B9DA109 MACHINE LEARNING AND PATTERN RECOGNITION (B9DA109_2223_TMD2)

SUPERVISED MACHINE LEARNING - REGRESSION

(DESIGN AND IMPLEMENTATION OF LINEAR REGRESSION METHODS IN PYTHON)

B9DA109_CA_One - Group Assessment

Student (s) Number as per your student card:

MUKESH KUMAR (10631945)

VAIBHAV RANE (10625216)

YOGESH BIRAJDAR (10629587)

Course Title: MASTER OF SCIENCE IN DATA ANALYTICS

Lecturer Name: Mr. SATYA PRAKASH

Module/Subject Title: B9DA109 MACHINE LEARNING AND PATTERN RECOGNITION (B9DA109_2223_TMD2)

Assignment Title: SUPERVISED MACHINE LEARNING - REGRESSION

No of Words: 1858

CONTENTS

1. Introduction: 1
2. Data Preparation: 2
   2.1 Variable Selection 2
   2.2 Analyzing 2
   2.3. Data Preparation 3
3. Feature selection: 8
4. Model Development and Evaluation: 9
5. Model Comparison: 9
6. Conclusion: 11
7. Individual Contribution – (Mukesh Kumar-10631945) 12
Bibliography: 13

1. Introduction:

Machine learning studies algorithms and statistical models that allow computers to learn from and predict data. Pattern recognition, on the other hand, finds data patterns. Pattern recognition identifies and classifies patterns in data, whereas machine learning develops algorithms that learn and improve data. Using machine learning and pattern recognition for sales data has various advantages, including anticipating future sales trends, recognizing consumer behavior patterns, and refining marketing techniques to boost income. These technologies also assist firms in making better judgments, increasing productivity, and streamlining processes. Machine learning algorithms and pattern recognition methods may be used to reveal hidden patterns in massive volumes of sales data, making it simpler to spot trends and support more focused sales tactics. This, in turn, may assist organizations in optimizing their sales efforts and increasing revenue growth. As we can see how machine learning is useful for sales analysis, we selected a data set for our group assignment from https://www.kaggle.com/datasets/kyanyoga/sample-sales-data. The selected data defines Sample Sales Data, Order Info, Sales, Customer, Shipping, etc., used for Segmentation, Customer Analytics, Clustering, and more. Inspired by retail analytics, this was originally used for Pentaho DI Kettle, But I found the set could be useful for Sales Simulation training. Its Originally Written by María Carina Roldán, Pentaho Community Member, BI consultant (Assert Solutions), Argentina. This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unparted License. Modified by Gus Segura June 2014.

The regression model that we developed in the research may be further refined to forecast Profit for Sales in each Category, Subcategory, Region, and City, among others. For now, we are doing predictive analysis for the following problem:

"Build a regression model that can predict sales of a product based on MSRP, quantity ordered, and price range."

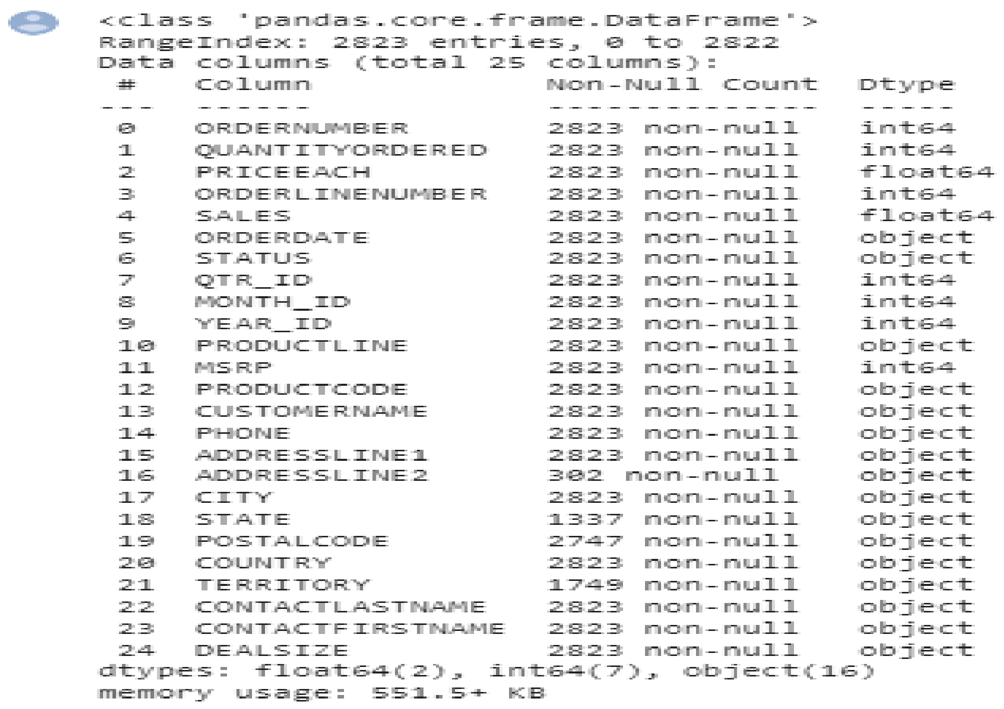
2. Data Preparation:

We did the following things with the above-mentioned data set, which includes both descriptive and predictive analysis:

2.1 Variable Selection, including the definition of independent and dependent variables, the selection of the appropriate kind of supervised regression techniques based on the variables, and the selection of the regression model to be used. "How to boost our total profit and sales" is a problem shared by all sales departments. As a result, "output" has been designated as the dependent variable, while the other variables will serve as independent variables, as output is directly or indirectly influenced by any change in their values.

Considering there is 1 dependent variable (1 output) and 21 independent variables (21 inputs), we can state that we are dealing because our target variable (SALES) is numerical and continuous, we use **supervised regression techniques**, specifically **multiple linear regression** and **Stochastic Gradient Descent Regressor (SGDRegressor)**, to build predictive models. As our output is not categorical, we will need to create a linear regression model.

