****

**NAME OF THE PROJECT**

**“**Big Data Mart Sales Analysis”

**Submitted by:**

**Leena chatterjee**

**ACKNOWLEDGMENT:**

* I have taken efforts in this project(preparing this blog). However, it would not have been possible without the kind support and help of each individual of DATA TRAINED organizations. I would like to extend my sincere thanks to all of them.
* I am highly indebted to all team of Data trained for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

**Bibliography**:

* <https://www.grammarly.com/>
* <https://www.google.com/>
* <https://in.zapmetasearch.com/>
* https://projects.datatrained.com/

**INTRODUCTION**

Problem Statement

Big Data Mart Sales Problem

The data scientists at Big Mart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store.

Using this model, Big Mart will try to understand the properties of products and stores which play a key role in increasing the sales of their products.\

The dataset includes two files:

- bigdatamart\_Train.csv: Use this file for the model building purpose.

- bigdatamart\_Test.csv: Use this file for getting predictions from the trained model.

The data scientists at Big Mart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store.

Using this model, Big Mart will try to understand the properties of products and stores which play a key role in increasing sales.

**Conceptual Background of the Domain Problem**

Data science is the field where we can predict the probability. Here basically we need to analyse a predictive model and find out the sales of each product at a particular store. Using this model, Big Mart will try to understand the properties of products and stores which play a key role in increasing the sales of their products

In this dataset we have two type of sub dataset. Those two dataset named as 1) Train dataset 2) Test dataset

Basic understanding of predicting a model using train dataset and test dataset

* Step 1---. Importing data, cleaning, handling missing values, skewness, outliers, standardization (all if needed)
* Step 2---. Building model, checking accuracy, find best model, cross validation and grid search cv and saving the model.

TRAIN part:

For TRAIN part we should be completing both step 1 and step 2. and save your model.

TEST part:

For test part we need to perform all the activities what have done with TRAIN data in step 1 only.

Now need to load saved model again which we have saved from train data and use the TEST data for predicting the values( output of prediction sales of the products present in big mart’s store)

In this dataset those variables are ====🡺

**Train dataset columns**:

* Item\_Outlet\_Sales---This is the target variable , as we found that it is continuous features we will predict the model using regression analysis
* Item Identifier--- It is unique code of identifying the product .In this data set its features is Length: 1559, dtype: int64
* Item Weight—In this columns Item weight is defined .it is also having continuous features
* Item\_Fat\_Content--- We are having 4 different type of category here in this column (Low Fat 5089 Regular 2889 LF 316 reg 117 low fat 112)
* Item Visibility—It is showing item Visibility which is continuous features
* Item Type'--- 'This is categorical column. We have category we mentioned below

(Ex--Fruits and Vegetables, Snack Foods , Household ,Frozen Foods , Dairy Canned ,Baking Goods ,Health and Hygiene ,Soft Drinks , Meat, Breads, Hard Drinks ,Others ,Starchy Foods ,Breakfast ,Seafood)

* Item\_MRP' - 'Item\_MRP' is basically item prize of each product, It is having continuous features

* Outlet Identifier',--  is a categorical column .here this columns content different outlet code
* Outlet\_Establishment\_Year—This is categorical column, here datas are categorized into 9 parts

(1985 1463

1987 932

1999 930

1997 930

2004 930

2002 929

2009 928

2007 926

1998 555)

* Outlet Size', -- In this dataset we have 3 different outlet size Medium , high , small .In this column we have NAN values as well which we will remove in next steps
* Outlet\_Location\_Type---------- Outlet\_Location\_Type we have 3 category of columns

(Ex- Tier1, Tire2, Tire-3)

* Outlet Type—we have 4 category in this column

(Ex-Supermarket Type1 Grocery Store Supermarket Type3 Supermarket Type2 )

Test dataset columns:

In this column we are having only input features, label (output) which is item outlet sales not present

* Item\_Outlet\_Sales---This is the target variable , as we found that it is continuous features we will predict the model using regression analysis
* Item Identifier--- It is unique code of identifying the product .In this data set its features is Length: 1559, dtype: int64
* Item Weight—In this columns Item weight is defined .it is also having continuous features
* Item\_Fat\_Content--- We are having 4 different type of category here in this column (Low Fat 5089 Regular 2889 LF 316 reg 117 low fat 112)
* Item Visibility—It is showing item Visibility which is continuous features
* Item Type'--- 'This is categorical column. We have category we mentioned below

(Ex--Fruits and Vegetables, Snack Foods , Household ,Frozen Foods , Dairy Canned ,Baking Goods ,Health and Hygiene ,Soft Drinks , Meat, Breads, Hard Drinks ,Others ,Starchy Foods ,Breakfast ,Seafood)

* Item\_MRP' - 'Item\_MRP' is basically item prize of each product, It is having continuous features

* Outlet Identifier',--  is a categorical column .here this columns content different outlet code
* Outlet\_Establishment\_Year—This is categorical column, here datas are categorized into 9 parts

(1985 1463

1987 932

1999 930

1997 930

2004 930

2002 929

2009 928

2007 926

1998 555)

* Outlet Size', -- In this dataset we have 3 different outlet size Medium , high , small .In this column we have NAN values as wwell which we will remove in next steps
* Outlet\_Location\_Type---------- Outlet\_Location\_Type we have 3 category of columns

( Ex- Tier1, Tire2, Tire-3)

* Outlet Type—we have 4 category in this column

( Ex-Supermarket Type1 Grocery Store Supermarket Type3 Supermarket Type2 )

**In this Model analysis we have imported Train data as df and test data as ds**

**Mathematical/ Analytical Modeling of the Problem**

supervised learning uses labelled input and output data Supervised learning (SL) is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.[ It infers a function from *labelled training data* consisting of a set of *training examples*.[In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances

Here our dataset output is continuous value which is part of supervised learning so we will analyse with Regression (Linear Regression)

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables. The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regressions

**Motivation for the Problem Undertaken**

We need to analyse a predictive model and find out the sales of each product at a particular store. Using this model, Big Mart will try to understand the properties of products and stores which play a key role in increasing the sales of their products

We study this model so this will help us to analyse any real world sales prediction example – if any running business product industry related wants to predict sales of year based on records like average sales, product MRP, product content will help us to find predictive sales so that business will not run in loss

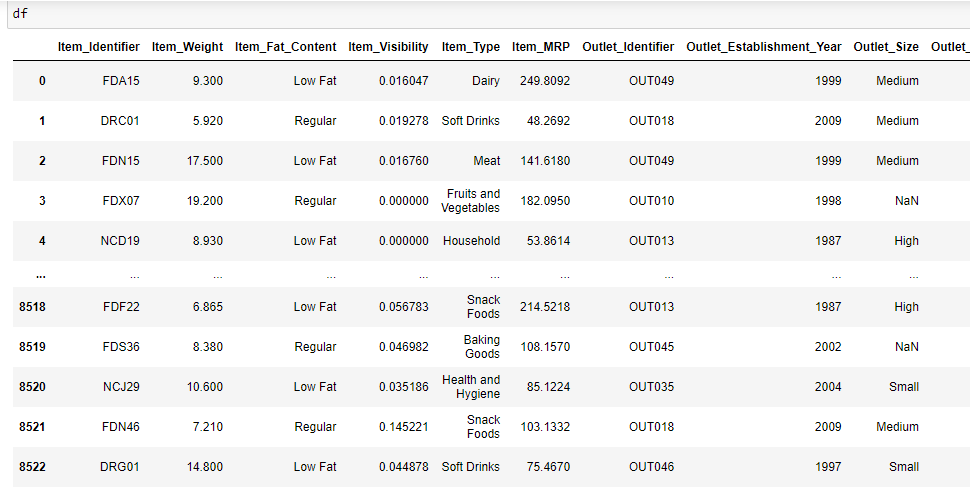
As we worked with real time data , we have gained knowledge that what are challenges has to face while working with real domain data( heavy data set ) , sometimes some information is uncertain so using this experience I believe we can work better on next project and that is being the best motivation behind this blog

