K L University
(Koneru Lakshmaiah Education Foundation)
Deemed to be University, Estd. u/s 3 of UGC Act, 1956
Accredited by NAAC as 'A' Grade University ✧ Approved by AICTE ✧ ISO 9001-2008 Certified
Campus: Greenfields, Vaddeswarm -522502, Guntur District, Andhra Pradesh, INDIA.
Phone: +91-863-2399999 Fax: +91-863- 2388999
Admin Off: 29-36-38, Museum Road, Governorpet, Vijayawada - 520 002, Ph: +91-866-2577715, Fax: +91-866-2577717.

Department of Electronics and Communication Engineering

18TP3101
TP&T-1
MINOR PROJECT-4

A Project Based Lab Report
On
"BIG MART SALES PREDICTION USING LOGISTIC REGRESSION AND EXTREME GRADIENT BOSTING REGRESSION "

SUBMITTED BY:
N.Satish
180040132
SEC NO:4

K L University

(Koneru Lakshmaiah Education Foundation)

Deemed to be University, Estd. u/s 3 of UGC Act, 1956

Accredited by NAAC as 'A' Grade University ✧ Approved by AICTE ✧ ISO 9001-2008 Certified

Campus: Greenfields, Vaddeswarm -522502, Guntur District, Andhra Pradesh, INDIA.

Phone: +91-863-2399999 Fax: +91-863- 2388999

Admin Off: 29-36-38, Museum Road, Governorpet, Vijayawada - 520 002, Ph: +91-866-2577715, Fax: +91-866-2577717.

Department of Electronics and Communication Engineering

BONAFIDE CERTIFICATE

This is to certify that the project-based laboratory Report Title "**BIG MART SALES PREDICTION USING LOGISTIC REGRESSION AND EXTREME GRADIENT BOSTING REGRESSION**" Submitted by: **N.SATISH** id number **180040132**. In the Department of Electronics and Communication Engineering, KL University in partial fulfillment of the requirements for the completion of a project-based Laboratory in course in III B Tech V Semester, is a bonafide record of the work carried out by her under during the academic year 2020 – 2021.

PROJECT SUPERVISOR **HEAD OF THE DEPARTMENT**

BIG MART SALES PREDICTION :

We will explore the problem in following stages:

1. **Hypothesis Generation** – understanding the problem better by brainstorming possible factors that can impact the outcome
2. **Data Exploration** – looking at categorical and continuous feature summaries and making inferences about the data.
3. **Data Cleaning** – imputing missing values in the data and checking for outliers
4. **Feature Engineering** – modifying existing variables and creating new ones for analysis
5. **Model Building** – making predictive models on the data

Store Level Hypotheses:

1. **City type:** Stores located in urban or Tier 1 cities should have higher sales because of the higher income levels of people there.
2. **Population Density:** Stores located in densely populated areas should have higher sales because of more demand.
3. **Store Capacity:** Stores which are very big in size should have higher sales as they act like one-stop-shops and people would prefer getting everything from one place
4. **Competitors:** Stores having similar establishments nearby should have less sales because of more competition.
5. **Marketing:** Stores which have a good marketing division should have higher sales as it will be able to attract customers through the right offers and advertising.
6. **Location:** Stores located within popular marketplaces should have higher sales because of better access to customers.
7. **Customer Behavior:** Stores keeping the right set of products to meet the local needs of customers will have higher sales.
8. **Ambiance:** Stores which are well-maintained and managed by polite and humble people are expected to have higher footfall and thus higher sales.

Product Level Hypotheses:

1. **Brand:** Branded products should have higher sales because of higher trust in the customer.
2. **Packaging:** Products with good packaging can attract customers and sell more.
3. **Utility:** Daily use products should have a higher tendency to sell as compared to the specific use products.
4. **Display Area:** Products which are given bigger shelves in the store are likely to catch attention first and sell more.
5. **Visibility in Store:** The location of product in a store will impact sales. Ones which are right at entrance will catch the eye of customer first rather than the ones in back.
6. **Advertising:** Better advertising of products in the store will should higher sales in most cases.
7. **Promotional Offers:** Products accompanied with attractive offers and discounts will sell more.

