, 0, 1, ..., 0, 0, 0])
```

```
Number of churn:  1193
```

Confusion Matrix:

```
[[19000    861]
[ 93    1100]]
```

Evaluation of model:

if we get 1 for all outputs of accuracy, precision, recall and F1 score, it means that the model's predictions are good. In other words, the model has correctly classified all the instances in the test set, and there are less false positives or false negatives.

```
Accuracy Score:95%
precision Score:95%
Recall score:95%
F1 Score:95%
```

Chapter-4

Summary/Conclusion

CONCLUSION

In order to predict the hunters e-grocery customer needs based on customer behaviour.

so we are using the data set propose business value for informative based descisionmaking , we developed an algorithm in which we applied machine learning algorithms in order to select thebest algorithm. In our data we used the algorithms with 25% on test and rest on train data.

The accuracy of Train and Test data are 95% and 95% respectively. Since,the accuracy of Train and Test data are similar, we can say that our random foreset classifier model can be used

Hence, we have applied the random forest classifier model:churn values with total number of churns

```
array([0, 0, 1, ..., 0, 1, 0])
```

```
Number of churn:  1193
```

Accuracy:

95% accuracy

Since the accuracy of train and test data are similar, this model is generalised. So, we used this model to predict the future data.

KNN . Below are the n-recommendations to user_ID 2 for train and test:

N-recommendations for user-id 2:

predictions is a list of tuples, where each tuple contains the ID of a recommended product and its similarity score (i.e., how similar the product is to the products that the user has already reordered).

```
[52, 18, 20, 4, 29, 14, 23, 75, 54, 0]
```

Apriori

Frequent-items

	support	itemsets
0	0.0460	(asian foods)
1	0.0440	(baby food formula)
2	0.0120	(bakery desserts)
3	0.0710	(baking ingredients)
4	0.0065	(baking supplies decor)
...	...	...

Rules:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(baking ingredients)	(asian foods)	0.0710	0.0460	0.0065	0.091549	1.990202	0.003234	1.050140
1	(asian foods)	(baking ingredients)	0.0460	0.0710	0.0065	0.141304	1.990202	0.003234	1.081873
2	(bread)	(asian foods)	0.1610	0.0460	0.0080	0.049689	1.080205	0.000594	1.003882
3	(asian foods)	(bread)	0.0460	0.1610	0.0080	0.173913	1.080205	0.000594	1.015632
4	(breakfast bakery)	(asian foods)	0.0725	0.0460	0.0055	0.075862	1.649175	0.002165	1.032313
...	...	...	...	...	...	...	...	...	...

APPENDIX

PYTHON – CODE

DATASET

BIBLIOGRAPHY

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
import os
```

In [3]:
```
pip install scikit-surprise
```

```
Requirement already satisfied: scikit-surprise in c:\users\nitya\anaconda3\lib\site-packages (1.1.3)
Requirement already satisfied: joblib>=1.0.0 in c:\users\nitya\anaconda3\lib\site-packages (from scikit-surprise) (1.1.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\nitya\anaconda3\lib\site-packages (from scikit-surprise) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in c:\users\nitya\anaconda3\lib\site-packages (from scikit-surprise) (1.10.0)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
from sklearn.decomposition import TruncatedSVD
```

In [5]:
```
from sklearn.metrics.pairwise import cosine_similarity
from surprise import Reader, Dataset
from surprise.prediction_algorithms import SVD
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [6]:
```
pip install mlxtend
```

In [7]:
```
import mlxtend
from mlxtend.frequent_patterns import association_rules, apriori
from mlxtend.preprocessing import TransactionEncoder
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from surprise import Dataset, Reader
from surprise import KNNWithMeans
import statsmodels.api as sm
import warnings
warnings.filterwarnings("ignore")
```

In [8]:
```
df=pd.read_csv("ecommerce.csv")
```

In [9]:
```
df
```

Out[9]:

	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id
0	2425083	49125	1	2	18	NaN	17	1	0	13
1	2425083	49125	1	2	18	NaN	91	2	0	16
2	2425083	49125	1	2	18	NaN	36	3	0	16
3	2425083	49125	1	2	18	NaN	83	4	0	4
4	2425083	49125	1	2	18	NaN	83	5	0	4
...	...	...	...	...	...	...	...	...	...	...
2019496	3390742	199430	16	3	18	5.0	83	8	0	4
2019497	458285	128787	42	2	19	3.0	115	1	1	7
2019498	458285	128787	42	2	19	3.0	32	2	1	4
2019499	458285	128787	42	2	19	3.0	32	3	1	4
2019500	458285	128787	42	2	19	3.0	123	4	1	4

2019501 rows × 12 columns

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
order_id                      0
user_id                       0
order_number                  0
order_dow                     0
order_hour_of_day             0
days_since_prior_order    124342
product_id                    0
add_to_cart_order             0
reordered                     0
department_id                 0
department                    0
product_name                  0
dtype: int64
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2019501 entries, 0 to 2019500
Data columns (total 12 columns):
 #   Column                  Dtype
---  ------                  -----
 0   order_id                int64
 1   user_id                 int64
 2   order_number            int64
 3   order_dow               int64
 4   order_hour_of_day       int64
 5   days_since_prior_order  float64
 6   product_id              int64
 7   add_to_cart_order       int64
 8   reordered               int64
 9   department_id           int64
 10  department              object
 11  product_name            object
dtypes: float64(1), int64(9), object(2)
```

In [12]:
```
df['days_since_prior_order'] = df['days_since_prior_order'].fillna(-1)# fill null values with -1 and change type
df['days_since_prior_order']=df['days_since_prior_order'].astype('int64')
```

