+ Code + Text

RAM Disk

Editing

Importing The Dependencies

```
[1] 1 import numpy as np
    2 import pandas as pd
    3 import matplotlib.pyplot as plt
    4 import seaborn as sns
    5 from sklearn.preprocessing import LabelEncoder
    6 from sklearn.model_selection import train_test_split
    7 from xgboost import XGBRegressor
    8 from sklearn import metrics
```

Data Collection & Analysis

```
1 # loading the dataset from csv file to a Pandas DataFrame
2 big_mart_data = pd.read_csv('/content/Train.csv')
```

```
1
```

+ Code + Text

RAM Disk

Editing

Data Collection & Analysis

```
[2] 1 # loading the dataset from csv file to a Pandas DataFrame
    2 big_mart_data = pd.read_csv('/content/Train.csv')
```

```
1 # first 5 rows of the dataframe
2 big_mart_data.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Ty
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tie
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tie
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tie
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tie
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tie

+ Code + Text

RAM Disk

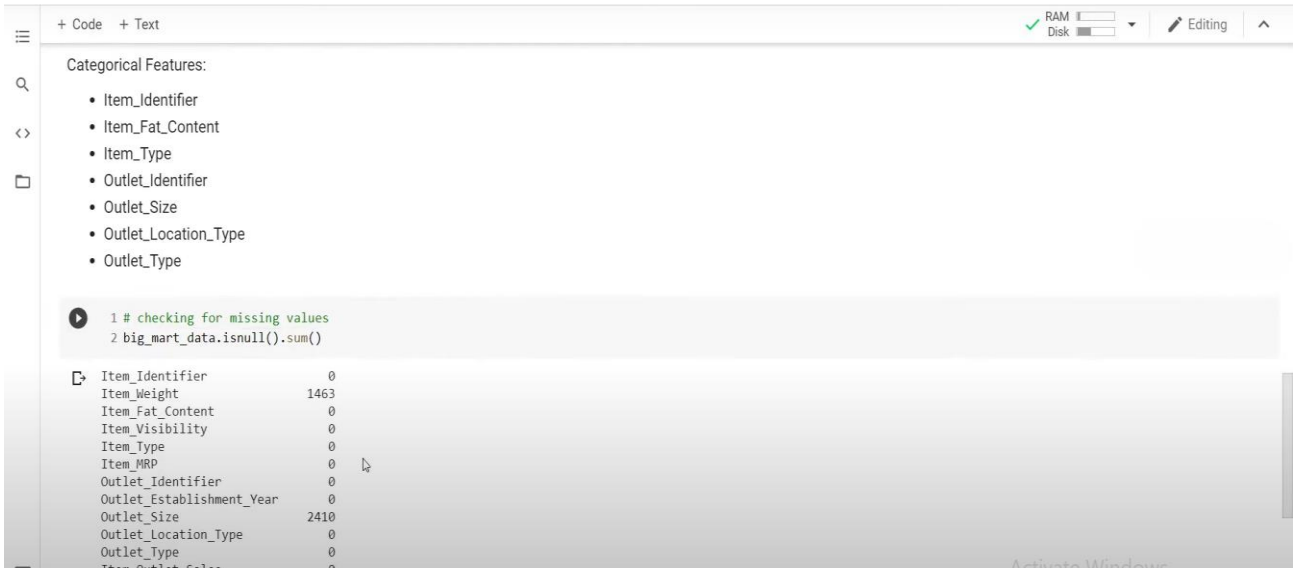
Editing

```
[4] 1 # number of data points & number of Features
    2 big_mart_data.shape
```

```
(8523, 12)
```

```
1 # getting some informations about the dataset
2 big_mart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
```



+ Code + Text

RAM Disk

Editing

Categorical Features:

- Item_Identifier
- Item_Fat_Content
- Item_Type
- Outlet_Identifier
- Outlet_Size
- Outlet_Location_Type
- Outlet_Type

```
1 # checking for missing values
2 big_mart_data.isnull().sum()
```

```
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
```

+ Code + Text

RAM Disk

Editing

```
[6] Item_MRP                      0
    Outlet_Identifier             0
    Outlet_Establishment_Year     0
    Outlet_Size                2410
    Outlet_Location_Type          0
    Outlet_Type                   0
    Item_Outlet_Sales             0
    dtype: int64
```

Handling Missing Values

Mean --> average value

Mode --> Most repeated value