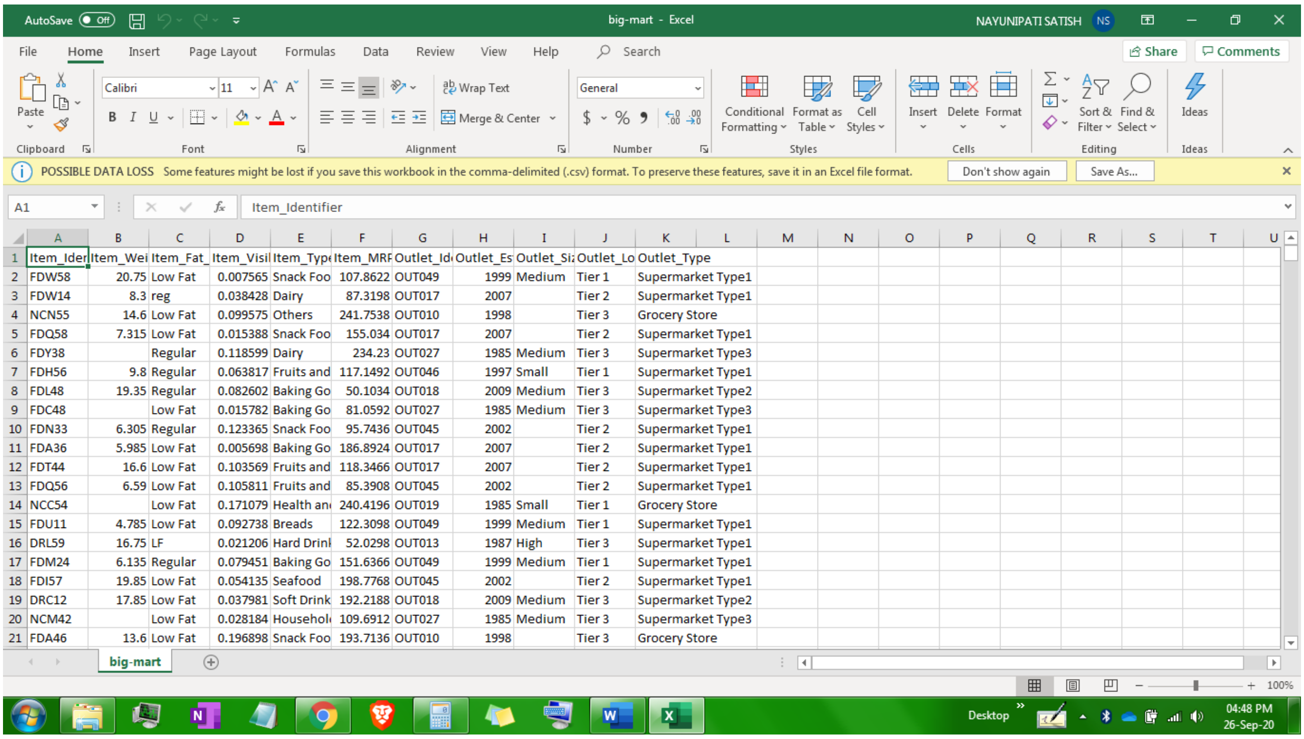
Problem:

The data scientists at BigMart 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, BigMart will try to understand the properties of products and stores which play a key role in increasing sales.

Loading The **BIG MART SALES PREDICTION** Data:

1. Open command prompt and type “jupyter notebook”
2. Press enter.
3. Then jupyter window will be opened and you can load the data set using the command read_csv(“filename.csv”).
4. Then the data is loaded.

