SALES PREDICTION USING MACHINE LEARNING

INTRODUCTION:

Businesses stand on their sales undoubtedly. When there is a large-scale marketplace to be covered by a single product manufacturing company, here comes the factor of individual product's sales. It is obvious that the overall sales in a business depend on the sales of their individual products. How these individual sales impact on the total sale rate is a big query to be resolved. The answer to this question can define how worthy a particular product is to boost the business value.

In this project, we are developing a Machine Learning regression model which would help us to find the total sales while giving it individual sales of different products. We have taken the Sales dataset available at Kaggle. This dataset consists of four columns and describes sales of different products and the Total sales as the target value. The column names are Jacket, Zipper, Sweater, and Sales (Total Sales). It is a pure Multiple Regression Problem and requires the relevant Machine Learning algorithm. We are using the simplest, yet sturdy algorithm i.e. Linear Regression.

PROBLEM STATEMENT:

As we know that there is an influence of every individual product's market value on the total sales of the business. Without having a proper idea about which product impacts how actively on the sales is not a good to go. For this purpose, businesses need such formulae to have an idea about the sales' fluctuation effected by their products and Machine Learning can be the best solution for this matter. As the problem is followed by the continuous data type of the sales feature, it is the Linear Regression problem and the dataset, having more than one train set / train features makes it a Multiple Linear Regression problem. What our linear regression model shall do is that it would give us accurate and precise estimation of how our products i.e. Jacket, Sweater and Zipper have put impact on the total sales.

TOOL AND UTILITIES:

In our project, we are using the Python Language as a Machine Learning tool for our task. It contains such powerful packages and libraries that one could easily solve a lot of problems beside Machine Learning and Artificial Intelligence as well. For the purpose of Machine Learning and Data Science, the Python language has built-in libraries which tend to solve any of the complex problems related to the three general Machine Learning categories i.e. Supervised Learning, Unsupervised Learning, and Clustering. Here, our project is purely a supervised learning problem as our Sales dataset has features as well as target values to train over a regression model. Now, for the execution of our project, there are certain Python libraries and utilities that we have used. Just taking a brief overview of them, they are defined as under:

Pandas:

- Pandas is an important data preprocessing library written for the Python language. It is an effectual tool to create and manipulate Data Frame objects.
- Pandas is sufficient to carry out data cleaning, data alignment, visualization and even handling of the missing data.
- We can also load the in-memory data files with different formats into the Python development environment

Matplotlib:

- Matplotlib is a data visualization library integrated with the Python language.
- We can produce a wide range of graphs according to the nature of our data.
- These different charts and graphs include line plots, scatter plots, bar graphs, pie charts, etc.

Seaborn:

- Seaborn library which is basically the advance version of Matplotlib.
- Seaborn library is the best for the plotting of statistical methods.
- Seaborn library is the all in one package for extensive and detailed data visualization including line plots, heatmaps, density plots, scatters, and many more utilities.

Scikit Learn:

- Sci-kit learn (or sklearn) is a Python library that is a complete package for machine learning algorithms and predictive analysis tasks.
- It deals with almost all the Machine Learning types including Regression, Classification, Clustering, Association Rules, etc.
- It contains such methods that are relevant for evaluation of the ML models. These methods include different accuracy and loss metrics.
- One can also deal with the preprocessing of data as there are different functions that help in encoding features, splitting datasets, and other useful utilities.

ABOUT LINEAR REGRESSION ALGORITHM:

The term "linearity" in algebra refers to a linear relationship between two or more variables. When this relationship is drawn on a graph, we get a straight line. As we know that equation of straight line is, $y = mx + b$. Here, b is the intercept and m is the slope. Linear Regression algorithm gives us the most optimal value for the intercept and the slope (in two dimensions). The y and x variables remain the same, since they are the data features and cannot be changed. The values that we can control are the intercept and slope. There can be multiple straight lines depending upon the values of intercept and slope. Basically what the linear regression algorithm does is it fits multiple lines on the data points and returns the line that results in the least error.