**Data Sources and their formats**

Here we found this data from Github (we believed it is real time data collected by github)

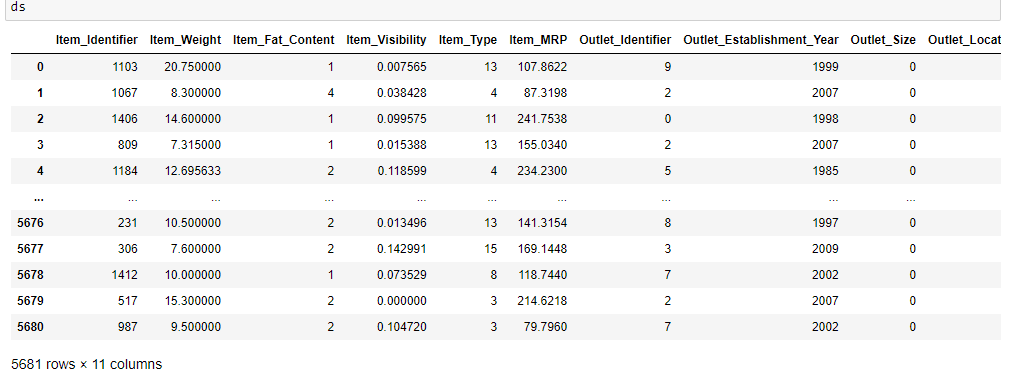
As we mentioned earlier we have two subset train data df and test data as ds

df:

****

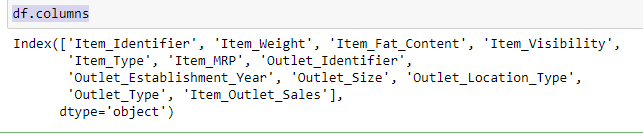
df having 8523 rows and 12 columns

ds:



Observation--ds having 5681 rows and 11 column.

Columns in df :



Types of Each column

df. Types provides us all the information of df.columns:

Item Identifier object

Item Weight float64

Item\_Fat\_Content object

Item\_Visibility float64

Item\_Type object

Item\_MRP float64

Outlet\_Identifier object

Outlet\_Establishment\_Year int64

Outlet\_Size object

Outlet\_Location\_Type object

Outlet\_Type object

Item\_Outlet\_Sales float64

dtype: object

**Null value detection :**

df.isna ().sum () using this command we can find all the null values present in dataset

Item Identifier 0

Item Weight 1463

Item\_Fat\_Content 0

Item\_Visibility 0

Item\_Type 0

Item\_MRP 0

Outlet\_Identifier 0

Outlet\_Establishment\_Year 0

Outlet\_Size 2410

Outlet\_Location\_Type 0

Outlet\_Type 0

Item\_Outlet\_Sales 0

**From here we found that Item Weight Outlet\_Size having null values so now we will calculate percentage of null value**

df.isna ().sum ()/len (df) --🡪 This is command to found percentage of null value

**Calculation is:**

Item Identifier 0.000000

Item Weight 0.171653

Item\_Fat\_Content 0.000000

Item\_Visibility 0.000000

Item\_Type 0.000000

Item\_MRP 0.000000

Outlet\_Identifier 0.000000

Outlet\_Establishment\_Year 0.000000

Outlet\_Size 0.282764

Outlet\_Location\_Type 0.000000

Outlet\_Type 0.000000

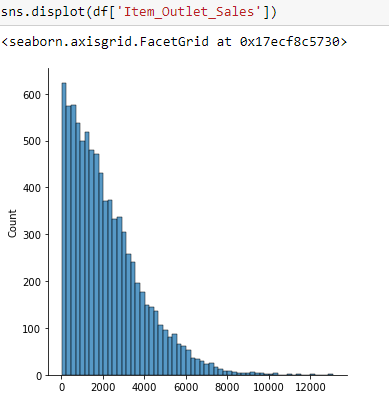
Item\_Outlet\_Sales 0.000000

Observation – Hence we found that having data loss of 17% and 28% , As data loss not more than 60% we will not drop we will replace by mean , median or mode as per these features we will replace null values after EDA analysis

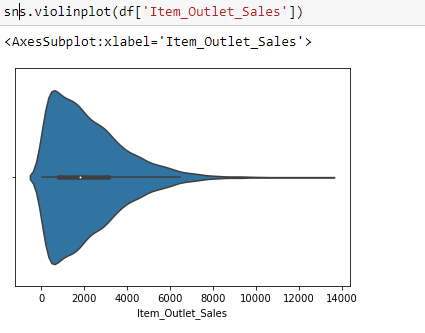
**EDA analysis :**

Univariate analysis:

First we will check target variable data analysis

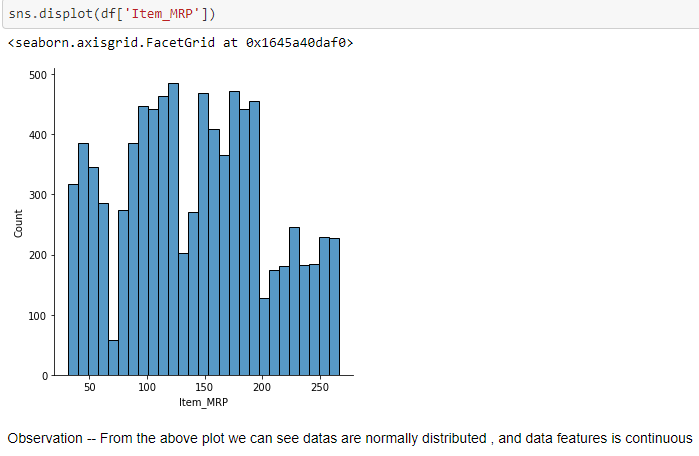


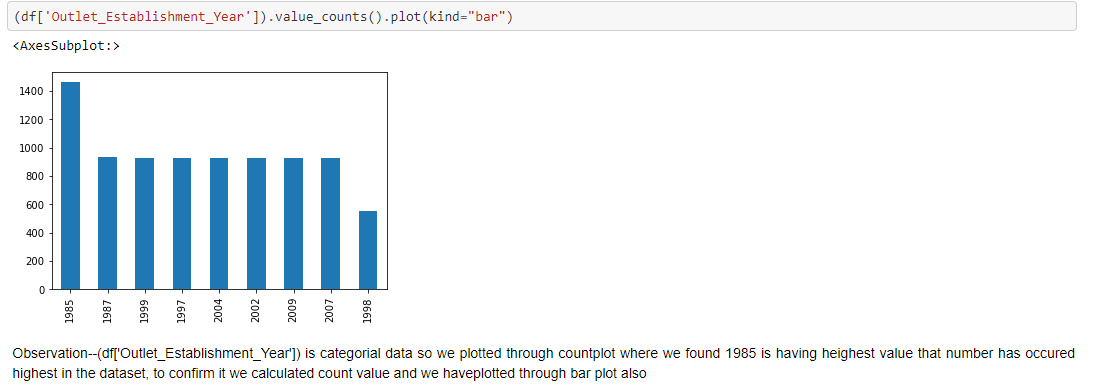
Observation -- From the above plot we do see data is not normally distributed. In our data set it is out target variable which is having continuous featured

