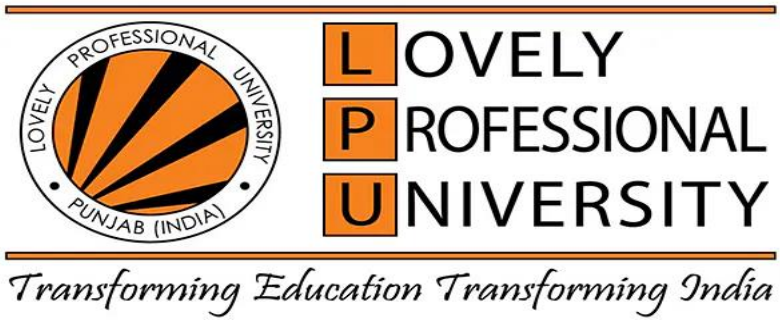


MACHINE LEARNING PROJECT REPORT

on

BIG MART SALES PREDICTION

Submitted by

BOORLA SAI TEJA

Registration Number: 12014428

Program Name: B. tech Data Science (ML and AI)

Under the Guidance of

Mr. VED PRAKASH CHAUBEY

School of Computer Science & Engineering
Lovely Professional University, Phagwara

Big Mart Sales

Platform: This dataset was chosen from Google Kaggle. A well-known website for all the data scientists.

Introduction: This is a Big Mart Sales dataset, a kind of a simulation where you perform extensive data analysis to deliver insights on how the company can increase its profits while minimizing the losses. The dataset provides the product details and the outlet information of the products purchased with their sales value split into a train set and test set.

Train set: It contains 8523 rows.

- It contains the item information with sales value in the Train dataset.

Test set: It contains 5681 rows.

- It contains the Item outlet combinations for which sales need to be forecasted.

Keywords: Big market Sales analysis, Machine Learning, Exploratory Data Analysis, Training the data, Testing the data, Prediction of the data, Regression Models.

Abstract: The Big Mart Sales contains the information of the products which have collected in the year of 2013 sales data for 1559 products across 10 stores in different cities.

Also, certain attributes of each product and store have been defined. And our aim is to build a predictive model and predict the sales of each product at a particular outlet. Using this model, the data will try to predict the output that gives the rise in their sales. The Train set contains 12 columns and Test set contains 11 columns.

It is commonly used for data analysis and visualization, as well as for machine learning and predictive modelling.

Description of the Data:

It is a Data set that contains the overview of the product. It 8523 rows and 12 columns -

- ProductID – Unique ID for the products.
- Weight – Weight of the products.
- FatContent – Fat content present in the product.
- Visibility - Percentage of total display area of all products in a store allocated to particular product.
- ProductType – Category of the product.
- MRP – Maximum Retail Price of the product.
- OutletID – Unique ID for the store.
- EstablishmentYear – Year of establishment of the outlets.
- OutletSize – The area covered of the store.
- LocationType – The type of city where the outlet is located.
- OutletType – The type of the outlet whether it is a grocery store or super market.
- OutletSales – Sales of the products in the outlet.

Methodology: We used a machine-learning approach to predict the price of used mobile phones. First, we performed data pre-processing by handling missing values, coding categorical variables, and scaling numeric variables. We then split the dataset into training and test sets. We trained several machines learning models, including linear regression, decision trees, random forests, and gradient boosting, and evaluated their performance using the mean square error (MSE), root mean square error (RMSE), and R-squared (R2) metrics.

Data Collection:
We have collected the data securely in accordance with an agreed methodology. The procedure for the collected data may differ from client to client and is dependent on the type, quantity, availability and need of data.

Data Cleaning and Processing:

The processed data is sent into a cleaning process so that to find the data is segregated properly or not.

And empty values in the data is filled with the aggregate numerical so that it will become easy for prediction.

Data Prediction:
Using XG boost regression and Linear regression and by finding the R Squared value we will calculate our predictive analysis.

And later they will be evaluated using the data.

Data Visualization Process:
Data Analysed is then further picturized to the customer providing insights of the data and how the variables are classified.

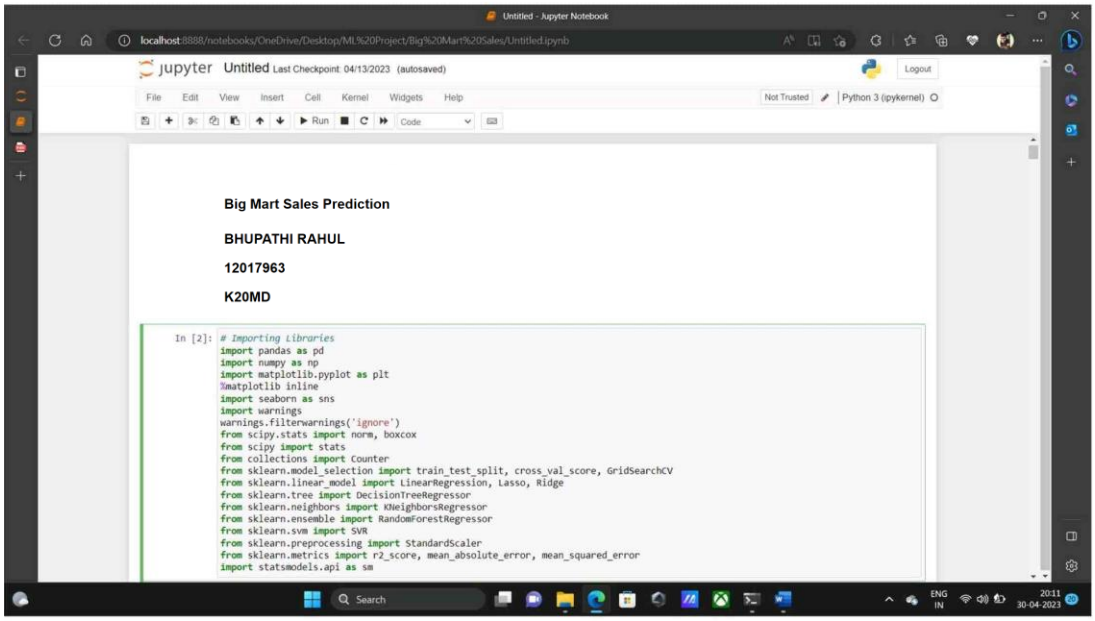
And in this we will see graphs like -

- Count Plot
- Scatter Plot
- Distplot

Corelation graphs which shows the insights of how each variable is connected to another variable.
Distplot contains the combination of both -

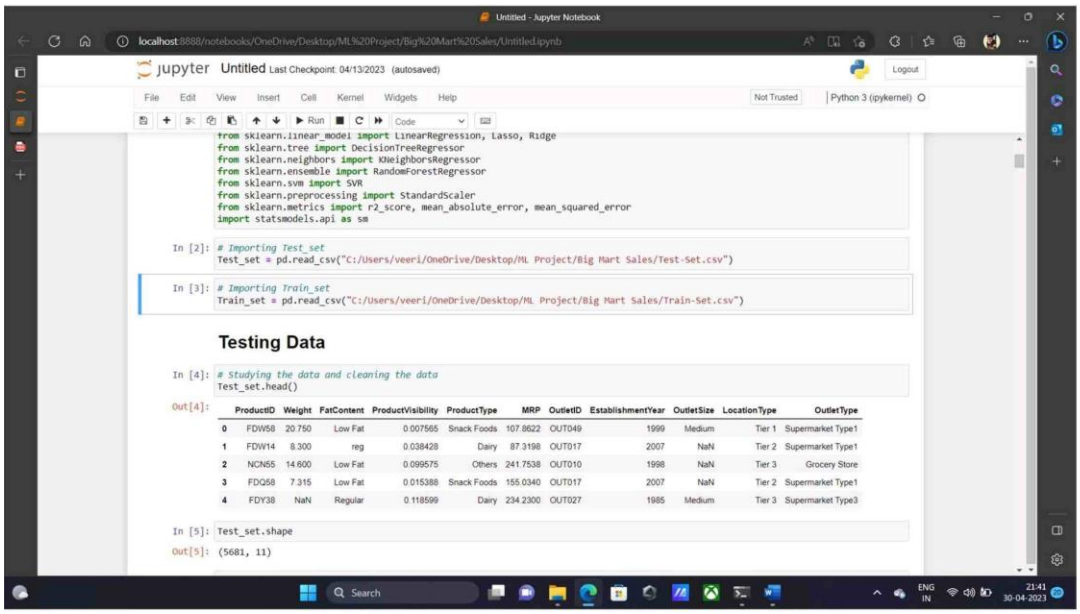
- Hist (Histogram)
- Dist (Density)

Importing Libraries:



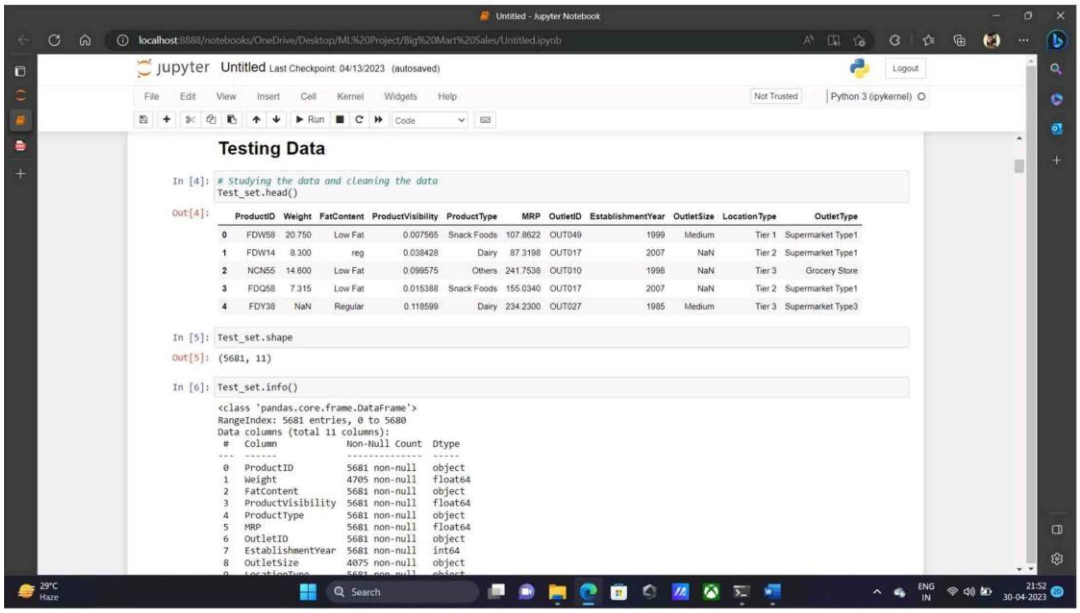
- Importing different python libraries which are used for the prediction of our dataset

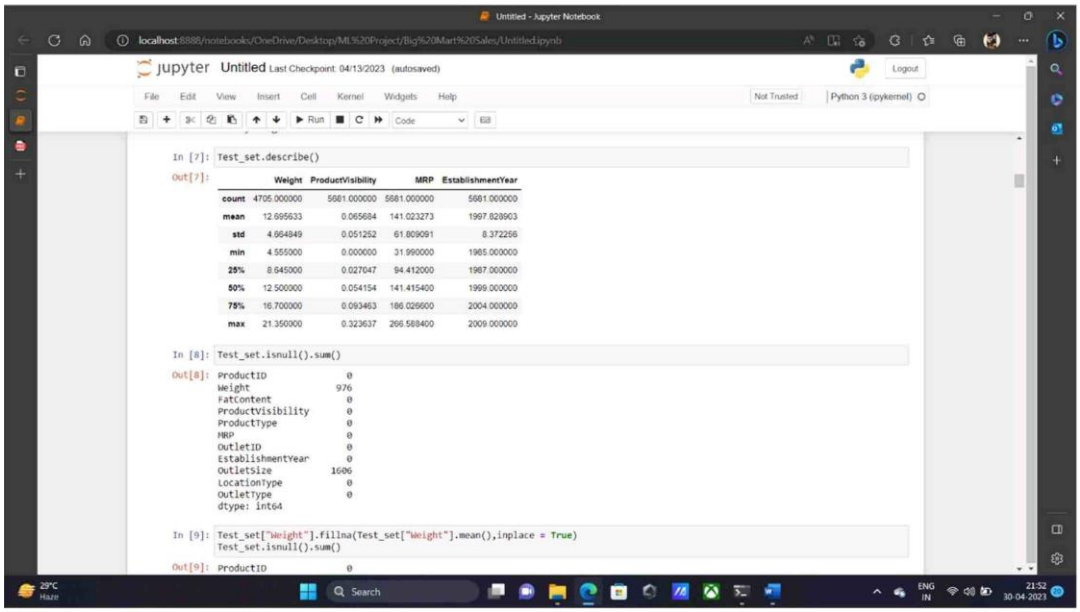
Importing the Test and Train dataset:

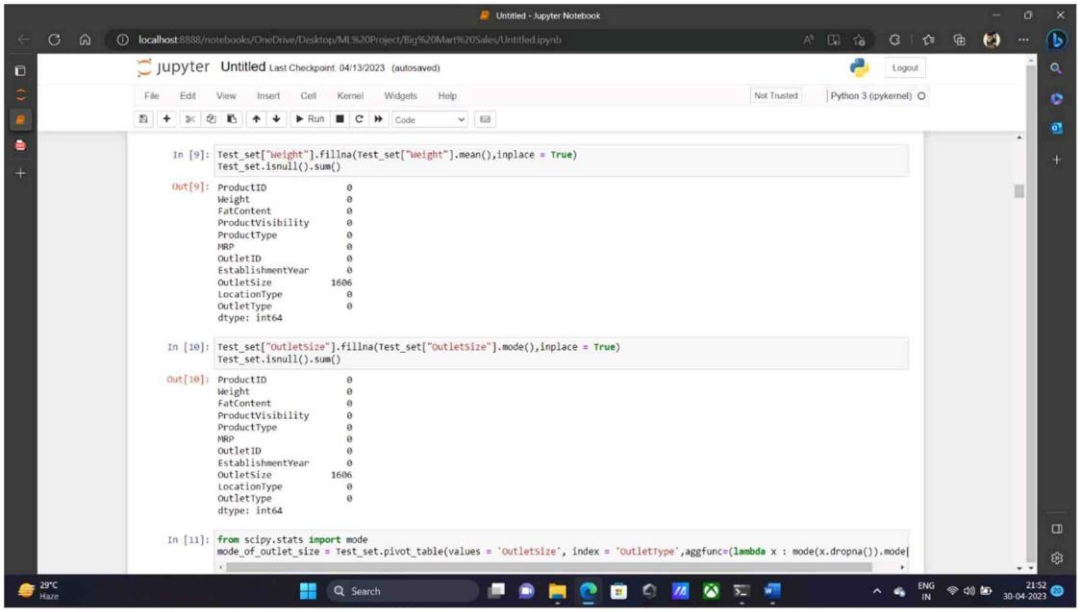


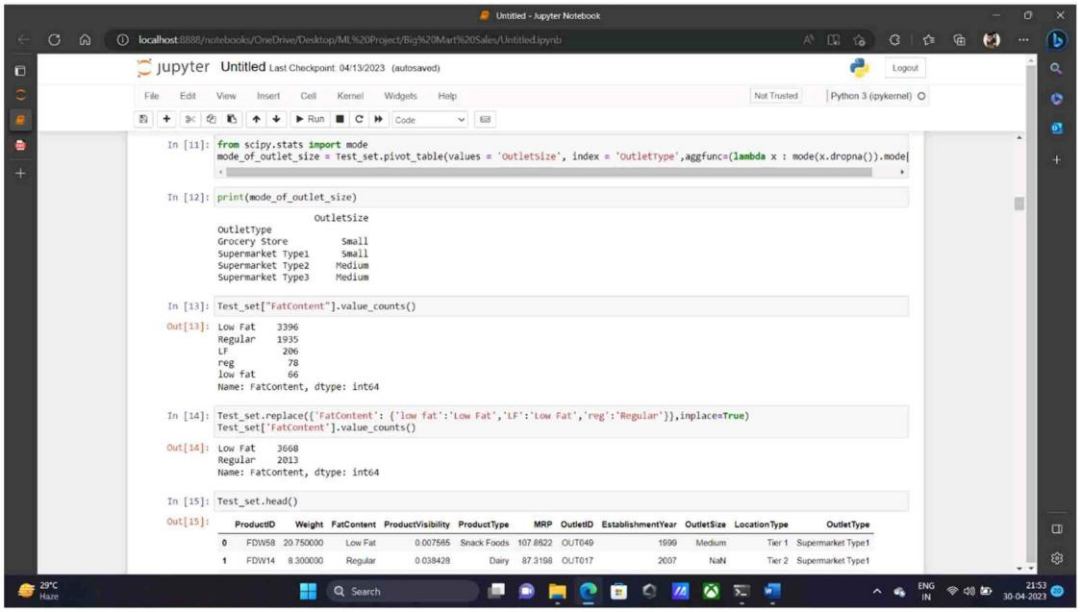
- Importing test and train model datasets into our jupyter notebook so we can analyse our data using python libraries.

Studying and Analysing our Test Dataset:

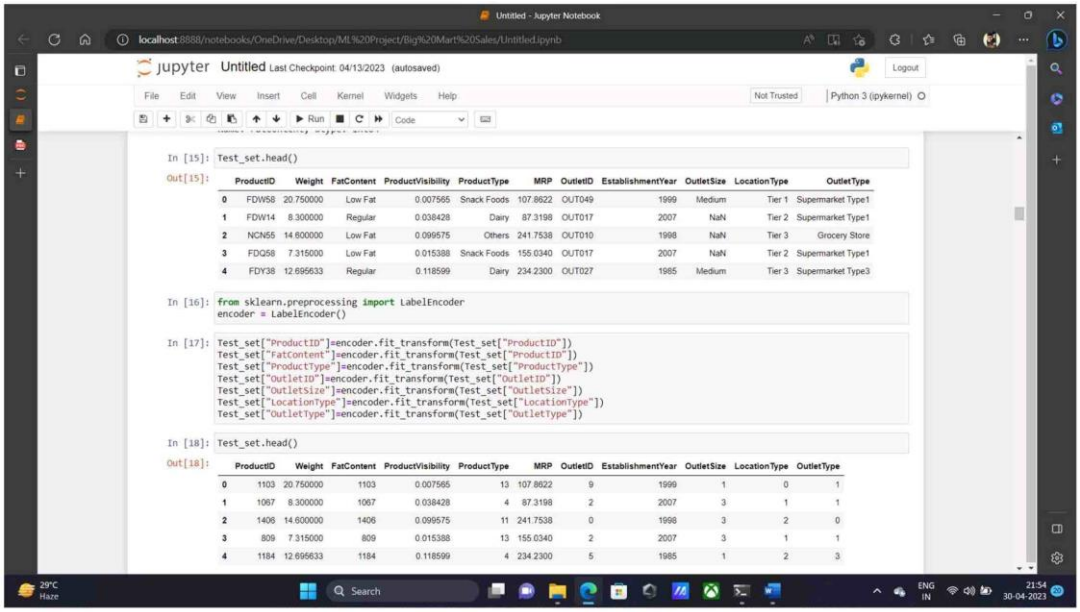








- We analysed the data and studied the whole dataset is about.
- We performed data cleaning method which includes removing null values, removing unwanted columns, removing duplicate values from the dataset.
- And we will perform that the columns which contains string value has changed to numerical value.



```
In [15]: Test_set.head()
```

