DAIRY GOODS SALES PREDICTION

Group Members:

VISHWANATH, LOVELY PROFESSIONAL UNIVERSITY, 12108780

Table of Contents

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty Prof. Arnab Chakraborty for his exemplary guidance, monitoring, and constant encouragement throughout the course of this project. The blessing, help and guidance given by him/her time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

VISHWANATH .R

Project Objective

In this project we have a shortened 'The Dairy Goods Sales Prediction' Dataset from Kaggle. In this dataset, the target attribute is the Approx_Total Revenue(INR) . So, in this project we need to do regression based on the attributes present in our dataset and predict sales of dairy products. Our objective in this project is to study the given dataset of 'The Dairy Goods Sales'. We might need to pre-process the given dataset if we need to. Then, we would train 4 models viz. 'Linear Regression model', 'Lasso model', 'Random Forest Regressor model' and 'Ridge model'. After training the aforementioned models, we will need to find out the r2_score,MSE. Our next step would be to use the trained models to predict the outcomes using the given test dataset and compare the outcome of each model. We would then choose the best model based on the accuracy score . Our methodology for solving the problems in the given project is described below:

• Load the required dataset.
• Study the dataset.
• Describe the dataset.
• Visualise the dataset.
• Find out if the dataset needs to be pre-processed. It will be determined on the basis of whether the dataset has null values or outliers or any such discrepancy that might affect the output of the models to be trained.
• If the dataset is required to be pre-processed, take the necessary steps to pre-process the data.
• Find out the principal attributes for training.
• Split the given dataset for training and testing purpose.
• Fit the previously split train data in the aforementioned 4 models.
• Calculate the accuracy of the 4 models.
• Plot the necessary graphs.
• Use each trained model to predict the outcomes of the given test dataset.
• Choose the best model among the 4 trained models bases on the accuracy and MSE

Project Scope

The broad scope of ‘Dairy Goods Sales Prediction’ project is given below:

• The given dataset has attributes based on which the status of the approval of the Dairy Sales will be predicted.

• It is a useful project as the Regression models can be used to quickly determine the dairy sales of large datasets.

• To enhance the keeping quality of products.

• To provide quality products at affordable prices to the consumers.

• The dataset given to us is a shortened form of the original dataset from Kaggle. So, the results might have some mismatch with the real-world applications. But that can be avoided if the models are trained accordingly.

Data Description

Source of the data: Kaggle.

- The given dataset is a shortened version of the original dataset in Kaggle.
- Data Description: The given train dataset has 150 rows and 15 columns.

Columns	Type	Description	Target Attribute
Product_Location	categorical	Location of product	No
Product_ID	Non-categorical	Id of product	No
Product_Name	categorical	Name of product	No
Brand	categorical	Brand of product	No
Quantity(liters/kg)	Non-categorical	Quantity of product	No
Price_per_Unit	Non-categorical	Price per unit of product	No
Total_Value	Non-categorical	Total value of product	No
Storage_Condition	categorical	Storage condition of product	No
Production_Date	Non-categorical	Production date of product	No
Expiration_Date	Non-categorical	Expiration date of product	No
Quantity_Sold	Non-categorical	Quantity sold	No
Price_per_Unit_Sold	Non-categorical	Price per unit sold	No
Approx_Total_Revenue(INR)	Non-categorical	Total revenue generated	Yes
Sales_Channel	categorical	Sales channels of product	No
Quantity_in_Stock_(liters/kg)	Non-categorical	Quantity in stock of product	No

Data description for numeric values:

	Product_ID	Quantity (liters/kg)	Price_per_Unit	Total_Value	Quantity_Sold	Price_per_Unit_Sold	Approx_Total Revenue(INR)	Quantity_in_Stock_(liters/kg)
count	150	150	150	150	147	147	150	150
mean	5.166667	497.2643	58.222733	28141.5851	268.639456	58.238844	15128.78673	225.413333
std	2.997575	289.3637	25.374899	21557.0132	230.703924	25.914929	15822.426	204.858805
min	1	8.47	11.58	722.0892	1	8.69	58.09	0
25%	3	253.1525	36.84	10256.7427	77	36.17	3946.06	48
50%	5	512.445	58.265	21891.7226	190	60.15	9408.025	195
75%	8	723.7075	82.18	41158.5913	415	81.615	21150.87	348
max	10	999.78	99.96	91387.7055	956	103.49	66719.24	929

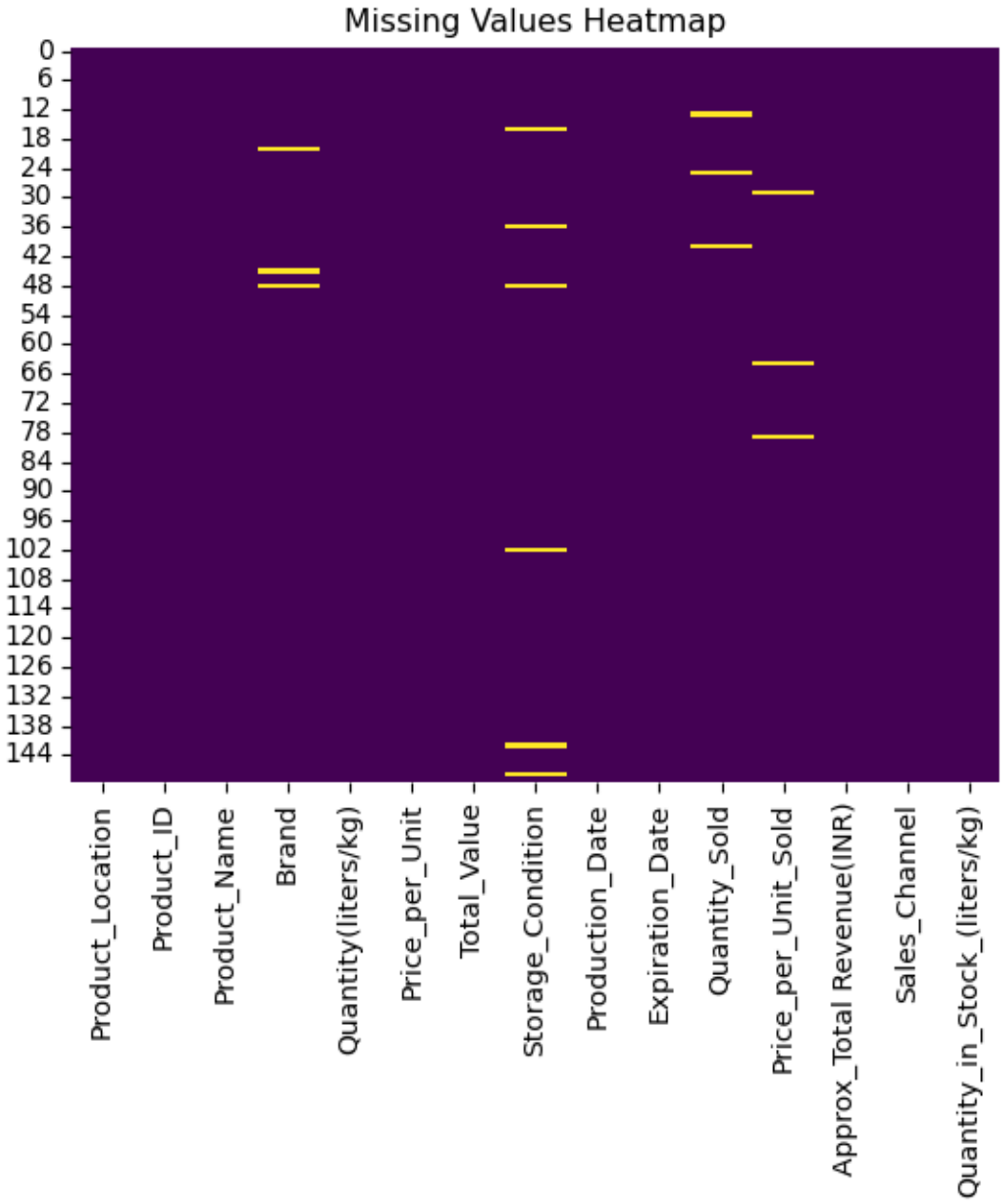
Data Pre-processing

Now we will pre-process the data. The methodology followed is given below:
We searched for null values in our dataset and formed the following table:

Column Name	Count of Null values
Product_Location	0
Product_ID	0
Product_Name	0
Brand	3
Quantity(liters/kg)	0
Price_per_Unit	0
Total_Value	0
Storage_Condition	6
Production_Date	0
Expiration_Date	0
Quantity_Sold	3
Price_per_Unit_Sold	3
Approx_Total Revenue(INR)	0
Sales_Channel	0
Quantity_in_Stock_(liters/kg)	0

• Checking for null values. o If null values are present, we will fill them or drop the row containing the null value based on the dataset.
• Checking for outliers. o If outliers are present, they will either be removed or replaced by following a suitable method depending on the dataset.
Or can be replaced using the module LabelEncoder from sklearn.

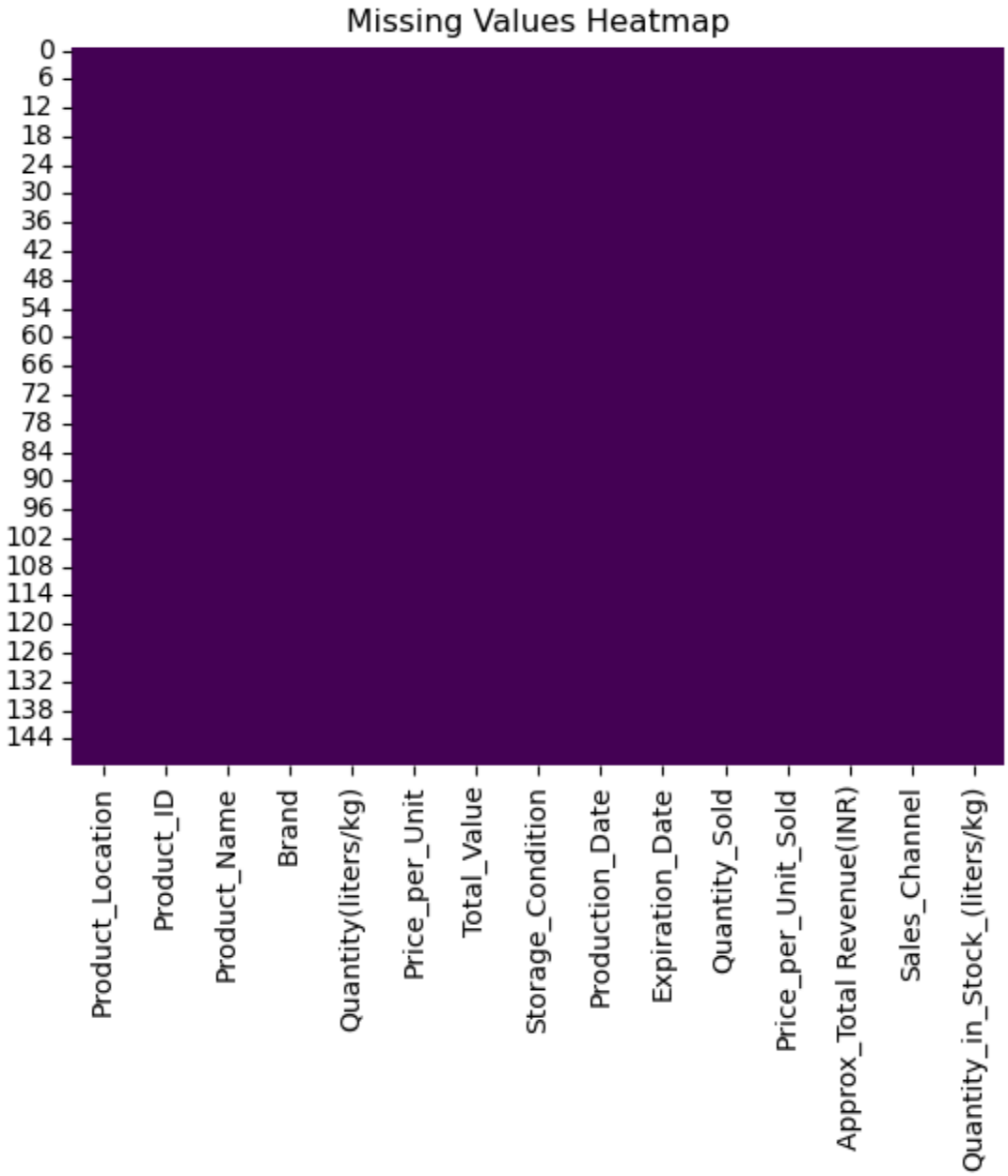
To visualise the null values, we made a heatmap plot using seaborn library function heatmap. The heatmap plot is given below:



To remove the null values, we had the following methodology:

• filling Null values in ‘Brand’,’storage_condition’ with mode as they are categorical values .
• filling Null values in ‘Quantity_Sold’,’ Price_per_Unit_Sold’ with median as they are non-categorical/Numeric values.