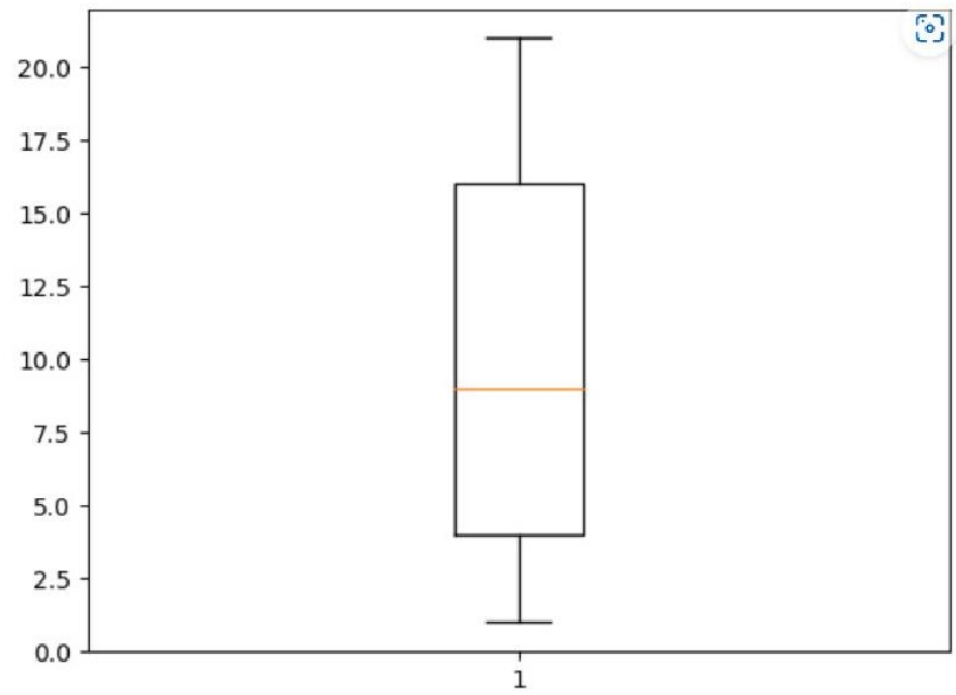
In [13]:
```
df.isnull().sum()
```

Out[13]:
```
order_id                  0
user_id                   0
order_number              0
order_dow                 0
order_hour_of_day         0
days_since_prior_order    0
product_id                0
add_to_cart_order         0
reordered                 0
department_id             0
department                0
product_name              0
dtype: int64
```