Out[15]:

	ProductID	Weight	FatContent	ProductVisibility	ProductType	MRP	OutletID	EstablishmentYear	OutletSize	LocationType	OutletType
0	FDW58	20.750000	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1
1	FDW14	8.300000	Regular	0.038428	Dairy	87.3198	OUT017	2007	NaN	Tier 2	Supermarket Type1
2	NCN55	14.600000	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	Tier 3	Grocery Store
3	FDQ58	7.315000	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	Tier 2	Supermarket Type1
4	FDY38	12.695633	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	Tier 3	Supermarket Type3

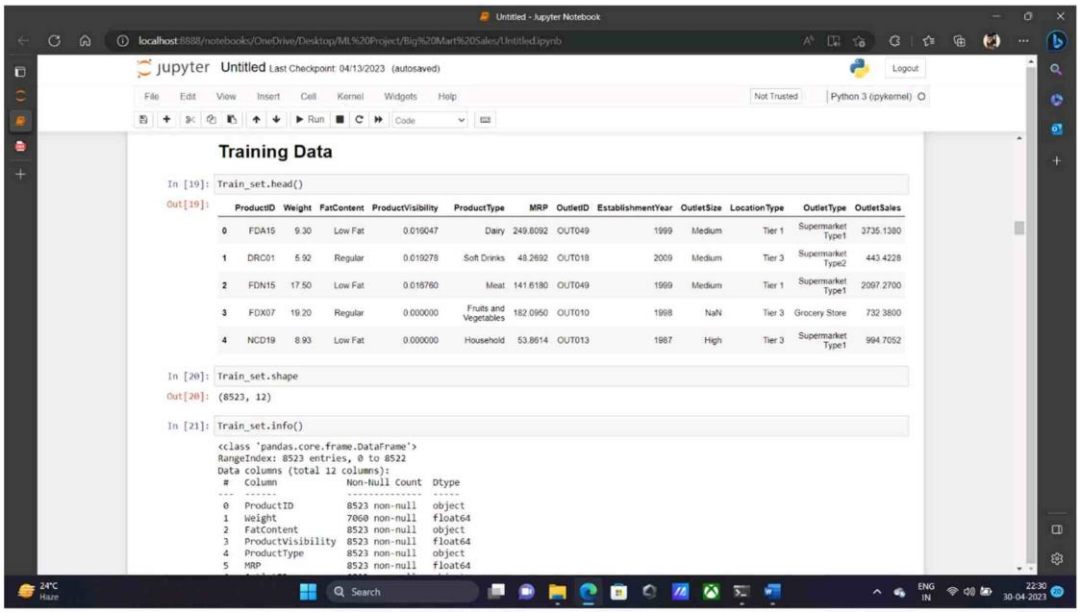
```
In [16]: from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

In [17]: Test_set["ProductID"]=encoder.fit_transform(Test_set["ProductID"])
Test_set["FatContent"]=encoder.fit_transform(Test_set["ProductID"])
Test_set["ProductType"]=encoder.fit_transform(Test_set["ProductType"])
Test_set["OutletID"]=encoder.fit_transform(Test_set["OutletID"])
Test_set["OutletSize"]=encoder.fit_transform(Test_set["OutletSize"])
Test_set["LocationType"]=encoder.fit_transform(Test_set["LocationType"])
Test_set["OutletType"]=encoder.fit_transform(Test_set["OutletType"])

In [18]: Test_set.head()
```

Out[18]:

	ProductID	Weight	FatContent	ProductVisibility	ProductType	MRP	OutletID	EstablishmentYear	OutletSize	LocationType	OutletType
0	1103	20.750000	1103	0.007565	13	107.8622	9	1999	1	0	1
1	1067	8.300000	1067	0.038428	4	87.3198	2	2007	3	1	1
2	1406	14.600000	1406	0.099575	11	241.7538	0	1998	3	2	0
3	809	7.315000	809	0.015388	13	155.0340	2	2007	3	1	1
4	1184	12.695633	1184	0.118599	4	234.2300	5	1985	1	2	3

- We can see that the string values in the column has changed to numerical values so that we can perform our analysis clearly and based on that we can do the prediction.

Studying and Analysing our Train Dataset:



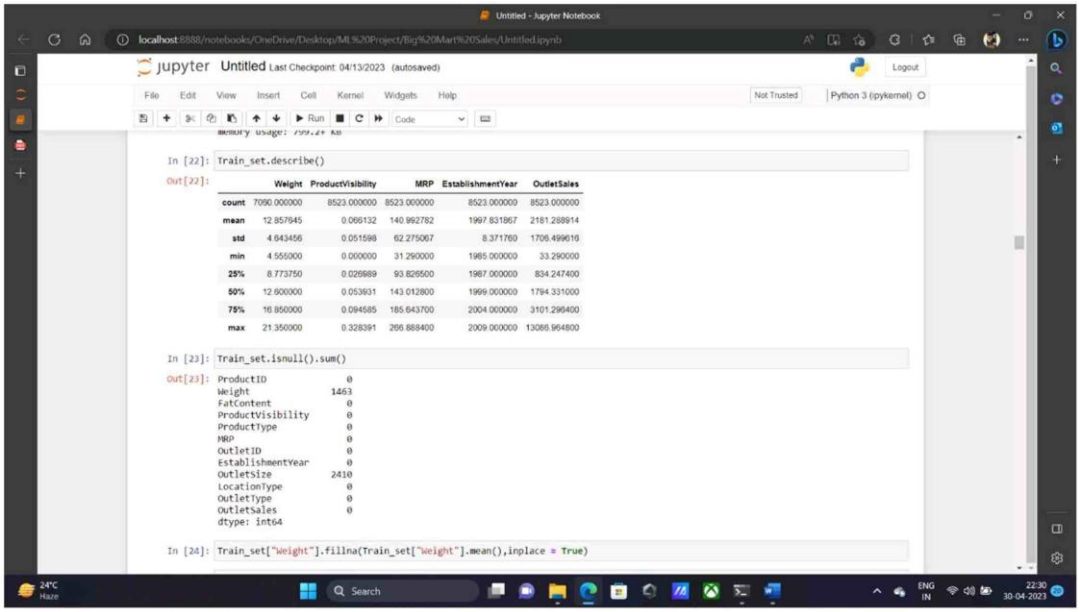
Training Data

```
In [19]: Train_set.head()
```

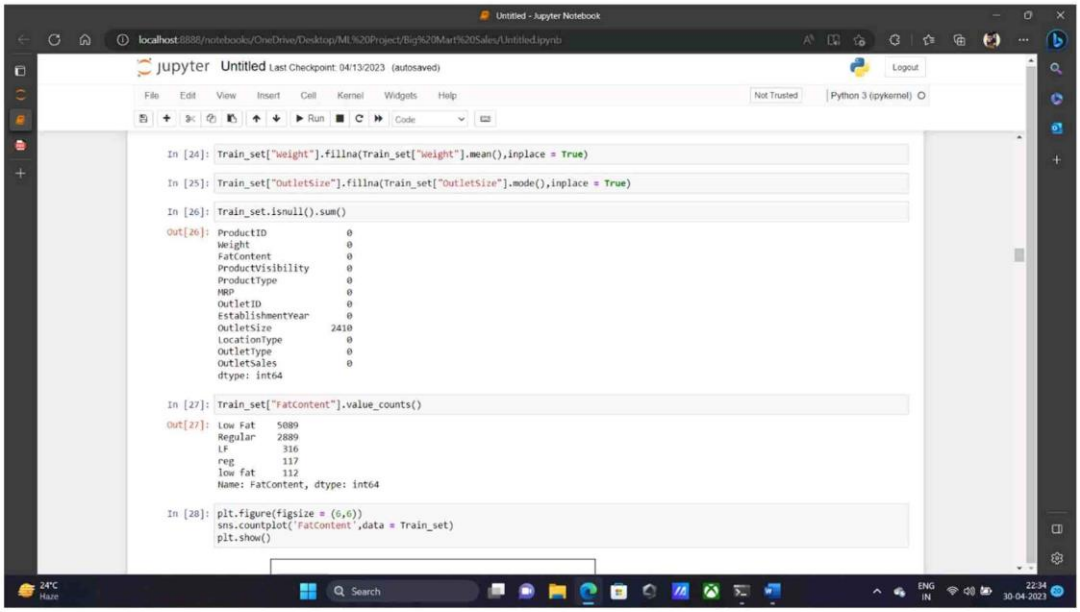
Out[19]:

	ProductID	Weight	FatContent	ProductVisibility	ProductType	MRP	OutletID	EstablishmentYear	OutletSize	LocationType	OutletType	OutletSales
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

```
In [20]: Train_set.shape
Out[20]: (8523, 12)

In [21]: Train_set.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ProductID          8523 non-null   object
 1   Weight             7060 non-null   float64
 2   FatContent         8523 non-null   object
 3   ProductVisibility  8523 non-null   float64
 4   ProductType        8523 non-null   object
 5   MRP                8523 non-null   float64
```