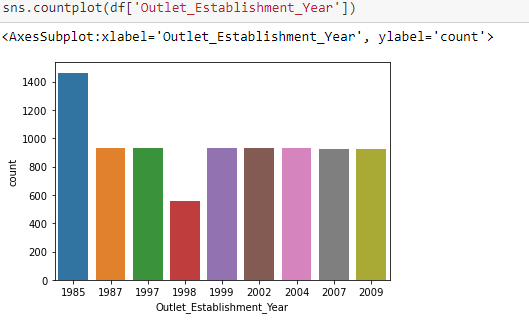


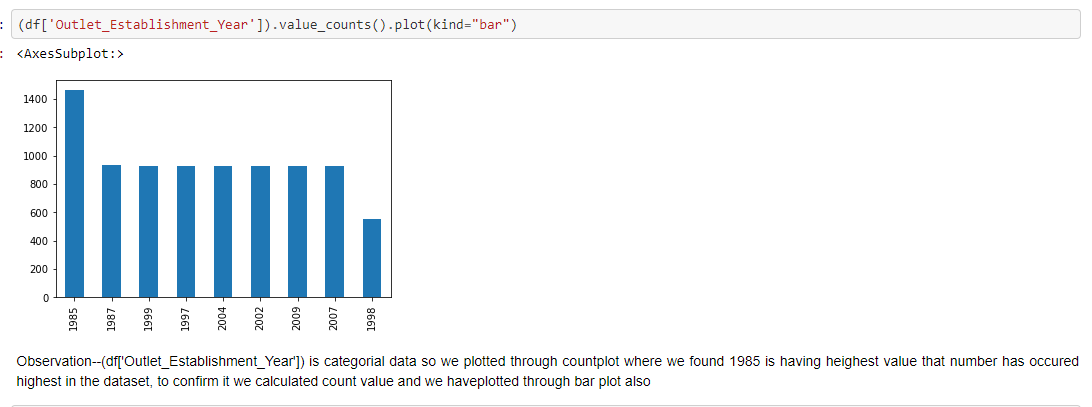
Observation -- Using violin plot we checked the kernel density in y axis and axis we plotted the dat

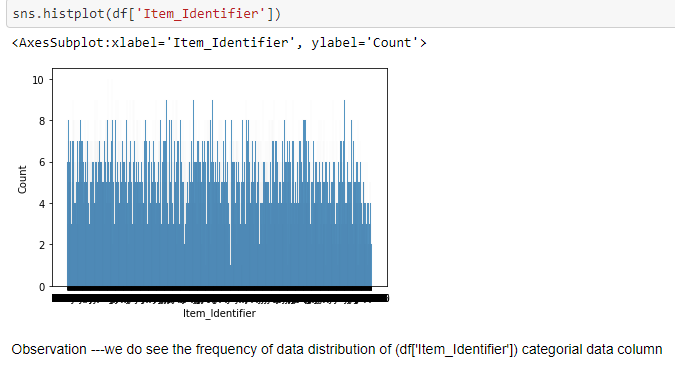
Now we will analysis input features through plot

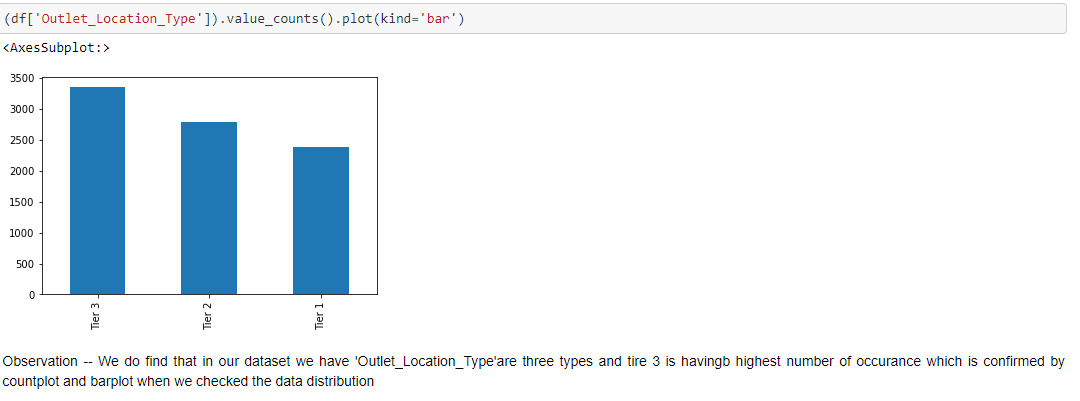


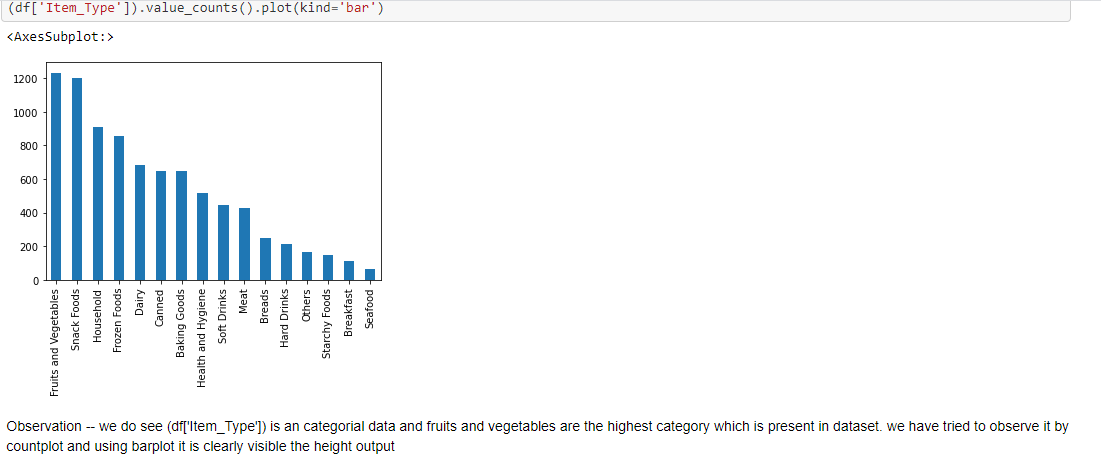


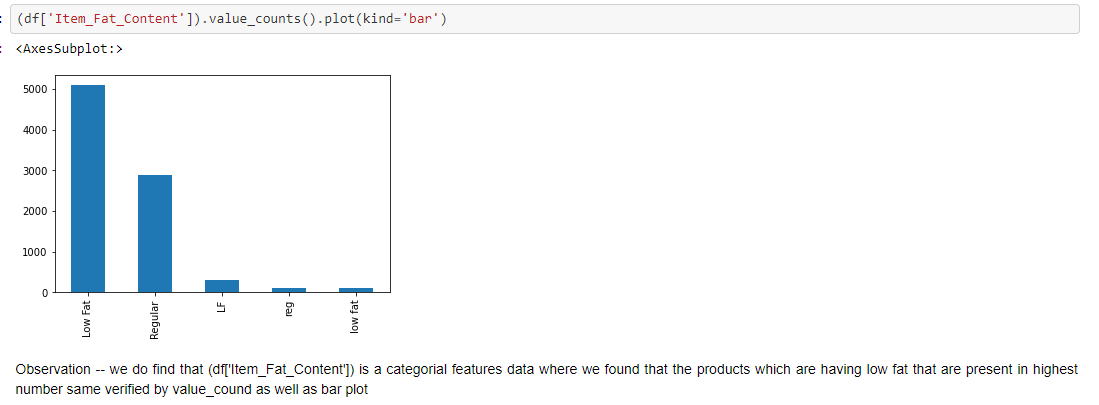


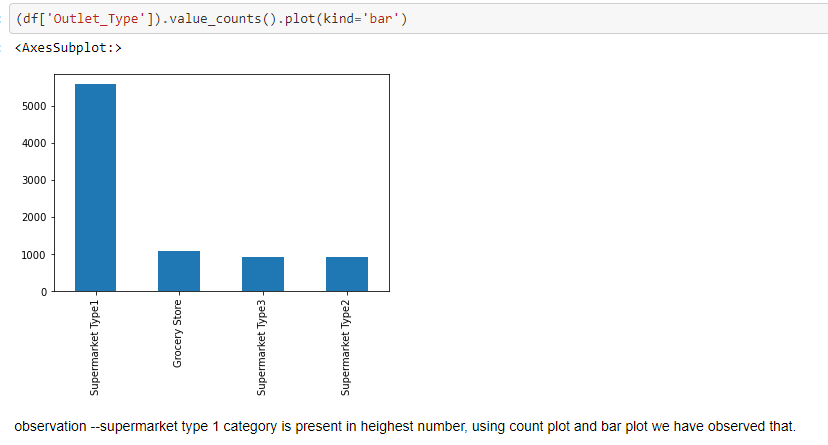












**Null value replacement:**

As the column 'Item Weight' data is continuous features equally distributed we are replacing by mean

mean = df['Item Weight'].mean()

df['Item Weight'] = df['Item Weight'].fillna(mean)