```
1 # mean value of "Item_Weight" column
2 big_mart_data['Item_Weight'].mean()
```

```
12.857645184136183
```

```
1
```

+ Code + Text

RAM Disk

Editing

Mode --> Most repeated value

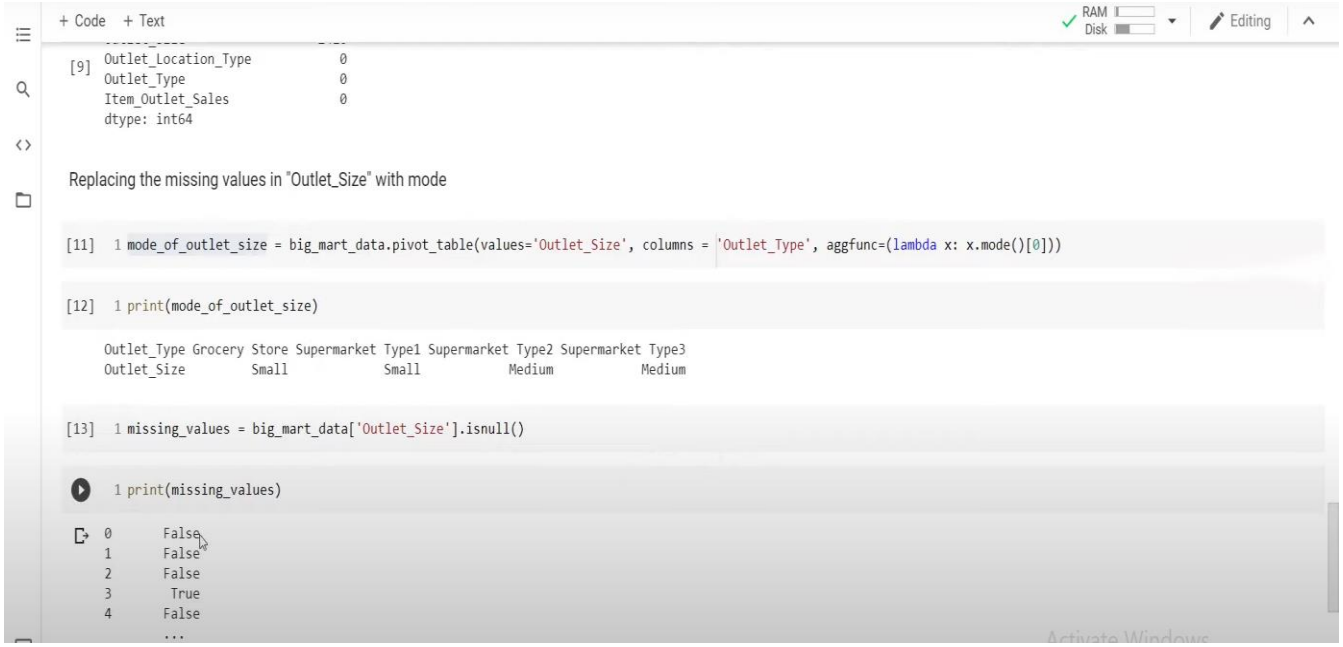
```
[7] 1 # mean value of "Item_Weight" column
    2 big_mart_data['Item_Weight'].mean()
```