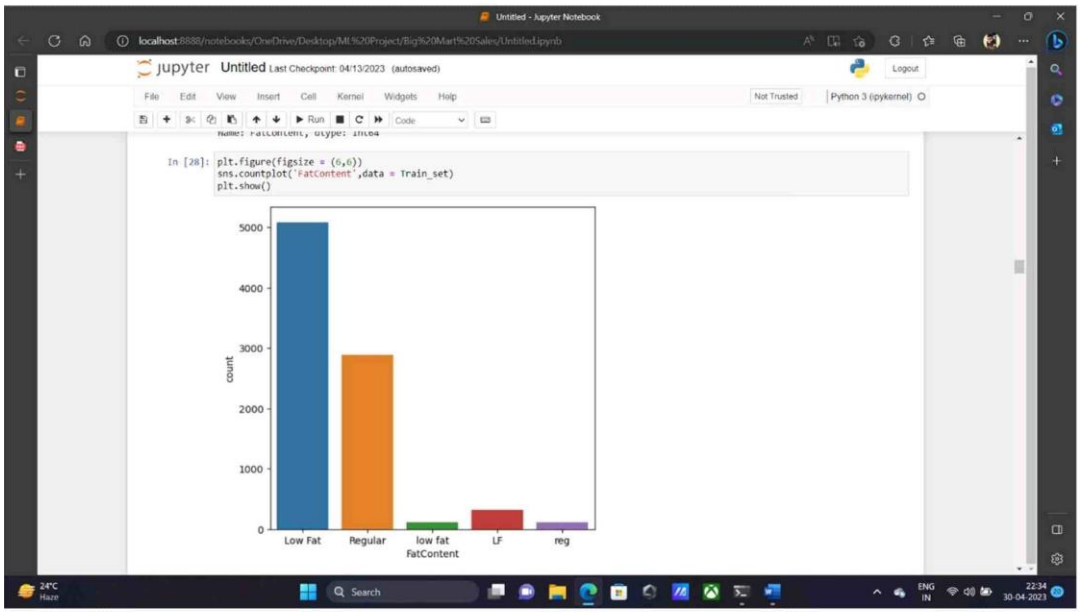


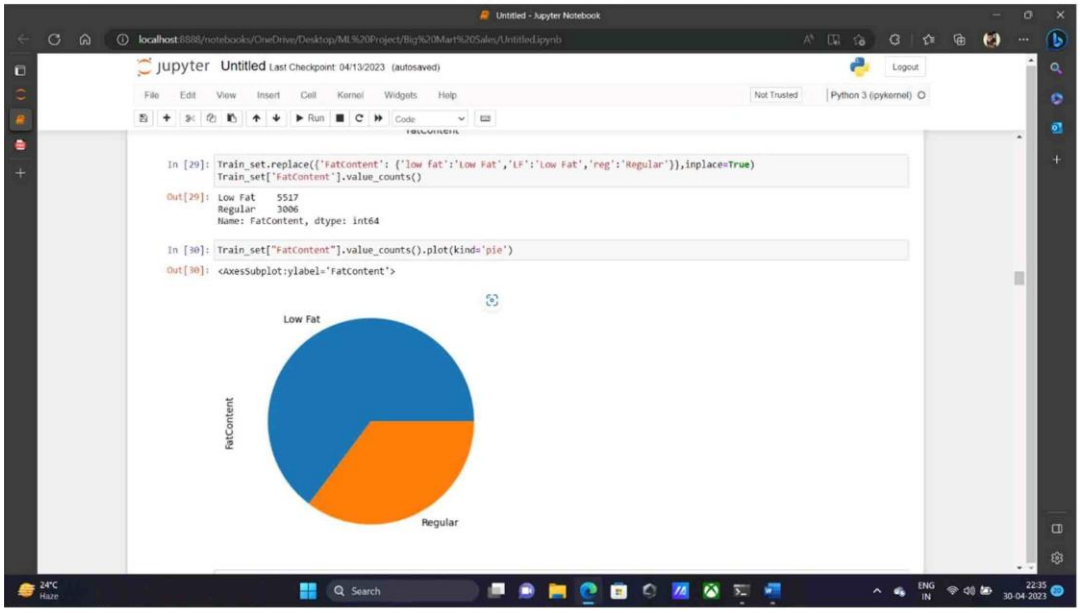
	Weight	ProductVisibility	MRP	EstablishmentYear	OutletSales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800



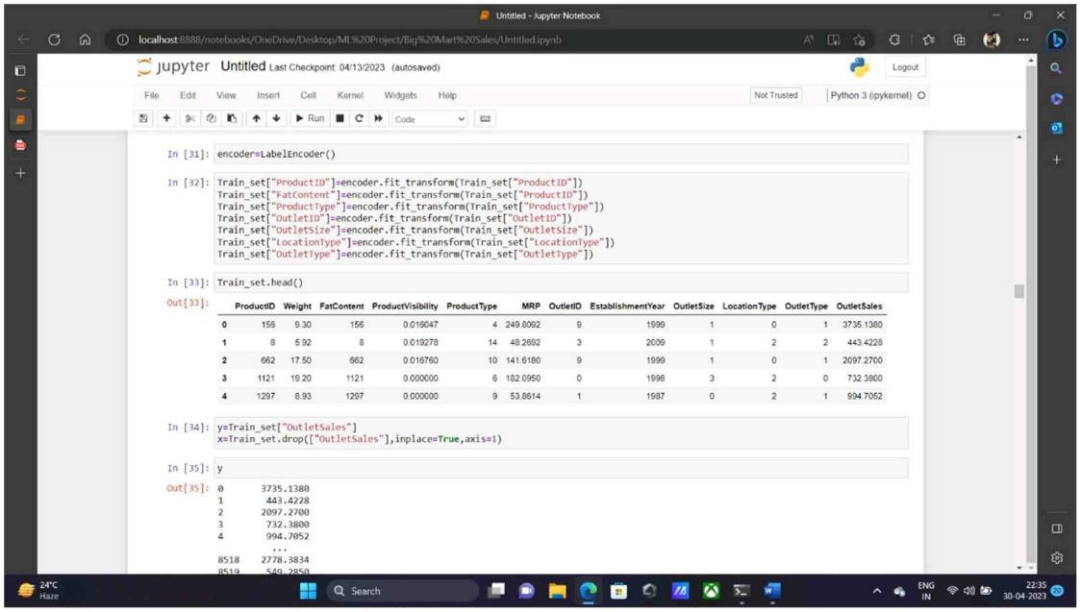
- We performed the same analytical statistics which we have performed for our test dataset.
- And now we will perform the count plot for Fat Content variable so that we can clearly observe that how many of them are duplicated values.

- And after that we will remove duplicate values and combine them into one value and visualize it using pie chart.





- We can see that the Fat Content variable values in our train data are different from that of test data.



```
In [31]: encoder=LabelEncoder()

In [32]: Train_set["ProductID"]=encoder.fit_transform(Train_set["ProductID"])
         Train_set["FatContent"]=encoder.fit_transform(Train_set["ProductID"])
         Train_set["ProductType"]=encoder.fit_transform(Train_set["ProductType"])
         Train_set["OutletID"]=encoder.fit_transform(Train_set["OutletID"])
         Train_set["OutletSize"]=encoder.fit_transform(Train_set["Outletsize"])
         Train_set["LocationType"]=encoder.fit_transform(Train_set["LocationType"])
         Train_set["OutletType"]=encoder.fit_transform(Train_set["OutletType"])

In [33]: Train_set.head()
```

Out[33]:

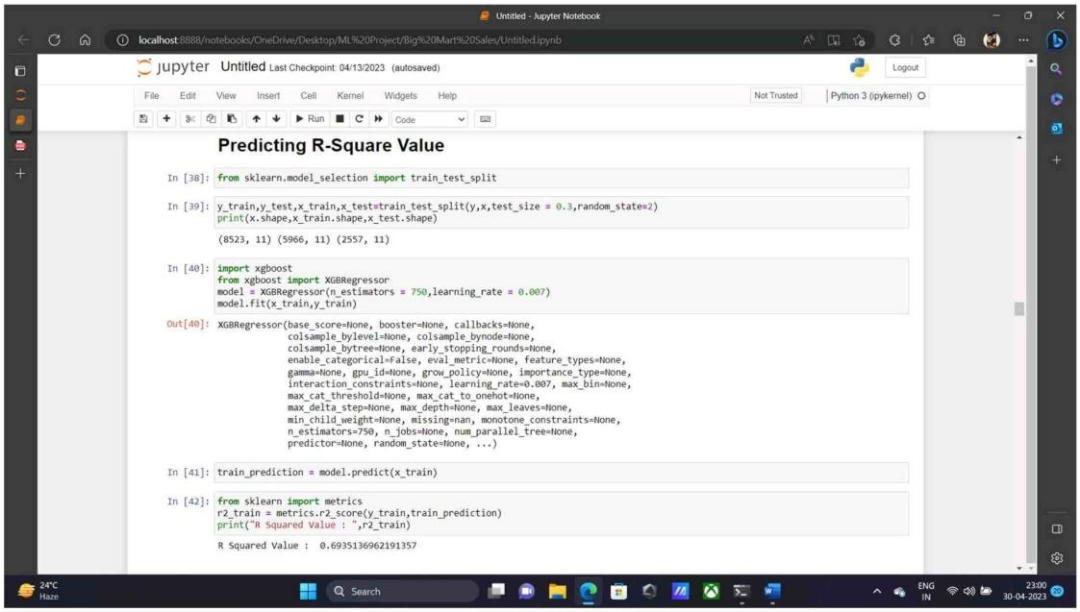
	ProductID	Weight	FatContent	ProductVisibility	ProductType	MRP	OutletID	EstablishmentYear	OutletSize	LocationType	OutletType	OutletSales
0	156	9.30	156	0.016047	4	249.8092	9	1999	1	0	1	3735.1380
1	8	5.92	8	0.019278	14	48.2692	3	2009	1	2	2	443.4228
2	662	17.50	662	0.016760	10	141.6180	9	1999	1	0	1	2097.2700
3	1121	19.20	1121	0.000000	6	182.0950	0	1998	3	2	0	732.3800
4	1297	8.93	1297	0.000000	9	53.8614	1	1987	0	2	1	994.7052

```
In [34]: y=Train_set["OutletSales"]
         x=Train_set.drop(["OutletSales"],inplace=True,axis=1)

In [35]: y
```

```
Out[35]: 0       3735.1380
         1        443.4228
         2       2097.2700
         3        732.3800
         4        994.7052
                   ...
         8518    2778.3834
         8519     549.2850
```

- Converted the string values into numerical values for our predictive analysis.
- And removed the outlet sales value and stored it aside so that we can calculate our R-Squared value which is used for predictive models.