As 'Outlet Size' is categorical column we will replace by Mode

mode = df['Outlet\_Size'].mode()

df['Outlet Size'] = df['Outlet Size'].fillna(mode**)**

**Now we will convert categorical data to numeric using label encoder**

from sklearn.preprocessing import Label Encoder

le=Label Encoder()

df['Item\_Type']=le.fit\_transform(df['Item\_Type'])

df['Item Identifier']=le.fit\_transform(df['Item Identifier'])

df['Outlet\_Identifier']=le.fit\_transform(df['Outlet\_Identifier'])

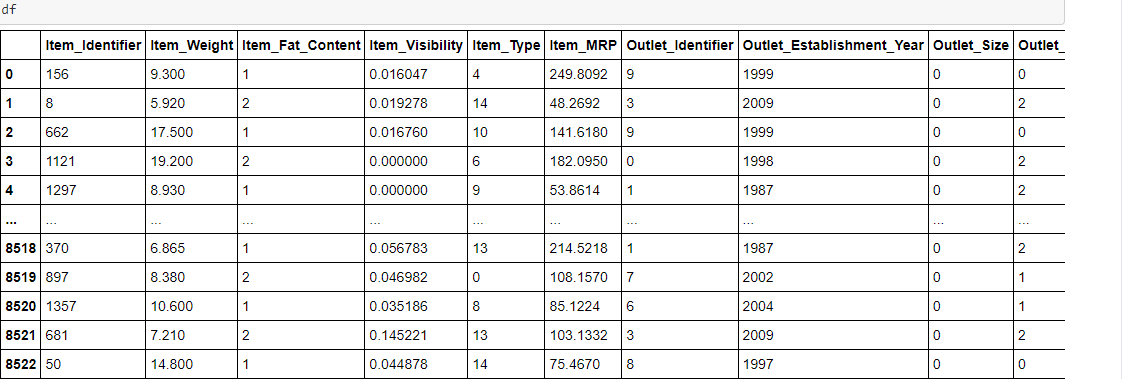
df['Outlet\_Size']=le.fit\_transform(df['Outlet\_Size'])

df['Outlet\_Location\_Type']=le.fit\_transform(df['Outlet\_Location\_Type'])

df['Outlet\_Type']=le.fit\_transform(df['Outlet\_Type'])

df['Item\_Fat\_Content']=le.fit\_transform(df['Item\_Fat\_Content'])

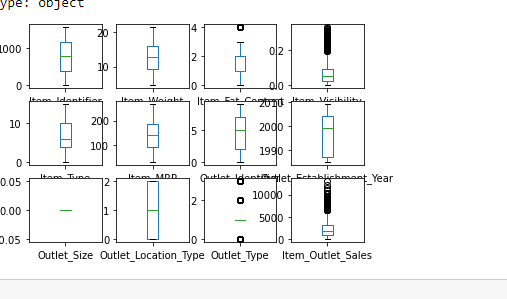
**Now new df is**

****

Outlier detection through boxplot

A **box plot** is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). ... It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is

df. Plot (kind='box' ,subplots=**True** ,layout=(3,4))



Observation –Where the value present above and below vertices showing as dots those are outliers

For checking correlationship of output label with other input features we will calculate cormatrix and sort the value ascending order

corr() is used to find the pair wise correlation of all columns in the data frame. Any na values are automatically excluded. For any non-numeric data type columns in the data frame it is ignored

Input---

cormatrix=df.corr()

cormatrix['Item\_Outlet\_Sales'].sort\_values(ascending= False)

Output---

Item\_Outlet\_Sales 1.000000

Item\_MRP 0.567574

Outlet\_Type 0.401522

Outlet\_Identifier 0.162325

Outlet\_Location\_Type 0.089367

Item\_Type 0.017048

Item Weight 0.011550

Item\_Fat\_Content 0.009800

Item Identifier 0.002869

Outlet\_Establishment\_Year -0.049135

Item\_Visibility -0.128625

Outlet\_Size NaN

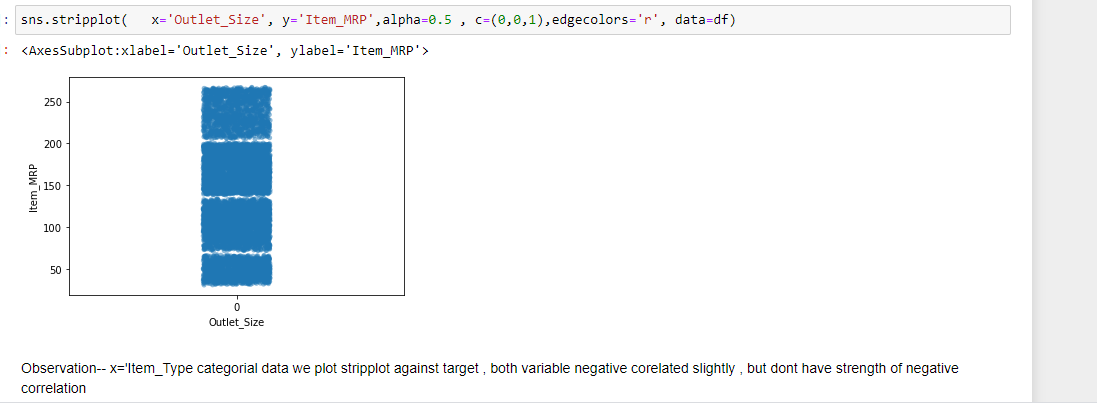
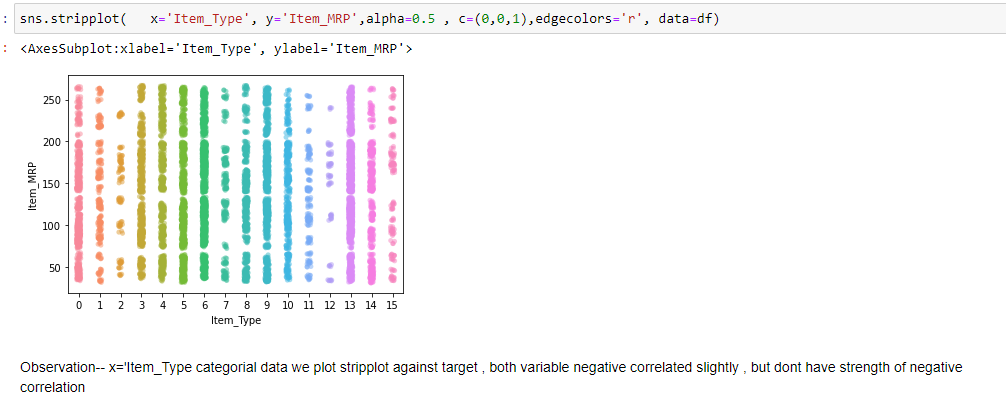
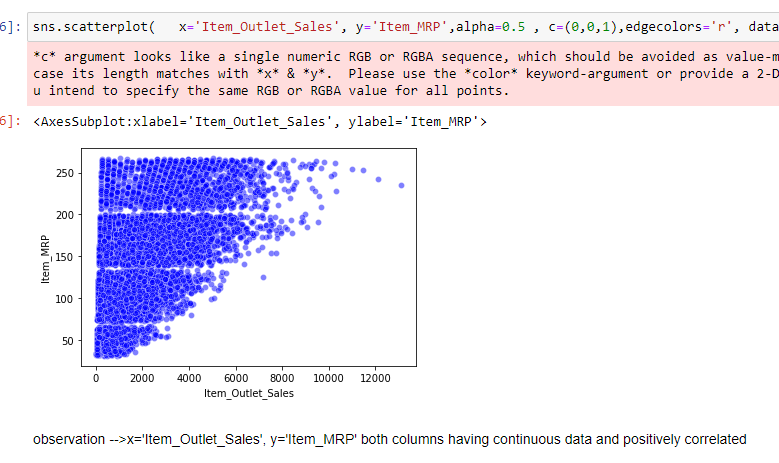
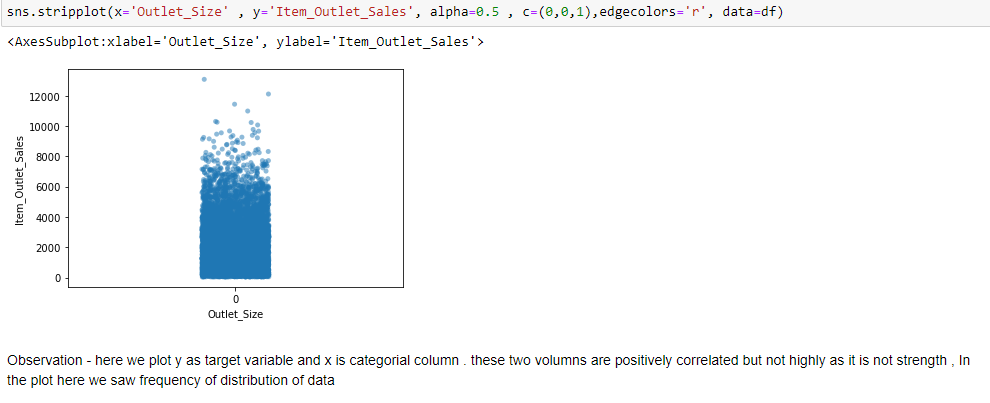
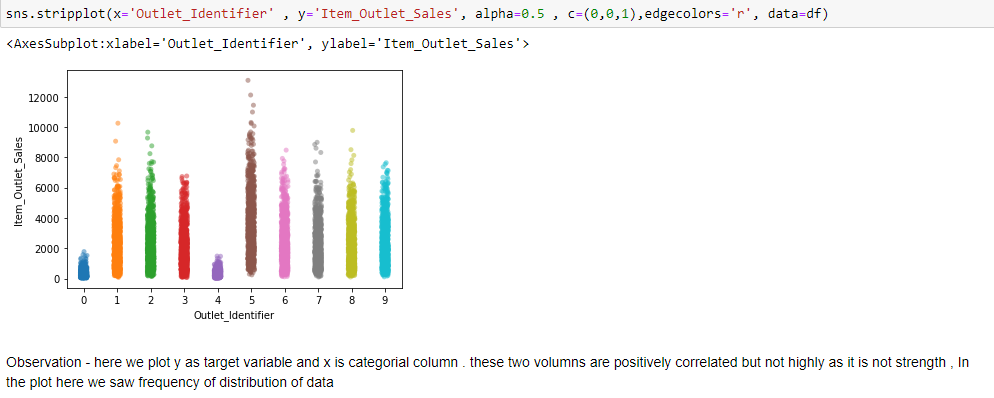
**Now we will check Bi variate analysis**