```
12.857645184136183
```

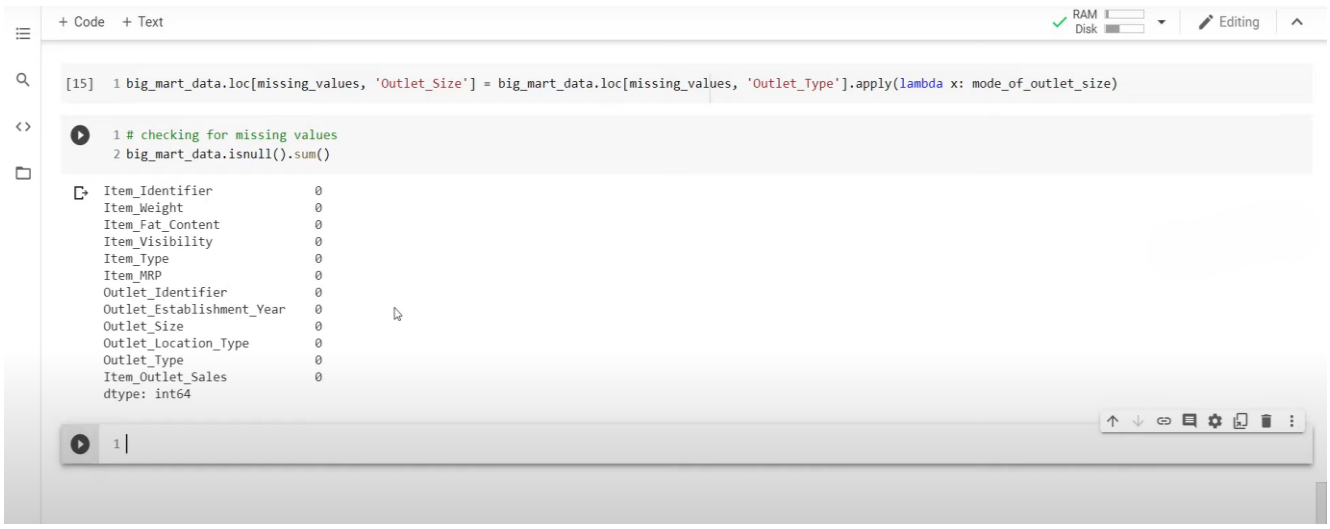
```
[8] 1 # filling the missing values in "Item_Weight" column with "Mean" value
    2 big_mart_data['Item_Weight'].fillna(big_mart_data['Item_Weight'].mean(), inplace = True)
```

```
1 # checking for missing values
2 big_mart_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                2410
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
```


+ Code + Text
RAM
Disk
Editing
[9]
Outlet_Location_Type 0
Outlet_Type 0
Item_Outlet_Sales 0
dtype: int64
Replacing the missing values in "Outlet_Size" with mode
[11] 1 mode_of_outlet_size = big_mart_data.pivot_table(values='Outlet_Size', columns = 'Outlet_Type', aggfunc=(lambda x: x.mode()[0]))
[12] 1 print(mode_of_outlet_size)
Outlet_Type Grocery Store Supermarket Type1 Supermarket Type2 Supermarket Type3
Outlet_Size Small Small Medium Medium
[13] 1 missing_values = big_mart_data['Outlet_Size'].isnull()
1 print(missing_values)
0 False
1 False
2 False
3 True
4 False
...


+ Code + Text
RAM
Disk
Editing
[14] 1 print(missing_values)
0 False
1 False
2 False
3 True
4 False
...
8518 False
8519 True
8520 False
8521 False
8522 False
Name: Outlet_Size, Length: 8523, dtype: bool
1 big_mart_data.loc[missing_values, 'Outlet_Size'] = big_mart_data.loc[missing_values, 'Outlet_Type'].apply(lambda x: mode_of_outlet_size)


+ Code + Text
RAM
Disk
Editing
[15] 1 big_mart_data.loc[missing_values, 'Outlet_Size'] = big_mart_data.loc[missing_values, 'Outlet_Type'].apply(lambda x: mode_of_outlet_size)
1 # checking for missing values
2 big_mart_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
Item_Outlet_Sales 0
dtype: int64
1

dtype: int64

Data Analysis

```
# statistical measures about the data
big_mart_data.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.226124	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	9.310000	0.026989	93.826500	1987.000000	834.247400
50%	12.857645	0.053931	143.012800	1999.000000	1794.331000
75%	16.000000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

```
1
```

[17]

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.226124	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	9.310000	0.026989	93.826500	1987.000000	834.247400
50%	12.857645	0.053931	143.012800	1999.000000	1794.331000
75%	16.000000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

Numerical Features

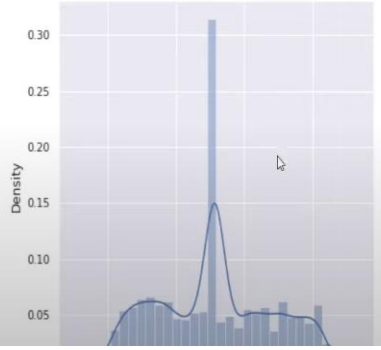
[18]
```
sns.set()
```

[21]
```
# Item_Weight distribution
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_Weight'])
plt.show()
```


[18]
```
sns.set()
```


```
# Item_Weight distribution
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_Weight'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Plea
  warnings.warn(msg, FutureWarning)
```