Data Set Used For **BIG MART SALES PREDICTION:**



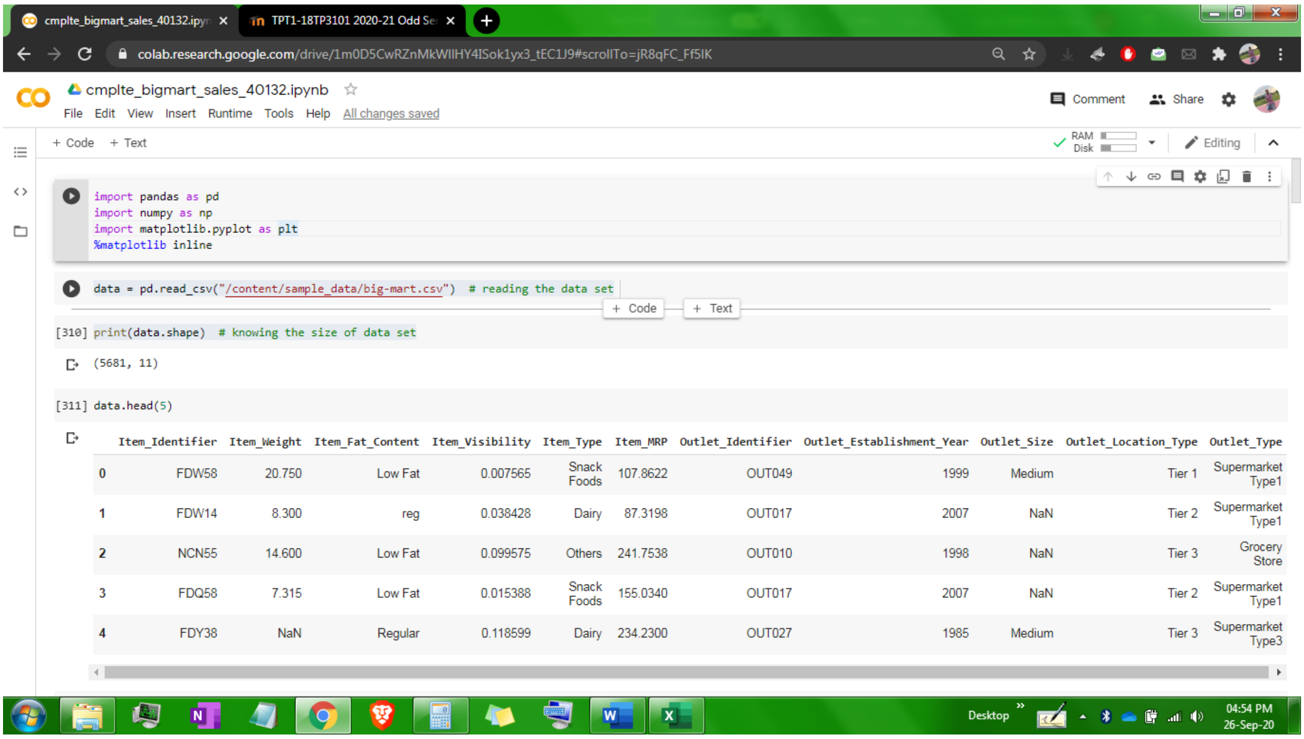
	A	B	C	D	E	F	G	H	I	J	K
1	Item_Ider	Item_Wei	Item_Fat_	Item_Visil	Item_Typ	Item_MRF	Outlet_Id	Outlet_Es	Outlet_Si	Outlet_Lo	Outlet_Type
2	FDW58	20.75	Low Fat	0.007565	Snack Foo	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1
3	FDW14	8.3	reg	0.038428	Dairy	87.3198	OUT017	2007		Tier 2	Supermarket Type1
4	NCN55	14.6	Low Fat	0.099575	Others	241.7538	OUT010	1998		Tier 3	Grocery Store
5	FDQ58	7.315	Low Fat	0.015388	Snack Foo	155.034	OUT017	2007		Tier 2	Supermarket Type1
6	FDY38		Regular	0.118599	Dairy	234.23	OUT027	1985	Medium	Tier 3	Supermarket Type3
7	FDH56	9.8	Regular	0.063817	Fruits and	117.1492	OUT046	1997	Small	Tier 1	Supermarket Type1
8	FDL48	19.35	Regular	0.082602	Baking Go	50.1034	OUT018	2009	Medium	Tier 3	Supermarket Type2
9	FDC48		Low Fat	0.015782	Baking Go	81.0592	OUT027	1985	Medium	Tier 3	Supermarket Type3
10	FDN33	6.305	Regular	0.123365	Snack Foo	95.7436	OUT045	2002		Tier 2	Supermarket Type1
11	FDA36	5.985	Low Fat	0.005698	Baking Go	186.8924	OUT017	2007		Tier 2	Supermarket Type1
12	FDT44	16.6	Low Fat	0.103569	Fruits and	118.3466	OUT017	2007		Tier 2	Supermarket Type1
13	FDQ56	6.59	Low Fat	0.105811	Fruits and	85.3908	OUT045	2002		Tier 2	Supermarket Type1
14	NCC54		Low Fat	0.171079	Health and	240.4196	OUT019	1985	Small	Tier 1	Grocery Store
15	FDU11	4.785	Low Fat	0.092738	Breads	122.3098	OUT049	1999	Medium	Tier 1	Supermarket Type1
16	DRL59	16.75	LF	0.021206	Hard Drinl	52.0298	OUT013	1987	High	Tier 3	Supermarket Type1
17	FDM24	6.135	Regular	0.079451	Baking Go	151.6366	OUT049	1999	Medium	Tier 1	Supermarket Type1
18	FDI57	19.85	Low Fat	0.054135	Seafood	198.7768	OUT045	2002		Tier 2	Supermarket Type1
19	DRC12	17.85	Low Fat	0.037981	Soft Drink	192.2188	OUT018	2009	Medium	Tier 3	Supermarket Type2
20	NCM42		Low Fat	0.028184	Househol	109.6912	OUT027	1985	Medium	Tier 3	Supermarket Type3
21	FDA46	13.6	Low Fat	0.196898	Snack Foo	193.7136	OUT010	1998		Tier 3	Grocery Store

Code:-

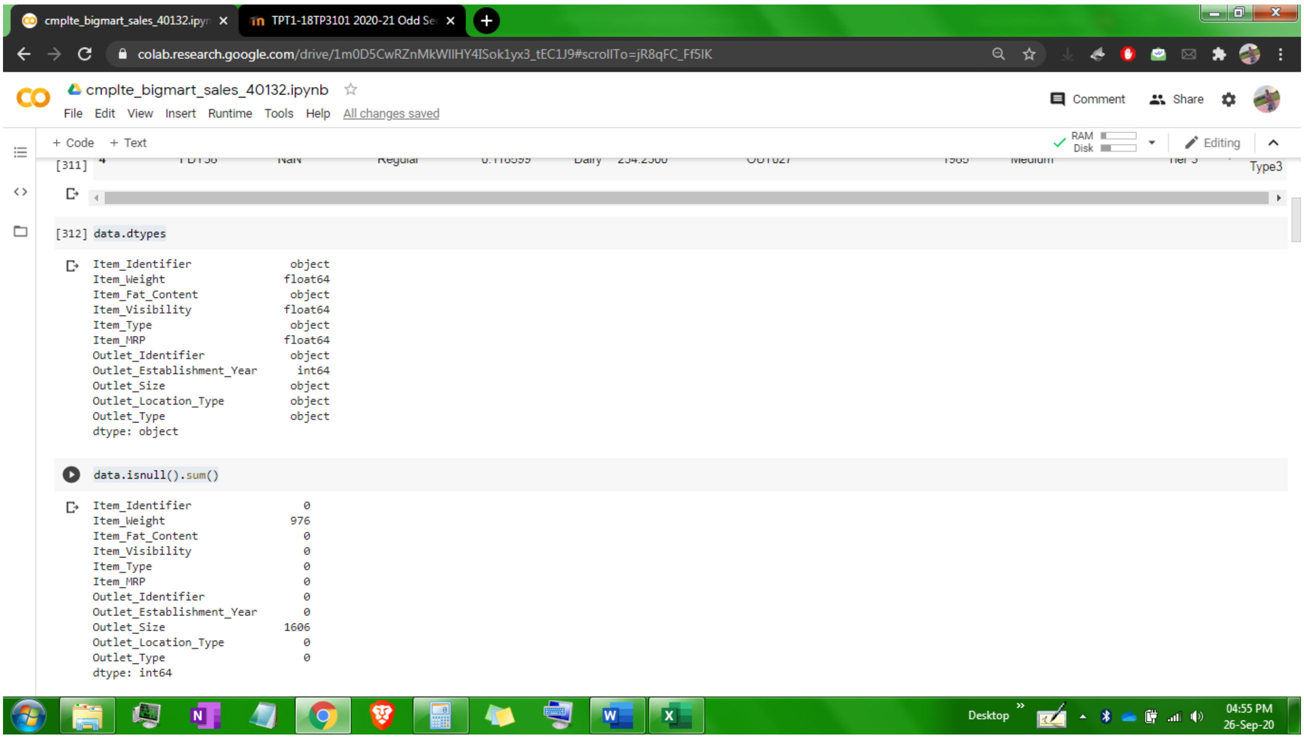
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
data = pd.read_csv("/content/sample_data/big-
mart.csv")  # reading the data set
print(data.shape)  # knowing the size of data set
data.head(5)
data.dtypes
data.isnull().sum()
data.Item_Weight = data.Item_Weight.fillna(data.Item_Weight.mean())  # fillin
g null values with mean() of weights
data['Outlet_Size'].value_counts()  # knowing the various sizes
data.Outlet_Size = data.Outlet_Size.fillna('Medium')   # filling null values
with medium
data.isnull().sum()
data.info() # overview of the dataset
data.describe()  #some basic statistics for numerical variables.
# combining the Item type , so that we can know types available .