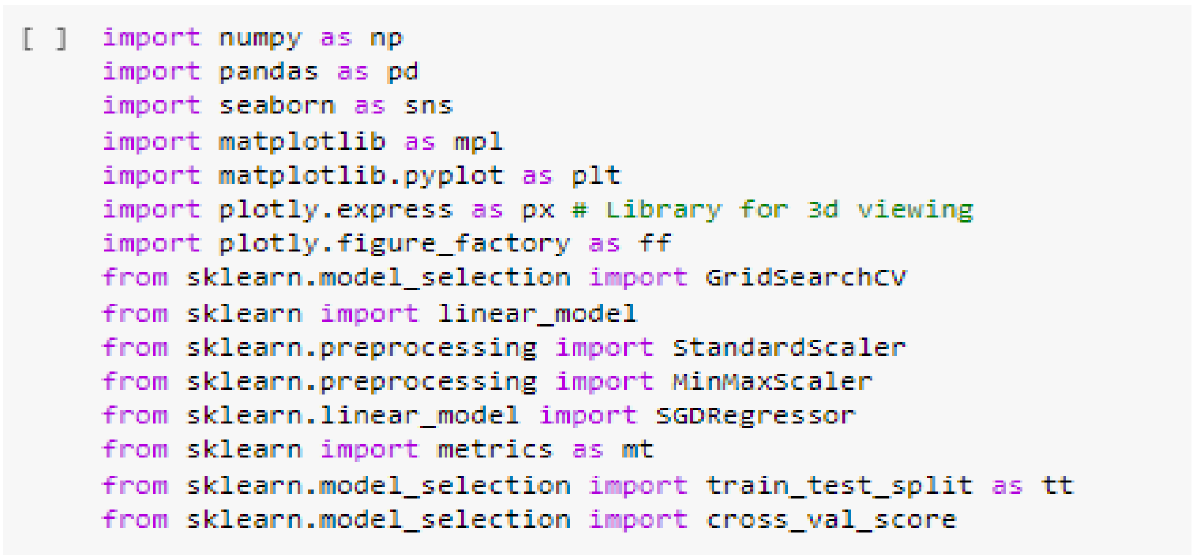
2.2 Analyzing the dataset using field descriptions, data types, duplicate data, and null data checks. By analyzing the Sales dataset, we discovered that there are 2823 rows and 25 columns, indicating that we have sufficient data to train a machine and estimate sales. The properties and description of the dataset are provided below:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      302 non-null    object
 17  CITY              2823 non-null   object
 18  STATE             1337 non-null   object
 19  POSTALCODE        2747 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         1749 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

2.3. Data Preparation, including the conversion of any null data to non-null data, locating category categories and turning them into numerical variables, and removing variables that are not necessary. The following procedures fall under the category of "Data Preparation," and we carry them out as follows:

Step 1: We are importing different libraries and their methods that we may utilize in our code for certain actions.

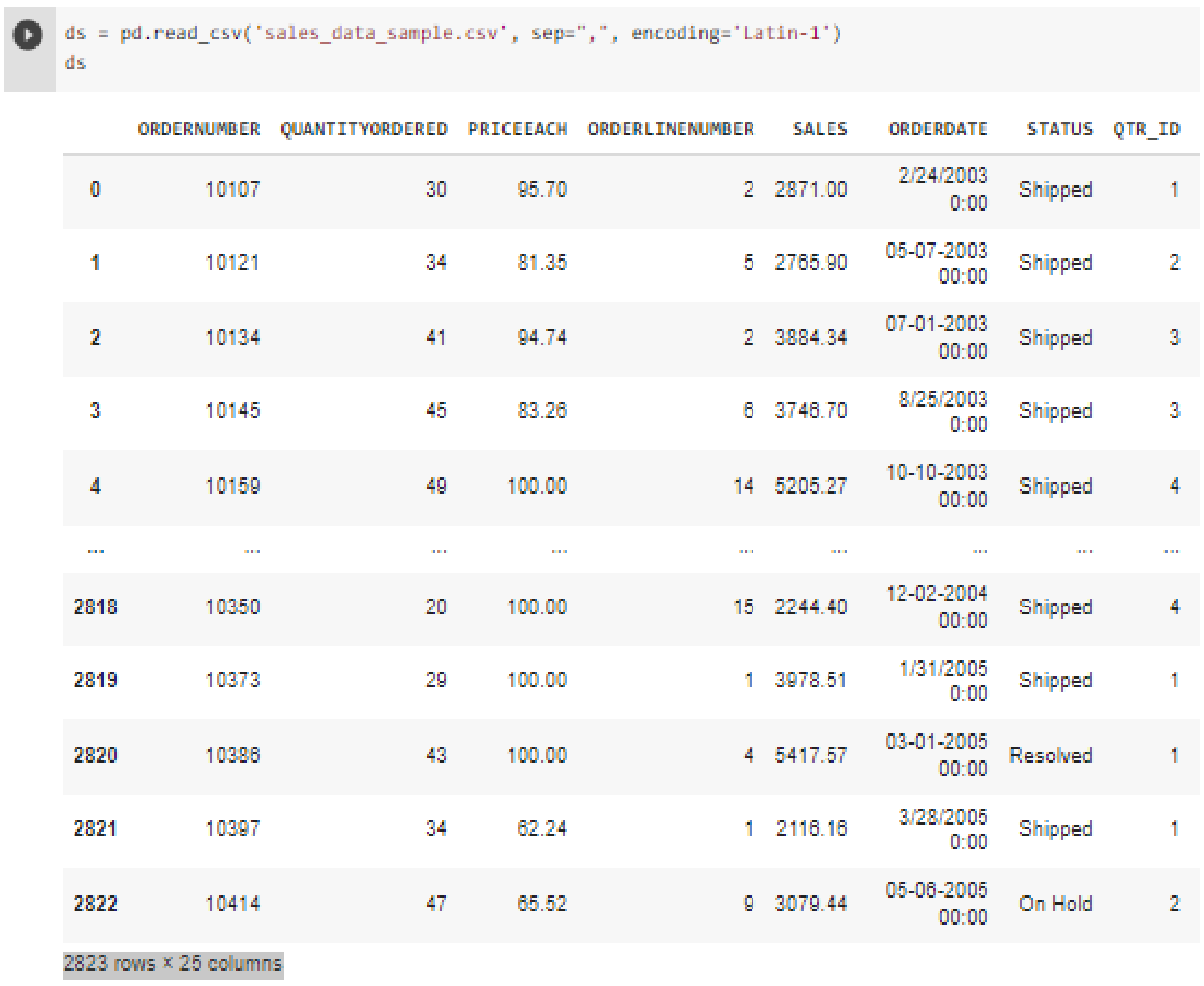


```
[ ]  import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib as mpl
     import matplotlib.pyplot as plt
     import plotly.express as px # Library for 3d viewing
     import plotly.figure_factory as ff
     from sklearn.model_selection import GridSearchCV
     from sklearn import linear_model
     from sklearn.preprocessing import StandardScaler
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.linear_model import SGDRegressor
     from sklearn import metrics as mt
     from sklearn.model_selection import train_test_split as tt
     from sklearn.model_selection import cross_val_score
```

Step 2:

```
ds = pd.read_csv('sales_data_sample.csv', sep=",", encoding='Latin-1')
```

function was used to read the dataset because it is in.csv format. There are 25 columns and 2823 rows, as shown by the result.