This same concept can be extended to the cases where there are more than two variables. This is called multiple linear regression. The dependent variable is dependent upon several independent variables. A regression model involving multiple variables can be represented as:

$$y = b_0 + m_1b_1 + m_2b_2 + m_3b_3 + \ldots \ldots m_nb_n$$

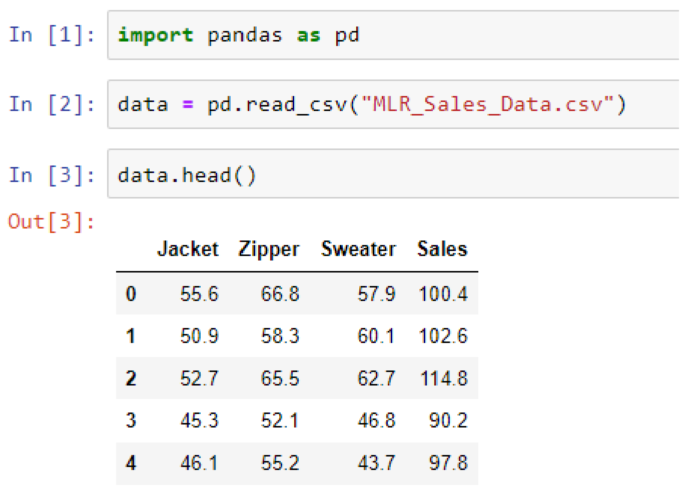
This is the equation of a hyper plane. Remember, a linear regression model in two dimensions is a straight line; in three dimensions it is a plane, and in more than three dimensions, a hyper plane.

METHODOLOGY:

The above model shows how our project step by step works. We have used the Jupyter Notebook as our development environment for the Sales Prediction project. Let's have a look on the details about each of the step involved in the project as displayed in the diagram:

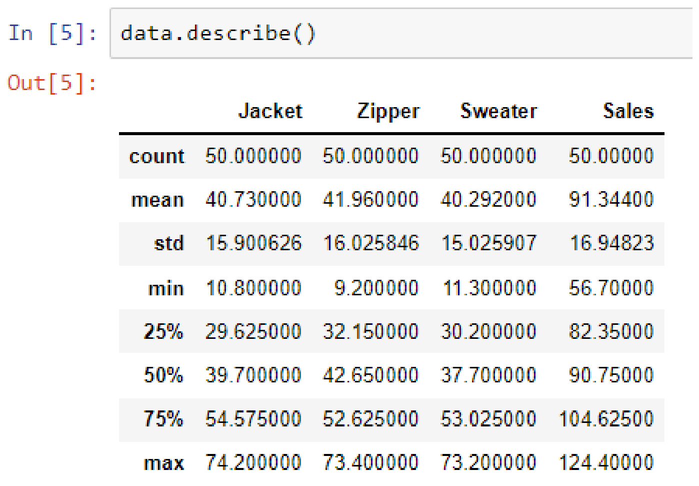
1) Loading Dataset:

In the first step of our project, we have used the Pandas library to load the dataset in the Jupyter Notebook environment. As our dataset file is in the .csv format, the function of read_csv() from Pandas library is relevant for this step. After successful loading of the sales data, we have carried out a brief information about this dataset through using the info() and describe() function from the Pandas library. The info() function shows that how many rows are there in the dataset. Here we have 50 rows. Whereas, through the describe() function, we have taken out some statistical calculations of our numeric features including their mean, median, standard deviations, minimum and maximum values. At the end of this step, we have also revealed the shape of our dataset i.e. (50,4). Here are the results of this step:



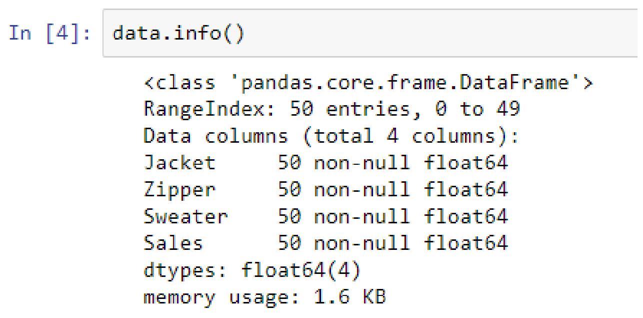
```
In [1]: import pandas as pd

In [2]: data = pd.read_csv("MLR_Sales_Data.csv")

In [3]: data.head()
Out[3]:
```