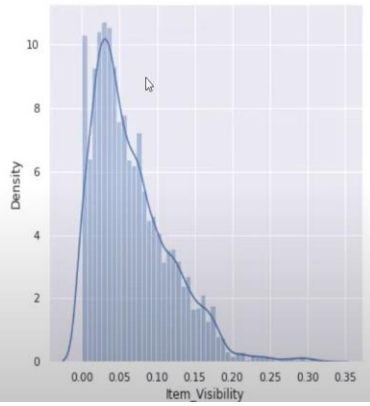


+ Code + Text

RAM Disk Editing

```
# Item_Visibility distribution
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_Visibility'])
plt.show()
```

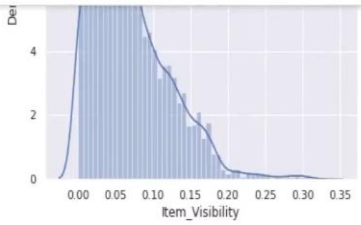
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Plea
  warnings.warn(msg, FutureWarning)
```



+ Code + Text

RAM Disk Editing

[22]




```
# Item_MRP distribution
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_MRP'])
plt.show()
```

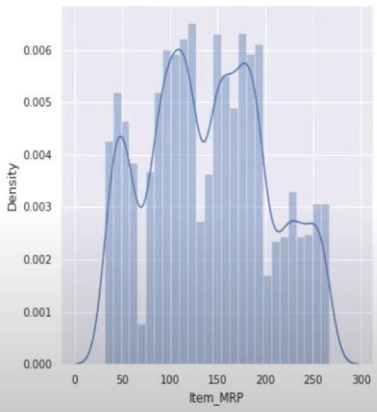
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Plea
  warnings.warn(msg, FutureWarning)
```

+ Code + Text

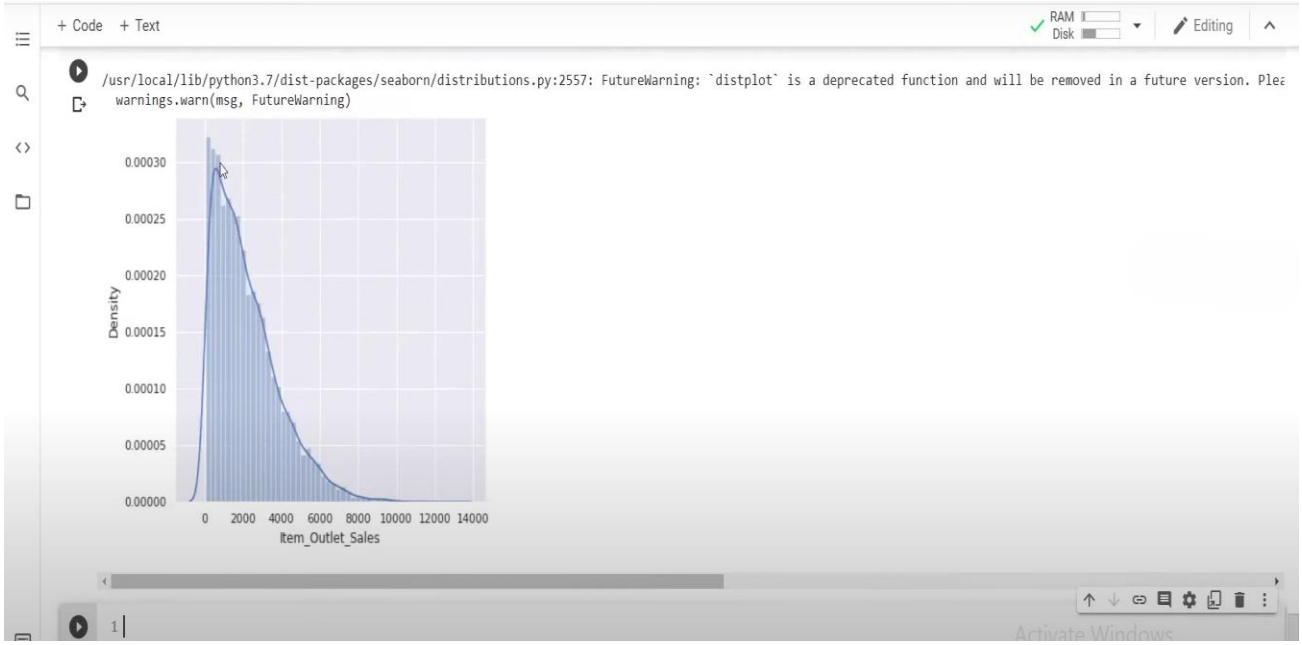
RAM Disk Editing

```
plt.figure(figsize=(6,6))
sns.distplot(big_mart_data['Item_MRP'])
plt.show()
```