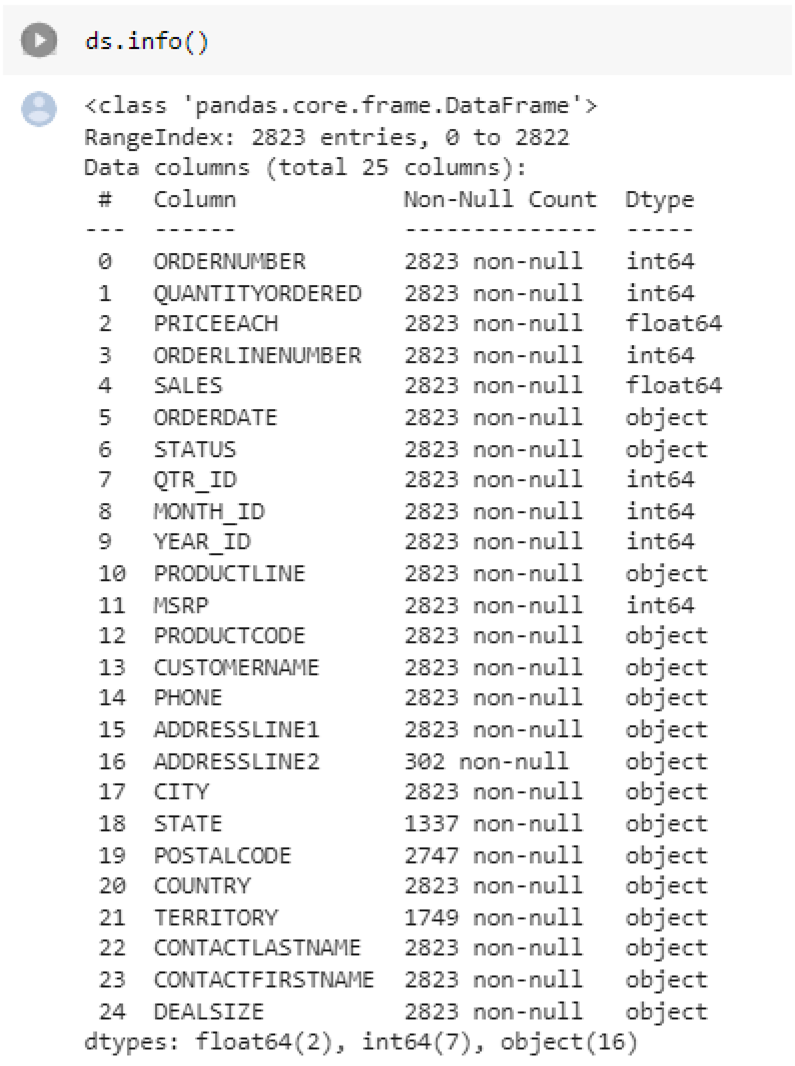


```python
ds = pd.read_csv('sales_data_sample.csv', sep=",", encoding='Latin-1')
ds
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1
1	10121	34	81.35	5	2765.90	05-07-2003 00:00	Shipped	2
2	10134	41	94.74	2	3884.34	07-01-2003 00:00	Shipped	3
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3
4	10159	49	100.00	14	5205.27	10-10-2003 00:00	Shipped	4
...	...	...	...	...	...	...	...	...
2818	10350	20	100.00	15	2244.40	12-02-2004 00:00	Shipped	4
2819	10373	29	100.00	1	3978.51	1/31/2005 0:00	Shipped	1
2820	10386	43	100.00	4	5417.57	03-01-2005 00:00	Resolved	1
2821	10397	34	62.24	1	2116.16	3/28/2005 0:00	Shipped	1
2822	10414	47	65.52	9	3079.44	05-06-2005 00:00	On Hold	2

2823 rows × 25 columns

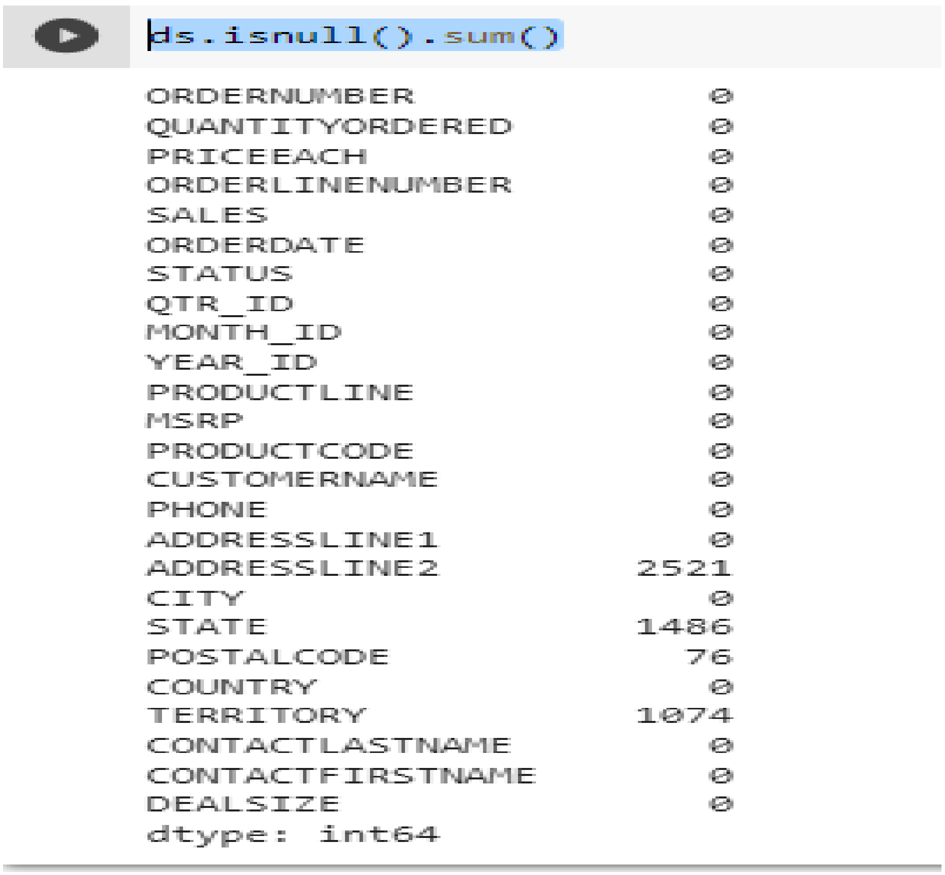
Step 3: Data information was defined by using the following function as: `ds.info()`, then the information received was as follows:



```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
```

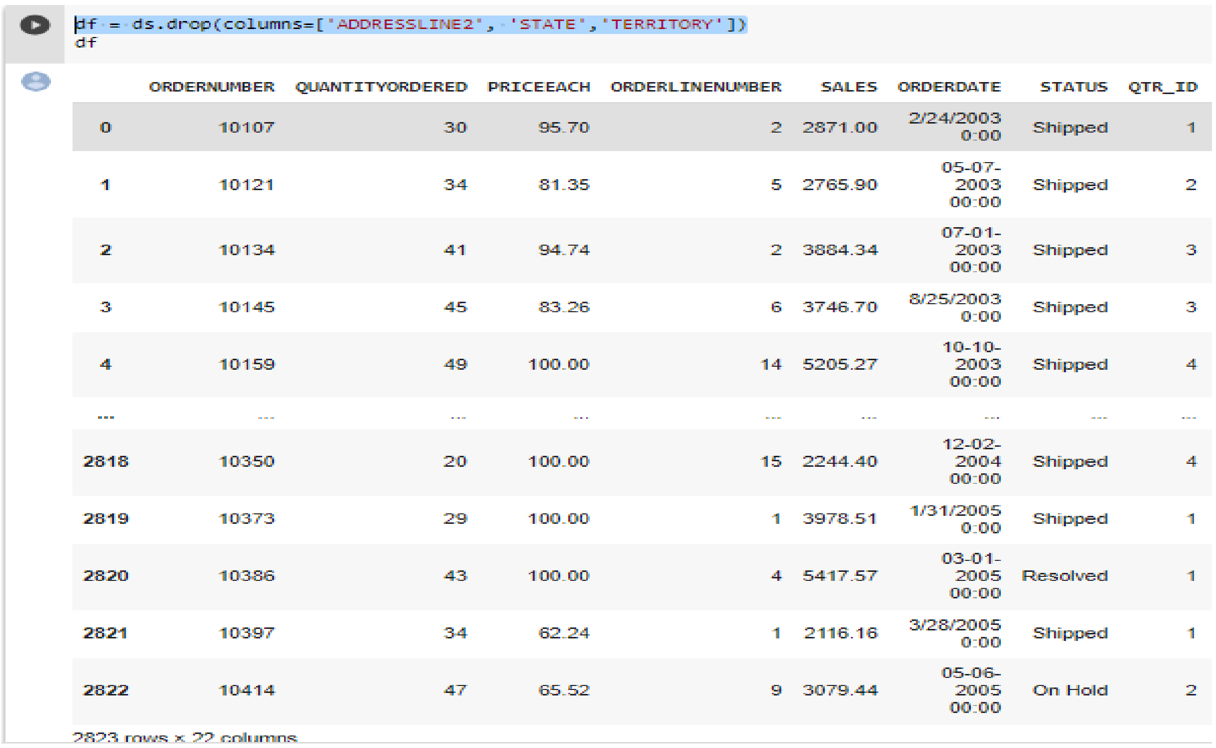
Step 4: Using `ds.isnull().sum()`, we found the null values as



```
ds.isnull().sum()
```

```
ORDERNUMBER            0
QUANTITYORDERED        0
PRICEEACH              0
ORDERLINENUMBER        0
SALES                  0
ORDERDATE              0
STATUS                 0
QTR_ID                 0
MONTH_ID               0
YEAR_ID                0
PRODUCTLINE            0
MSRP                   0
PRODUCTCODE            0
CUSTOMERNAME           0
PHONE                  0
ADDRESSLINE1           0
ADDRESSLINE2        2521
CITY                   0
STATE               1486
POSTALCODE            76
COUNTRY                0
TERRITORY           1074
CONTACTLASTNAME        0
CONTACTFIRSTNAME       0
DEALSIZE               0
dtype: int64
```

Step 5: It was clear that ADDRESSLINE2, STATE and TERRITORY have the maximum null values in the data set. So, these were removed by using the function as `df = ds.drop(columns=['ADDRESSLINE2', 'STATE','TERRITORY'])`



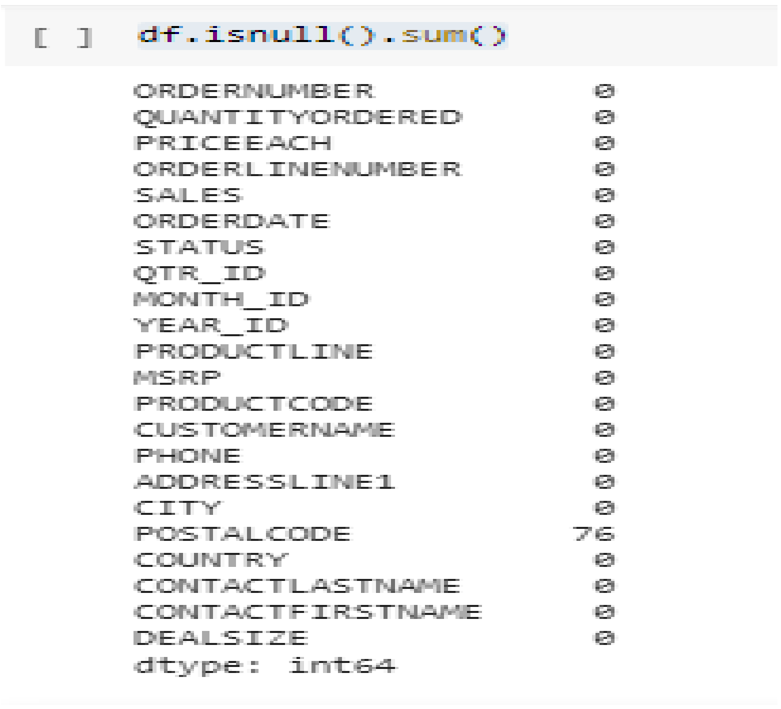
```
df = ds.drop(columns=['ADDRESSLINE2', 'STATE','TERRITORY'])
df
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1
1	10121	34	81.35	5	2765.90	05-07-2003 00:00	Shipped	2
2	10134	41	94.74	2	3884.34	07-01-2003 00:00	Shipped	3
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3
4	10159	49	100.00	14	5205.27	10-10-2003 00:00	Shipped	4
...	...	...	...	...	...	...	...	...
2818	10350	20	100.00	15	2244.40	12-02-2004 00:00	Shipped	4
2819	10373	29	100.00	1	3978.51	1/31/2005 0:00	Shipped	1
2820	10386	43	100.00	4	5417.57	03-01-2005 00:00	Resolved	1
2821	10397	34	62.24	1	2116.16	3/28/2005 0:00	Shipped	1
2822	10414	47	65.52	9	3079.44	05-06-2005 00:00	On Hold	2