After removing the null values, the following heatmap was obtained:



Conversion of non-categorical values is done by sklearn.LabelEncoder in this project

```
# Preprocessing Task before model building
# Label Encoding

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Product_Location']=le.fit_transform(df['Product_Location'])
df['Product_Name']=le.fit_transform(df['Product_Name'])
df['Brand']=le.fit_transform(df['Brand'])
df['Storage_Condition']=le.fit_transform(df['Storage_Condition'])
df['Sales_Channel']=le.fit_transform(df['Sales_Channel'])
df.head()
```

	Product_Location	Product_Name	Brand	Quantity(liters/kg)	Price_per_Unit	Total_Value	Storage_Condition	Quantity_Sold	Price_per_Unit_Sold	Approx_Total Revenue(INR)
0	3	5	1	222.40	85.72	19064.1280	0	7.0	82.24	575.68
1	4	7	0	687.48	42.61	29293.5228	1	558.0	39.24	21895.92
2	3	9	1	503.48	36.50	18377.0200	1	256.0	33.81	8655.36
3	3	2	0	823.36	26.52	21835.5072	0	601.0	28.92	17380.92
4	1	1	1	147.77	83.85	12390.5145	1	145.0	83.07	12045.15

Now we have successfully handled Null values and converted non-numeric values to Numeric values. We didn't drop the rows with null values as we have a small dataset (only 150 entries). So, we are moving on to find if there are any outliers in our data and find the correlations of different attributes to our target i.e. 'Approx_Total Revenue(INR)' column in the dataset. The following table gives the correlation value of each attribute with our target attribute i.e. 'Approx_Total Revenue(INR):

Columns	**Correlation Value**
Quantity(liters/kg)	0.611
Price_per_Unit	0.262
Total_Value	0.738
Quantity_Sold	0.870
Price_per_Unit_Sold	0.261
Quantity_in_Stock_(liters/kg)	0.034
Product_Location	0.000
Product_Name	0.098
Brand	0.154
Storage_Condition	0.220
Sales_Channel	0.140

Model Building

Splitting data for training and testing purpose We split the given train dataset into two parts for training and testing purpose. The split ratio we used is 0.75 which indicates we used 75% data for training purpose and 25% data for testing purpose. We will be using the same split ratio for all the models trained.

Models used:

- Linear Regression
- Lasso
- Random Forest Regressor
- Ridge

LINEAR REGRESSION

Linear Regression is an algorithm that belongs to supervised Machine Learning. It tries to apply relations that will predict the outcome of an event based on the independent variable data points. The relation is usually a straight line that best fits the different data points as close as possible.
Linear regression can be expressed mathematically as:

$$y= \beta 0+ \beta 1x+ \varepsilon$$

Here,

- Y= Dependent Variable
- X= Independent Variable
- β 0= intercept of the line
- β1 = Linear regression coefficient (slope of the line)
- ε = random error

Types of Linear Regression

Linear Regression can be broadly classified into two types of algorithms:

1. Simple Linear Regression

A simple straight-line equation involving slope (dy/dx) and intercept (an integer/continuous value) is utilized in simple Linear Regression. Here a simple form is:

y=mx+c where y denotes the output x is the independent variable, and c is the intercept when x=0. With this equation, the algorithm trains the model of machine learning and gives the most accurate output

2. Multiple Linear Regression

When a number of independent variables more than one, the governing linear equation applicable to regression takes a different form like:

y= c+m1x1+m2x2… mnxn where represents the coefficient responsible for impact of different independent variables x1, x2 etc. This machine learning algorithm, when applied, finds the values of coefficients m1, m2, etc., and gives the best fitting line.

3. Non-Linear Regression

When the best fitting line is not a straight line but a curve, it is referred to as Non-Linear Regression.

LASSO

The full form of LASSO is the Least Absolute Shrinkage and Selection Operation. As the name suggests, LASSO uses the “shrinkage” technique in which coefficients are determined, which get shrunk towards the central point as the mean.

The LASSO regression in regularization is based on simple models that posses fewer parameters. We get a better interpretation of the models due to the shrinkage process. The shrinkage process also enables the identification of variables strongly associated with variables corresponding to the target.

Lasso regression is also called Penalized regression method. This method is usually used in machine learning for the selection of the subset of variables. It provides greater prediction accuracy as compared to other regression models. Lasso Regularization helps to increase model interpretation.

The less important features of a dataset are penalized by the lasso regression. The coefficients of this dataset are made zero leading to their elimination. The dataset with high dimensions and correlation is well suited for lasso regression.

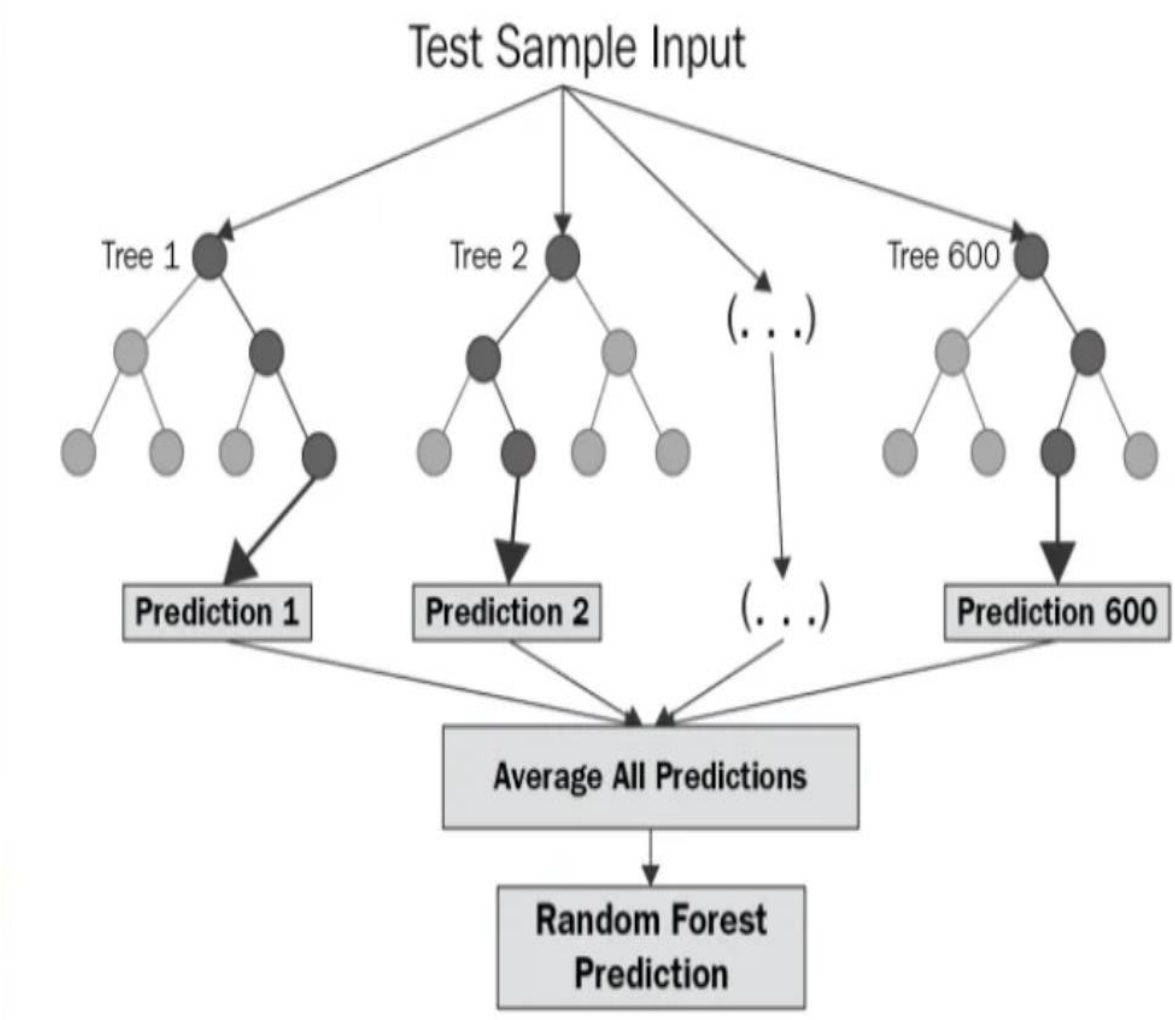
Lasso Regression Formula:

D= Residual Sum of Squares or Least Squares Lambda * Aggregate of absolute values of coefficients

Lambda denotes the amount of shrinkage in the lasso regression equation.

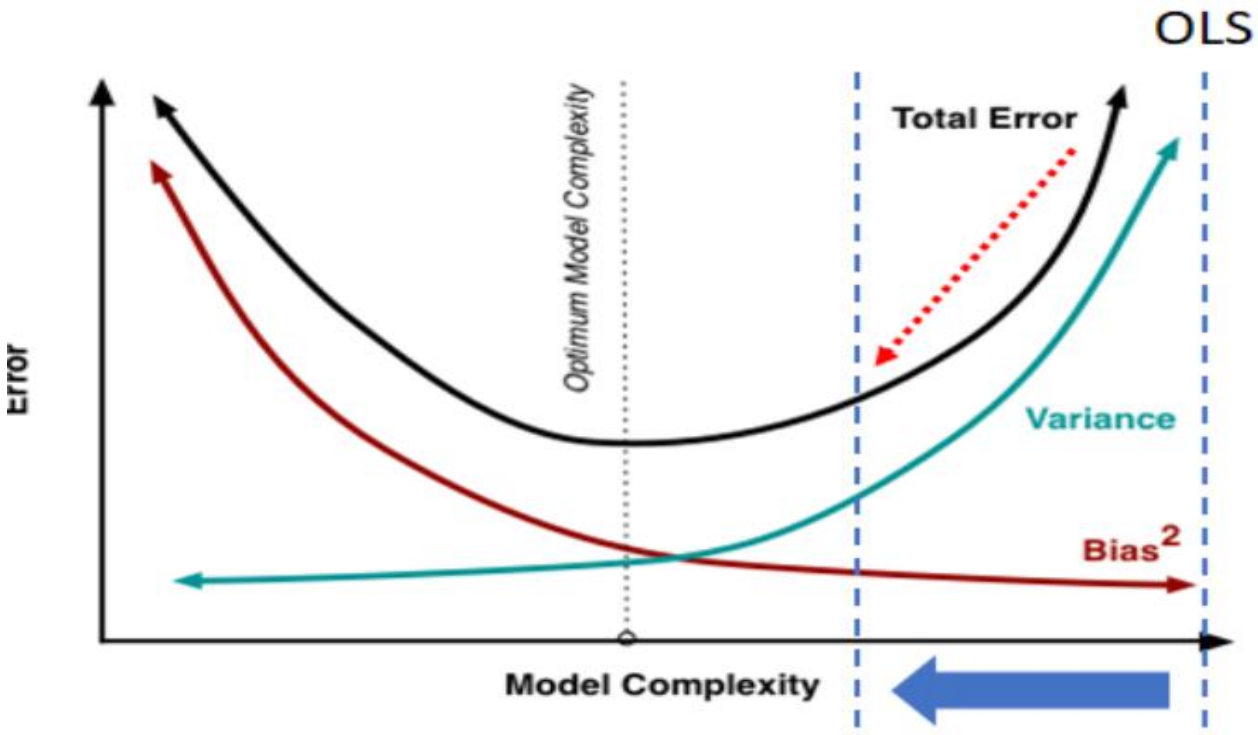
RANDOM FOREST REGRESSOR

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.