data['Item_Identifier'].value_counts()   # finding no.of different types of i
dentifiers
data['Item_Type_Combined'] = data['Item_Identifier'].apply(lambda x: x[0:2])
 # lamda is a function like normal function ,

   # it returns the value after , :


data['Item_Type_Combined'] = data['Item_Type_Combined'].map({'FD':'Food',
                                                              'NC':'Non-
Consumable',
                                                              'DR':'Drinks'})
    # mapping the different identifiers as names
d1=data['Item_Type_Combined'].value_counts()
print(d1)
d1.plot()
import seaborn as sns
sns.countplot(data['Outlet_Location_Type'],hue=data['Outlet_Size'])
#Import library:
from sklearn.preprocessing import LabelEncoder #, OneHotEncoder
le = LabelEncoder()     # label encoder = categorical features into numeric v
alues

#New variable for outlet

data['Outlet'] = le.fit_transform(data['Outlet_Identifier'])
```

```
var_mod = ['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Item_Type
_Combined','Outlet_Type','Outlet'] # we are passing all the coloumns to chang
e into variable


for i in var_mod:
    data[i] = le.fit_transform(data[i])

data.head()

# creating dummies in cegrating the types .
data = pd.get_dummies(data, columns=['Item_Fat_Content','Outlet_Location_Type
','Outlet_Size','Outlet_Type','Item_Type_Combined','Outlet'])
data.head()

data.dtypes   # finding the data types for better clarification
data.size
X = data.values
train = X[0:59650] # 59650 data as train data  // nearly 70 of data is using
for train
test = X[59650:]  # 13884 data as test data
predictions = []
data=pd.get_dummies(data)

y=data['Item_MRP']    # considering mrp as main component.
x=data.drop(['Item_MRP'],axis='columns')
y
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
reg.score(x_test,y_pred)
from xgboost.sklearn import XGBRegressor  # importing extreme gradient boosti
ng
xgb_reg=XGBRegressor()
xgb_reg.fit(x_train,y_train)
xgb_pred=xgb_reg.predict(x_test)
xgb_pred
from sklearn import metrics
print(metrics.r2_score(y_test, xgb_pred))
print(np.log(metrics.mean_squared_error(y_test, xgb_pred)))
model.score(x_test,y_test)
```

Outputs:



cmplte_bigmart_sales_40132.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
data = pd.read_csv("/content/sample_data/big-mart.csv") # reading the data set
[310] print(data.shape) # knowing the size of data set
(5681, 11)
[311] data.head(5)
Item_Identifier Item_Weight Item_Fat_Content Item_Visibility Item_Type Item_MRP Outlet_Identifier Outlet_Establishment_Year Outlet_Size Outlet_Location_Type Outlet_Type
0 FDW58 20.750 Low Fat 0.007565 Snack Foods 107.8622 OUT049 1999 Medium Tier 1 Supermarket Type1
1 FDW14 8.300 reg 0.038428 Dairy 87.3198 OUT017 2007 NaN Tier 2 Supermarket Type1
2 NCN55 14.600 Low Fat 0.099575 Others 241.7538 OUT010 1998 NaN Tier 3 Grocery Store
3 FDQ58 7.315 Low Fat 0.015388 Snack Foods 155.0340 OUT017 2007 NaN Tier 2 Supermarket Type1
4 FDY38 NaN Regular 0.118599 Dairy 234.2300 OUT027 1985 Medium Tier 3 Supermarket Type3
04:54 PM
26-Sep-20



cmplte_bigmart_sales_40132.ipynb
File Edit View Insert Runtime Tools Help All changes saved
[312] data.dtypes
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
dtype: object
data.isnull().sum()
Item_Identifier 0
Item_Weight 976
Item_Fat_Content 0
Item_Visibility 0
Item_Type 0
Item_MRP 0
Outlet_Identifier 0
Outlet_Establishment_Year 0
Outlet_Size 1606
Outlet_Location_Type 0
Outlet_Type 0
dtype: int64
04:55 PM
26-Sep-20