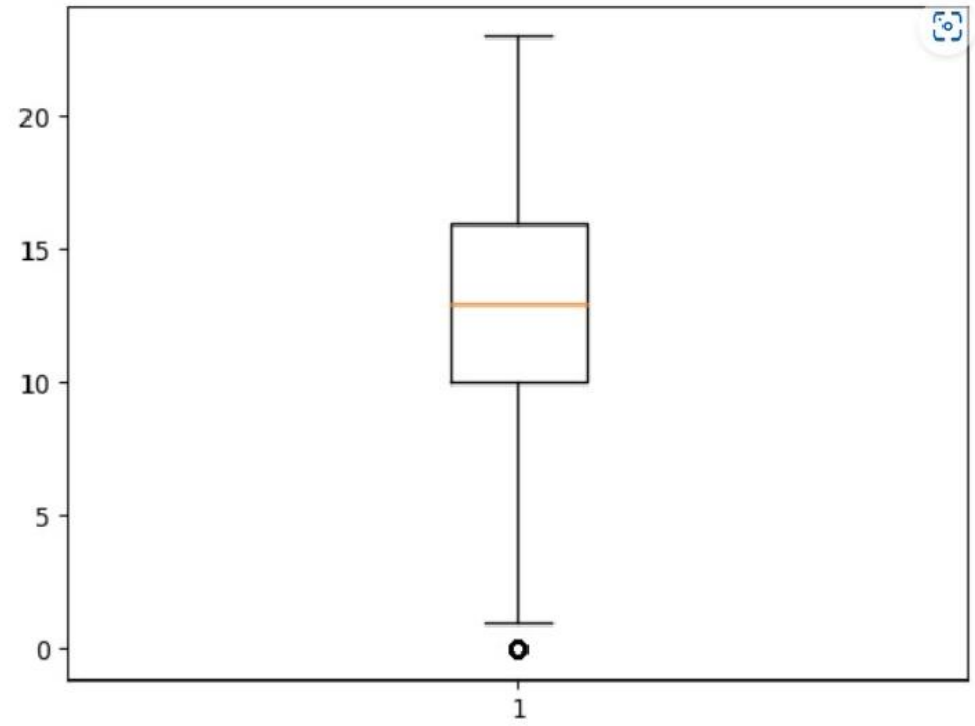
In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2019501 entries, 0 to 2019500
Data columns (total 12 columns):
 #   Column                  Dtype
---  ------                  -----
 0   order_id                int64
 1   user_id                 int64
 2   order_number            int64
 3   order_dow               int64
 4   order_hour_of_day       int64
 5   days_since_prior_order  int64
 6   product_id              int64
 7   add_to_cart_order       int64
 8   reordered               int64
 9   department_id           int64
 10  department              object
 11  product_name            object
dtypes: int64(10), object(2)
memory usage: 184.9+ MB
```

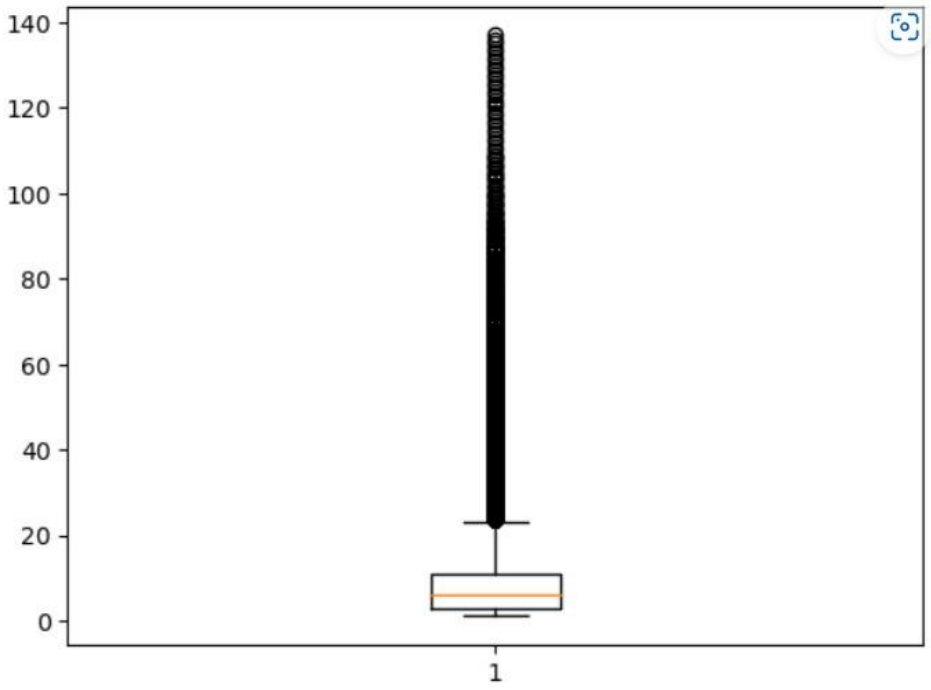
In [15]:
```
plt.boxplot(df['department_id'])
plt.show()
```



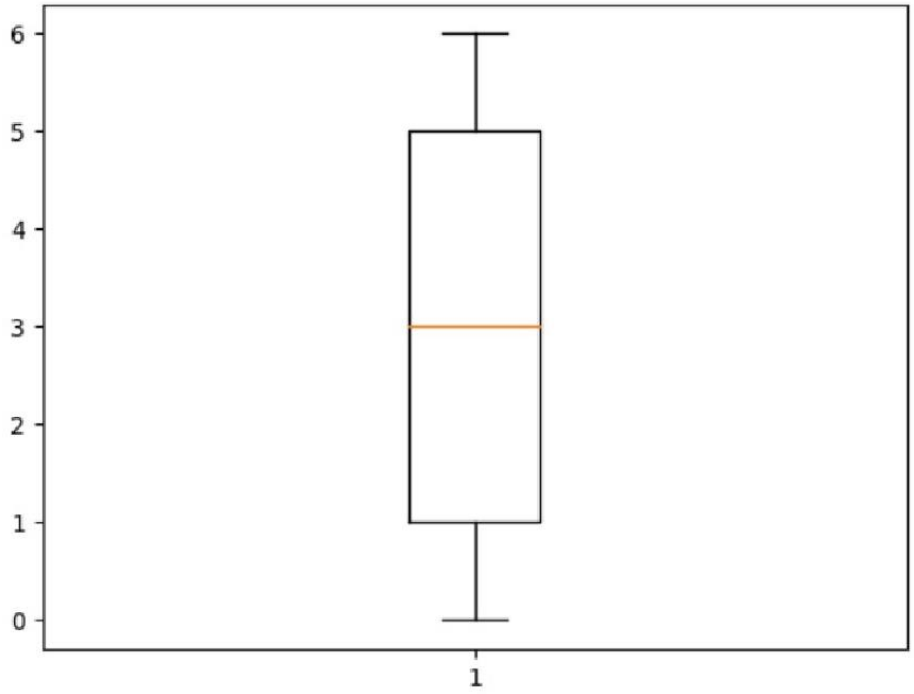
In [16]:
```
plt.boxplot(df['order_hour_of_day'])
plt.show()
```



In [17]:
```
plt.boxplot(df['add_to_cart_order'])
plt.show()
```



In [18]:
```
plt.boxplot(df['order_dow'])
plt.show()
```



In [19]:
```
# OUTLIER customers based on add_to_cart_order

# group the data by user_id and calculate the mean value for each user
df_mean = df.groupby('user_id').mean()
# calculate the IQR for the addtocart_order feature
Q1 = df_mean['add_to_cart_order'].quantile(0.25)
Q3 = df_mean['add_to_cart_order'].quantile(0.75)
IQR = Q3 - Q1
# find the outlier threshold
outlier_threshold = Q3 + 1.5 * IQR
outlier_customers = df_mean[df_mean['add_to_cart_order'] > outlier_threshold]
#outlier_customers = outlier_customers.reset_index()
outlier_customers
```

Out[19]:

user_id	order_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id
10	8.339500e+04	4.000000	3.000000	15.000000	14.00000	60.000000	15.500000	0.433333	10.400000
133	2.415920e+05	11.000000	3.000000	13.000000	5.00000	68.133333	15.500000	0.400000	9.033333
197	2.357070e+05	7.000000	6.000000	14.000000	6.00000	74.206897	15.000000	0.793103	11.551724
216	1.500908e+06	5.000000	5.000000	15.000000	24.00000	68.448276	15.000000	0.172414	13.655172
264	3.022479e+06	29.000000	2.000000	9.000000	9.00000	79.551724	15.000000	0.827586	11.103448
...	...	...	...	...	...	...	...	...	...
205789	3.418993e+06	13.000000	6.000000	12.000000	7.00000	63.900000	15.500000	0.600000	9.466667
205926	2.385091e+06	11.000000	1.000000	15.000000	6.00000	73.828571	18.000000	0.514286	10.171429
205956	4.562900e+05	3.000000	3.000000	14.000000	30.00000	69.151515	17.000000	0.303030	10.909091
206146	3.185446e+06	2.000000	2.000000	9.000000	30.00000	72.256410	20.000000	0.025641	11.230769

In [20]:
```
# OUTLIER customers in the purchases made by each customer