RIDGE

Ridge regression is a specialized technique used to analyze multiple regression data that is multi collinear in nature. It is a fundamental regularization technique, but it is not used very widely because of the complex science behind it. However, it is fairly easy to explore the science behind ridge regression in r if you have an overall idea of the concept of multiple regression. Regression stays the same, but in regularization, the way the model coefficients are determined is different.



- *Ridge regression* penalize the size of the regression coefficients based on their l^2 norm:

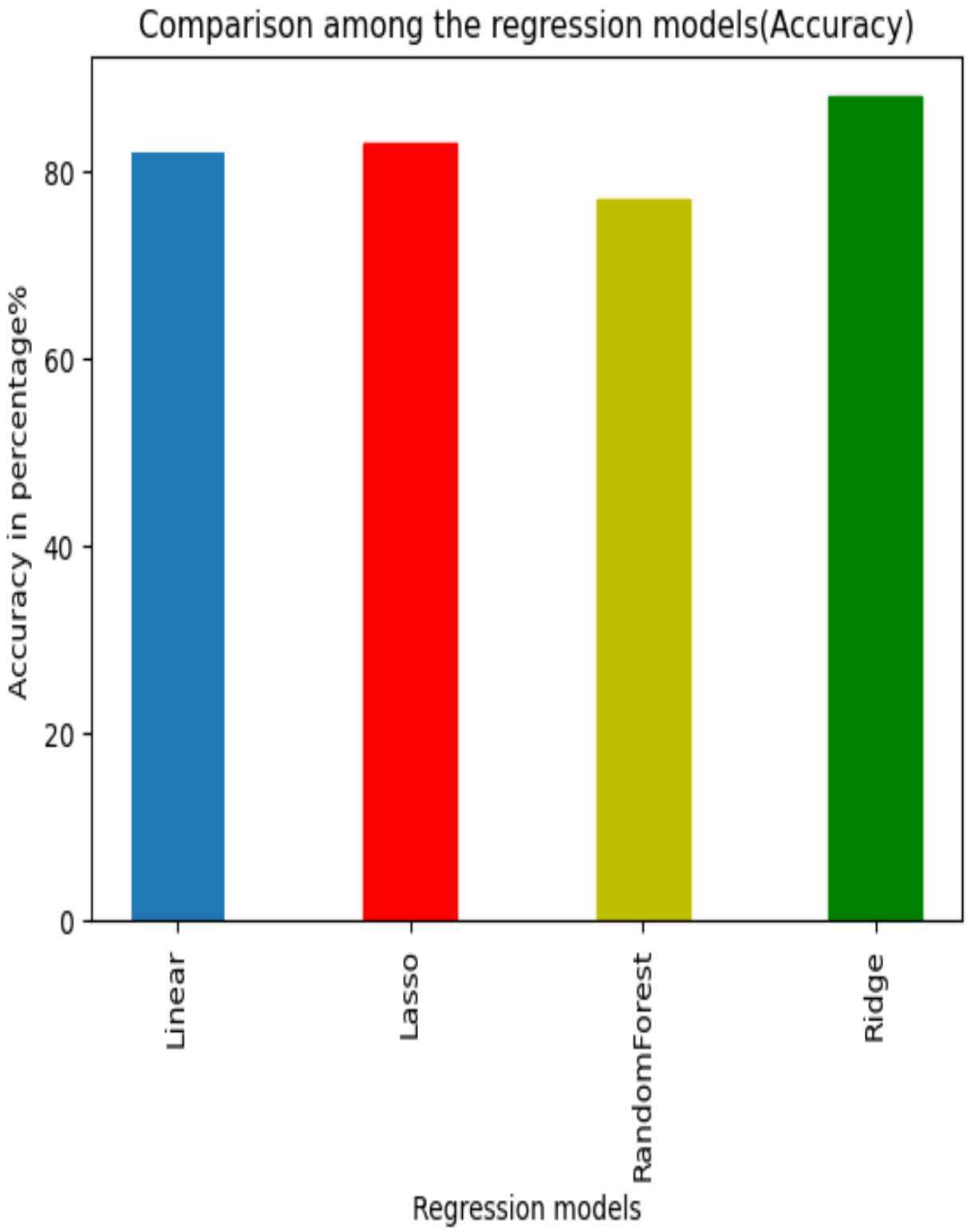
$$argmin_{\beta} \sum_{i} (y_i - \boldsymbol{\beta}' \boldsymbol{x}_i)^2 + \lambda \sum_{k=1}^{K} \beta_k^2$$

Comparison of the Models trained We trained 4 models using the 4 algorithms viz.

1. Linear Regression
2. Lasso
3. Random Forest Regressor
4. Ridge

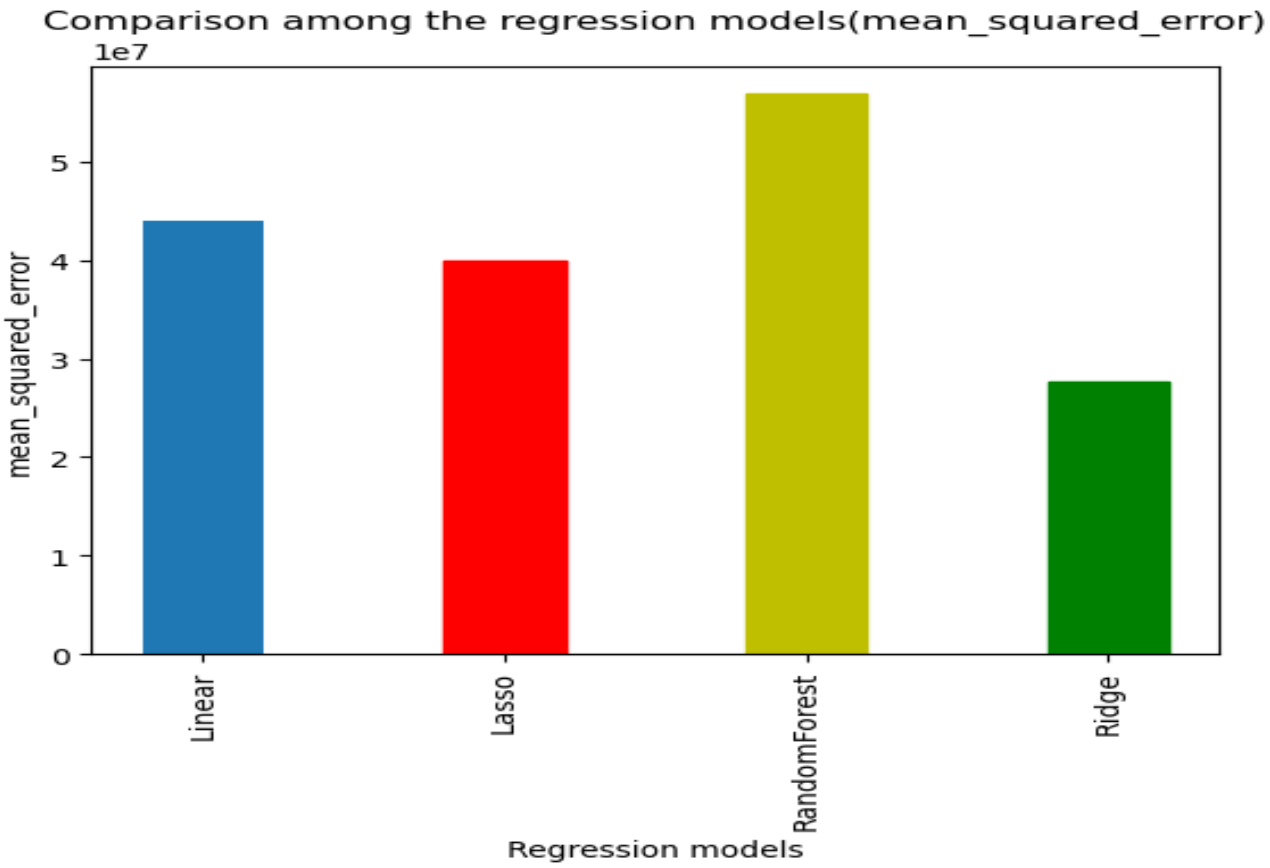
The 4 models had different accuracy. The comparison of the accuracies of the models are given below:

Model	Accuracy%
Linear Regression	82%
Lasso	83%
Random Forest	76%
Ridge	88%


Comparison among the regression models(Accuracy)
Accuracy in percentage%
0
20
40
60
80
Linear
Lasso
RandomForest
Ridge
Regression models

The comparison of the mean squared error of the models are given below:

Model	Mean Squared Error
Linear Regression	43966520.5580716
Lasso	39862068.07829146
Random Forest	56832382.3657430
Ridge	27670732.3551490



- So we select Ridge model to predict the dataset as it has the highest accuracy and low MSE .

Code

Dairy Goods Sales Prediction

```
# importing the libraries

import pandas as pd #for handling data
import matplotlib.pyplot as plt #for plotting graphs
import numpy as np # for numpy function and ndarrays
import seaborn as sns # for plotting graphs
```

```
# load the dataset

data=pd.read_csv('dairy_dataset.csv')
df=data.copy()
df.describe(include='all')
```

	Product_Location	Product_ID	Product_Name	Brand	Quantity(liters/kg)	Price_per_Unit	Total_Value	Storage_Condition	Production_Date
count	150	150.000000	150	147	150.000000	150.000000	150.000000	144	150
unique	5	NaN	10	2	NaN	NaN	NaN	2	147
top	Haryana	NaN	Milk	Amul	NaN	NaN	NaN	Refrigerated	06-08-2022
freq	34	NaN	23	84	NaN	NaN	NaN	108	2
mean	NaN	5.166667	NaN	NaN	497.264333	58.222733	28141.585181	NaN	NaN
std	NaN	2.997575	NaN	NaN	289.363745	25.374899	21557.013218	NaN	NaN
min	NaN	1.000000	NaN	NaN	8.470000	11.580000	722.089200	NaN	NaN
25%	NaN	3.000000	NaN	NaN	253.152500	36.840000	10256.742750	NaN	NaN
50%	NaN	5.000000	NaN	NaN	512.445000	58.265000	21891.722650	NaN	NaN
75%	NaN	8.000000	NaN	NaN	723.707500	82.180000	41158.591300	NaN	NaN
max	NaN	10.000000	NaN	NaN	999.780000	99.960000	91387.705500	NaN	NaN

Total_Value	Storage_Condition	Production_Date	Expiration_Date	Quantity_Sold	Price_per_Unit_Sold	Approx_Total Revenue(INR)	Sales_Channel	Quantity_in_Stock_(liters/kg)
150.000000	144	150	150	147.000000	147.000000	150.000000	150	150.000000
NaN	2	147	144	NaN	NaN	NaN	3	NaN
NaN	Refrigerated	06-08-2022	31-10-2022	NaN	NaN	NaN	Online	NaN
NaN	108	2	2	NaN	NaN	NaN	54	NaN
8141.585181	NaN	NaN	NaN	268.639456	58.238844	15128.786733	NaN	225.413333
1557.013218	NaN	NaN	NaN	230.703924	25.914929	15822.426521	NaN	204.858805
722.089200	NaN	NaN	NaN	1.000000	8.690000	58.090000	NaN	0.000000
0256.742750	NaN	NaN	NaN	77.000000	36.170000	3946.060000	NaN	48.000000
1891.722650	NaN	NaN	NaN	190.000000	60.150000	9408.025000	NaN	195.000000
1158.591300	NaN	NaN	NaN	415.000000	81.615000	21150.870000	NaN	348.000000
1387.705500	NaN	NaN	NaN	956.000000	103.490000	66719.240000	NaN	929.000000

```
data.head()
```

	Product_Location	Product_ID	Product_Name	Brand	Quantity(liters/kg)	Price_per_Unit	Total_Value	Storage_Condition	Production_Date	Expiration_Date	Q
0	Tamil Nadu	5	Ice Cream	Mother Dairy	222.40	85.72	19064.1280	Frozen	27-12-2021	21-01-2022	
1	West Bengal	1	Milk	Amul	687.48	42.61	29293.5228	Refrigerated	03-10-2021	25-10-2021	
2	Tamil Nadu	4	Yogurt	Mother Dairy	503.48	36.50	18377.0200	Refrigerated	14-01-2022	13-02-2022	
3	Tamil Nadu	3	Cheese	Amul	823.36	26.52	21835.5072	Frozen	15-05-2019	26-07-2019	
4	Haryana	8	Buttermilk	Mother Dairy	147.77	83.85	12390.5145	Refrigerated	17-10-2020	28-10-2020	

```
data.head()
```