```
[313] Outlet_Location_Type         0
      Outlet_Type                  0
      dtype: int64
```

```
[314] data.Item_Weight = data.Item_Weight.fillna(data.Item_Weight.mean())  # filling null values with mean() of weights
```

```
[315] data['Outlet_Size'].value_counts()  # knowing the various sizes
```

```
Medium    1862
Small     1592
High       621
Name: Outlet_Size, dtype: int64
```

```
[316] data.Outlet_Size = data.Outlet_Size.fillna('Medium')   # filling null values with medium
```

```
data.isnull().sum()
```

```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
dtype: int64
```

now we dont have any null values in a data set .

```
Data columns (total 11 columns):
[318] #   Column                     Non-Null Count  Dtype
      ---  ------                    --------------  -----
       0   Item_Identifier           5681 non-null   object
       1   Item_Weight               5681 non-null   float64
       2   Item_Fat_Content          5681 non-null   object
       3   Item_Visibility           5681 non-null   float64
       4   Item_Type                 5681 non-null   object
       5   Item_MRP                  5681 non-null   float64
       6   Outlet_Identifier         5681 non-null   object
       7   Outlet_Establishment_Year 5681 non-null   int64
       8   Outlet_Size               5681 non-null   object
       9   Outlet_Location_Type      5681 non-null   object
       10  Outlet_Type               5681 non-null   object
      dtypes: float64(3), int64(1), object(7)
      memory usage: 488.3+ KB
```

```
data.describe()   #some basic statistics for numerical variables.
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year
count	5681.000000	5681.000000	5681.000000	5681.000000
mean	12.695633	0.065684	141.023273	1997.828903
std	4.245189	0.051252	61.809091	8.372256
min	4.555000	0.000000	31.990000	1985.000000
25%	9.195000	0.027047	94.412000	1987.000000
50%	12.695633	0.054154	141.415400	1999.000000
75%	15.850000	0.093463	186.026600	2004.000000
max	21.350000	0.323637	266.588400	2009.000000

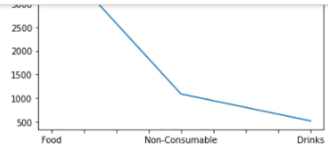
75%	15.850000	0.093463	186.026600	2004.000000
max	21.350000	0.323637	266.588400	2009.000000

```python
# combining the Item type , so that we can know types available .

data['Item_Identifier'].value_counts()   # finding no.of different types of identifiers
data['Item_Type_Combined'] = data['Item_Identifier'].apply(lambda x: x[0:2])  # lamda is a function like normal function ,
                                                                             # it returns the value after , :

data['Item_Type_Combined'] = data['Item_Type_Combined'].map({'FD':'Food',
                                                             'NC':'Non-Consumable',
                                                            'DR':'Drinks'})    # mapping the different identifiers as names
d1=data['Item_Type_Combined'].value_counts()
print(d1)
```

```
Food              4076
Non-Consumable    1087
Drinks             518
Name: Item_Type_Combined, dtype: int64
```

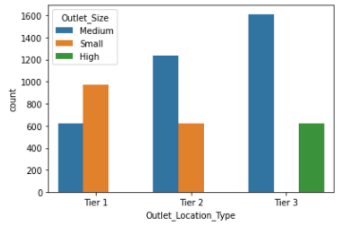
[321]
```python
d1.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f072aa22d30>
```

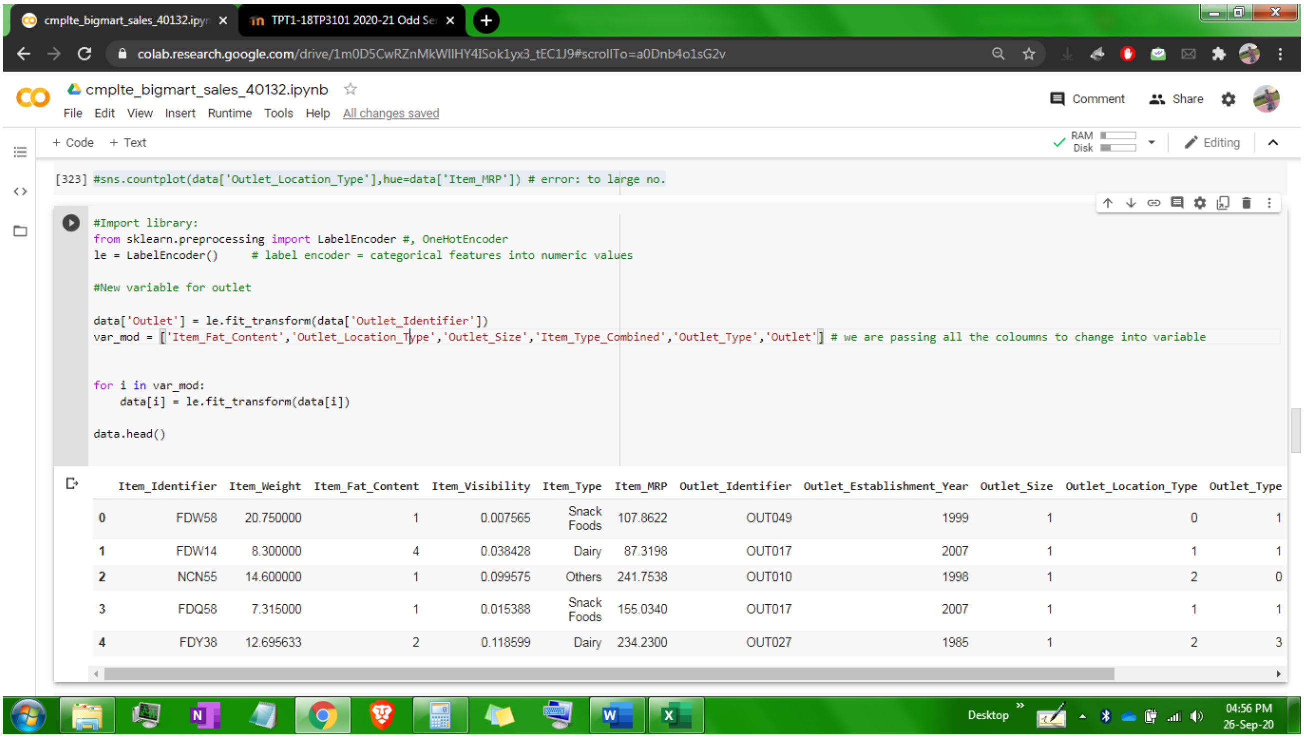



```python
import seaborn as sns
sns.countplot(data['Outlet_Location_Type'],hue=data['Outlet_Size'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f073bd94128>
```