Predicting R-Squared value:



Predicting R-Square Value

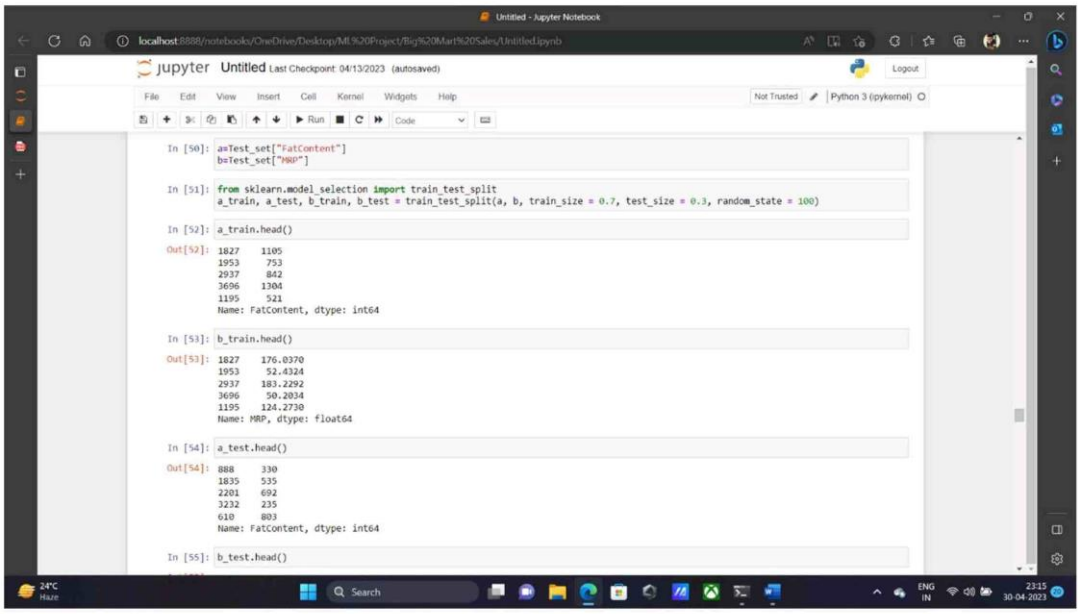
```
In [38]: from sklearn.model_selection import train_test_split

In [39]: y_train,y_test,x_train,x_test=train_test_split(y,x,test_size = 0.3,random_state=2)
         print(x.shape,x_train.shape,x_test.shape)

         (8523, 11) (5966, 11) (2557, 11)

In [40]: import xgboost
         from xgboost import XGBRegressor
         model = XGBRegressor(n_estimators = 750,learning_rate = 0.007)
         model.fit(x_train,y_train)
```

```
Out[40]: XGBRegressor(base_score=None, booster=None, callbacks=None,
                      colsample_bylevel=None, colsample_bynode=None,
                      colsample_bytree=None, early_stopping_rounds=None,
                      enable_categorical=False, eval_metric=None, feature_types=None,
                      gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
                      interaction_constraints=None, learning_rate=0.007, max_bin=None,
                      max_cat_threshold=None, max_cat_to_onehot=None,
                      max_delta_step=None, max_depth=None, max_leaves=None,
                      min_child_weight=None, missing=nan, monotone_constraints=None,
                      n_estimators=750, n_jobs=None, num_parallel_tree=None,
                      predictor=None, random_state=None, ...)
```

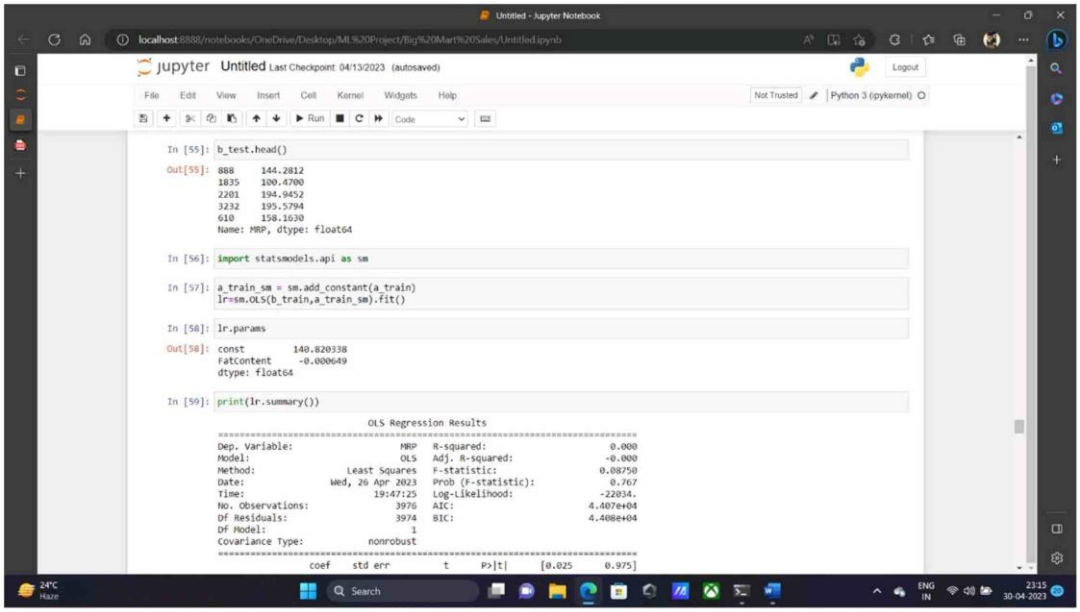
```
In [41]: train_prediction = model.predict(x_train)

In [42]: from sklearn import metrics
         r2_train = metrics.r2_score(y_train,train_prediction)
         print("R Squared Value : ",r2_train)

         R Squared Value :  0.6935136962191357
```

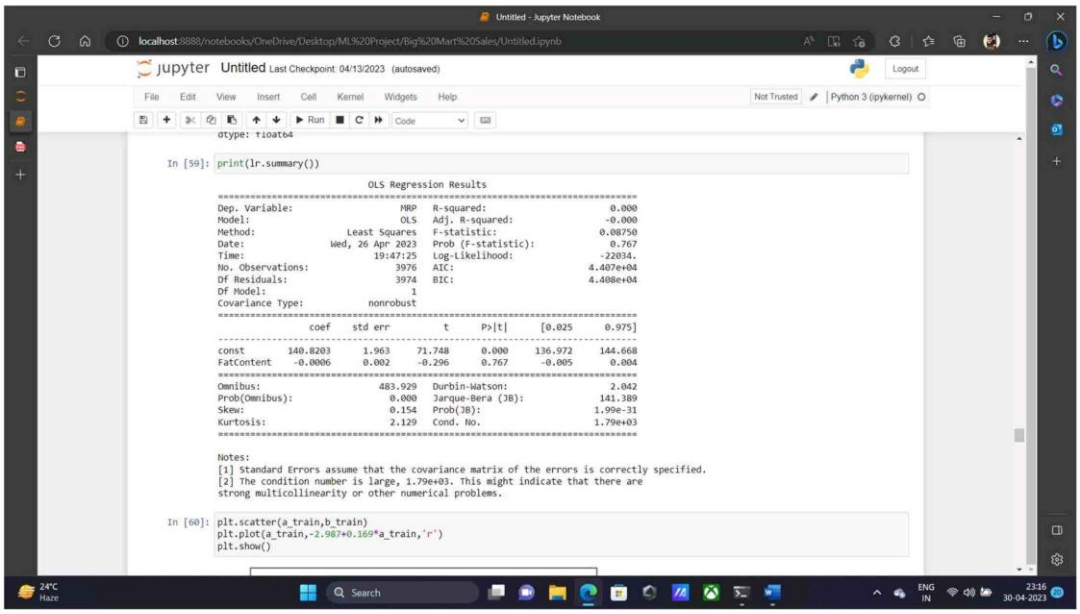
- By Using the above data cleaning and data prediction for train set and test set we found the R-Squared value.
- With R-Squared value we can define that it is a goodness-of-fit measure for linear regression models. And from this value we can see the best fit line for our linear regression model which is used for the prediction analysis.