Total_Value	Storage_Condition	Production_Date	Expiration_Date	Quantity_Sold	Price_per_Unit_Sold	Approx_Total Revenue(INR)	Sales_Channel	Quantity_in_Stock_(liters/kg)
19064.1280	Frozen	27-12-2021	21-01-2022	7.0	82.24	575.68	Wholesale	215
29293.5228	Refrigerated	03-10-2021	25-10-2021	558.0	39.24	21895.92	Wholesale	129
18377.0200	Refrigerated	14-01-2022	13-02-2022	256.0	33.81	8655.36	Online	247
21835.5072	Frozen	15-05-2019	26-07-2019	601.0	28.92	17380.92	Online	222
12390.5145	Refrigerated	17-10-2020	28-10-2020	145.0	83.07	12045.15	Retail	2

```
# Checking the dataset for any null values

print("null values count:\n",df.isnull().sum())
```

```
null values count:
 Product_Location              0
Product_ID                     0
Product_Name                   0
Brand                          3
Quantity(liters/kg)            0
Price_per_Unit                 0
Total_Value                    0
Storage_Condition              6
Production_Date                0
Expiration_Date                0
Quantity_Sold                  3
Price_per_Unit_Sold            3
Approx_Total Revenue(INR)      0
Sales_Channel                  0
Quantity_in_Stock_(liters/kg)  0
dtype: int64
```

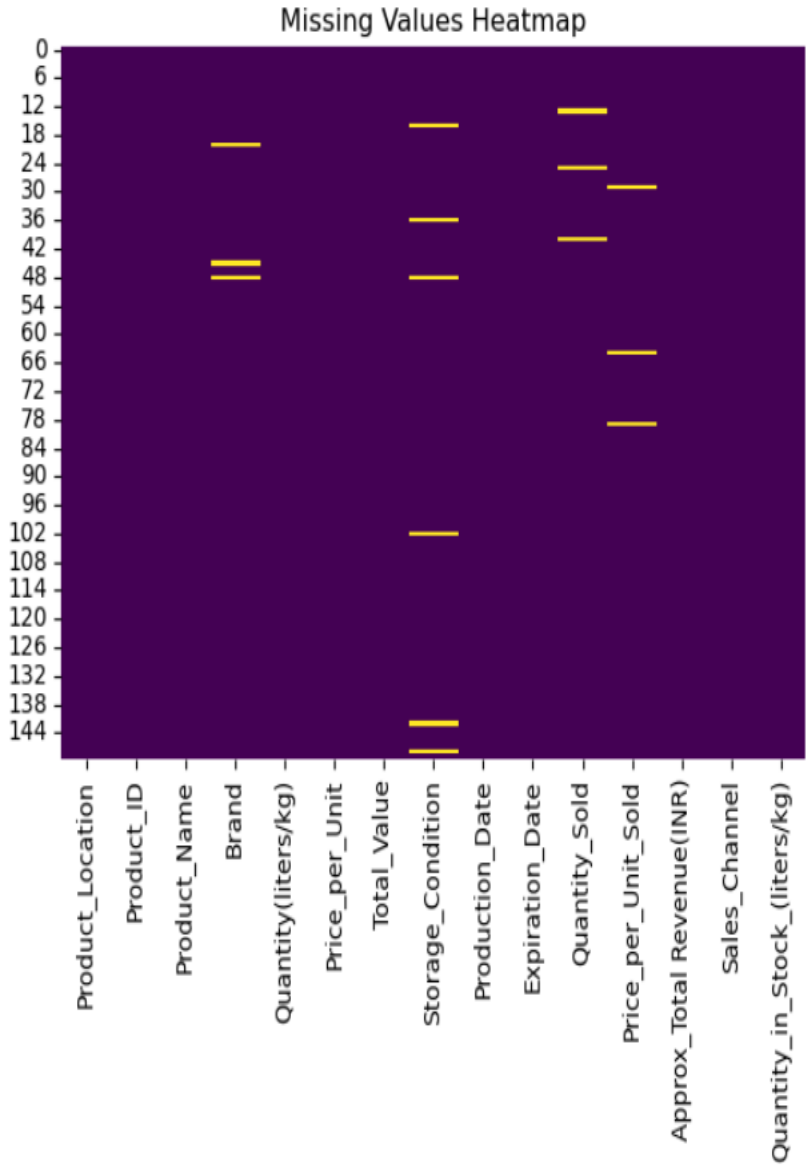
```
# Shape of the dataset

df.shape
```

```
(150, 15)
```

```
# Info of the dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Product_Location            150 non-null    object 
 1   Product_ID                  150 non-null    int64  
 2   Product_Name                150 non-null    object 
 3   Brand                       147 non-null    object 
 4   Quantity(liters/kg)         150 non-null    float64
 5   Price_per_Unit              150 non-null    float64
 6   Total_Value                 150 non-null    float64
 7   Storage_Condition           144 non-null    object 
 8   Production_Date             150 non-null    object 
 9   Expiration_Date             150 non-null    object 
 10  Quantity_Sold               147 non-null    float64
 11  Price_per_Unit_Sold         147 non-null    float64
 12  Approx_Total Revenue(INR)   150 non-null    float64
 13  Sales_Channel               150 non-null    object 
 14  Quantity_in_Stock_(liters/kg)  150 non-null    int64  
dtypes: float64(6), int64(2), object(7)
memory usage: 17.7+ KB
```

```
# Visualizing null values using heatmap

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Values Heatmap')
plt.show()
```



```
df.describe()
```

	Product_ID	Quantity(liters/kg)	Price_per_Unit	Total_Value	Quantity_Sold	Price_per_Unit_Sold	Approx_Total Revenue(INR)	Quantity_in_Stock_(liters/kg)
count	150.000000	150.000000	150.000000	150.000000	147.000000	147.000000	150.000000	150.000000
mean	5.166667	497.264333	58.222733	28141.585181	268.639456	58.238844	15128.786733	225.413333
std	2.997575	289.363745	25.374899	21557.013218	230.703924	25.914929	15822.426521	204.858805
min	1.000000	8.470000	11.580000	722.089200	1.000000	8.690000	58.090000	0.000000
25%	3.000000	253.152500	36.840000	10256.742750	77.000000	36.170000	3946.060000	48.000000
50%	5.000000	512.445000	58.265000	21891.722650	190.000000	60.150000	9408.025000	195.000000
75%	8.000000	723.707500	82.180000	41158.591300	415.000000	81.615000	21150.870000	348.000000
max	10.000000	999.780000	99.960000	91387.705500	956.000000	103.490000	66719.240000	929.000000

```
df['Quantity_Sold'].describe()
```

```
count    147.000000
mean     268.639456
std      230.703924
min        1.000000
25%       77.000000
50%      190.000000
75%      415.000000
max      956.000000
Name: Quantity_Sold, dtype: float64
```

```
df['Price_per_Unit_Sold'].describe()
```

```
count    147.000000
mean      58.238844
std       25.914929
min        8.690000
25%       36.170000
50%       60.150000
75%       81.615000
max      103.490000
Name: Price_per_Unit_Sold, dtype: float64
```

```
# Filling the numeric null values in dataset with mean imputation

df['Quantity_Sold'].fillna(df['Quantity_Sold'].mean(),inplace=True)
df['Price_per_Unit_Sold'].fillna(df['Price_per_Unit_Sold'].mean(),inplace=True)
df.isnull().sum()
```

```
Product_Location              0
Product_ID                    0
Product_Name                  0
Brand                         3
Quantity(liters/kg)           0
Price_per_Unit                0
Total_Value                   0
Storage_Condition             6
Production_Date               0
Expiration_Date               0
Quantity_Sold                 0
Price_per_Unit_Sold           0
Approx_Total Revenue(INR)     0
Sales_Channel                 0
Quantity_in_Stock_(liters/kg)    0
dtype: int64
```

```
df['Storage_Condition'].describe()
```

```
count             144
unique              2
top       Refrigerated
freq              108
Name: Storage_Condition, dtype: object
```

```
df['Brand'].describe()
```

```
count      147
unique       2
top       Amul
freq        84
Name: Brand, dtype: object
```

```
df['Brand'].mode()
```

```
0    Amul
Name: Brand, dtype: object
```

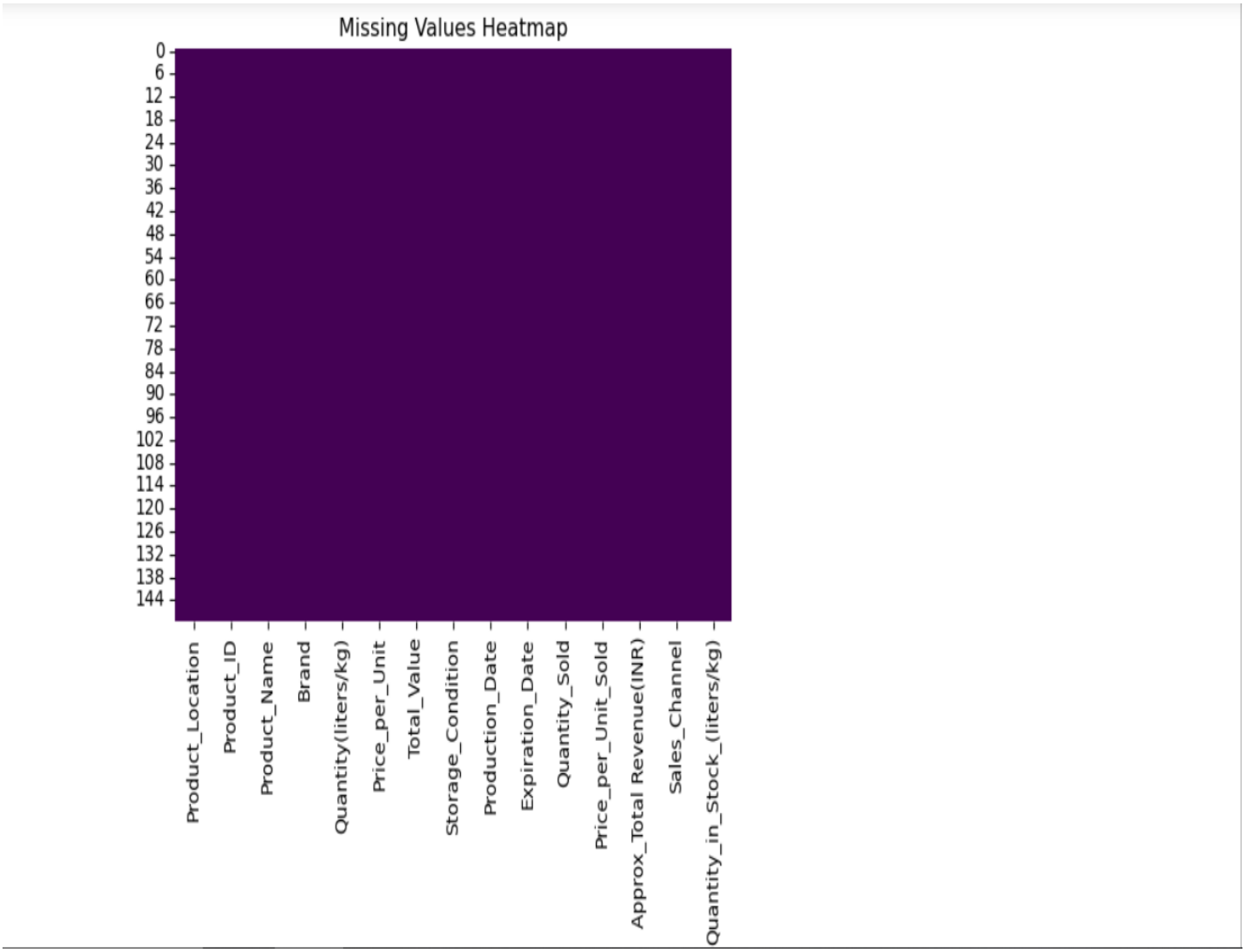
```
df['Storage_Condition'].mode()
```

```
0    Refrigerated
Name: Storage_Condition, dtype: object
```

```
# Filling the categorical null values in dataset with mode imputation

df['Brand'].fillna(df['Brand'].mode()[0],inplace=True)
df['Storage_Condition'].fillna(df['Storage_Condition'].mode()[0],inplace=True)
df.isnull().sum()
```

```
Product_Location              0
Product_ID                    0
Product_Name                  0
Brand                         0
Quantity(liters/kg)           0
Price_per_Unit                0
Total_Value                   0
Storage_Condition             0
Production_Date               0
Expiration_Date               0
Quantity_Sold                 0
Price_per_Unit_Sold           0
Approx_Total Revenue(INR)     0
Sales_Channel                 0
Quantity_in_Stock_(liters/kg) 0
dtype: int64
```

```
# Checking all null values are filled using heatmap

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Values Heatmap')
plt.show()
```




```
# Selecting based on general requirements

df.drop(['Product_ID'],axis=1,inplace=True)
df.drop(['Production_Date'],axis=1,inplace=True)
df.drop(['Expiration_Date'],axis=1,inplace=True)
```

```
df.head()
```