	Jacket	Zipper	Sweater	Sales
0	55.6	66.8	57.9	100.4
1	50.9	58.3	60.1	102.6
2	52.7	65.5	62.7	114.8
3	45.3	52.1	46.8	90.2
4	46.1	55.2	43.7	97.8



```
In [5]: data.describe()
Out[5]:
```

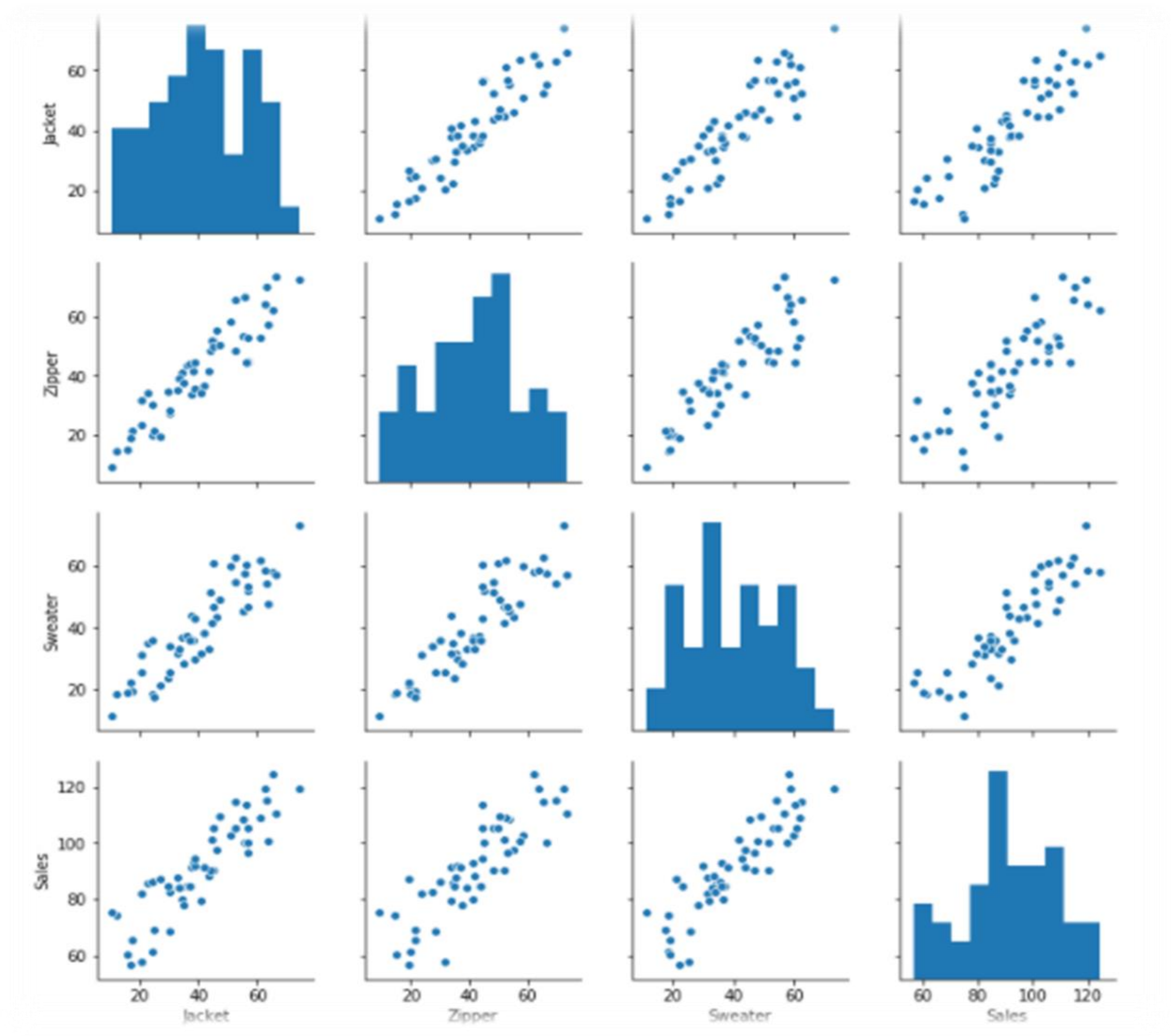
	Jacket	Zipper	Sweater	Sales
count	50.000000	50.000000	50.000000	50.00000
mean	40.730000	41.960000	40.292000	91.34400
std	15.900626	16.025846	15.025907	16.94823
min	10.800000	9.200000	11.300000	56.70000
25%	29.625000	32.150000	30.200000	82.35000
50%	39.700000	42.650000	37.700000	90.75000
75%	54.575000	52.625000	53.025000	104.62500
max	74.200000	73.400000	73.200000	124.40000