Performing Linear Regression model on Test data:

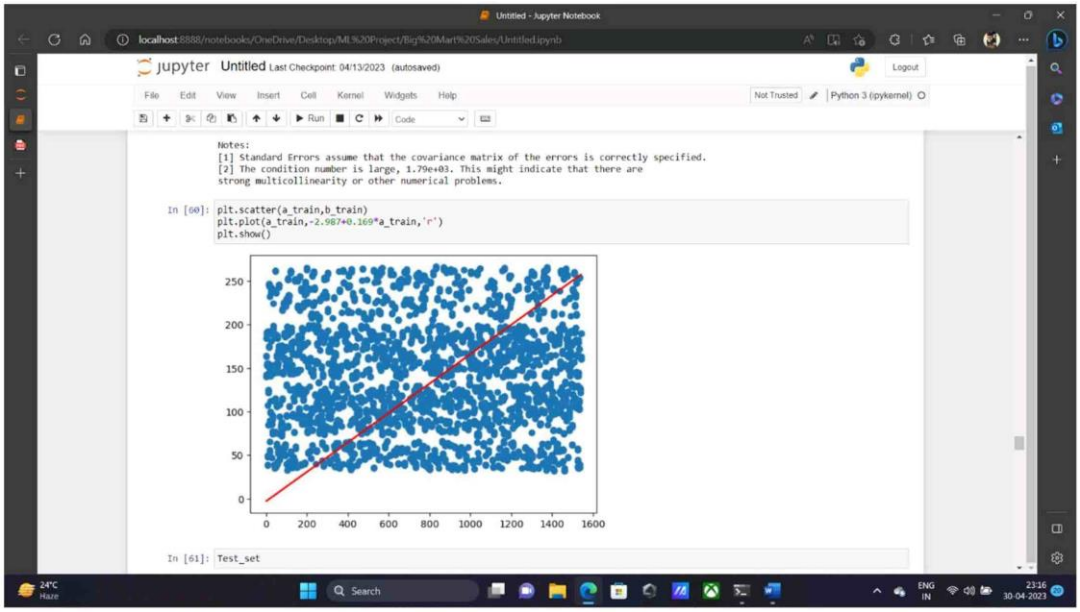


- For our prediction in Linear Regression model, we will use our Fat Content variable and MRP Sales of the dataset.
- By this we can see the best fit line for our Linear Regression model which is used for predictive analysis.
- And we will be seeing T-Test value, P-Value, F-Score value and all other statistical values for our Linear Regression model.



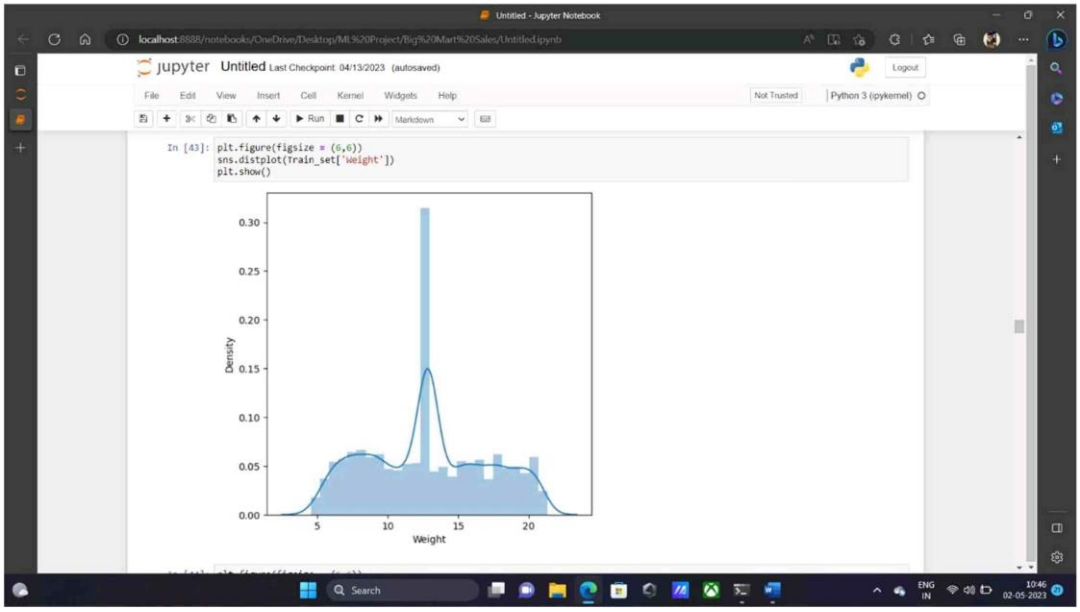


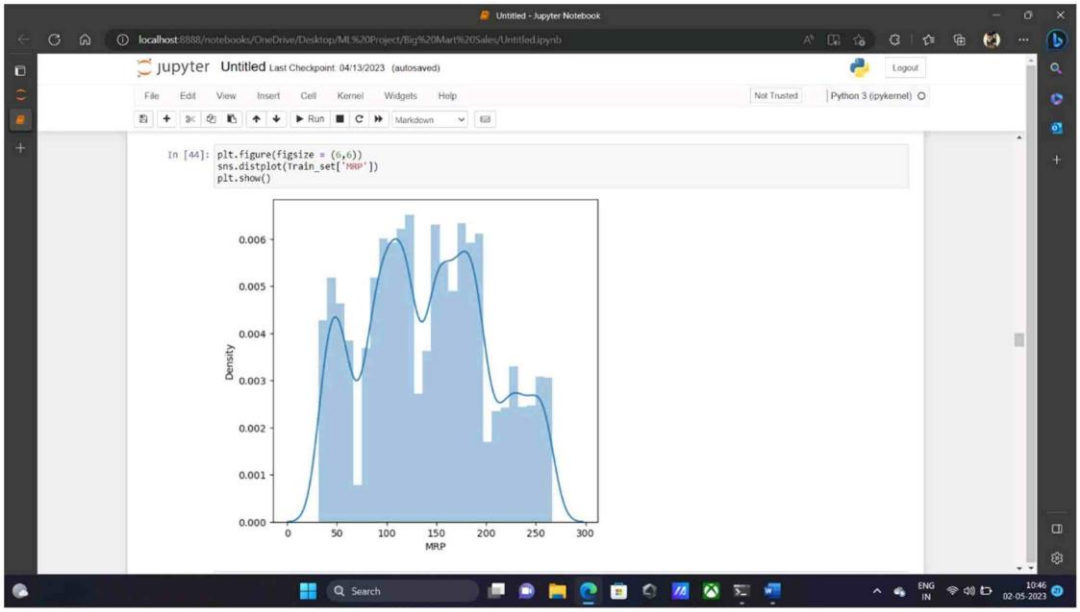
- From the above we can see all the statistical values which are used for Hypothesis Testing –
- If the P-value is greater than 0.05 then we will accept the null hypothesis.
- If the P-value is less than 0.05 then we will accept the alternate hypothesis.

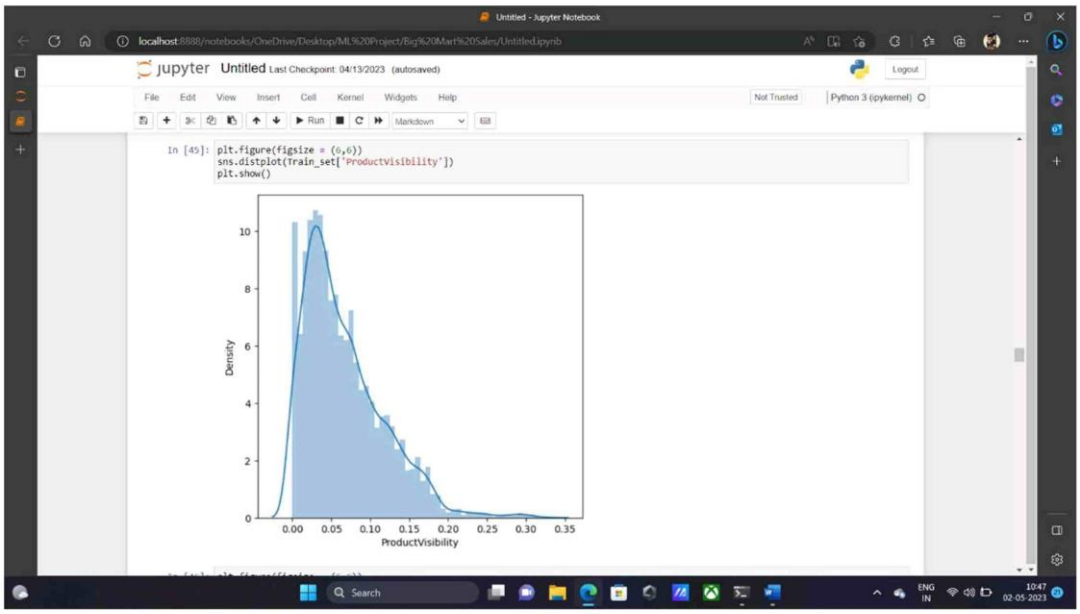


- We can see that best fit line for our Linear Regression model.
- From that we can say that the MRP Sales increases if there is decrease in Fat Content in the products.