	Product_Location	Product_Name	Brand	Quantity(liters/kg)	Price_per_Unit	Total_Value	Storage_Condition	Quantity_Sold	Price_per_Unit_Sold
0	Tamil Nadu	Ice Cream	Mother Dairy	222.40	85.72	19064.1280	Frozen	7.0	82.24
1	West Bengal	Milk	Amul	687.48	42.61	29293.5228	Refrigerated	558.0	39.24
2	Tamil Nadu	Yogurt	Mother Dairy	503.48	36.50	18377.0200	Refrigerated	256.0	33.81
3	Tamil Nadu	Cheese	Amul	823.36	26.52	21835.5072	Frozen	601.0	28.92
4	Haryana	Buttermilk	Mother Dairy	147.77	83.85	12390.5145	Refrigerated	145.0	83.07

```
df.head()
```

Quantity(liters/kg)	Price_per_Unit	Total_Value	Storage_Condition	Quantity_Sold	Price_per_Unit_Sold	Approx_Total Revenue(INR)	Sales_Channel	Quantity_in_Stock_(liters/kg)
222.40	85.72	19064.1280	Frozen	7.0	82.24	575.68	Wholesale	215
687.48	42.61	29293.5228	Refrigerated	558.0	39.24	21895.92	Wholesale	129
503.48	36.50	18377.0200	Refrigerated	256.0	33.81	8655.36	Online	247
823.36	26.52	21835.5072	Frozen	601.0	28.92	17380.92	Online	222
147.77	83.85	12390.5145	Refrigerated	145.0	83.07	12045.15	Retail	2

```
# EDA(Exploratory Data Analysis) using Pandas Profiling

from pandas_profiling import ProfileReport
profile=ProfileReport(df,title="Pandas Profiling Report")
```

```
profile
```

Summarize dataset: 100% 70/70 [00:08<00:00, 6.74it/s, Completed]

Generate report structure: 100% 1/1 [00:04<00:00, 4.63s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.44s/it]

Pandas Profiling Report | Overview | Variables | Interactions | Correlations | Missing values | Sample

Overview

Overview | Alerts 9 | Reproduction

Dataset statistics

Number of variables	12
Number of observations	150
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	14.2 KiB
Average record size in memory	96.9 B

Variable types

Categorical	5
Numeric	7

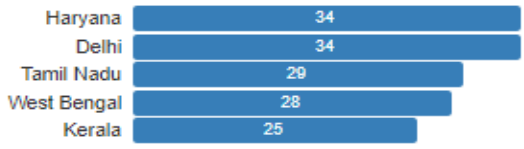
Variables

Select Columns

Product_Location

Categorical

Distinct	5
Distinct (%)	3.3%
Missing	0
Missing (%)	0.0%
Memory size	1.3 KiB

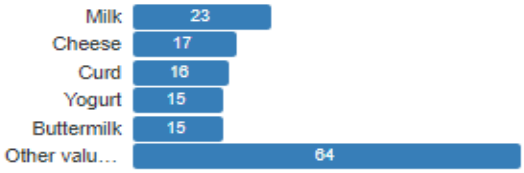


More details

Product_Name

Categorical

Distinct	10
Distinct (%)	6.7%
Missing	0
Missing (%)	0.0%
Memory size	1.3 KiB

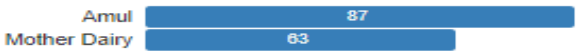


More details

Brand

Categorical

Distinct	2
Distinct (%)	1.3%
Missing	0
Missing (%)	0.0%
Memory size	1.3 KiB

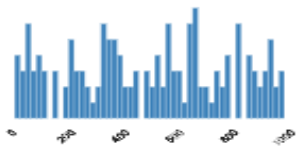


More details

Quantity(liters/kg)

Real number (ℝ)

Distinct	149	Minimum	8.47
Distinct (%)	99.3%	Maximum	999.78
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	497.26433	Memory size	1.3 KiB

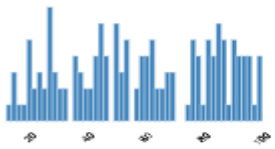


More details

Price_per_Unit

Real number (ℝ)

Distinct	149	Minimum	11.58
Distinct (%)	99.3%	Maximum	99.96
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	58.222733	Memory size	1.3 KiB

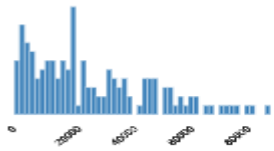


More details

Total_Value

Real number (ℝ)

Distinct	149	Minimum	722.0892
Distinct (%)	99.3%	Maximum	91387.705
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	28141.585	Memory size	1.3 KiB



Storage_Condition

Categorical

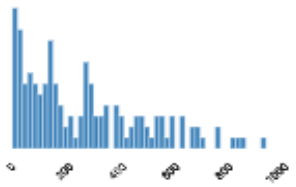
Distinct	2
Distinct (%)	1.3%
Missing	0
Missing (%)	0.0%
Memory size	1.3 KiB

More details

Quantity_Sold

Real number (ℝ)

Distinct	126	Minimum	1
Distinct (%)	84.0%	Maximum	956
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	268.63946	Memory size	1.3 KiB

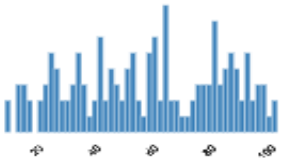


More details

Price_per_Unit_Sold

Real number (ℝ)

Distinct	147	Minimum	8.69
Distinct (%)	98.0%	Maximum	103.49
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	58.238844	Memory size	1.3 KiB

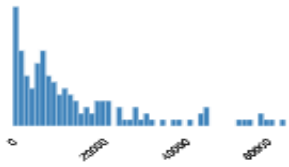


More details

Approx_Total Revenue(INR)

Real number (ℝ)

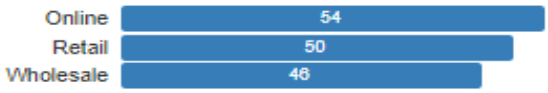
Distinct	149	Minimum	58.09
Distinct (%)	99.3%	Maximum	66719.24
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	15128.787	Memory size	1.3 KiB



Sales_Channel

Categorical

Distinct	3
Distinct (%)	2.0%
Missing	0
Missing (%)	0.0%
Memory size	1.3 KiB

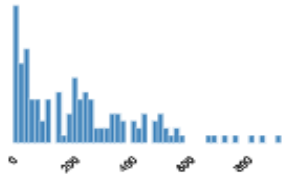


More details

Quantity_in_Stock_(liters/kg)

Real number (ℝ)

Distinct	130	Minimum	0
Distinct (%)	86.7%	Maximum	929
Missing	0	Zeros	1
Missing (%)	0.0%	Zeros (%)	0.7%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	225.41333	Memory size	1.3 KiB



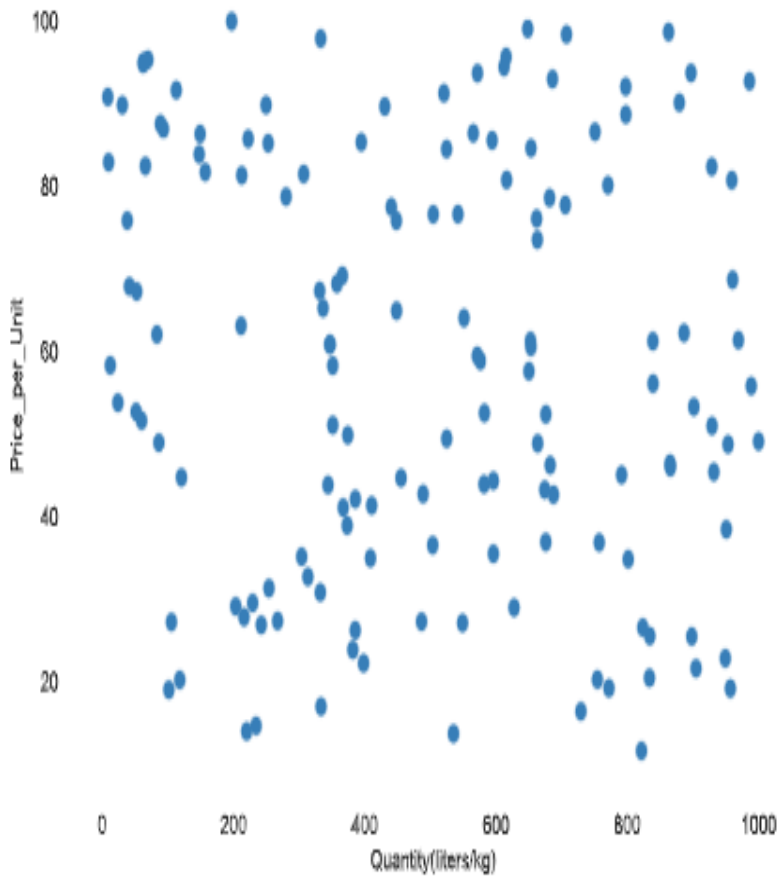
More details

Quantity(liters/kg) Price_per_Unit Total_Value Quantity_Sold Price_per_Unit_Sold

Approx_Total Revenue(INR) Quantity_in_Stock_(liters/kg)

Quantity_in_Stock_(liters/kg) Quantity(liters/kg) Price_per_Unit Total_Value

Quantity_Sold Price_per_Unit_Sold Approx_Total Revenue(INR)

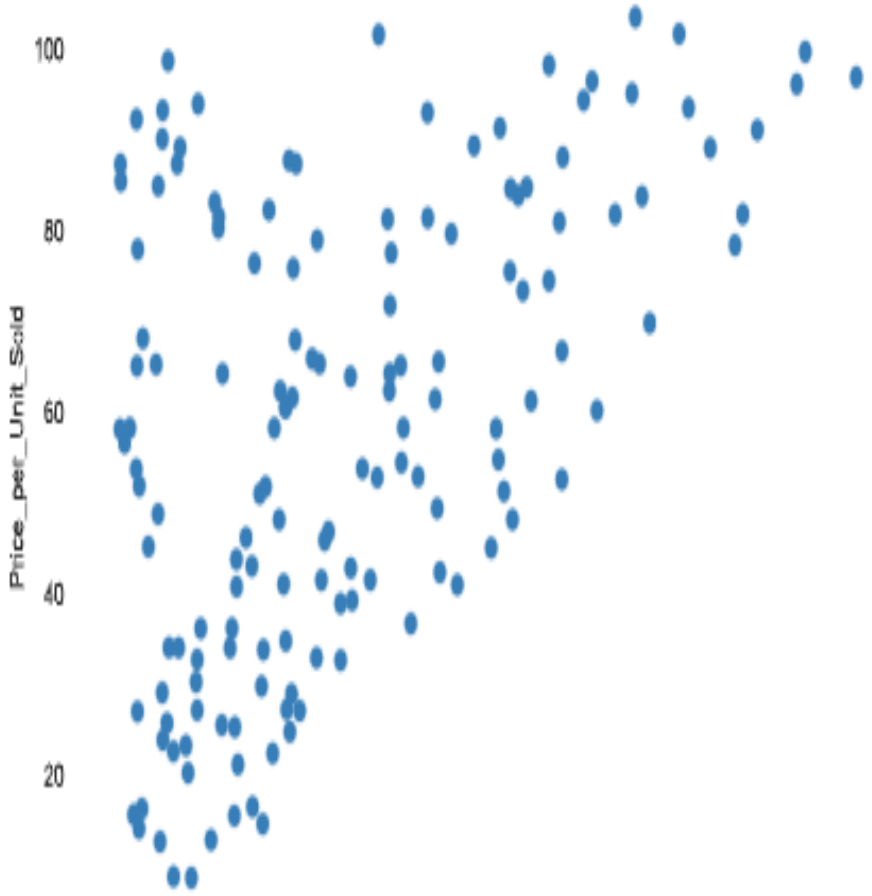


Quantity(liters/kg) Price_per_Unit Total_Value Quantity_Sold Price_per_Unit_Sold

Approx_Total Revenue(INR) Quantity_in_Stock_(liters/kg)