# Group the data by user_id
grouped_data = df.groupby("user_id")["order_number"].agg(["mean", "std"])

# Calculate the lower and upper bounds for outliers
lower_bound = grouped_data["mean"] - 3 * grouped_data["std"]
lower_bound.name = 'order_number_lower'
upper_bound = grouped_data["mean"] + 3 * grouped_data["std"]
upper_bound.name = 'order_number_upper'
# join the lower and upper bounds to the original dataframe on user_id
df = df.join(lower_bound, on='user_id')
df = df.join(upper_bound, on='user_id')
# Identify the outliers
outliers = df[(df["order_number"] < df['order_number_lower']) | (df["order_number"] > df['order_number_upper'])]
outliers
```

Out[20]:

	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id
2538	677735	175561	2	3	12	16	32	1	0	4
8764	2018358	185514	22	4	17	17	128	1	0	3
8765	2018358	185514	22	4	17	17	54	2	1	17
15536	2545752	185279	9	5	9	12	84	1	0	16
16290	1365995	136421	1	3	10	-1	88	1	0	13
...	...	...	...	...	...	...	...	...	...	...
2010592	1097536	97028	5	0	16	7	24	1	1	4
2010593	1097536	97028	5	0	16	7	24	2	0	4
2015806	1581072	93695	5	4	17	30	97	1	0	13
2016905	1373400	139870	11	2	12	30	116	1	0	1
2016906	1373400	139870	11	2	12	30	32	2	0	4

1282 rows × 14 columns

In [21]:
```
df=df.drop(outlier_customers.index)
df
```

Out[21]:

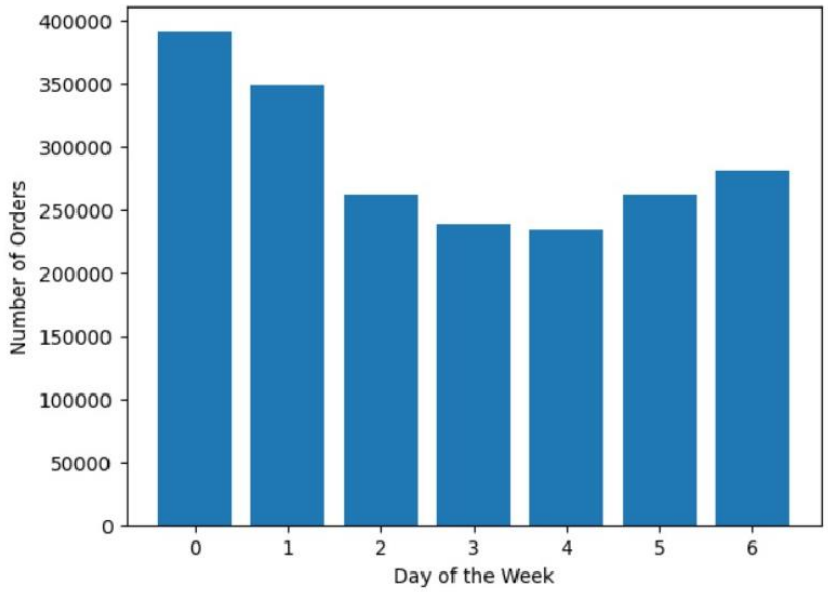
	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id
0	2425083	49125	1	2	18	-1	17	1	0	13
1	2425083	49125	1	2	18	-1	91	2	0	16
2	2425083	49125	1	2	18	-1	36	3	0	16
3	2425083	49125	1	2	18	-1	83	4	0	4
4	2425083	49125	1	2	18	-1	83	5	0	4
...	...	...	...	...	...	...	...	...	...	...
2019496	3390742	199430	16	3	18	5	83	8	0	4
2019497	458285	128787	42	2	19	3	115	1	1	7
2019498	458285	128787	42	2	19	3	32	2	1	4
2019499	458285	128787	42	2	19	3	32	3	1	4
2019500	458285	128787	42	2	19	3	123	4	1	4

2016558 rows × 14 columns

In [23]:

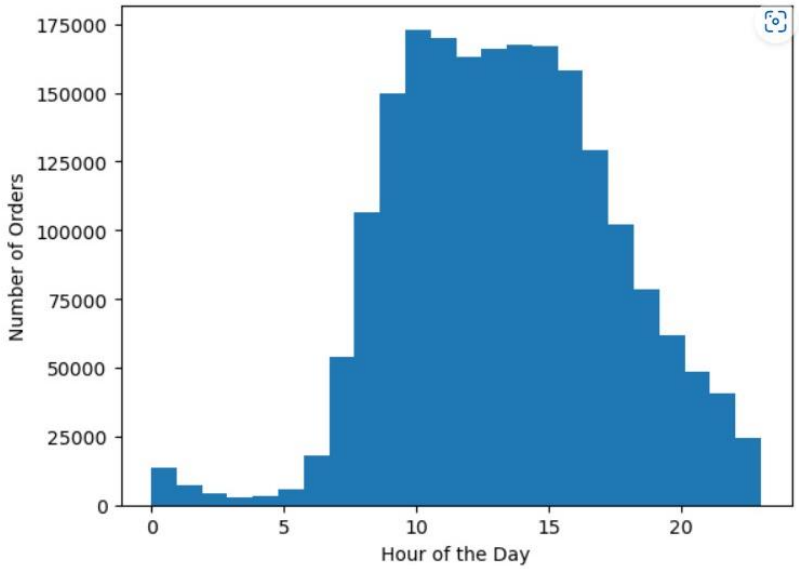
```
# distribution of orders by day of the week
order_dow_counts = df.groupby('order_dow').size()