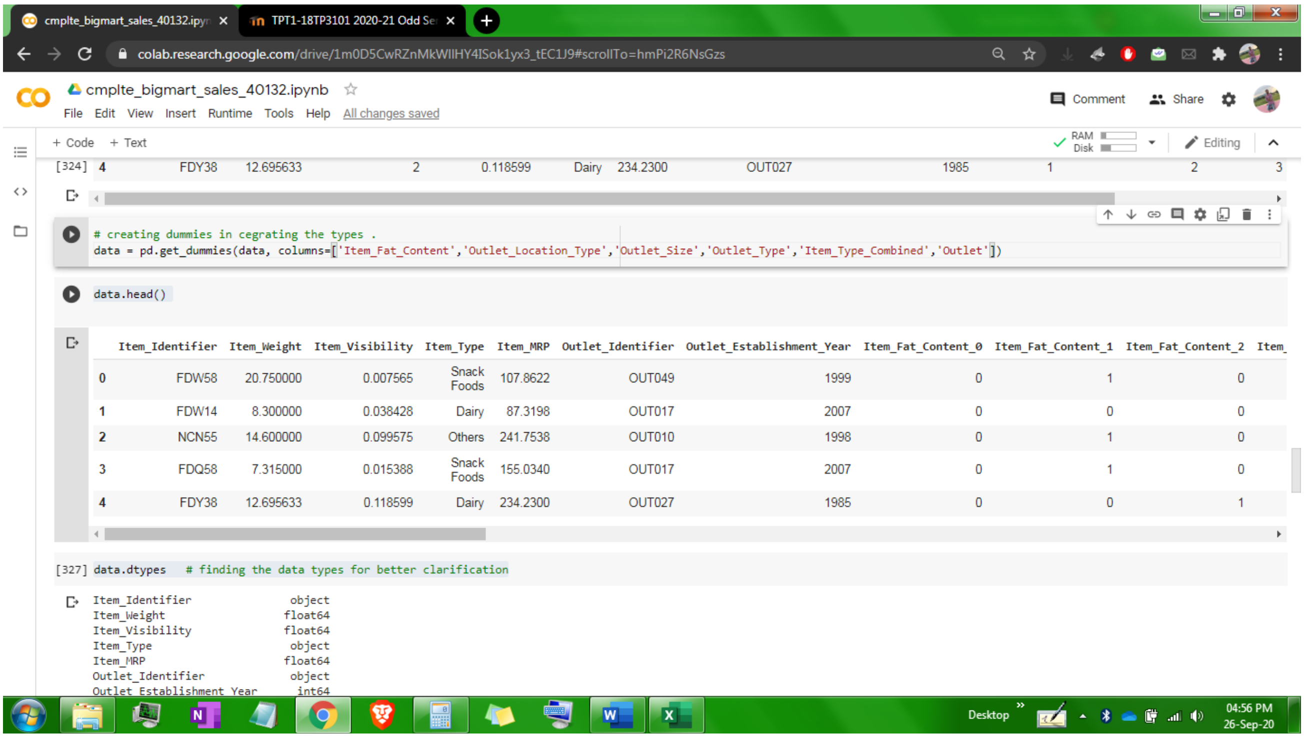
[323]
```python
#sns.countplot(data['Outlet_Location_Type'],hue=data['Item_MRP']) # error: to large no.
```



```
[323] #sns.countplot(data['Outlet_Location_Type'],hue=data['Item_MRP']) # error: to large no.
```

```
#Import library:
from sklearn.preprocessing import LabelEncoder #, OneHotEncoder
le = LabelEncoder()      # label encoder = categorical features into numeric values

#New variable for outlet

data['Outlet'] = le.fit_transform(data['Outlet_Identifier'])
var_mod = ['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Item_Type_Combined','Outlet_Type','Outlet'] # we are passing all the coloumns to change into variable


for i in var_mod:
    data[i] = le.fit_transform(data[i])

data.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
0	FDW58	20.750000	1	0.007565	Snack Foods	107.8622	OUT049	1999	1	0	1
1	FDW14	8.300000	4	0.038428	Dairy	87.3198	OUT017	2007	1	1	1
2	NCN55	14.600000	1	0.099575	Others	241.7538	OUT010	1998	1	2	0
3	FDQ58	7.315000	1	0.015388	Snack Foods	155.0340	OUT017	2007	1	1	1
4	FDY38	12.695633	2	0.118599	Dairy	234.2300	OUT027	1985	1	2	3