Quantity_in_Stock_(liters/kg) Quantity(liters/kg) Price_per_Unit Total_Value

Quantity_Sold Price_per_Unit_Sold Approx_Total Revenue(INR)

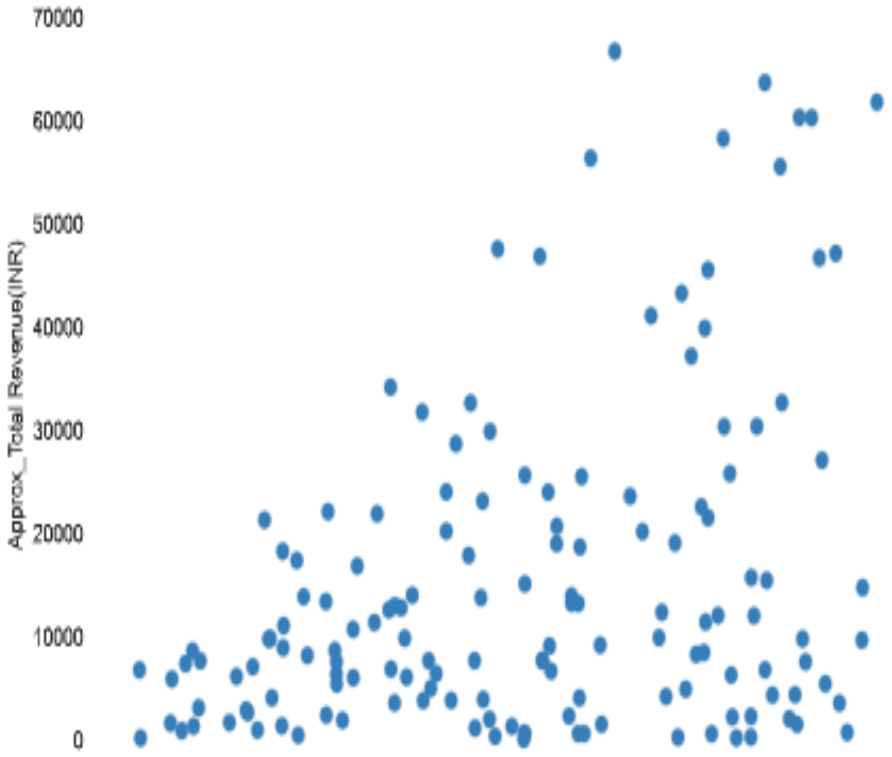


Quantity(liters/kg) Price_per_Unit Total_Value Quantity_Sold Price_per_Unit_Sold

Approx_Total Revenue(INR) Quantity_in_Stock_(liters/kg)

Quantity_in_Stock_(liters/kg) Quantity(liters/kg) Price_per_Unit Total_Value

Quantity_Sold Price_per_Unit_Sold Approx_Total Revenue(INR)

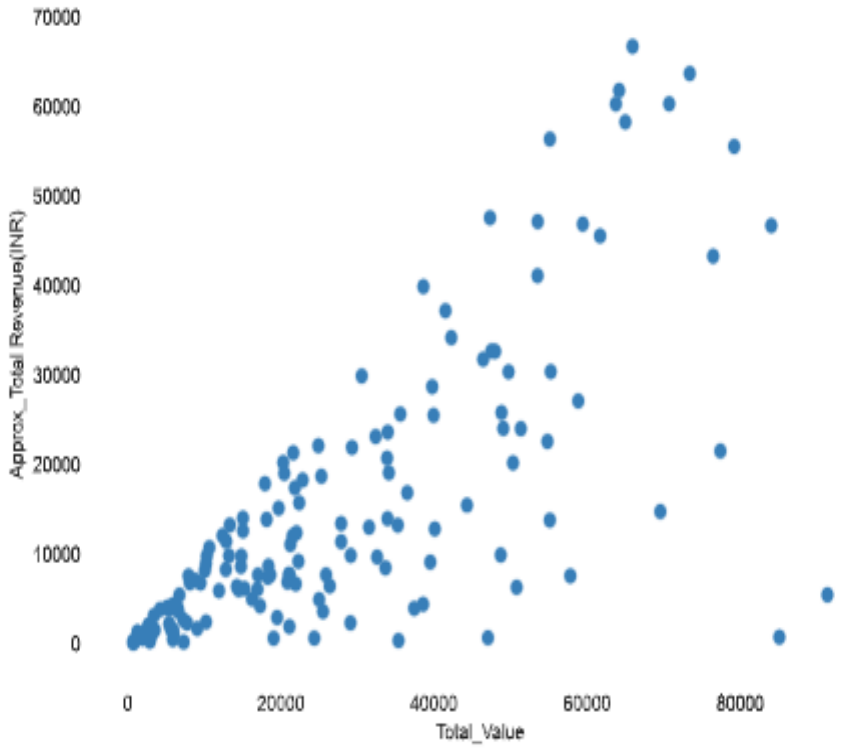


Quantity(liters/kg) Price_per_Unit Total_Value Quantity_Sold Price_per_Unit_Sold

Approx_Total Revenue(INR) Quantity_in_Stock_(liters/kg)

Quantity_in_Stock_(liters/kg) Quantity(liters/kg) Price_per_Unit Total_Value

Quantity_Sold Price_per_Unit_Sold Approx_Total Revenue(INR)

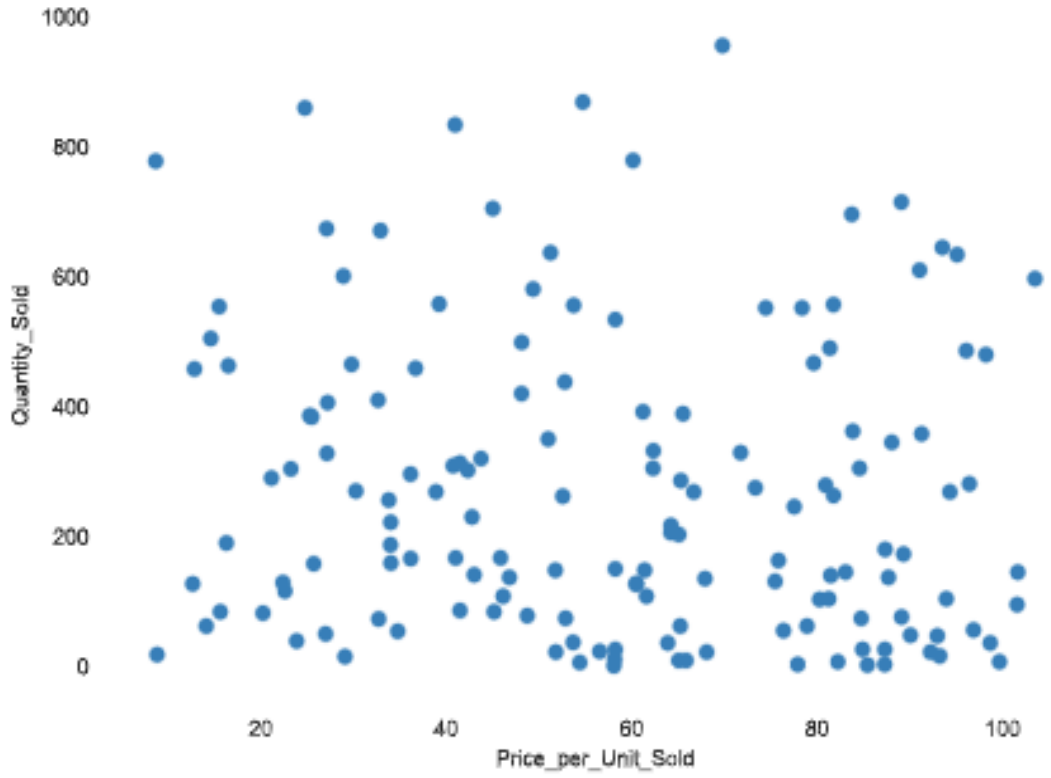


Quantity(liters/kg) Price_per_Unit Total_Value Quantity_Sold Price_per_Unit_Sold

Approx_Total Revenue(INR) Quantity_in_Stock_(liters/kg)

Quantity_in_Stock_(liters/kg) Quantity(liters/kg) Price_per_Unit Total_Value

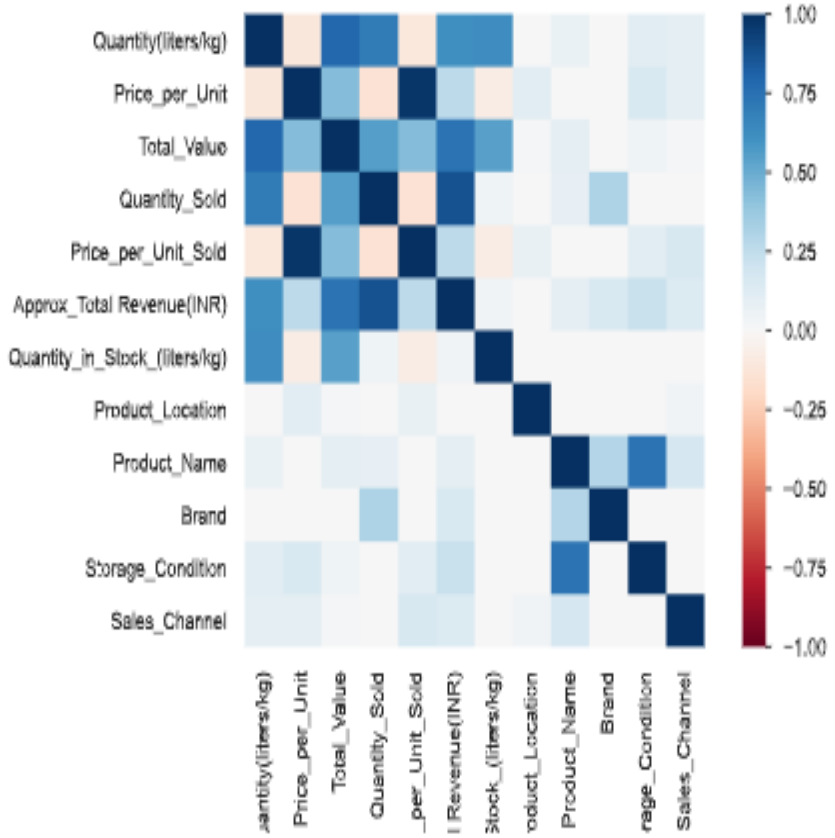
Quantity_Sold Price_per_Unit_Sold Approx_Total Revenue(INR)



Correlations

Auto

Heatmap | Table



Correlations

Auto

Heatmap | Table

	Quantity(liters/kg)	Price_per_Unit	Total_Value	Quant
Quantity(liters/kg)	1.000	-0.114	0.785	0.703
Price_per_Unit	-0.114	1.000	0.430	-0.141
Total_Value	0.785	0.430	1.000	0.548
Quantity_Sold	0.703	-0.141	0.548	1.000
Price_per_Unit_Sold	-0.105	0.973	0.431	-0.141
Approx_Total Revenue(INR)	0.611	0.262	0.738	0.870
Quantity_in_Stock_(liters/kg)	0.624	-0.078	0.546	0.040
Product_Location	0.000	0.106	0.017	0.000
Product_Name	0.064	0.000	0.099	0.079

```
# Preprocessing Task before model building
# Label Encoding

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Product_Location']=le.fit_transform(df['Product_Location'])
df['Product_Name']=le.fit_transform(df['Product_Name'])
df['Brand']=le.fit_transform(df['Brand'])
df['Storage_Condition']=le.fit_transform(df['Storage_Condition'])
df['Sales_Channel']=le.fit_transform(df['Sales_Channel'])
df.head()
```