plt.bar(order_dow_counts.index, order_dow_counts.values)
plt.xlabel('Day of the Week')
plt.ylabel('Number of Orders')
plt.show()
```

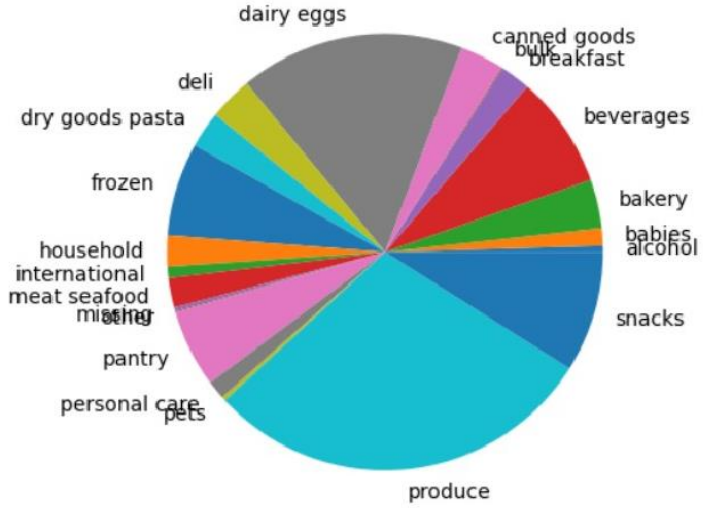


In [24]:

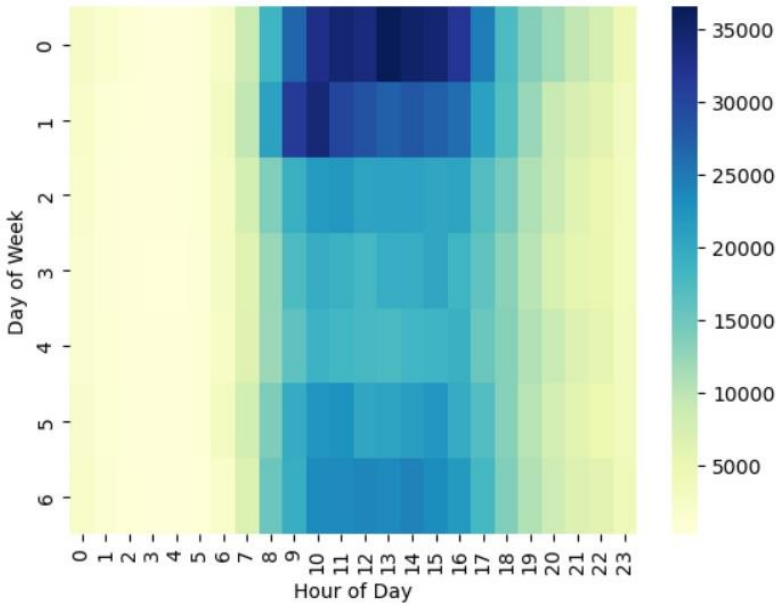
```
# distribution of orders by hour of the day
plt.hist(df['order_hour_of_day'], bins=24)
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Orders')
plt.show()
```



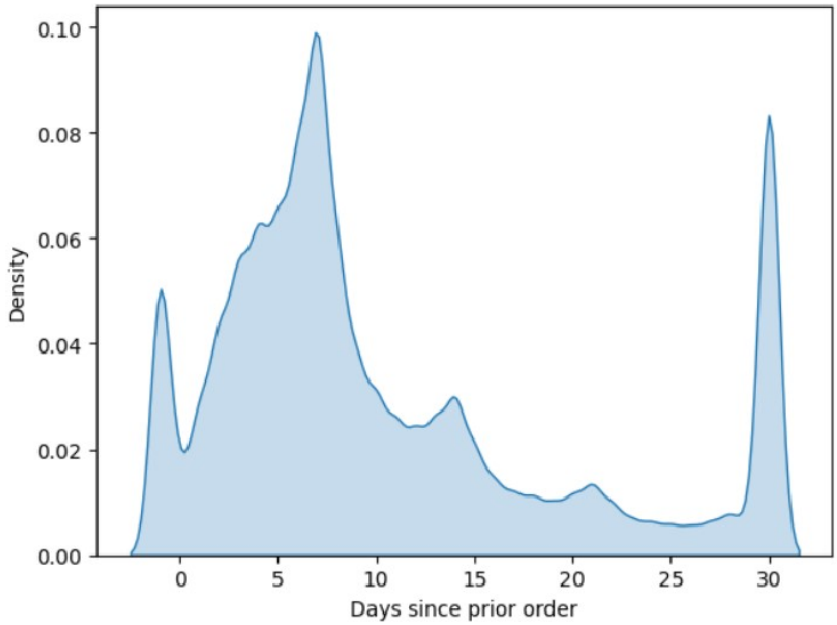
In [25]:
```
# distribution of orders by department
department_counts = df.groupby('department').size()
plt.pie(department_counts.values, labels=department_counts.index)
plt.show()
```



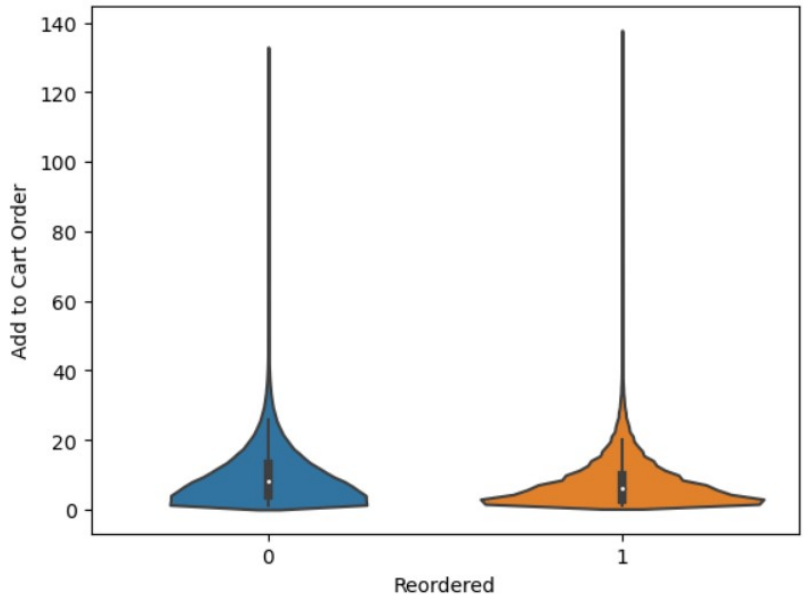
In [26]:
```
# relationship between the day of the week and hour of the day for orders
pivot = df.pivot_table(values='order_id', index='order_dow', columns='order_hour_of_day', aggfunc='count')
sns.heatmap(pivot, cmap='YlGnBu')
plt.xlabel('Hour of Day')
plt.ylabel('Day of Week')
plt.show()
```



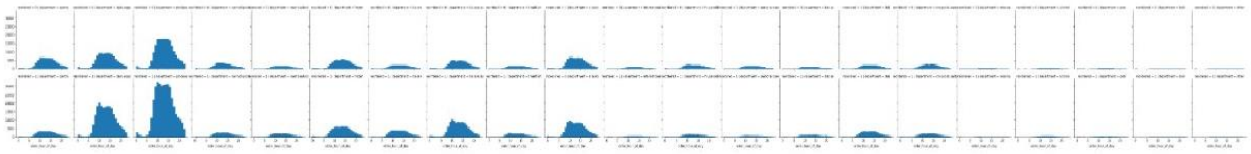
In [27]:
```
# distribution of days since prior order
sns.kdeplot(df['days_since_prior_order'], shade=True)
plt.xlabel('Days since prior order')
plt.show()
```



In [28]:
```
# relationship between the number of items added to cart and the reorder status
sns.violinplot(x='reordered', y='add_to_cart_order', data=df)
plt.xlabel('Reordered')
plt.ylabel('Add to Cart Order')
plt.show()
```



In [29]:
```
# distribution of reorders by department and hour of the day
g = sns.FacetGrid(df, col="department", row="reordered")
g = g.map(plt.hist, "order_hour_of_day",  bins=24)
plt.show()
```



In [30]:
```
# 1 Product Reorder Prediction