2823 rows × 22 columns

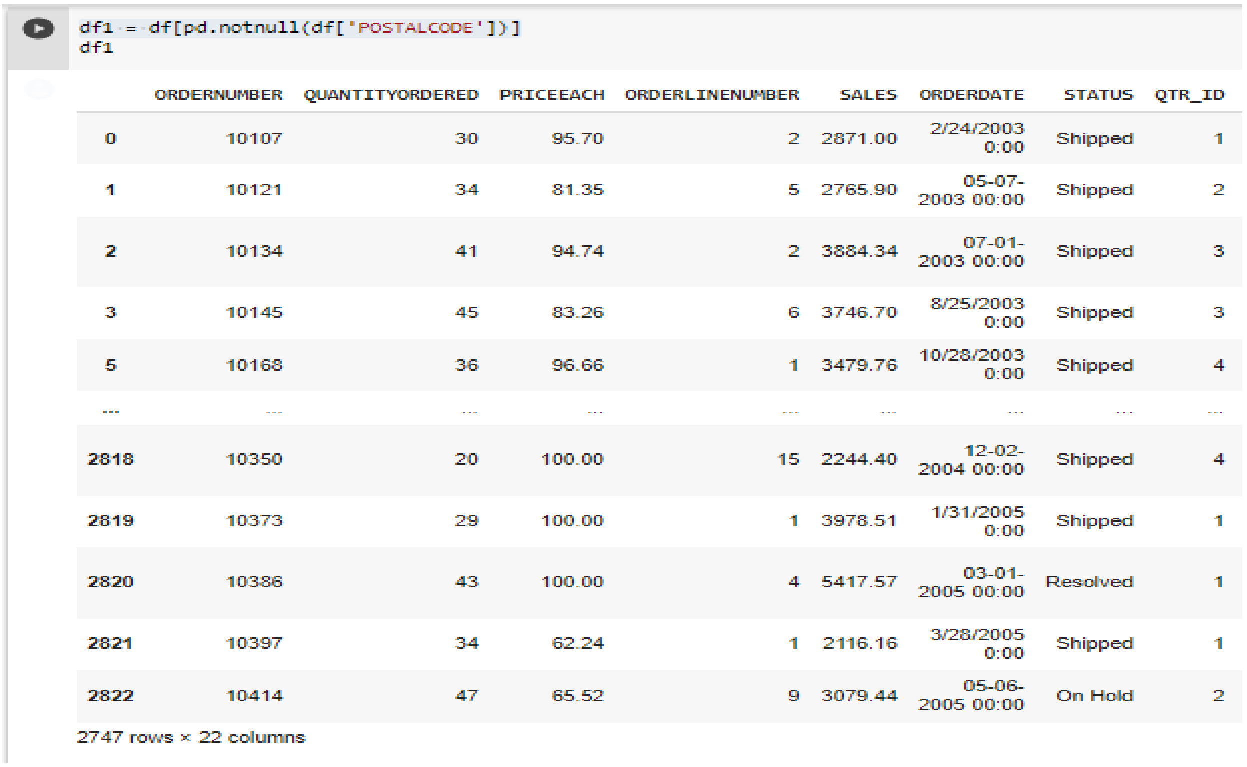
Step 6: It was also clear that postal code has 76 null values in different rows, so we also removed these rows using following functions as: `df.isnull().sum()`

```
df1 = df[pd.notnull(df['POSTALCODE'])]
```



```
[ ]  df.isnull().sum()

ORDERNUMBER           0
QUANTITYORDERED       0
PRICEEACH             0
ORDERLINENUMBER       0
SALES                 0
ORDERDATE             0
STATUS                0
QTR_ID                0
MONTH_ID              0
YEAR_ID               0
PRODUCTLINE           0
MSRP                  0
PRODUCTCODE           0
CUSTOMERNAME          0
PHONE                 0
ADDRESSLINE1          0
CITY                  0
POSTALCODE           76
COUNTRY               0
CONTACTLASTNAME       0
CONTACTFIRSTNAME      0
DEALSIZE              0
dtype: int64
```



```
df1 = df[pd.notnull(df['POSTALCODE'])]
df1
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1
1	10121	34	81.35	5	2765.90	05-07-2003 00:00	Shipped	2
2	10134	41	94.74	2	3884.34	07-01-2003 00:00	Shipped	3
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3
5	10168	36	96.66	1	3479.76	10/28/2003 0:00	Shipped	4
...	...	...	...	...	...	...	...	...
2818	10350	20	100.00	15	2244.40	12-02-2004 00:00	Shipped	4
2819	10373	29	100.00	1	3978.51	1/31/2005 0:00	Shipped	1
2820	10386	43	100.00	4	5417.57	03-01-2005 00:00	Resolved	1
2821	10397	34	62.24	1	2116.16	3/28/2005 0:00	Shipped	1
2822	10414	47	65.52	9	3079.44	05-06-2005 00:00	On Hold	2

2747 rows × 22 columns

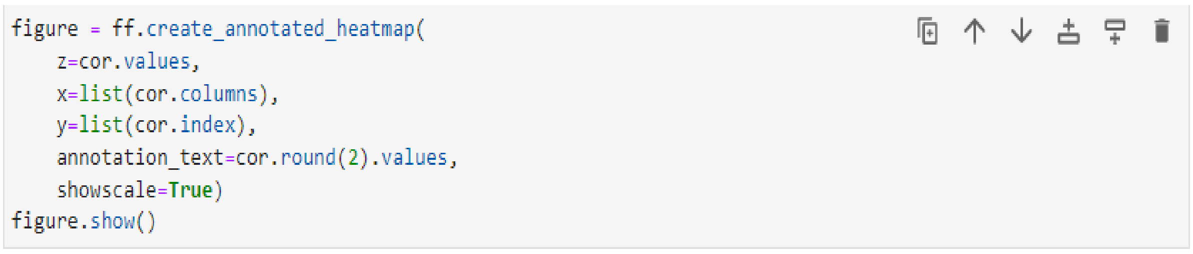
Step 7: As Duplicate values are so problematic, we used the function `df1.duplicated().count()` to remove duplicates and then prepare the final data for the next process.

3. Feature selection:

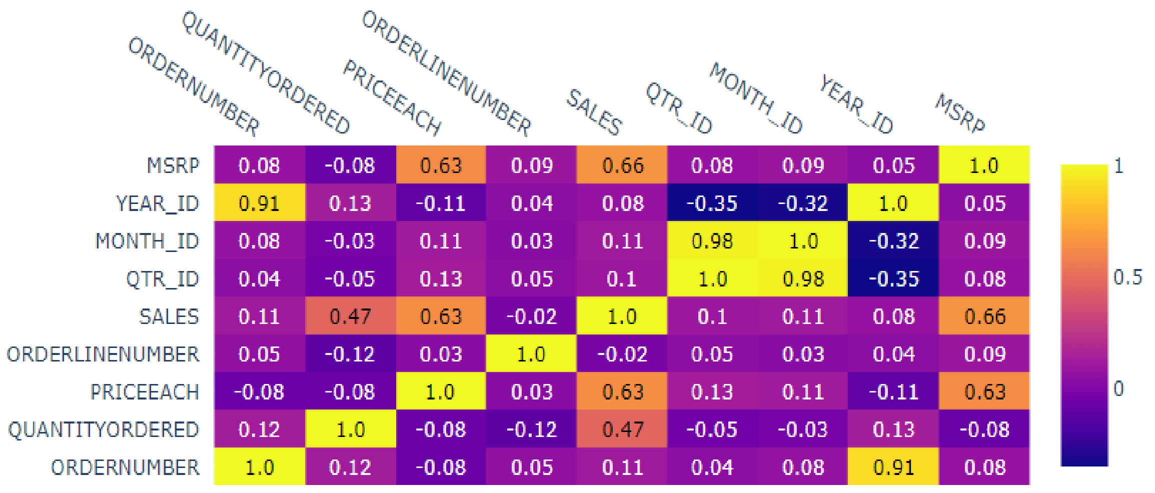
It is a method for standardizing independent data characteristics within a specified range. It is conducted during the preliminary processing of data

After removing non-contributing and categorical variables, the final selected numeric features used for modelling were **QUANTITYORDERED, PRICEEACH, and MSRP**.

We dropped ADDRESSLINE2, STATE, and TERRITORY. Most of them yield no addition to our problem statement. POSTELCODE also had some null values in some rows and without these values it's not useful to perform as a valid data set, so we removed all the rows which have null values for POSTELCODE. As a further step after data preparation for modelling, we used data visualization, which was shown in the following forms:



```
figure = ff.create_annotated_heatmap(
    z=cor.values,
    x=list(cor.columns),
    y=list(cor.index),
    annotation_text=cor.round(2).values,
    showscale=True)
figure.show()
```



	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	QTR_ID	MONTH_ID	YEAR_ID	MSRP
MSRP	0.08	-0.08	0.63	0.09	0.66	0.08	0.09	0.05	1.0
YEAR_ID	0.91	0.13	-0.11	0.04	0.08	-0.35	-0.32	1.0	0.05
MONTH_ID	0.08	-0.03	0.11	0.03	0.11	0.98	1.0	-0.32	0.09
QTR_ID	0.04	-0.05	0.13	0.05	0.1	1.0	0.98	-0.35	0.08
SALES	0.11	0.47	0.63	-0.02	1.0	0.1	0.11	0.08	0.66
ORDERLINENUMBER	0.05	-0.12	0.03	1.0	-0.02	0.05	0.03	0.04	0.09
PRICEEACH	-0.08	-0.08	1.0	0.03	0.63	0.13	0.11	-0.11	0.63
QUANTITYORDERED	0.12	1.0	-0.08	-0.12	0.47	-0.05	-0.03	0.13	-0.08
ORDERNUMBER	1.0	0.12	-0.08	0.05	0.11	0.04	0.08	0.91	0.08

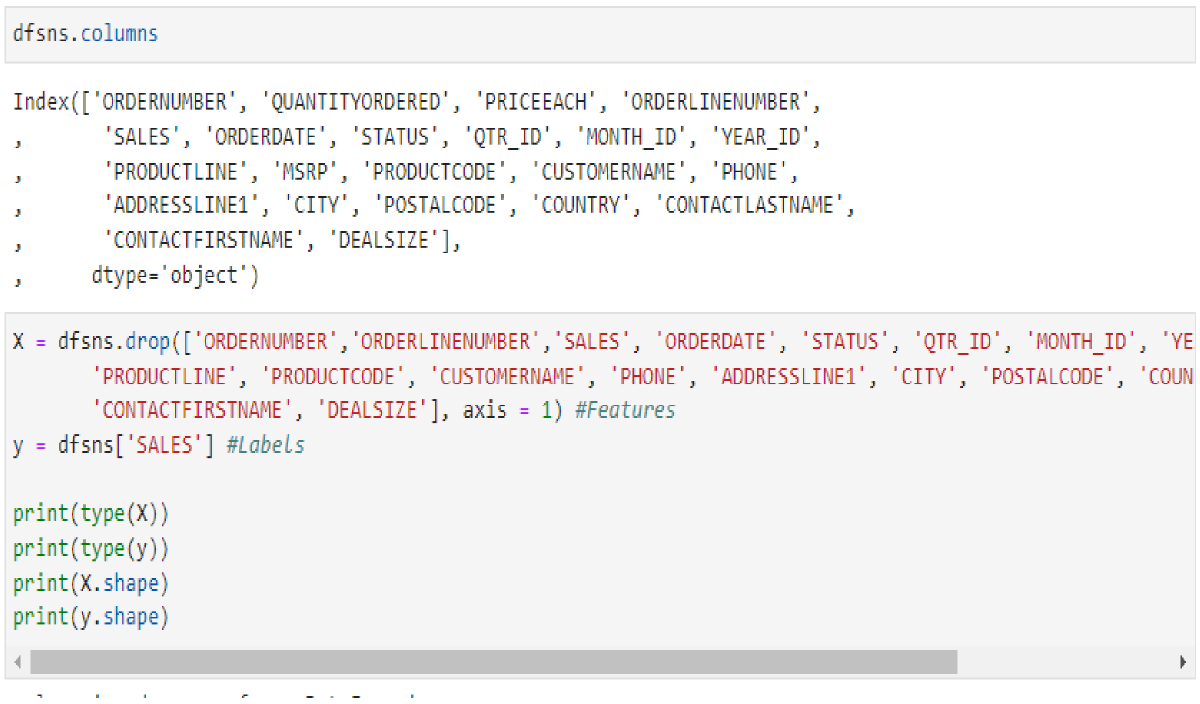
4. Model Development and Evaluation:

After segregating features and labels, created model by first Scaling filtered dataset by using. Applied StandardScaler to normalize QUANTITYORDERED, PRICEEACH, and MSRP because linear regression performs better when features have comparable scales. The dataset was split into 80% training and 20% testing using train_test_split() to evaluate how well the model generalizes to unseen data. Used sklearn's LinearRegression model to learn coefficients for MSRP, QUANTITYORDERED, and PRICEEACH.

```
scale = StandardScaler()

scaledX = scale.fit_transform(X)
```

then dividing dataset further into Training & Testing sets to train machine with 80% dataset and test the developed model with remaining 20%.



```
dfsns.columns
```

```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
,       'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
,       'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
,       'ADDRESSLINE1', 'CITY', 'POSTALCODE', 'COUNTRY', 'CONTACTLASTNAME',
,       'CONTACTFIRSTNAME', 'DEALSIZE'],
,      dtype='object')
```

```
X = dfsns.drop(['ORDERNUMBER','ORDERLINENUMBER','SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YE
       'PRODUCTLINE', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE', 'ADDRESSLINE1', 'CITY', 'POSTALCODE', 'COUN
       'CONTACTFIRSTNAME', 'DEALSIZE'], axis = 1) #Features
y = dfsns['SALES'] #Labels