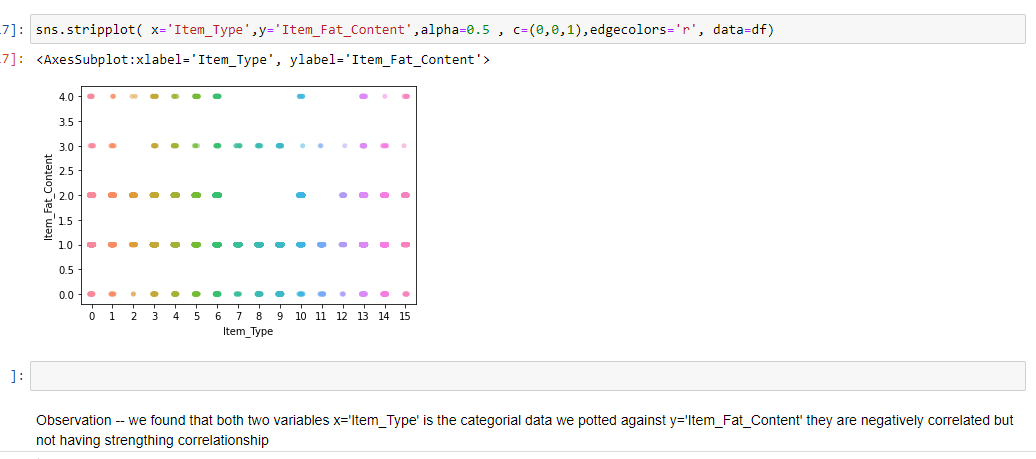
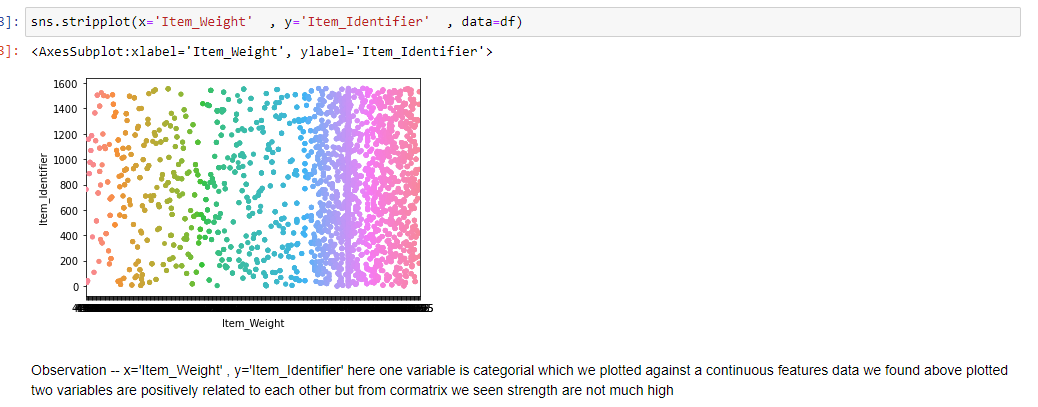
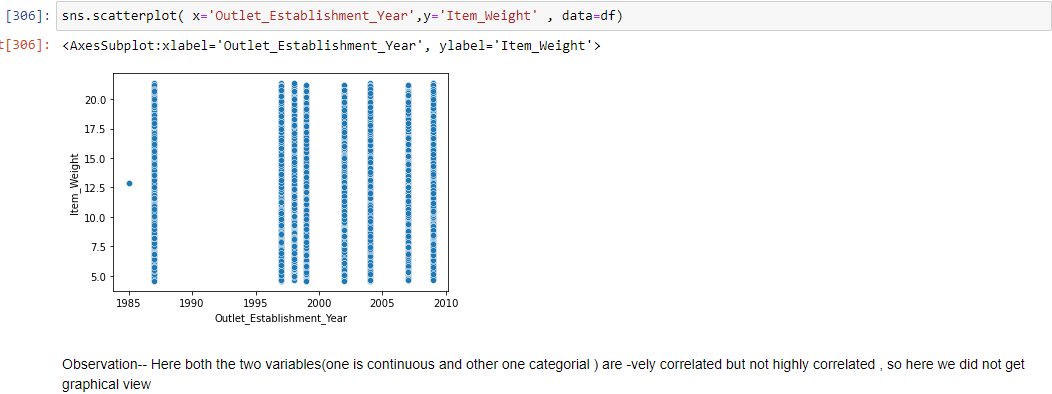
**For bivariate analysis we will plot scatter plot or strip plot**

Bivariate analysis is one of the simplest forms of quantitative analysis. It involves the analysis of two variables, for the purpose of determining the empirical relationship between them. Bivariate analysis can be helpful in testing simple hypotheses of association.

Bivariate analysis is one of the simplest forms of quantitative analysis. It involves the analysis of two variables, for the purpose of determining the empirical relationship between them. Bivariate analysis can be helpful in testing simple hypotheses of association.