# Group the data by user_id and product_id, calculating the mean of the 'reordered' column for each group
reorder_prob = df.groupby(['user_id','product_id'])['reordered'].mean().reset_index()
# Rename the columns
reorder_prob = reorder_prob.rename(columns={'reordered':'reorder_prob'})
reorder_prob
```

Out[30]:

	user_id	product_id	reorder_prob
0	2	1	0.000000
1	2	21	0.000000
2	2	23	1.000000
3	2	24	0.666667
4	2	67	0.000000
...	...	...	...
1108531	206209	112	0.500000
1108532	206209	116	0.000000
1108533	206209	120	0.000000
1108534	206209	121	0.000000
1108535	206209	123	1.000000

1108536 rows × 3 columns

In [31]:
```
# 2 Department Popularity

# Count number of products ordered per department
department_popularity = df.groupby('department')['product_id'].count().reset_index()
# Rename the column
department_popularity = department_popularity.rename(columns={'product_id':'product_count'})
# Sort the data by product_count
department_popularity = department_popularity.sort_values('product_count', ascending=False)
department_popularity
```

Out[31]:

	department	product_count
19	produce	588125
7	dairy eggs	336440
20	snacks	180450
3	beverages	167909
10	frozen	139316
16	pantry	116101
2	bakery	72881
6	canned goods	65951
8	deli	65059
9	dry goods pasta	53956
11	household	46380
4	breakfast	44543
13	meat seafood	44206
17	personal care	28093
1	babies	25898
12	international	16711
0	alcohol	9428
18	pets	6000
14	missing	4743
15	other	2235
5	bulk	2133

In [32]:
```
# 3 Product Affinity

# Prepare the data for the Apriori algorithm
basket = (df.groupby(['order_id', 'product_name'])['product_name']
          .count().unstack().reset_index().fillna(0)
          .set_index('order_id'))
basket_sets = basket.applymap(lambda x: 0 if x<=0 else 1)
```

In [33]:
```
basket_sets = basket_sets.sample(frac=0.01)# Take a sample
```

In [34]:
```
# Run the Apriori algorithm to find frequent item sets
frequent_itemsets = apriori(basket_sets, min_support=0.005, use_colnames=True)
```

In [35]:
```
# Generate the association rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
frequent_itemsets
```

Out[36]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(baking ingredients)	(asian foods)	0.0710	0.0460	0.0065	0.091549	1.990202	0.003234	1.050140
1	(asian foods)	(baking ingredients)	0.0460	0.0710	0.0065	0.141304	1.990202	0.003234	1.081873
2	(bread)	(asian foods)	0.1610	0.0460	0.0080	0.049689	1.080205	0.000594	1.003882
3	(asian foods)	(bread)	0.0460	0.1610	0.0080	0.173913	1.080205	0.000594	1.015632
4	(breakfast bakery)	(asian foods)	0.0725	0.0460	0.0055	0.075862	1.649175	0.002165	1.032313
...	...	...	...	...	...	...	...	...	...
145607	(fresh vegetables)	(packaged vegetables fruits, yogurt, milk, fro...	0.4670	0.0090	0.0060	0.012848	1.427552	0.001797	1.003898
145608	(milk)	(packaged vegetables fruits, yogurt, fresh veg...	0.2480	0.0085	0.0060	0.024194	2.846300	0.003892	1.016083
145609	(frozen produce)	(packaged vegetables fruits, yogurt, fresh veg...	0.1210	0.0165	0.0060	0.049587	3.005259	0.004004	1.034813
145610	(fresh fruits)	(packaged vegetables fruits, yogurt, fresh veg...	0.5715	0.0060	0.0060	0.010499	1.749781	0.002571	1.004546
145611	(packaged cheese)	(packaged vegetables fruits, yogurt, fresh veg...	0.2460	0.0095	0.0060	0.024390	2.567394	0.003663	1.015263

145612 rows × 9 columns

In [37]:
```
# 4 for finding similar product which customers has reordered