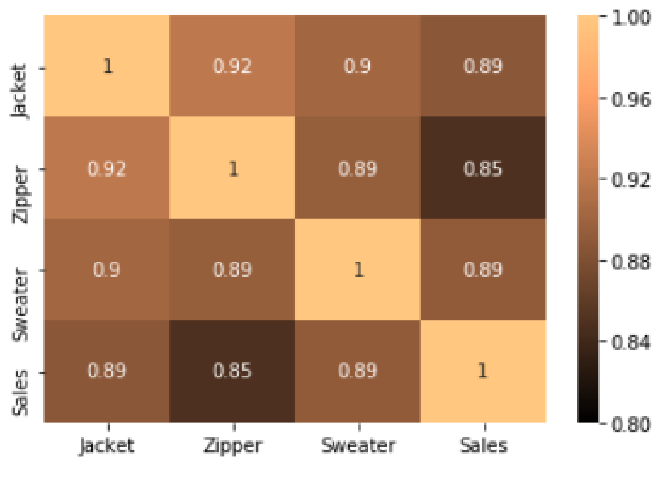
```
In [4]: data.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 50 entries, 0 to 49
        Data columns (total 4 columns):
        Jacket      50 non-null float64
        Zipper      50 non-null float64
        Sweater     50 non-null float64
        Sales       50 non-null float64
        dtypes: float64(4)
        memory usage: 1.6 KB
```

2) Plotting Data:

In the next step which is all about data visualization of our sales data, we have utilized the Seaborn library. Through the function of pairplot, we have shown the relationship among the features in our dataset. Here are the pairplots that were being carried out.



The above pairplot clearly states that the relationship among our features is linear and highly correlated. To check up the correlations among our features, we have developed a heat map matrix through the Seaborn library. Here is the correlation matrix:

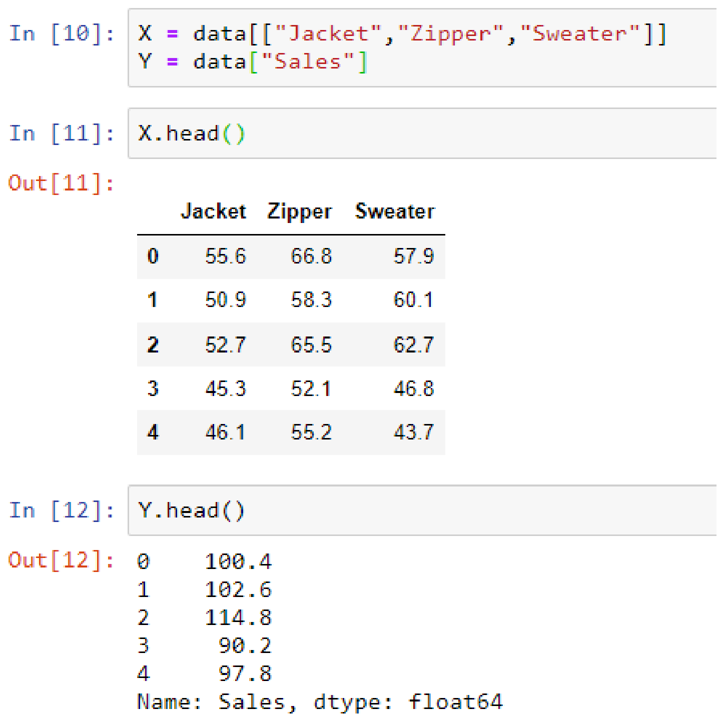


	Jacket	Zipper	Sweater	Sales
Jacket	1	0.92	0.9	0.89
Zipper	0.92	1	0.89	0.85
Sweater	0.9	0.89	1	0.89
Sales	0.89	0.85	0.89	1

The matrix shows that according to our dataset, there is a big influence of every individual product (Jacket, Sweater, and Zipper) on the Total Sales. As the correlation values of three features / products are of 0.85, 0.89 and 0.89 (all in positive), it can be concluded that the dataset is highly correlated with respect to target function overall.