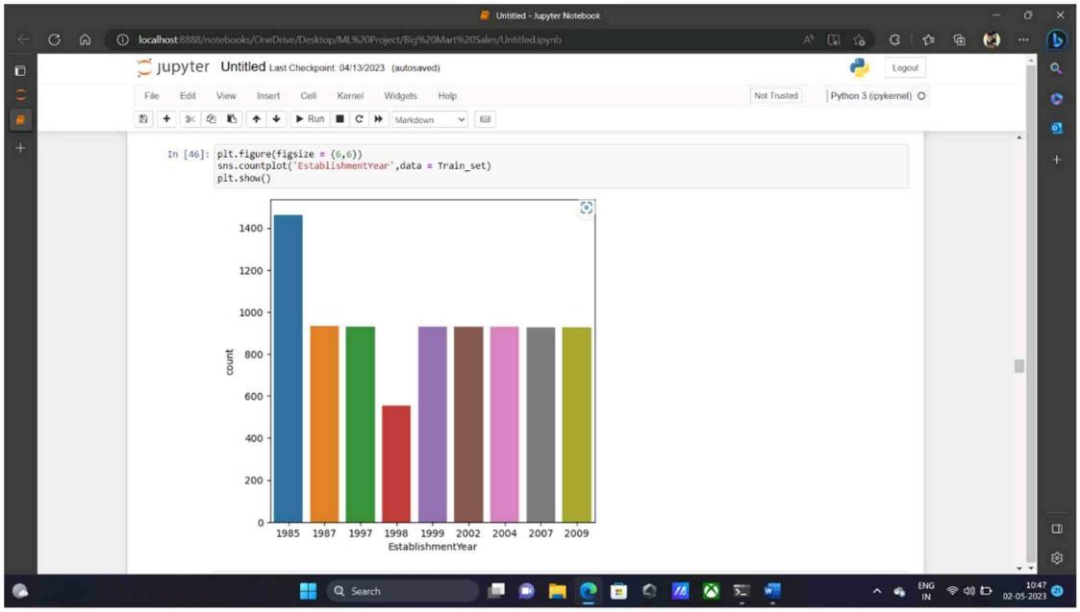
Visualization of the dataset:



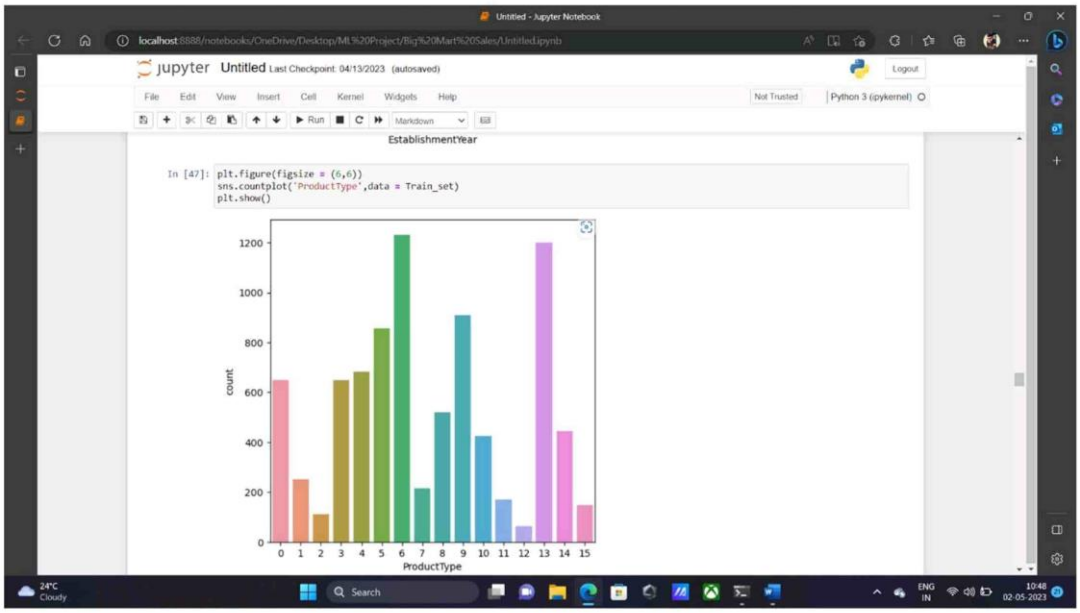




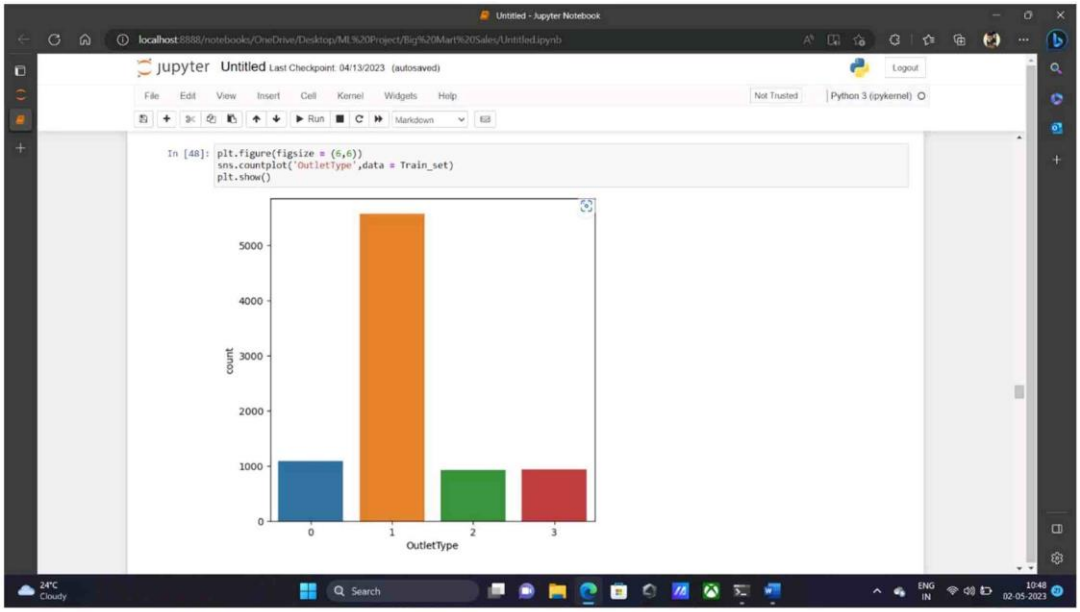
- These three graphs are the distplot which represents the density and hist of the given variable.
- And here we checked how the density is distributed throughout the data for the Weight, MRP, Product Visibility.
- We can observe that MRP is fluctuating here and when we check for product visibility it is decreasing rapidly.



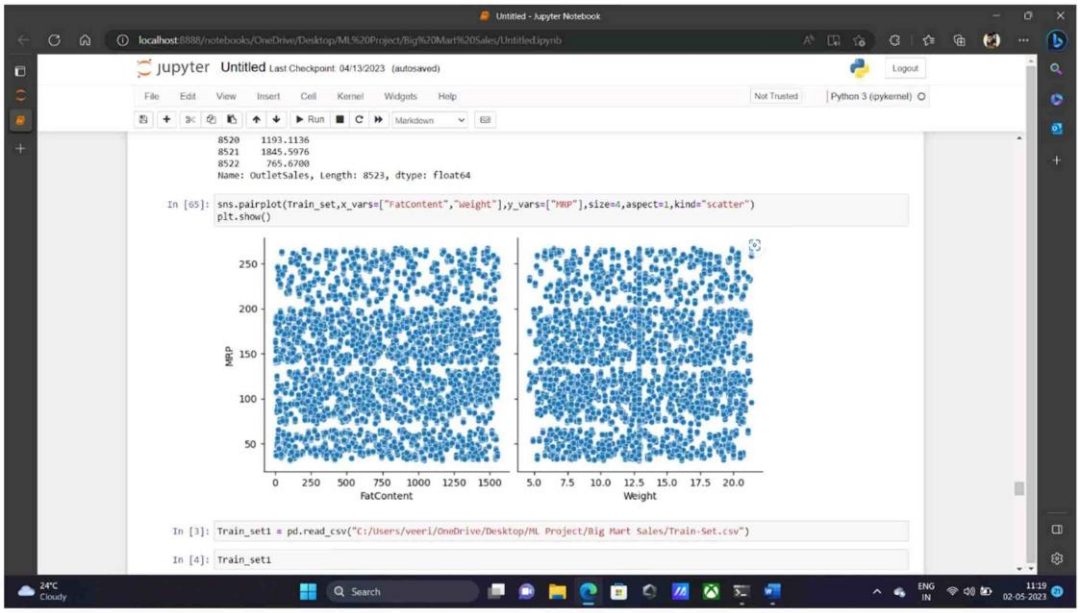
- From the count plot of the establishment year we can see that the most of the outlets are established in the year of 1985 and from preceding year they are establishing equal number of outlets except in the year of 1998.



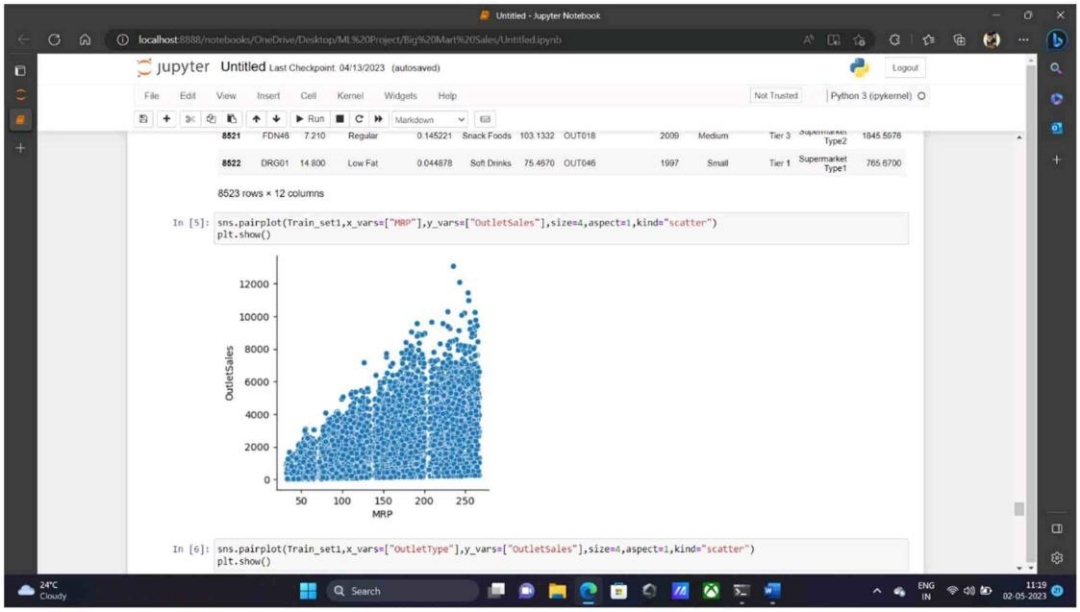
- From the count plot of product type, we can see that the 6, 13 are having the greater number of sales as compared to others.