```
[324] 4 FDY38 12.695633 2 0.118599 Dairy 234.2300 OUT027 1985 1 2 3
```

```
# creating dummies in cegrating the types .
data = pd.get_dummies(data, columns=['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Outlet_Type','Item_Type_Combined','Outlet'])
```

```
data.head()
```

	Item_Identifier	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Item_Fat_Content_0	Item_Fat_Content_1	Item_Fat_Content_2	Item_
0	FDW58	20.750000	0.007565	Snack Foods	107.8622	OUT049	1999	0	1	0	
1	FDW14	8.300000	0.038428	Dairy	87.3198	OUT017	2007	0	0	0	
2	NCN55	14.600000	0.099575	Others	241.7538	OUT010	1998	0	1	0	
3	FDQ58	7.315000	0.015388	Snack Foods	155.0340	OUT017	2007	0	1	0	
4	FDY38	12.695633	0.118599	Dairy	234.2300	OUT027	1985	0	0	1	

```
[327] data.dtypes   # finding the data types for better clarification
```

```
Item_Identifier               object
Item_Weight                  float64
Item_Visibility              float64
Item_Type                     object
Item_MRP                     float64
Outlet_Identifier             object
Outlet_Establishment_Year      int64
```

```
[330] data.size

198835
```

```
X = data.values
train = X[0:59650] # 59650 data as train data  // nearly 70 of data is using for train
test = X[59650:]  # 13884 data as test data
predictions = []
```

```
[332] data=pd.get_dummies(data)
```

```
[333]
y=data['Item_MRP']    # considering mrp as main component.
x=data.drop(['Item_MRP'],axis='columns')
y

0       107.8622
1        87.3198
2       241.7538
3       155.0340
4       234.2300
          ...
5676    141.3154
5677    169.1448
5678    118.7440
5679    214.6218
5680     79.7960
Name: Item_MRP, Length: 5681, dtype: float64
```

```
[334] from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

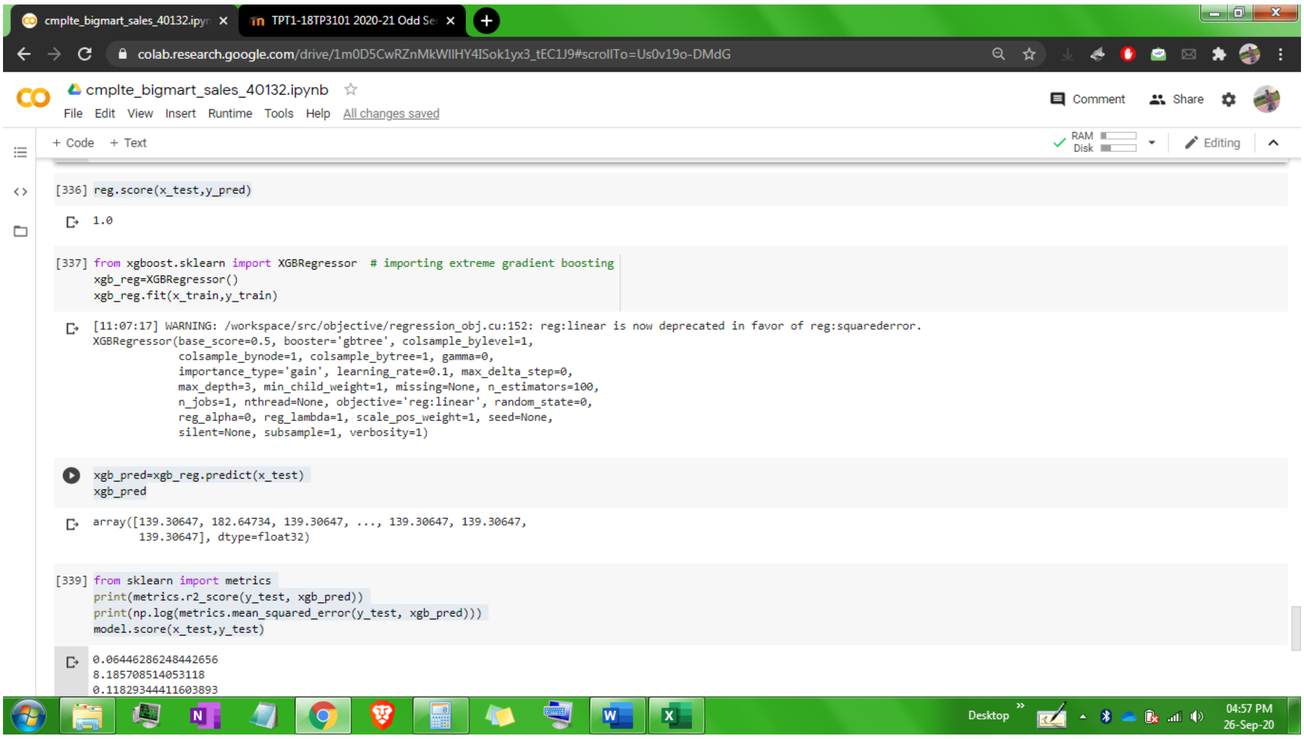
```
[333] 2       241.7538
3       155.0340
4       234.2300
          ...
5676    141.3154
5677    169.1448
5678    118.7440
5679    214.6218
5680     79.7960
Name: Item_MRP, Length: 5681, dtype: float64
```

```
[334] from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
```

```
[336] reg.score(x_test,y_pred)

1.0
```