3) Splitting Data In X and Y:

Every regression problem requires a Coefficient (X) as an independent variable and an intercept (Y) as a dependent variable. Here, we have divided our Sales dataset into two parts naming them X and Y. The X variable contains the features whereas the Y variable is our target function as shown in the figure:



```
In [10]: X = data[["Jacket","Zipper","Sweater"]]
         Y = data["Sales"]

In [11]: X.head()

Out[11]:
```

	Jacket	Zipper	Sweater
0	55.6	66.8	57.9
1	50.9	58.3	60.1
2	52.7	65.5	62.7
3	45.3	52.1	46.8
4	46.1	55.2	43.7

```
In [12]: Y.head()

Out[12]: 0    100.4
         1    102.6
         2    114.8
         3     90.2
         4     97.8
         Name: Sales, dtype: float64
```

Note: The head() function in the above figure prints first five rows of the data.

4) Making Train and Test Sets:

In order to fit our data into a Machine Learning model, we need to divide both, X and the Y sets into two parts to get our training and testing sets. As a result, we get x_train, x_test, y_train, and y_test. Here,

- x_train contains the training features extracted from X sets,

- y_train contains the labels/ target values for x_train
- x_test contains the features that have to be evaluated by our model,
- y_test contains targets/ labels of the original test sets that are to be compared with the predicted y instances.

This whole work steps are carried out through the sklearn library's model_selection package which gives a method namely train_test_split() for this purpose. There are some parameters of this method i.e. train_size, and random_state. We have to hard code the percentage of our train sets in the first parameter, while the second one is for setting the shuffle order of the data provided.

For our Sales dataset, we have split our features into training and testing sets through the above discussed method. In addition, we have considered 0.7 (70%) of our data in the training portion, while remaining 30% is for testing. The random state shuffle that we have fixed depicts the 3rd order. So, our training and testing data have shapes as under:

x_train – (35, 4)	x_test – (15, 4)	y_train – (35,)	y_test – (15,)

5) Creating Linear Regression Model and Fitting Train Sets:

From the sklearn library's linear models, we have selected the Linear Regression algorithm and initialized its instance as "lr". There are some parameters for this algorithm to be tuned but, we have preferred all the default values of them for our Sales dataset. After creating the model instance, we have put in the train values (x_train and y_train) in our model through the fit() function. Now that we have fitted the values over the Linear Regression model, it works through finding the best coefficients and intercepts in order to give accurate results for our data. The next step is the prediction phase as our model has been trained.

6) Predicting Test Sets:

Through the predict() function of our prepared model, we have carried out our predictions. We have put x_test as our parameter here, hence, getting our predicted values in the y_pred variable. After saving our predictions, it's time to jump upon the model evaluation and results.

RESULTS AND EVALUATIONS:

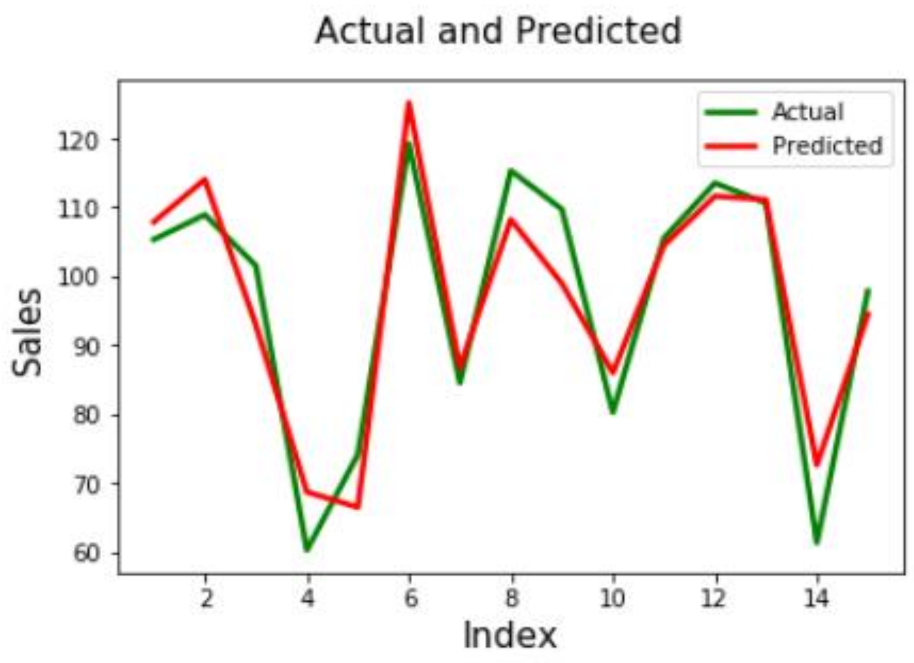
The results and evaluations are bounded by the comparison of the predicted y (y_pred) and original test y (y_test). In order to accomplish this assessment, we have called the evaluation functions from sklearn library's package namely metrics. There are two evaluation functions that we have used in this Sales prediction problem, the one is R2 Score (for accuracy score of our model), and the other is Mean Absolute Error (for loss score of our model). Both of them are relevant for the regression problems.