Product_Location	Product_Name	Brand	Quantity(liters/kg)	Price_per_Unit	Total_Value	Storage_Condition	Quantity_Sold	Price_per_Unit_Sold	Approx_Total Revenue(INR)
3	5	1	222.40	85.72	19064.1280	0	7.0	82.24	575.68
4	7	0	687.48	42.61	29293.5228	1	558.0	39.24	21895.92
3	9	1	503.48	36.50	18377.0200	1	256.0	33.81	8655.36
3	2	0	823.36	26.52	21835.5072	0	601.0	28.92	17380.92
1	1	1	147.77	83.85	12390.5145	1	145.0	83.07	12045.15

```
# Splitting our data into train and test

from sklearn.model_selection import train_test_split

X=df.drop('Approx_Total Revenue(INR)',axis=1)
y=df['Approx_Total Revenue(INR)']
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=101, test_size=0.2)
```

```
X_train
```

	Product_Location	Product_Name	Brand	Quantity(liters/kg)	Price_per_Unit	Total_Value	Storage_Condition	Quantity_Sold	Price_per_Unit_Sold	Sales_Char
104	2	7	1	105.47	27.19	2867.7293	1	50.0	27.00	
89	4	4	0	729.21	16.37	11937.1677	1	458.0	12.86	
116	1	1	0	682.38	46.20	31525.9560	1	313.0	41.50	
82	3	3	0	336.47	65.23	21947.9381	1	108.0	61.59	
112	1	2	0	448.41	64.90	29101.8090	1	36.0	63.90	
...	...	...	...	...	...	...	...	...	...	
63	1	4	0	838.85	61.21	51346.0085	1	392.0	61.22	
70	1	4	1	455.19	44.64	20319.6816	1	420.0	48.13	
81	4	0	0	705.44	77.71	54819.7424	1	279.0	80.91	
11	1	6	0	653.04	84.60	55247.1840	1	345.0	88.05	
95	4	7	0	280.10	78.72	22049.4720	1	163.0	75.83	

120 rows × 11 columns

```
X_test
```

	Product_Location	Product_Name	Brand	Quantity(liters/kg)	Price_per_Unit	Total_Value	Storage_Condition	Quantity_Sold	Price_per_Unit_Sold	Sales_Cha
33	3	4	0	581.50	43.88	25516.2200	1	86.000000	41.490000	
16	0	4	1	663.34	48.83	32390.8922	1	438.000000	52.790000	
43	1	5	0	675.77	52.35	35376.5595	0	6.000000	54.410000	
129	2	8	0	374.12	49.82	18638.6584	1	148.000000	51.770000	
50	3	1	1	410.48	41.31	16956.9288	1	141.000000	43.020000	
123	4	1	0	833.58	20.43	17030.0394	1	463.000000	16.510000	
68	2	9	1	548.97	27.07	14860.6179	1	386.000000	25.290000	
53	0	0	0	30.69	89.80	2755.9620	0	22.000000	92.230000	
146	4	9	1	365.86	69.14	25295.5604	1	286.000000	65.310000	
1	4	7	0	687.48	42.61	29293.5228	1	558.000000	39.240000	
147	0	7	1	448.09	75.81	33969.7029	1	329.000000	71.750000	
32	0	8	1	595.79	44.30	26393.4970	1	137.000000	46.820000	
31	0	0	0	985.95	92.69	91387.7055	1	58.000000	96.880000	
122	1	0	1	771.93	19.17	14797.8981	0	554.000000	15.520000	
127	0	6	0	333.33	16.94	5646.6102	1	127.000000	12.670000	
74	0	7	0	373.21	38.88	14510.4048	1	166.000000	36.180000	
88	2	0	1	535.10	13.66	7309.4660	0	18.000000	8.830000	
96	2	6	0	331.30	67.30	22296.4900	1	135.000000	67.910000	
42	3	6	0	797.52	92.04	73403.7408	1	715.000000	89.080000	
134	3	5	1	249.49	89.82	22409.1918	0	180.000000	87.330000	
80	2	5	0	69.59	95.30	6631.9270	0	36.000000	98.670000	
48	1	6	0	681.45	78.56	53534.7120	1	552.000000	74.450000	
90	1	1	0	149.16	86.32	12875.4912	1	140.000000	81.470000	
65	4	7	1	62.79	95.03	5966.9337	1	16.000000	93.220000	
97	0	7	0	229.17	29.49	6758.2233	1	159.000000	34.040000	
64	0	0	1	791.17	45.01	35610.5617	0	534.000000	58.238844	
93	0	3	1	865.53	45.95	39771.1035	1	581.000000	49.380000	
114	0	5	0	950.71	38.42	36526.2782	0	459.000000	36.690000	
25	2	8	0	886.35	62.21	55139.8335	1	268.639456	66.700000	
41	0	6	1	112.80	91.62	10334.7360	1	104.000000	93.930000	

```
y_train
```

```
104     1350.00
89      5889.88
116    12989.50
82      6651.72
112     2300.40
         ...
63     23998.24
70     20214.60
81     22573.89
11     30377.25
95     12360.29
Name: Approx_Total Revenue(INR), Length: 120, dtype: float64
```

```
y_test
```

```
33      3568.14
16     23122.02
43       326.46
129     7661.96
50      6065.82
123     7644.13
68      9761.94
53      2029.06
146    18678.66
1      21895.92
147    23605.75
32      6414.34
31      5424.16
122     8598.08
127     1609.09
74      6005.88
88       158.94
96      9167.85
42     63692.20
134    15719.40
80      3552.12
48     41096.40
90     11405.80
65      1491.52
97      5412.36
64     25637.34
93     28689.78
114    16840.71
25     56361.50
41      9768.72
Name: Approx_Total Revenue(INR), dtype: float64
```

```
# Standarization


from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train_std=sc.fit_transform(X_train)
X_test_std=sc.transform(X_test)
```

```
X_train_std
```

```
array([[ 0.00590497,  0.89131387,  1.18321596, ..., -1.25317491,
         1.36825105, -0.83519599],
       [ 1.42309876, -0.16072873, -0.84515425, ..., -1.81363133,
         1.36825105,  0.23586477],
       [-0.70269192, -1.21277133, -0.84515425, ..., -0.67844944,
         1.36825105,  0.721809  ],
       ...,
       [ 1.42309876, -1.5634522 , -0.84515425, ...,  0.88361475,
         0.15202789,  1.00445004],
       [-0.70269192,  0.540633  , -0.84515425, ...,  1.1666175 ,
        -1.06419526,  0.41933351],
       [ 1.42309876,  0.89131387, -0.84515425, ...,  0.68226265,
         0.15202789, -0.52776188]])
```

```
X_test_std
```

```
array([[ 0.71450187, -0.16072873, -0.84515425,  0.30362616, -0.5865793 ,
        -0.14000487,  0.55167728, -0.7947494 , -0.6788458 ,  1.36825105,
         1.34659444],
       [-1.41128881, -0.16072873,  1.18321596,  0.58539627, -0.3890633 ,
         0.17564943,  0.55167728,  0.71910821, -0.2309563 ,  0.15202789,
         0.0077685 ],
       [-0.70269192,  0.18995214, -0.84515425,  0.628192  , -0.24860747,
         0.31273796, -1.81265393, -1.13880795, -0.16674559,  1.36825105,
         2.20939339],
       [ 0.00590497,  1.24199473, -0.84515425, -0.41037046, -0.3495601 ,
        -0.45579184,  0.55167728, -0.52810403, -0.27138526,  1.36825105,
         0.01272711],
       [ 0.71450187, -1.21277133,  1.18321596, -0.28518521, -0.68912801,
        -0.53300937,  0.55167728, -0.55820915, -0.61820236, -1.06419526,
         0.22594754],
       [ 1.42309876, -1.21277133, -0.84515425,  1.17152215, -1.52228642,
        -0.52965245,  0.55167728,  0.8266265 , -1.66895906,  0.15202789,
         0.72676762],
       [ 0.00590497,  1.5926756 ,  1.18321596,  0.19162737, -1.25733566,
        -0.62926262,  0.55167728,  0.49547015, -1.32095288, -1.06419526,
        -0.30462423],
       [-1.41128881, -1.5634522 , -0.84515425, -1.5927789 ,  1.24573077,
        -1.18505446, -1.81265393, -1.06999624,  1.33229698,  0.15202789,
        -1.06825088],
       [ 1.42309876,  1.5926756 ,  1.18321596, -0.43880914,  0.42135085,
        -0.15013658,  0.55167728,  0.06539697,  0.26528942, -1.06419526,
        -0.71618924],
       [ 1.42309876,  0.89131387, -0.84515425,  0.66850881, -0.63725512,
         0.03343204,  0.55167728,  1.23519603, -0.76802734,  1.36825105,
        -0.46825851],
       [-1.41128881,  0.89131387,  1.18321596, -0.15569629,  0.68749867,
         0.24814138,  0.55167728,  0.25032844,  0.5205468 ,  1.36825105,
        -0.51784465],
       [-1.41128881,  1.24199473,  1.18321596,  0.35282576, -0.56982037,
        -0.09972422,  0.55167728, -0.57541208, -0.46758465,  1.36825105,
         1.1631257 ],
       [-1.41128881, -1.5634522 , -0.84515425,  1.69612272,  1.36104819,
         2.8845201 ,  0.55167728, -0.92377135,  1.51581277,  0.15202789,
         3.49863319],
       [-0.70269192, -1.5634522 ,  1.18321596,  0.95926498, -1.57256322,
        -0.63214243, -1.81265393,  1.2179931 , -1.70819894,  0.15202789,
        -0.03190042],
       [-1.41128881,  0.540633  , -0.84515425, -0.55080793, -1.66154518,
        -1.05232878,  0.55167728, -0.61841939, -1.82116222,  1.36825105,
        -0.08644518],
       [-1.41128881,  0.89131387, -0.84515425, -0.41350354, -0.78609041,
        -0.64534284,  0.55167728, -0.45069085, -0.88931423,  0.15202789,
        -0.08148657],
       [ 0.00590497, -1.5634522 ,  1.18321596,  0.14387381, -1.79242447,
        -0.97597785, -1.81265393, -1.08719916, -1.97336538,  0.15202789,
         1.45568397],
       [ 0.00590497,  0.540633  , -0.84515425, -0.5577971 ,  0.34793076,
        -0.28784052,  0.55167728, -0.58401354,  0.36834364,  0.15202789,
        -0.13603133],
       [ 0.71450187,  0.540633  , -0.84515425,  1.04736978,  1.33511175,
         2.05877661,  0.55167728,  1.91041092,  1.20744282,  1.36825105,
        -0.7013134 ],
```

Model Building

```
# LinearRegression Model

from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train_std,y_train)
```

▾ LinearRegression

LinearRegression()

```
Y_pred_lr=lr.predict(X_test_std)
```

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
print(r2_score(y_test,Y_pred_lr))
h1=mean_absolute_error(y_test,Y_pred_lr)
print(h1)
g1=mean_squared_error(y_test,Y_pred_lr)
print(g1)
score = r2_score(y_test,Y_pred_lr)
print("The accuracy of the model is {}%".format(round(score, 2) *100))
```

```
0.8158321311411573
3610.290759970679
43966520.5580716
The accuracy of the model is 82.0%
```

```
# Lasso Model

from sklearn.linear_model import Lasso
ls=Lasso()
ls.fit(X_train_std,y_train)
```

▾ Lasso

Lasso()

```
Y_pred_ls=ls.predict(X_test_std)
```

```
print(r2_score(y_test,Y_pred_ls))
h2=mean_absolute_error(y_test,Y_pred_ls)
print(h2)
g2=mean_squared_error(y_test,Y_pred_ls)
print(g2)
score1 = r2_score(y_test,Y_pred_ls)
print("The accuracy of the model is {}%".format(round(score1, 2) *100))
```

```
0.8330249464114738
3525.013044916586
39862068.07829146
The accuracy of the model is 83.0%
```

```
# RandomForestRegression


from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()
rfr.fit(X_train_std,y_train)
```

```
▾ RandomForestRegressor

RandomForestRegressor()
```

```
Y_pred_rfr=rfr.predict(X_test_std)
```

```
print(r2_score(y_test,Y_pred_rfr))
h3=mean_absolute_error(y_test,Y_pred_rfr)
print(h3)
g3=mean_squared_error(y_test,Y_pred_rfr)
print(g3)
score2 = r2_score(y_test,Y_pred_rfr)
print("The accuracy of the model is {}%".format(round(score2, 2) *100))
```