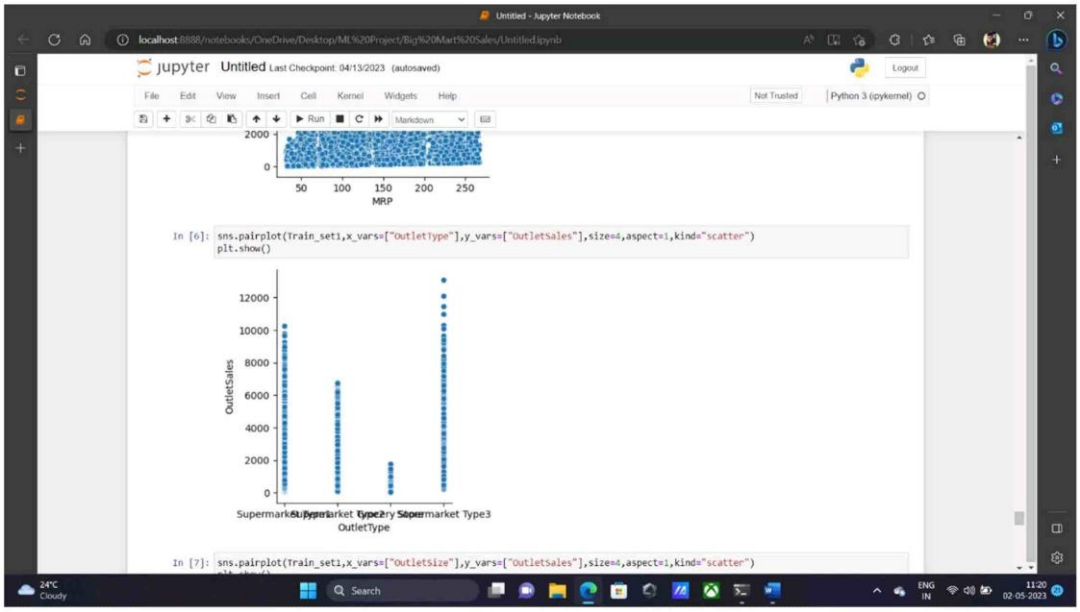
- From the count plot of outlet type we can see that the type 1 is greater than other outlets which means supermarkets are selling more than the grocery stores.

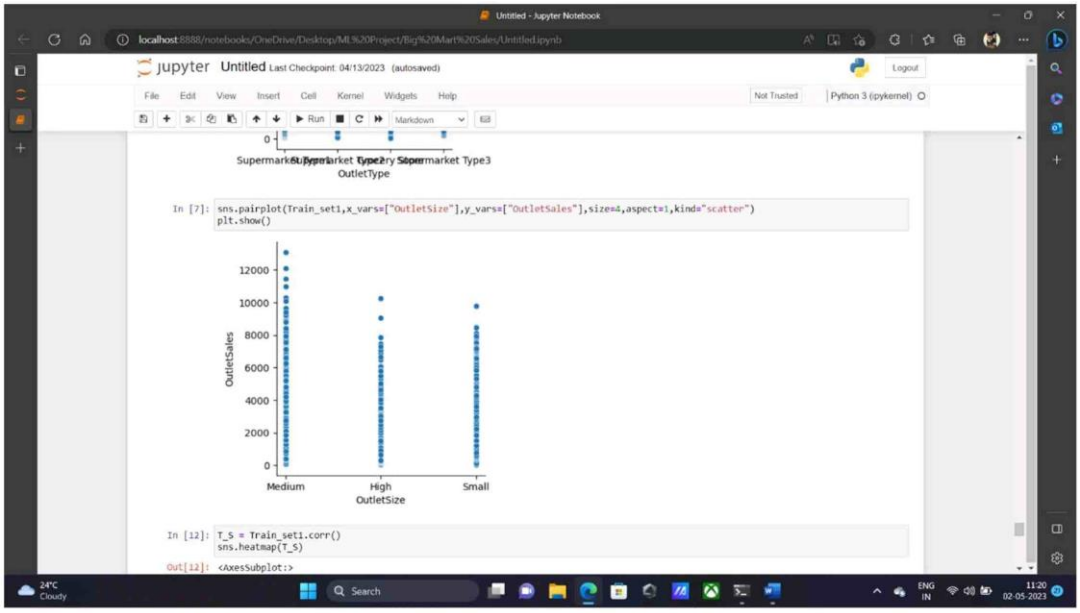


- Scatter plot for Fat Content and weight for train data depending upon the MRP sales and from this we can see it is also same as the test data.

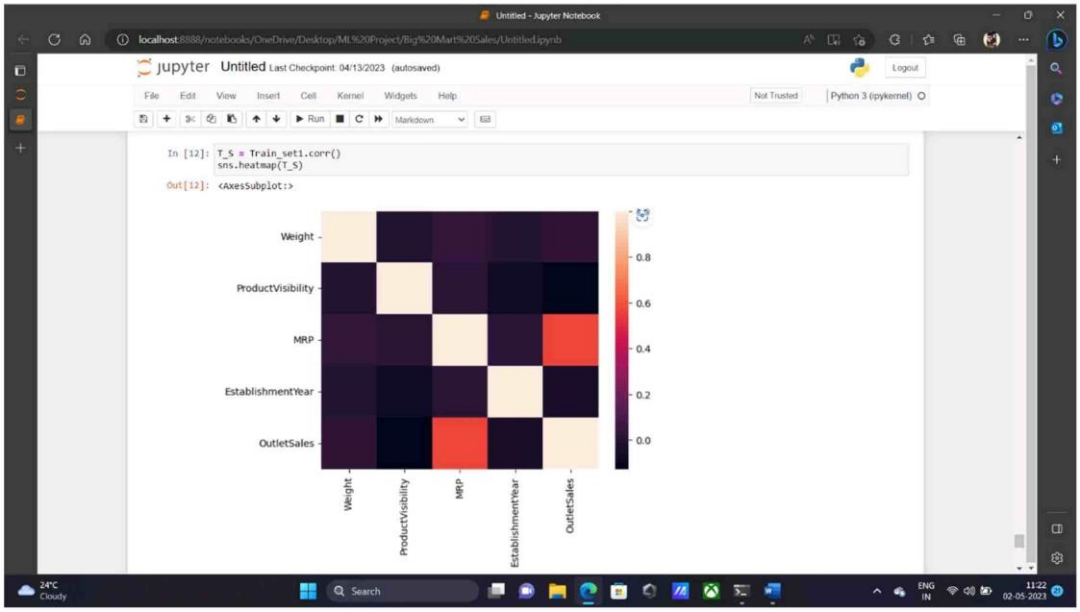


- From the above scatter plot we have checked the outlet sales depending upon the MRP sales and we can observe that the outlet sales are increasing rapidly in increase of the MRP sales.





- We can see here that the outlet sales do depend on the size of the outlet and here medium outlet size has a greater number of sales than high and small outlet size and we can say it does not depend whether the size is big or small it depends on the location.



Conclusion:

We have applied four algorithms XGBoost, Linear Regression and Decision Tree. From the results, we can conclude that among all the four algorithms XGBoost has the highest accuracy of 61.14% when distinguished together. Hence, we can say that XGBoost is the better algorithm for efficient sales analysis. This methodology is primarily used by shopping marts, groceries, Brand outlets etc. The data analysis applied to the predictive machine learning models provides a very effective way to manage sales, it also generously contributes to better decisions and plan strategies based on future demands.This approach is very much encouraged in today's world since it aids many companies, enterprises, researchers and brands for outcomes that lead to management of their profits, sales, inventory management, data research and customer demand.

From the above Statistical Analysis, Virtualization, Linear Regression Model we can conclude that our data is normally distributed.

And the null hypothesis is accepted.

We can also conclude that the outlet sales for the train data depends upon the Location, Outlet Type, Fat Content and MRP price of the product

We can clearly see that sales are high for the products which has low fat content.

And outlet sales do not depend upon the outlet size but it will depend upon the outlet type.

The big mart with medium size has more sales.

The big mart with small size has more size than big outlet size.

From the Linear Regression graph we can clearly see that the MRP increases in the future when there is increase in products which has low Fat Content.

Due to increase in MRP the Outlet Sales will be automatically increased.

References:

Find Open Datasets and Machine Learning Projects | Kaggle

- Gopal Behere, Neeta Nain (2019). Grid Search Optimization (GSO) Based Future Sales Prediction for Big Mart. 2019 International Conference on Signal-Image Technology & Internet-Based Systems (SITIS).

- Chandel, A. Dubey, S. Dhawale and M. Ghuge,;Sales Prediction System using Machine Learning; International Journal of Scientific Research and Engineering Development, vol. 2, no. 2, pp. 1-4, 2019. [Accessed 27 January 2020].

- Heramb Kadam, Rahul Shevade, Prof. Deven Ketkar, Mr.

  Sufiyan Rajguru (2018). A Forecast for Big Mart Sales Based on

  Random Forests and Multiple Linear Regression. (UEDR).