If both two variable are continuous we check data distribution through scatter plot and if one is categorical, other one is continuous we will check through stripplot .below we have shown some bivariate analysis



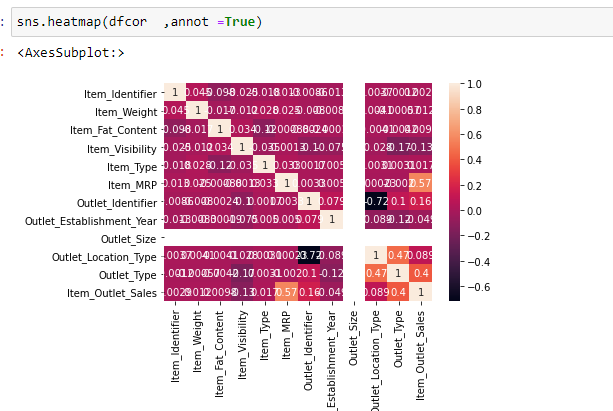


**Multivariate analysis**:--

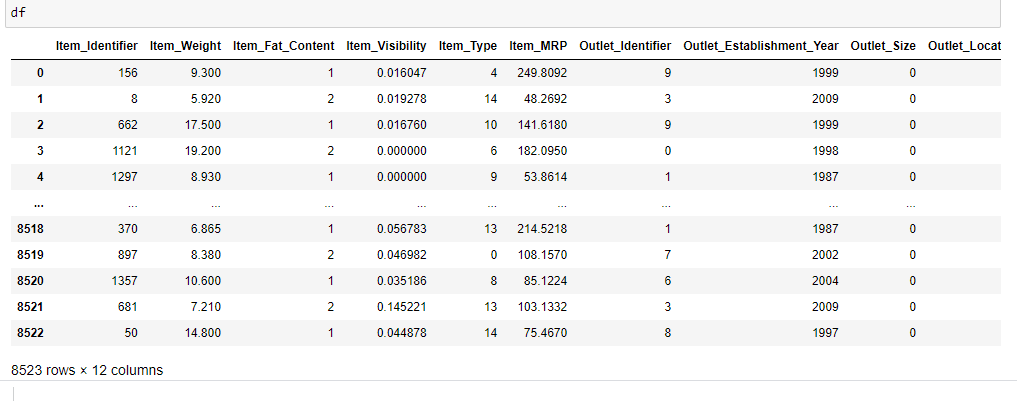
Multivariate statistics is a subdivision of statistics encompassing the simultaneous observation and analysis of more than one outcome variable. Multivariate statistics concerns understanding the different aims and background of each of the different forms of multivariate analysis, and how they relate to each other

Pair plot and heat map we can use to analysis Multivariate analysis

dfcor= df.corr()



**As we found by plotting heat map that outlet size having constant value of data frame so we will drop it**

****

Dropping outlet\_Size

Input--

* df.drop(('Outlet\_Size'),axis=1, inplace=True)

result—now Outlet\_Size will not exist in dataframe

VIF calculation as data set having high variance , we will face underfitting issue

A variance inflation factor (VIF) detects multicollinearity in regression analysis. Multicollinearity is when there’s correlation between predictors (i.e. independent variables) in a model. it’s presence can adversely affect your regression results. The VIF estimates how much the variance of a regression coefficient is inflated due to multicollinearity in the model.

from stats models.stats.outliers\_influence import variance\_inflation\_factor

def CalculateVIF (df):

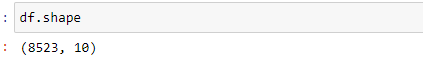
vif = dict ()

vif["FeatureColumns"] =df.columns

vif["VIF"] = [variance\_inflation\_factor(df.values, i) for i in range(df.shape[1])]

return(pd.DataFrame(vif))

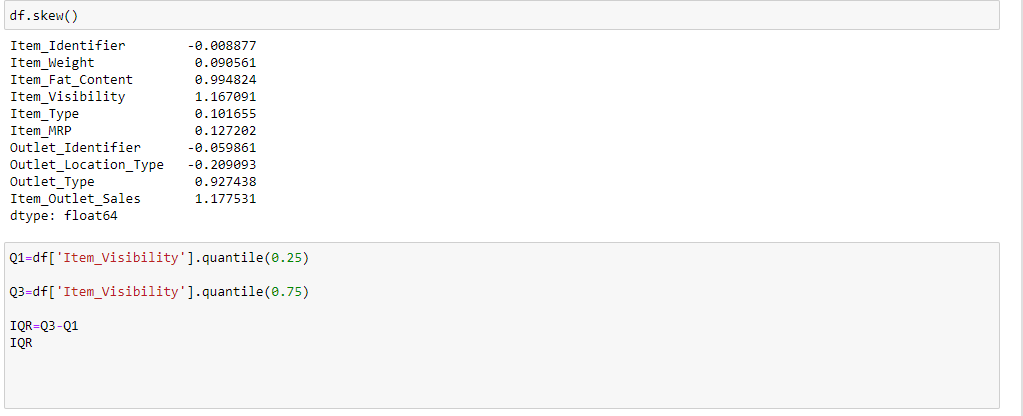
****

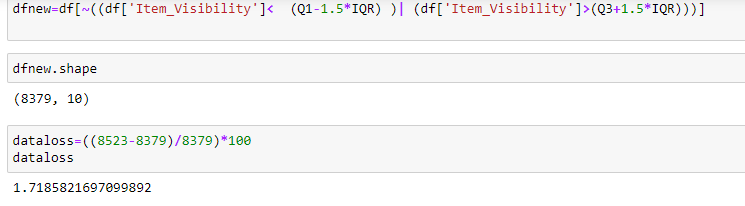
**Now the new df shape we will check **

**Outlier detection and skewness detection and removal**

Skew---A data is called as **skewed** when curve appears distorted or **skewed** either to the left or to the right, in a statistical distribution. In a normal distribution, the graph appears symmetry meaning that there are about as many data values on the left side of the median as on the right side.

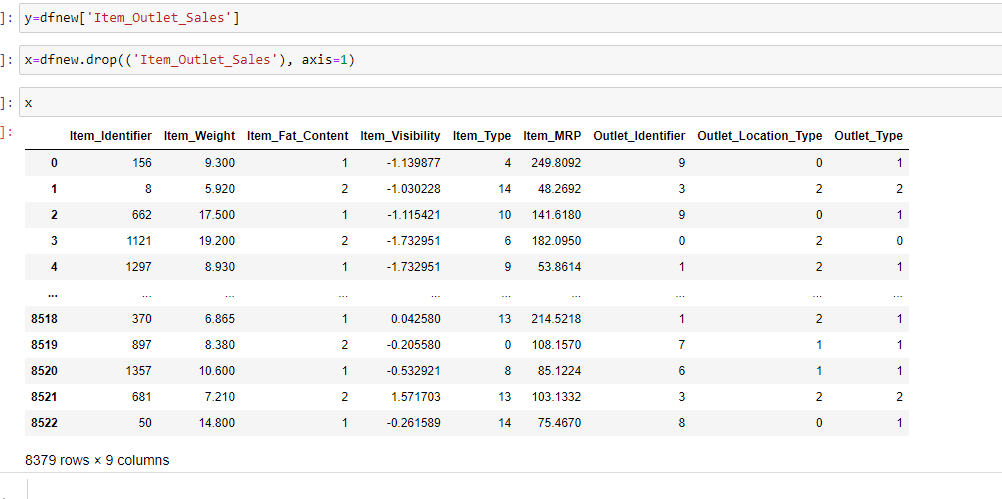
Outlier- In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses

****

****

Observation--We found the data loss is 1.71 % , so we can go ahead with model analysis and our new dataset is dfnew

**Data pre-processing**

****

**StandardScaler**. **StandardScaler** standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation. ... **StandardScaler** makes the mean of the distribution 0. About 68% of the values will lie be between -1 and 1.