```
0.7721721035375838
3312.4050899999975
54389508.63461696
The accuracy of the model is 77.0%
```

```
# Ridge model

from sklearn.linear_model import Ridge
r=Ridge()
r.fit(X_train_std,y_train)
```

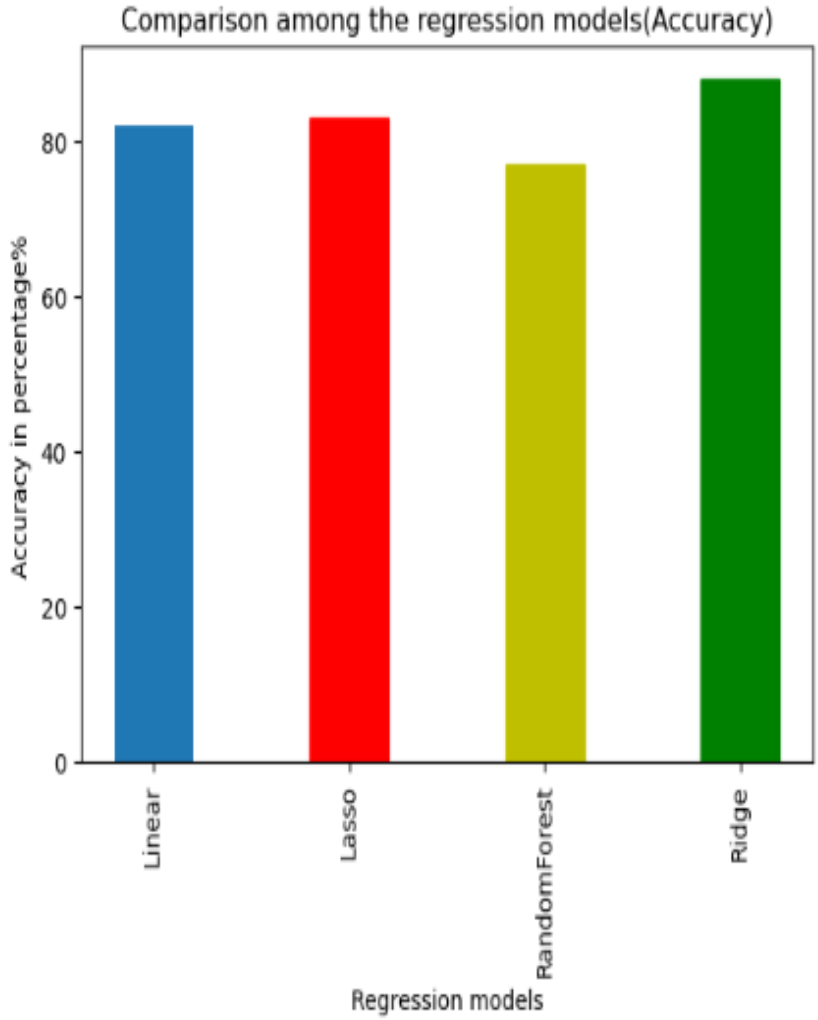
```
Ridge
Ridge()
```

```
Y_pred_r=r.predict(X_test_std)
```

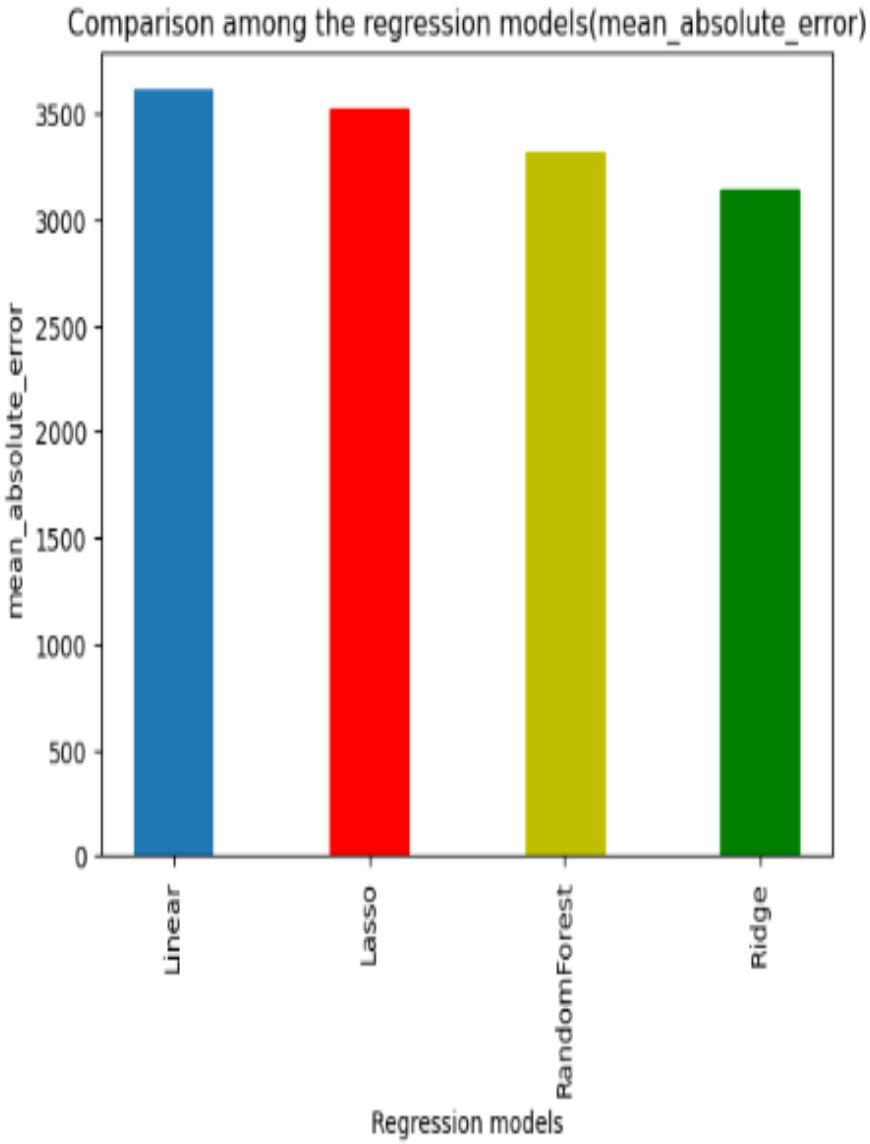
```
print(r2_score(y_test,Y_pred_r))
h4=mean_absolute_error(y_test,Y_pred_r)
print(h4)
g4=mean_squared_error(y_test,Y_pred_r)
print(g4)
score3 = r2_score(y_test,Y_pred_r)
print("The accuracy of the model is {}%".format(round(score3, 2) *100))
```

```
0.8840922651388735
3146.1586018305175
27670732.35514907
The accuracy of the model is 88.0%
```

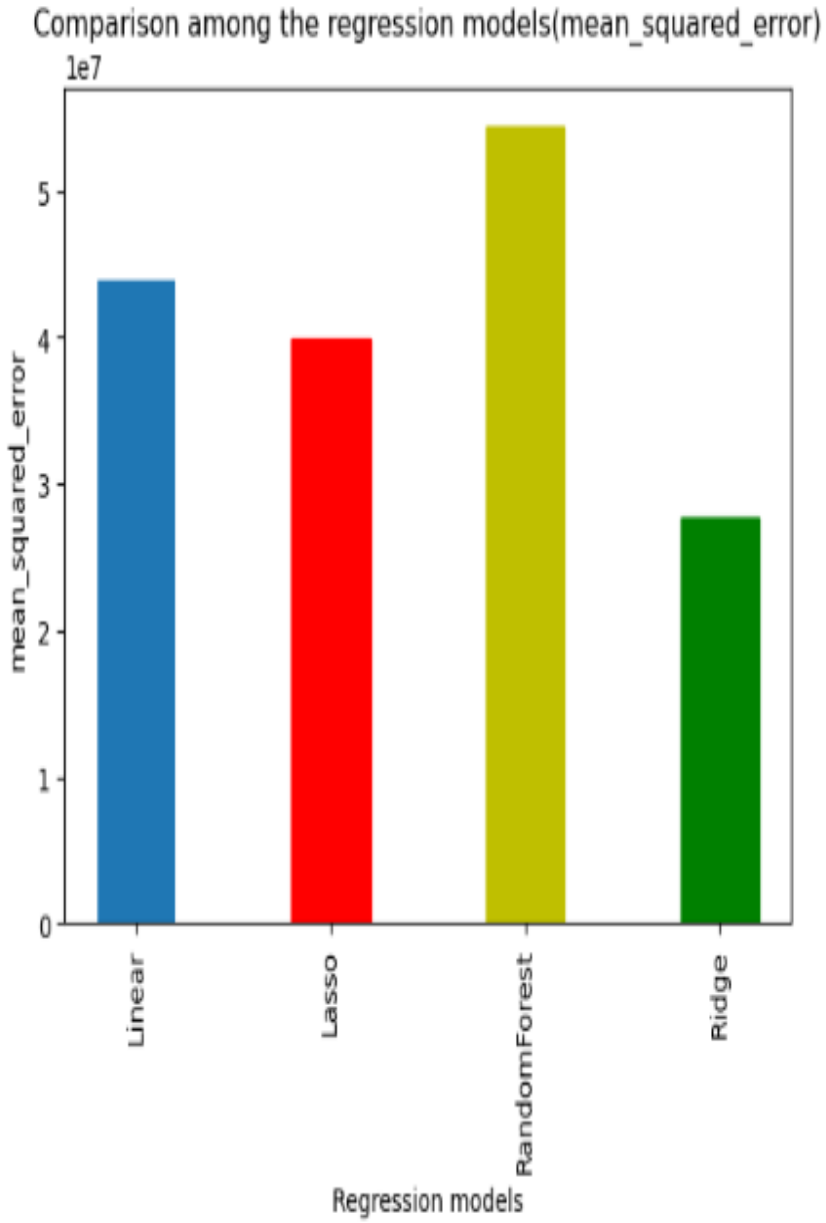
```
# comparison for all the models

data = {'Linear':82, 'Lasso':83, 'RandomForest':77,
        'Ridge':88}
models = list(data.keys())
accuracy = list(data.values())
barlist=plt.bar(models, accuracy,width = 0.4)
barlist[1].set_color('r')
barlist[2].set_color('y')
barlist[3].set_color('g')
plt.xlabel("Regression models")
plt.xticks(rotation=90)
plt.ylabel("Accuracy in percentage%")
plt.title("Comparison among the regression models(Accuracy)")
plt.show()
```



```
data = {'Linear':h1, 'Lasso':h2, 'RandomForest':h3,
        'Ridge':h4}
models = list(data.keys())
accuracy = list(data.values())
barlist=plt.bar(models, accuracy,width = 0.4)
barlist[1].set_color('r')
barlist[2].set_color('y')
barlist[3].set_color('g')
plt.xlabel("Regression models")
plt.xticks(rotation=90)
plt.ylabel("mean_absolute_error")
plt.title("Comparison among the regression models(mean_absolute_error)")
plt.show()
```



```
data = {'Linear':g1, 'Lasso':g2, 'RandomForest':g3,
        'Ridge':g4}
models = list(data.keys())
accuracy = list(data.values())
barlist=plt.bar(models, accuracy,width = 0.4)
barlist[1].set_color('r')
barlist[2].set_color('y')
barlist[3].set_color('g')
plt.xlabel("Regression models")
plt.xticks(rotation=90)
plt.ylabel("mean_squared_error")
plt.title("Comparison among the regression models(mean_squared_error)")
plt.show()
```



Future Scope of Improvements

Various product based companies use these models and modify them according to their needs to use in sales. This will reduce the manual labour and time spent.

• Companies who intend to sell products online, retail or wholesale can use these trained models to check whether their sales prediction is correct or not. The trained models would be required to be implemented in a platform or interface easily accessible as well as with an easy GUI.

• And more improvement can be done in this application by adding customer detail more precisely and fulfilling there needs accordingly with their past order records.

•Post delivery of the product customer reviews/feedback survey information should be collected and attached in database for future requirements.

Certificate

This is to certify that Mr Vishwanath.R of Lovely Professional University, registration number: 12108780, has successfully completed a project on Dairy Goods Sales Prediction using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Arnab Chakraborty