print(type(X))
print(type(y))
print(X.shape)
print(y.shape)
```

We tried predicting a model not only with the use of a test dataset but also used a random value to generate a Sales prediction.

5. Model Comparison:

Once we had the predictions, we checked the random value by putting the coefficient and intercept values into the equation and seeing if our predicted sales value met the below equation or not –

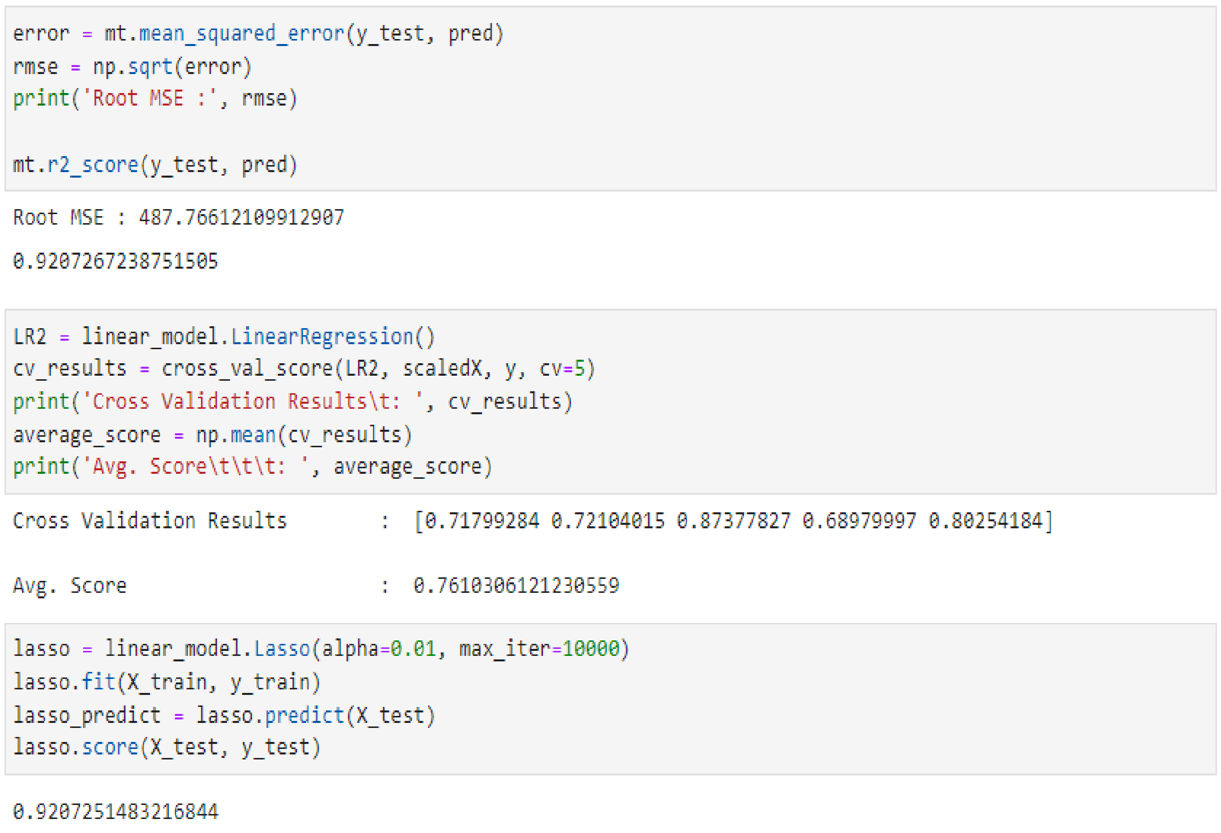
$$y = m_1x_1 + m_2x_2 + m_3x_3 + c$$

SALES=m1 (MSRP)+m2 (QUANTITYORDERED)+m3 (PRICEEACH)+c

Where:

- **y = SALES** (dependent variable)
- **X_1 = MSRP**
- **X_2 = QUANTITYORDERED**
- **X_3 = PRICEEACH**
- **m_1, m_2, m_3** = regression coefficients
- **c** = intercept.

Model performance was evaluated using R^2, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and 5-fold cross-validation. These metrics help assess prediction accuracy and detect overfitting.



```python
error = mt.mean_squared_error(y_test, pred)
rmse = np.sqrt(error)
print('Root MSE :', rmse)

mt.r2_score(y_test, pred)
```

```
Root MSE : 487.76612109912907
0.9207267238751505
```

```python
LR2 = linear_model.LinearRegression()
cv_results = cross_val_score(LR2, scaledX, y, cv=5)
print('Cross Validation Results\t: ', cv_results)
average_score = np.mean(cv_results)
print('Avg. Score\t\t\t: ', average_score)
```

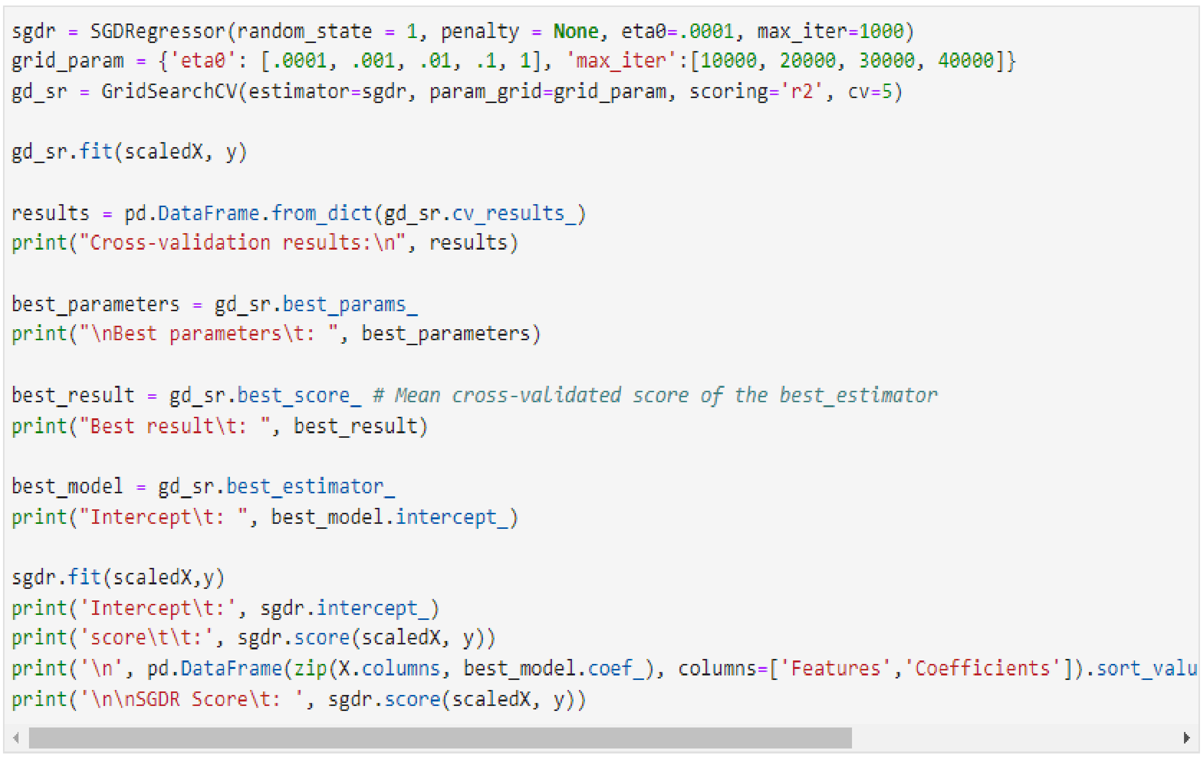
```
Cross Validation Results       :  [0.71799284 0.72104015 0.87377827 0.68979997 0.80254184]

Avg. Score                     :  0.7610306121230559
```

```python
lasso = linear_model.Lasso(alpha=0.01, max_iter=10000)
lasso.fit(X_train, y_train)
lasso_predict = lasso.predict(X_test)
lasso.score(X_test, y_test)
```

```
0.9207251483216844
```

SGDRegressor was used as an alternative regression method. Through GridSearchCV, we tested multiple learning rates and iteration values to find the best performing SGD-based model. While creating a regressor using SGD, we set the learning rate to 0.0001 and the maximum iterations to 1000. To locate the optimal coefficients, we put our regressor through a few additional learning rates of.0001,.00001,.01, and.1 with maximum iterations of 10000, 20000, 30000, and 40000 correspondingly.