# Define a reader object
reader = Reader(rating_scale=(1, 1))
# Load the data into a Dataset object
data = Dataset.load_from_df(df[['user_id', 'product_id', 'reordered']], reader)
# Split the data into training and test sets
from surprise.model_selection import train_test_split
trainset, testset = train_test_split(data, test_size=.25, random_state=123)
# Train the model
algo = KNNWithMeans(k=50, sim_options={'name': 'pearson_baseline', 'user_based': False})
algo.fit(trainset)
# Make recommendations for a specific product ID
user_id = 2
n_recommendations = 10
predictions = algo.get_neighbors(user_id, k=n_recommendations)
predictions
```

```
Estimating biases using als...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.
```

Out[37]:
```
[104, 132, 26, 84, 124, 54, 110, 103, 75, 17]
```

In [39]:
```
# 5 Churn Prediction
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score
# Prepare the data
df_churn = df.groupby('user_id').agg({'order_number':'max','days_since_prior_order':'mean'}).reset_index()
df_churn['churn'] = (df_churn['order_number'] < df_churn['order_number'].quantile(0.1)).astype(int)
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df_churn[['days_since_prior_order']], df_churn['churn'], test_size=0.2, r
clf = RandomForestClassifier(n_estimators=100, max_depth=4, random_state=123)
clf.fit(X_train, y_train)
# Make predictions
y_pred = clf.predict(X_test)

y_pred
```

Out[39]:
```
array([0, 0, 1, ..., 0, 1, 0])
```

In [40]:
```
num_churn = sum(y_pred)
print("Number of churn: ", num_churn)
```

```
Number of churn:  1193
```

In [41]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
y_test_pred = clf.predict(X_test)
test_accuracy = accuracy_score(y_test, y_test_pred)
test_precision = precision_score(y_test, y_test_pred)
test_recall = recall_score(y_test, y_test_pred)
test_f1_score = f1_score(y_test, y_test_pred)
```

```
Accuracy Score:95%
precision Score:95%
Recall score:95%
F1 Score:95%
```

In [42]:
```
from sklearn.metrics import confusion_matrix

# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