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_std = sc.fit\_transform(x)

Scaling the input---

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_std, y,test\_size=.22 , random\_state=11)

**Linear regression:**

lg=LinearRegression()

lg.fit(x\_train,y\_train)

lg.score(x\_train,y\_train)

lg.fit(x\_train,y\_train)

lg.score(x\_train,y\_train)

**Regularization**

In mathematics, statistics, finance, computer science, particularly in machine learning and inverse problems, regularization is the process of adding information in order to solve an ill-posed problem

ls=Lasso(0.001)

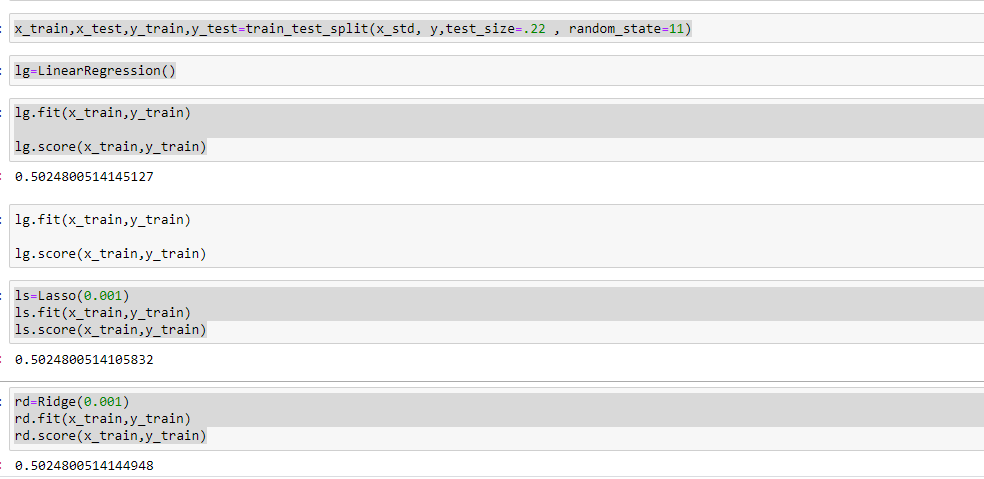
ls.fit(x\_train,y\_train)

ls.score(x\_train,y\_train)

rd=Ridge(0.001)

rd.fit(x\_train,y\_train)

rd.score(x\_train,y\_train)

****

We have performed train test where we have send data to model ( some data for training and some for testing ). We have used 5 model to

* DecisionTreeRegressor
* KNeighborsRegressor()
* RandomForestRegressor()
* XGBRegressor
* AdaBoostRegressor

**ALGORITHIM**

**DecisionTreeRegressor**

dtr=DecisionTreeRegressor()

dtr.fit(x\_train,y\_train)

preddtc=dtr.predict(x\_test)

print("mse==",mean\_squared\_error(y\_test,preddtc))

print("R2SCORE" ,r2\_score(y\_test,preddtc))

mse== 2425170.6748097194

R2SCORE 0.21088065809974277

**KNeighborsRegressor**

kmn=KNeighborsRegressor()

kmn.fit(x\_train,y\_train)

predkmn=kmn.predict(x\_test)

print("mse==",mean\_squared\_error(y\_test,predkmn))

print("R2SCORE" ,r2\_score(y\_test,predkmn))

**RandomForestRegressor**

rvr=RandomForestRegressor()

rvr.fit(x\_train,y\_train)

predrvr=rvr.predict(x\_test)

print("mse==",mean\_squared\_error(y\_test,predrvr))

print("R2SCORE" ,r2\_score(y\_test,predrvr))

mse== 139434545838

R2SCORE 0.5462979995447108

* **AdaBoostRegressor**

from sklearn.ensemble import AdaBoostRegressor

ad=AdaBoostRegressor()

ad.fit(x\_train,y\_train)

predrvr=ad.predict(x\_test)

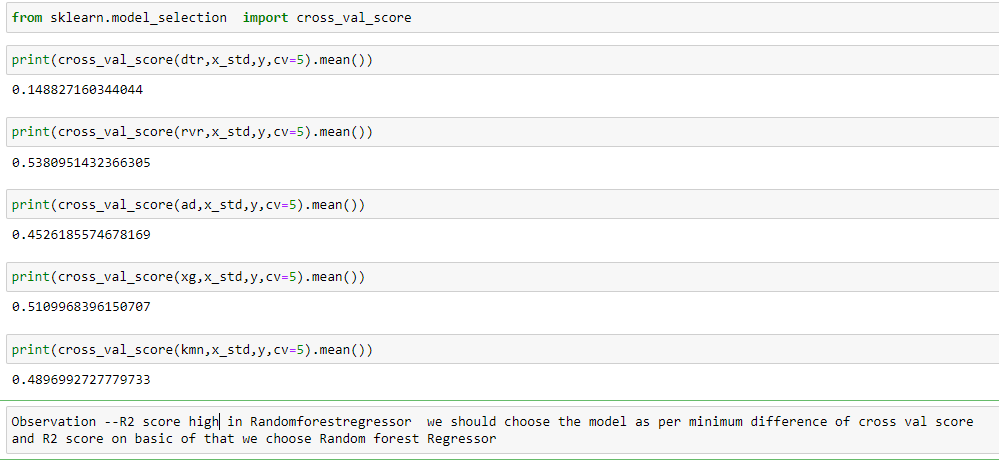
print("mse==",mean\_squared\_error(y\_test,predrvr))

print("R2SCORE" ,r2\_score(y\_test,predrvr))

mse== 1598439.7926385542

R2SCORE 0.4798882526760363

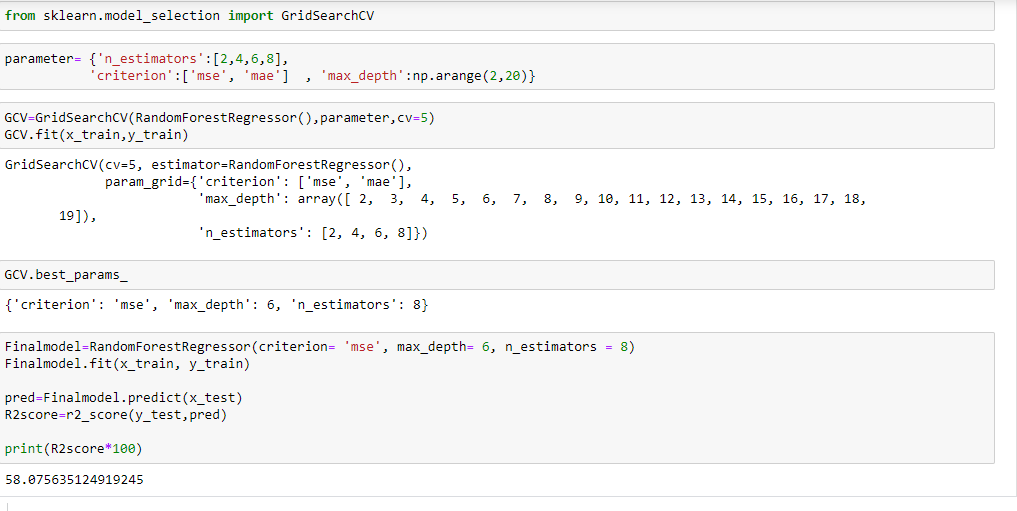
**Cross validation of each model :**

****

As R2 score high in Randomforestregressor we should choose the model as per minimum difference of cross val score and R2 score on basic of that we choose Random forest Repressor

**Hyper Parameter tuning:**

Tuning is the process of maximizing a model's performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate “hyper parameters.”