```
sgdr = SGDRegressor(random_state = 1, penalty = None, eta0=.0001, max_iter=1000)
grid_param = {'eta0': [.0001, .001, .01, .1, 1], 'max_iter':[10000, 20000, 30000, 40000]}
gd_sr = GridSearchCV(estimator=sgdr, param_grid=grid_param, scoring='r2', cv=5)

gd_sr.fit(scaledX, y)

results = pd.DataFrame.from_dict(gd_sr.cv_results_)
print("Cross-validation results:\n", results)

best_parameters = gd_sr.best_params_
print("\nBest parameters\t: ", best_parameters)

best_result = gd_sr.best_score_ # Mean cross-validated score of the best_estimator
print("Best result\t: ", best_result)

best_model = gd_sr.best_estimator_
print("Intercept\t: ", best_model.intercept_)

sgdr.fit(scaledX,y)
print('Intercept\t:', sgdr.intercept_)
print('score\t\t:', sgdr.score(scaledX, y))
print('\n', pd.DataFrame(zip(X.columns, best_model.coef_), columns=['Features','Coefficients']).sort_valu
print('\n\nSGDR Score\t: ', sgdr.score(scaledX, y))
```

In addition, we re-forecasted the profit for the same amount of random sales, arriving at the following SGDR Score: -0.21338791312720384

6. Conclusion:

All the preceding phases of predictive analysis led us to the conclusion that we have developed a linear regression model capable of predicting sales based on MSRP, quantity ordered, and price per unit. During the process, we developed a model that trained our machine and enabled it to predict values based on a given dataset. In the equation of the linear regression model, we also included cross validation and random value inputs to validate our model. When data is produced for a sample only, we can also deduce that this prediction is not perfect and that it will get more accurate as we train our machine with additional data over the original data over a longer period, say 10 years. In addition, it was determined that there is a great deal of room for prediction in the sample sales dataset and that machines can be trained to make many more predictions from the same dataset, such as increasing overall revenue through sales of a particular category, subcategory, quantity, region, and city, and answering many other predictive questions.

7. Individual Contribution – (Vaibhav Rane-10625216)

Working on this group project gave me the opportunity to apply machine learning techniques in a practical context and to contribute meaningfully across multiple stages of the assignment. My role involved extensive technical work, conceptual clarification, and ensuring that the final implementation aligned with the principles of supervised regression modelling. Throughout the project, I took responsibility for the data preparation pipeline, model development, performance evaluation, and theoretical corrections that were necessary to complete the assignment successfully.

1. Understanding and Preparing the Dataset

My first contribution was to thoroughly explore and understand the sales dataset. I examined the structure, data types, distribution of values, and presence of missing or inconsistent entries. I identified variables such as ADDRESSLINE2, STATE, and TERRITORY that contained substantial missing data and therefore needed to be removed to maintain the integrity of the dataset. I also located rows with missing postal codes and implemented row-level filtering to ensure all remaining entries were usable for modelling.

I checked for duplicate records using Pandas functions and removed them to avoid bias or distortion during training. These steps ensured that our final dataset was clean, reliable, and suitable for statistical modelling. This data-preparation stage was essential, as the accuracy of any machine learning model depends heavily on the quality of the input data.

2. Feature Selection and Initial Analysis

After cleaning the data, I led the analysis to determine which features were most relevant for predicting SALES. Using correlation heatmaps, scatter plots, and explorative visualizations, I evaluated how different attributes of the dataset related to the target variable. Through this process, I identified that **QUANTITYORDERED**, **PRICEEACH**, and **MSRP** were the most meaningful predictors for a regression model.

I then prepared the dataset by constructing the feature matrix (X) and the label vector (y), selecting only numerical features suitable for regression. I applied StandardScaler() to normalize the feature values so that the linear regression algorithm could learn coefficient weights more effectively. My focus during this stage was to ensure the feature set reflected a logical and statistically justified relationship with SALES.

3. Developing and Training the Regression Models

A major portion of my contribution involved constructing the machine learning models in Python. I implemented multiple regression approaches, including:

- **Linear Regression** using scikit-learn
- **Lasso Regression** for regularization
- **SGDRegressor** to explore optimization through stochastic gradient descent

I split the dataset into training and testing subsets using an 80/20 ratio and trained the models using the scaled dataset. I generated predictions from the trained model and compared them with actual sales values to evaluate performance.

To validate the correctness of the model, I also manually substituted random input values into the regression equation using the learned coefficients and intercept. This helped ensure that the mathematical structure of the model aligned with the predictions generated by the machine.

4. Performance Evaluation and Cross-Validation

I assessed model performance using several evaluation metrics, such as:

- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**
- **R² Score**
- **5-fold Cross-Validation**

This gave us a complete understanding of how well the model generalized to new data. I also interpreted the strengths and limitations of each regression technique, noting how linear regression produced stable coefficients while SGDRegressor required hyperparameter tuning.

To optimize SGDRegressor, I used GridSearchCV, testing multiple learning rates and iteration values. This allowed me to identify the best hyperparameters for the model and provided insights into how gradient-based methods behave during training.

Bibliography:

Data set Selection (https://www.kaggle.com/datasets/kyanyoga/sample-sales-data.)
Data and Issues with Data (https://elearning.dbs.ie/pluginfile.php/1779034/mod_resource/content/0/Data_2.pdf)
Linear Regression (https://elearning.dbs.ie/mod/resource/view.php?id=1352699)
Machine Learning And Pattern Recognition (https://elearning.dbs.ie/mod/resource/view.php?id=1343958)
Overflow problem solving (https://stackoverflow.com/questions/57598962/python-pandas-unicode-decode-error-on-read-csv)
Python Notebook for Beginners (https://elearning.dbs.ie/mod/folder/view.php?id=1349142)
Scikit Learn Documentation - Linear Regression - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression.score

Scikit Learn Documentation - SGD Regressor - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html?highlight=sgd regressor#sklearn.linear_model.SGDRegressor