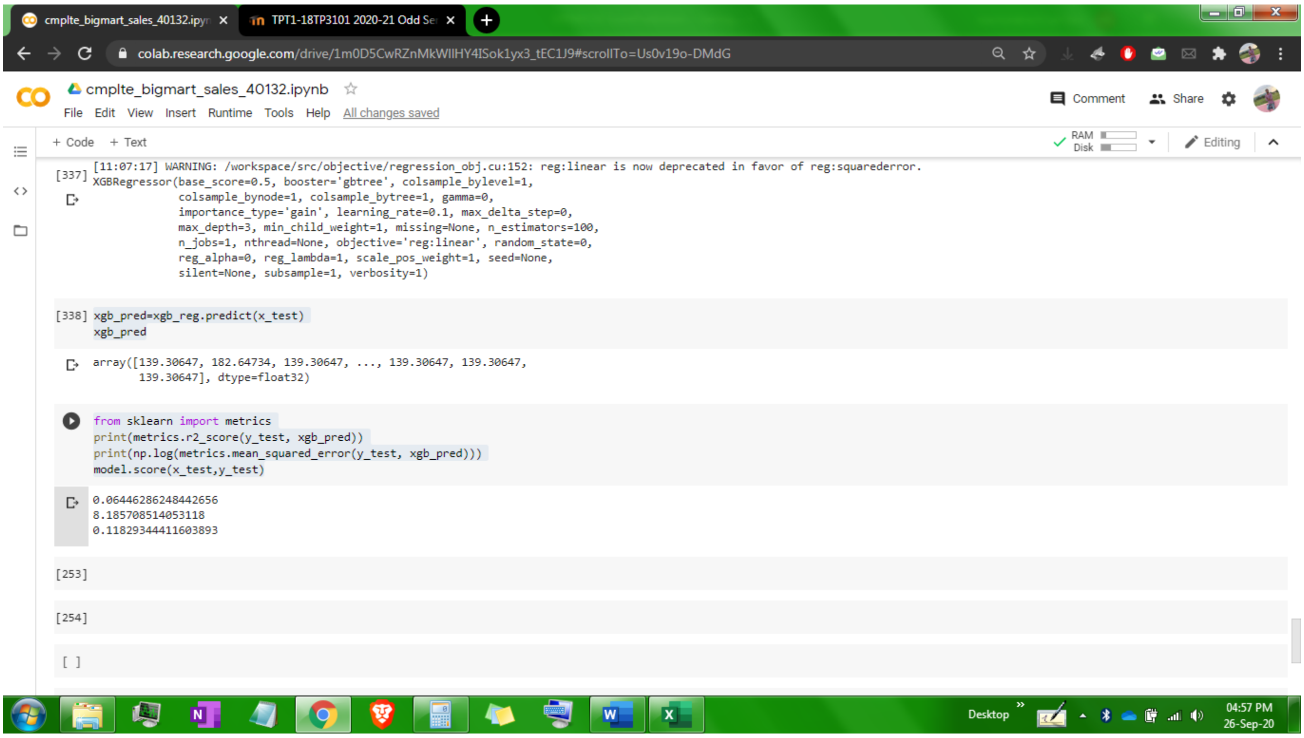
```
[337] from xgboost.sklearn import XGBRegressor  # importing extreme gradient boosting
xgb_reg=XGBRegressor()
xgb_reg.fit(x_train,y_train)

[11:07:17] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
```



cmplte_bigmart_sales_40132.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
[336] reg.score(x_test,y_pred)

1.0

[337] from xgboost.sklearn import XGBRegressor  # importing extreme gradient boosting
      xgb_reg=XGBRegressor()
      xgb_reg.fit(x_train,y_train)

[11:07:17] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=1, verbosity=1)

      xgb_pred=xgb_reg.predict(x_test)
      xgb_pred

array([139.30647, 182.64734, 139.30647, ..., 139.30647, 139.30647,
       139.30647], dtype=float32)

[339] from sklearn import metrics
      print(metrics.r2_score(y_test, xgb_pred))
      print(np.log(metrics.mean_squared_error(y_test, xgb_pred)))
      model.score(x_test,y_test)

0.06446286248442656
8.185708514053118
0.11829344411603893
```



cmplte_bigmart_sales_40132.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
[337]
[11:07:17] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=1, verbosity=1)

[338] xgb_pred=xgb_reg.predict(x_test)
      xgb_pred

array([139.30647, 182.64734, 139.30647, ..., 139.30647, 139.30647,
       139.30647], dtype=float32)

      from sklearn import metrics
      print(metrics.r2_score(y_test, xgb_pred))
      print(np.log(metrics.mean_squared_error(y_test, xgb_pred)))
      model.score(x_test,y_test)

0.06446286248442656
8.185708514053118
0.11829344411603893

[253]

[254]

[ ]
```