```
[[19000    861]
 [ 93    1100]]
```

DATASET

order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id	department	product_name
2425083	49125	1	2	18	2	17	1	0	13	pantry	baking ingredients
2425083	49125	1	2	18	2	91	2	0	16	dairy eggs	soy lactosefree
2425083	49125	1	2	18	2	36	3	0	16	dairy eggs	butter
2425083	49125	1	2	18	2	83	4	0	4	produce	fresh vegetables
2425083	49125	1	2	18	2	83	5	0	4	produce	fresh vegetables
2425083	49125	1	2	18	2	91	6	0	16	dairy eggs	soy lactosefree
2425083	49125	1	2	18	2	120	7	0	16	dairy eggs	yogurt
2425083	49125	1	2	18	2	59	8	0	15	canned goods	canned meals beans
2425083	49125	1	2	18	2	35	9	0	12	meat seafood	poultry counter
1944304	162867	1	3	17	2	37	1	0	1	frozen	ice cream ice
1944304	162867	1	3	17	2	24	2	0	4	produce	fresh fruits
1944304	162867	1	3	17	2	83	3	0	4	produce	fresh vegetables
1944304	162867	1	3	17	2	84	4	0	16	dairy eggs	milk
1944304	162867	1	3	17	2	91	5	0	16	dairy eggs	soy lactosefree
1944304	162867	1	3	17	2	24	6	0	4	produce	fresh fruits
1944304	162867	1	3	17	2	24	7	0	4	produce	fresh fruits
1944304	162867	1	3	17	2	24	8	0	4	produce	fresh fruits
1944304	162867	1	3	17	2	21	9	0	16	dairy eggs	packaged cheese
1944304	162867	1	3	17	2	112	10	0	3	bakery	bread
1944304	162867	1	3	17	2	24	11	0	4	produce	fresh fruits
1944304	162867	1	3	17	2	24	12	0	4	produce	fresh fruits
1944304	162867	1	3	17	2	24	13	0	4	produce	fresh fruits
1201011	147243	14	0	16	3	94	1	0	7	beverages	tea
1201011	147243	14	0	16	3	83	2	0	4	produce	fresh vegetables
1201011	147243	14	0	16	3	83	3	1	4	produce	fresh vegetables
1201011	147243	14	0	16	3	24	4	1	4	produce	fresh fruits
1201011	147243	14	0	16	3	120	5	1	16	dairy eggs	yogurt

1201011	147243	14	0	16	3	120	6	1	16	dairy eggs	yogurt
1201011	147243	14	0	16	3	91	7	1	16	dairy eggs	soy lactosefree
1201011	147243	14	0	16	3	120	8	1	16	dairy eggs	yogurt
1201011	147243	14	0	16	3	120	9	1	16	dairy eggs	yogurt
1201011	147243	14	0	16	3	8	10	0	3	bakery	bakery desserts
1201011	147243	14	0	16	3	24	11	1	4	produce	fresh fruits
1201011	147243	14	0	16	3	52	12	0	1	frozen	frozen breakfast
1201011	147243	14	0	16	3	36	13	1	16	dairy eggs	butter
1201011	147243	14	0	16	3	121	14	0	14	breakfast	cereal
1201011	147243	14	0	16	3	86	15	0	16	dairy eggs	eggs
1906860	195275	81	1	11	3	83	1	1	4	produce	fresh vegetables
1906860	195275	81	1	11	3	86	2	1	16	dairy eggs	eggs
1906860	195275	81	1	11	3	24	3	1	4	produce	fresh fruits
1906860	195275	81	1	11	3	24	4	1	4	produce	fresh fruits
1906860	195275	81	1	11	3	43	5	1	3	bakery	buns rolls
1906860	195275	81	1	11	3	120	6	1	16	dairy eggs	yogurt
1906860	195275	81	1	11	3	24	7	1	4	produce	fresh fruits
1906860	195275	81	1	11	3	83	8	1	4	produce	fresh vegetables
1906860	195275	81	1	11	3	83	9	1	4	produce	fresh vegetables
1906860	195275	81	1	11	3	83	10	1	4	produce	fresh vegetables
1906860	195275	81	1	11	3	24	11	1	4	produce	fresh fruits
2766469	58222	3	6	11	6	53	1	1	16	dairy eggs	cream
2766469	58222	3	6	11	6	86	2	1	16	dairy eggs	eggs
2766469	58222	3	6	11	6	115	3	1	7	beverages	water seltzer sparkling water
2766469	58222	3	6	11	6	120	4	0	16	dairy eggs	yogurt
2766469	58222	3	6	11	6	17	5	0	13	pantry	baking ingredients
2766469	58222	3	6	11	6	17	6	1	13	pantry	baking ingredients
2766469	58222	3	6	11	6	17	7	0	13	pantry	baking ingredients
2766469	58222	3	6	11	6	24	8	0	4	produce	fresh fruits
2766469	58222	3	6	11	6	110	9	0	13	pantry	pickled goods olives
2766469	58222	3	6	11	6	49	10	0	12	meat seafood	packaged poultry
2766469	58222	3	6	11	6	83	11	0	4	produce	fresh vegetables
2766469	58222	3	6	11	6	21	12	0	16	dairy eggs	packaged cheese
2766469	58222	3	6	11	6	21	13	0	16	dairy eggs	packaged cheese
2766469	58222	3	6	11	6	83	14	0	4	produce	fresh vegetables

2766469	58222	3	6	11	6	108	15	0	16	dairy eggs	other creams cheeses
3195784	120094	11	5	17	7	29	1	0	13	pantry	honeys syrups nectars
3195784	120094	11	5	17	7	26	2	1	7	beverages	coffee
3195784	120094	11	5	17	7	91	3	1	16	dairy eggs	soy lactosefree
3195784	120094	11	5	17	7	31	4	1	7	beverages	refrigerated
3195784	120094	11	5	17	7	24	5	1	4	produce	fresh fruits
3195784	120094	11	5	17	7	3	6	1	19	snacks	energy granola bars
1378095	73110	1	6	10		77	1	0	7	beverages	soft drinks
1378095	73110	1	6	10		30	2	0	6	international	latino foods
1304367	53959	5	5	1	30	111	1	1	17	household	plates bowls cups flatware
1304367	53959	5	5	1	30	54	2	1	17	household	paper goods
1304367	53959	5	5	1	30	91	3	1	16	dairy eggs	soy lactosefree
1304367	53959	5	5	1	30	20	4	1	11	personal care	oral hygiene
1304367	53959	5	5	1	30	56	5	1	18	babies	diapers wipes
1304367	53959	5	5	1	30	85	6	0	17	household	food storage
1304367	53959	5	5	1	30	85	7	0	17	household	food storage
1304367	53959	5	5	1	30	85	8	0	17	household	food storage
1304367	53959	5	5	1	30	24	9	1	4	produce	fresh fruits
1304367	53959	5	5	1	30	117	10	1	19	snacks	nuts seeds dried fruit
1304367	53959	5	5	1	30	25	11	1	11	personal care	soap
1304367	53959	5	5	1	30	25	12	0	11	personal care	soap
1670129	176782	12	0	8	20	24	1	1	4	produce	fresh fruits
1670129	176782	12	0	8	20	24	2	1	4	produce	fresh fruits
1670129	176782	12	0	8	20	24	3	1	4	produce	fresh fruits
1670129	176782	12	0	8	20	120	4	1	16	dairy eggs	yogurt
1670129	176782	12	0	8	20	123	5	1	4	produce	packaged vegetables fruits
1670129	176782	12	0	8	20	106	6	1	12	meat seafood	hot dogs bacon sausage
1670129	176782	12	0	8	20	96	7	1	20	deli	lunch meat
1670129	176782	12	0	8	20	83	8	1	4	produce	fresh vegetables
1670129	176782	12	0	8	20	107	9	1	19	snacks	chips pretzels
1670129	176782	12	0	8	20	77	10	1	7	beverages	soft drinks
1670129	176782	12	0	8	20	123	11	1	4	produce	packaged vegetables fruits
1670129	176782	12	0	8	20	122	12	1	12	meat seafood	meat counter

1670129	176782	12	0	8	20	24	13	1	4	produce	fresh fruits
1670129	176782	12	0	8	20	49	14	1	12	meat seafood	packaged poultry
1670129	176782	12	0	8	20	35	15	1	12	meat seafood	poultry counter
1670129	176782	12	0	8	20	67	16	0	20	deli	fresh dips tapenades

As the data is Complex and big, it has 2019501 rows and 12 columns. I have given only 100 rows.

BIBLIOGRAPHY

1. Python Data science hand book – Jake vanderplas ,O'REILLY Publications

2. Introduction to Machine learning to python – Andreas C Muller,Sarah Guido, O'REILLY publications

3. Python Machine Learning second edition-Sebastianraschka, PACKT publications

4. Python Machine Learning- Leonardlee

5. Machine Learning using python – Manaranjan Pradhan,U Dinesh Kumar,WILEY publications

6. Data Science Projects with Python – Stephen Klosterman, PACKT publications

7. Machine Learning in python – Michael Bowles, WILEY publications

8. Python:Data Analytics and Visualization – Ashish kumar, Kirthi Raman, PACKT publications

9. Learning Predictive Analytics with Python – Ashish Kumar, PACKT publications