R2_Score (Accuracy):

The R2 score states, “the proportion of the variance in the dependent variable that is predictable from the independent variable(s).”

The value of R2 Score can vary from 0 to 1 in float. The higher the value of R2 Score, the best our model is. However, the value 1 which depicts 100% accuracy causes the model to overfit and conclusions become invalid.

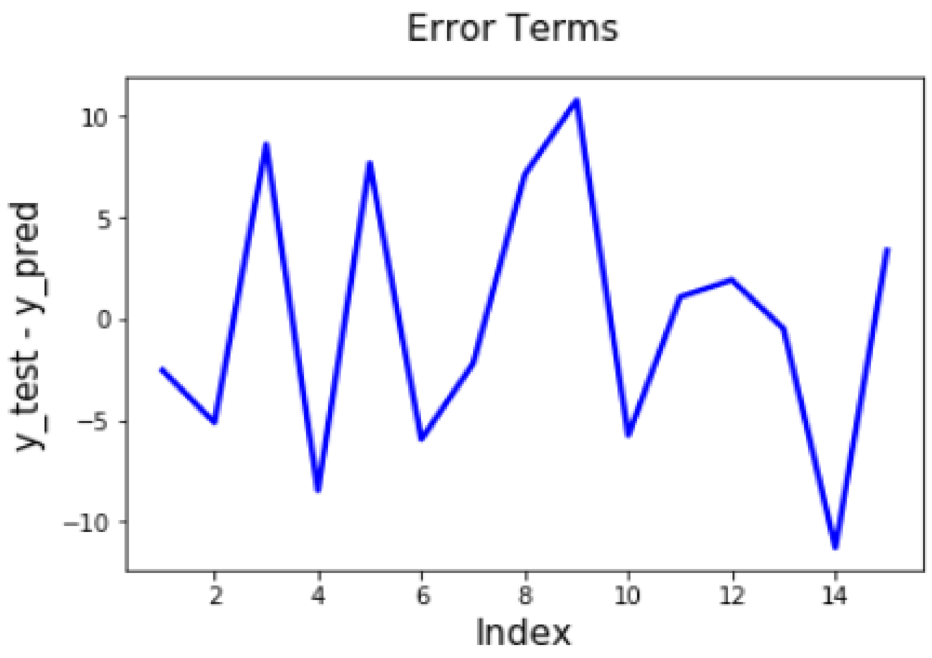
As we predicted the Sales values and evaluated our Linear Regression model, we got R2 Score as 0.8839 or we can say 88.39 %. Here is the graph of the Y_test and Y_pred values comparison:



Mean Absolute Error (Loss Score):

The Mean Absolute Error defines the average of deviation in the original Y values and predicted Y values. It takes the differences as absolute values and gives mean of those absolute differences, thus, putting the loss score of our model in front.

In our Sales prediction model, we have got the MAE loss score as 5.48. Here is the graph which shows the error terms of differences in the values of Y_test and Y_pred:



FINAL WORDS:

We have successfully created a Machine Learning model for our Sales dataset. The optimal accuracy and reduced loss metrics makes it clear that the model is not much weak to predict garbage value or irrelevant output for the unseen data. Although, it was just a short project where we had only three products. This problem can be solved in a generalized way on such a Sales dataset that has more product features and a large number of instances or tuples.