****

**Final model accuracy came 58.7 which is near about 59%**

**Test data:**

**Converting to numeric**

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

ds['Item\_Type']=le.fit\_transform(ds['Item\_Type'])

ds['Item\_Identifier']=le.fit\_transform(ds['Item\_Identifier'])

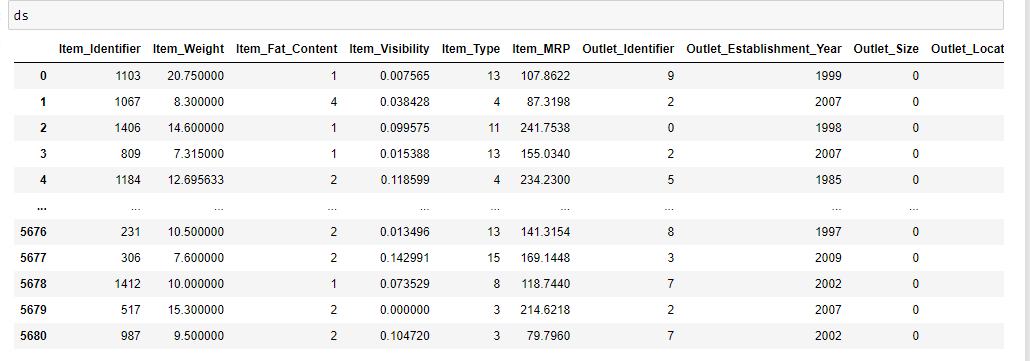
ds['Outlet\_Identifier']=le.fit\_transform(ds['Outlet\_Identifier'])

ds['Outlet\_Size']=le.fit\_transform(ds['Outlet\_Size'])

ds['Outlet\_Location\_Type']=le.fit\_transform(ds['Outlet\_Location\_Type'])

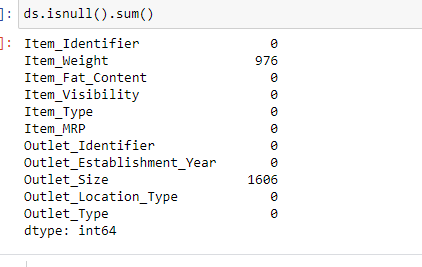
ds['Outlet\_Type']=le.fit\_transform(ds['Outlet\_Type'])

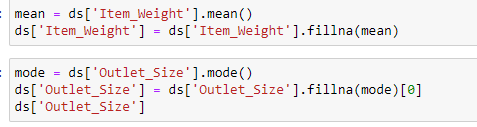
ds['Item\_Fat\_Content']=le.fit\_transform(ds['Item\_Fat\_Content'])

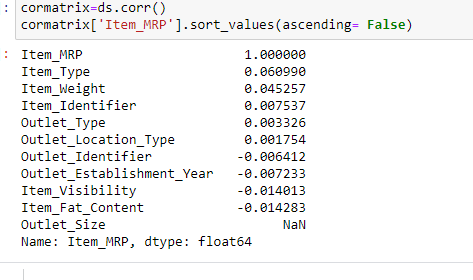
****

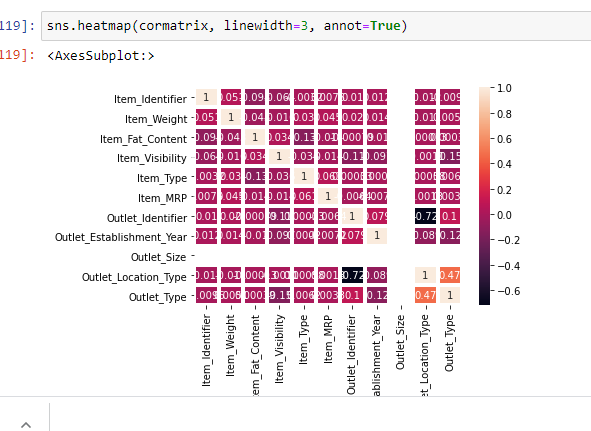
**Same way we have cleaned test data as we cleaned train**

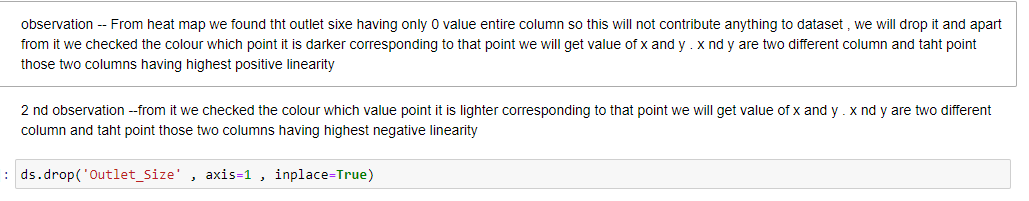
**Null value detection and replacement**

****

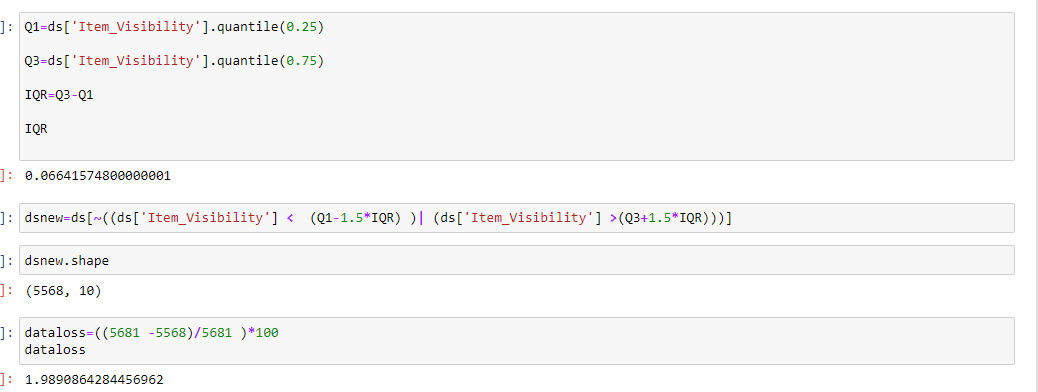
****

**Correlation calculation: **

****

****

**We have removed outlier and skew**

****

****

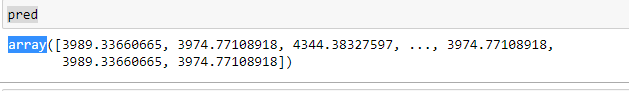
**Final output:**

Now we are calculating test data output which is the sales of each product at a particular store.

pred=Finalmodel.predict (dsnew)

pred

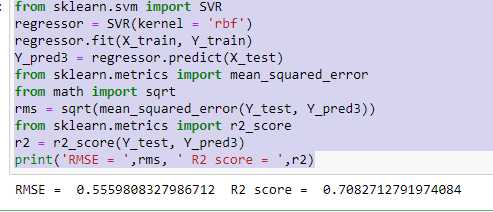
**output –**

****

Best model selection

We have calculated cross validation score of each model. Cross-validation is a statistical method used to estimate the skill of machine learning models and we found RandomForestRegressor has having less difference between R2 and cross validation score .So as per logic RandomForestRegressor is our best model

**Conclusion**

* we have transform categorical data to numeric using Label Encoder
* We have plotted graphical view of each column to understand data distribution , kernel density, as well as for finding outlier concept
* AS R2 score came too low because model under fitting
* We tried to improve it using min-max scalar while scaling input data
* 

**Hardware and Software Requirements and Tools Used**

Hardware

* Good performance PC [Minimum – 8gb RAM +SSD]
* Enough space in hard disk drive

Software requirements

* jupyter note book
* Sometimes you may need Google colab to cross check the output

Package

* Numpy ---import numpy as np ( For calculation )
* **P**anda-import pandas as pd (read data frame )
* Imblearn----- For class sampling

Here the list of some other function

* For plotting- 1>import seaborn as sns 2> import matplotlib.pyplot as plt
* For ignore new version warning--- import warnings

warnings.filterwarnings('ignore')’

import statistics as st

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error ,r2\_score import statistics as st

from sklearn.tree import DecisionTreeRegressor

from sklearn.svm import SVR

from sklearn.ensemble import RandomForestRegressor

from sklearn.neighbors import KNeighborsRegressor

**Limitation:**-

* The data could be incomplete. even the lack of a section or a substantial part of the data, could limit its usability.
* We don’t get always accurate information as data might be not totally completed .
* As it is real time data , it is complex data, took long time to execute