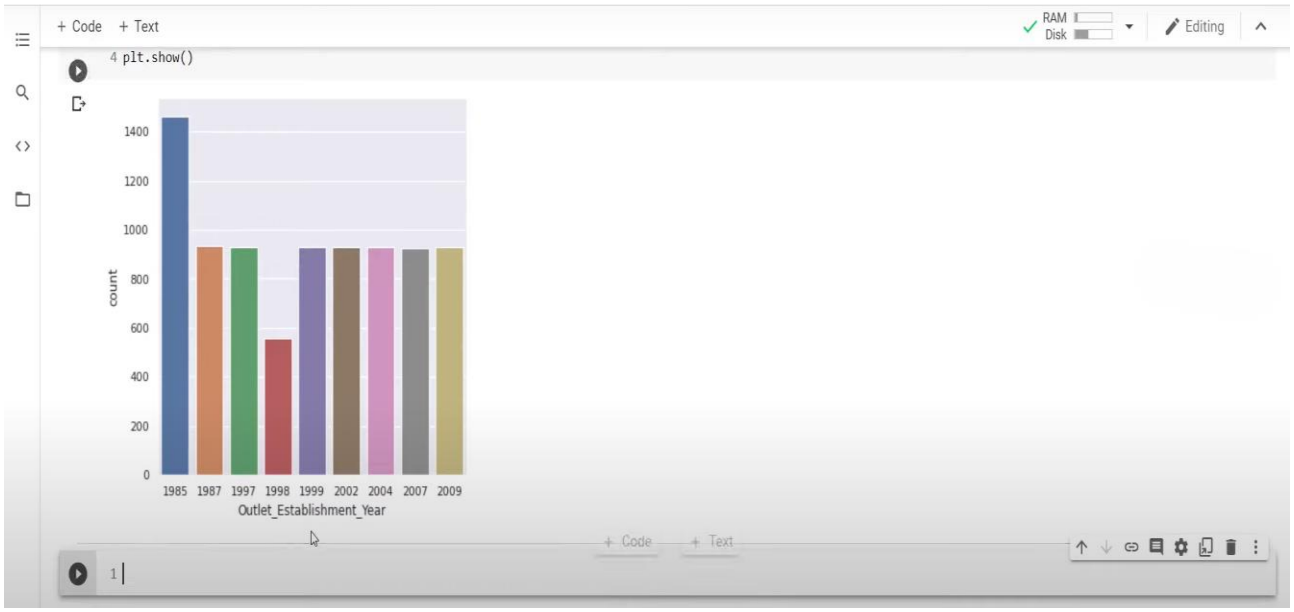
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Plea
  warnings.warn(msg, FutureWarning)
```

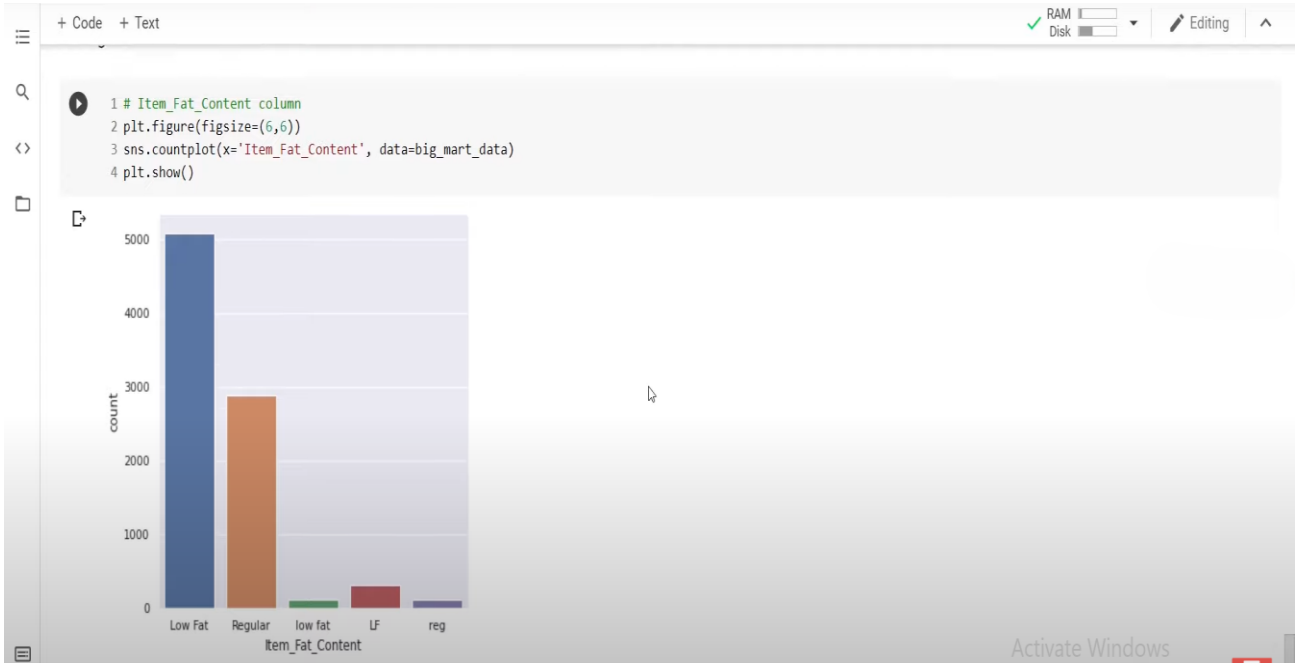


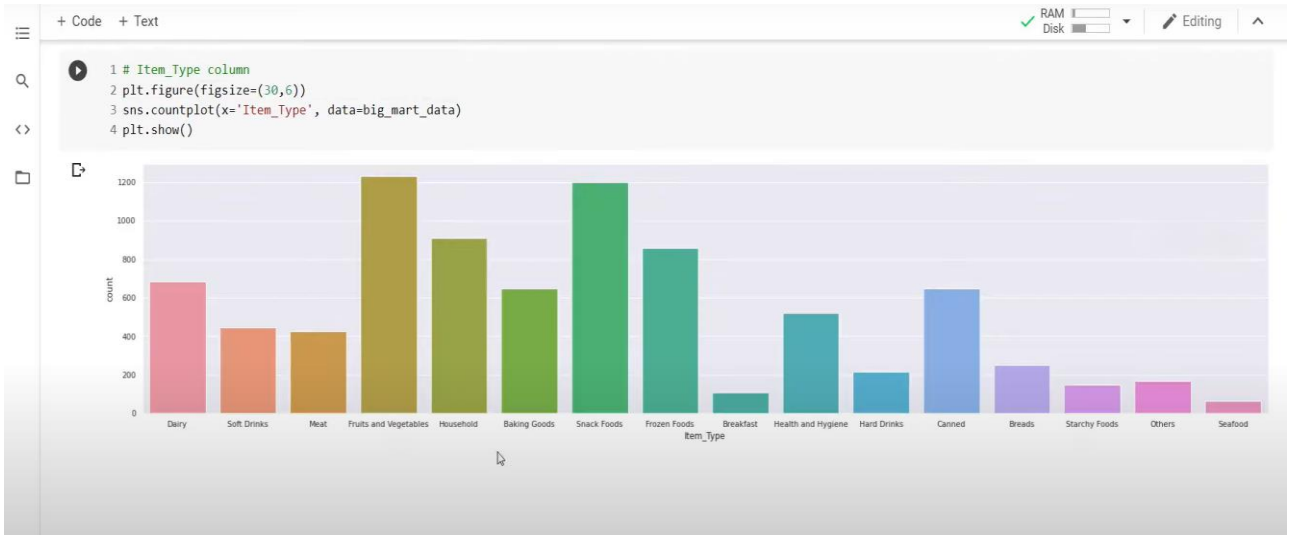

+ Code
+ Text
RAM
Disk
Editing
[23]
0.001
0.000
0
50
100
150
200
250
300
Item_MRP
1 # Item_Outlet_Sales distribution
2 plt.figure(figsize=(6,6))
3 sns.distplot(big_mart_data['Item_Outlet_Sales'])
4 plt.show()
1

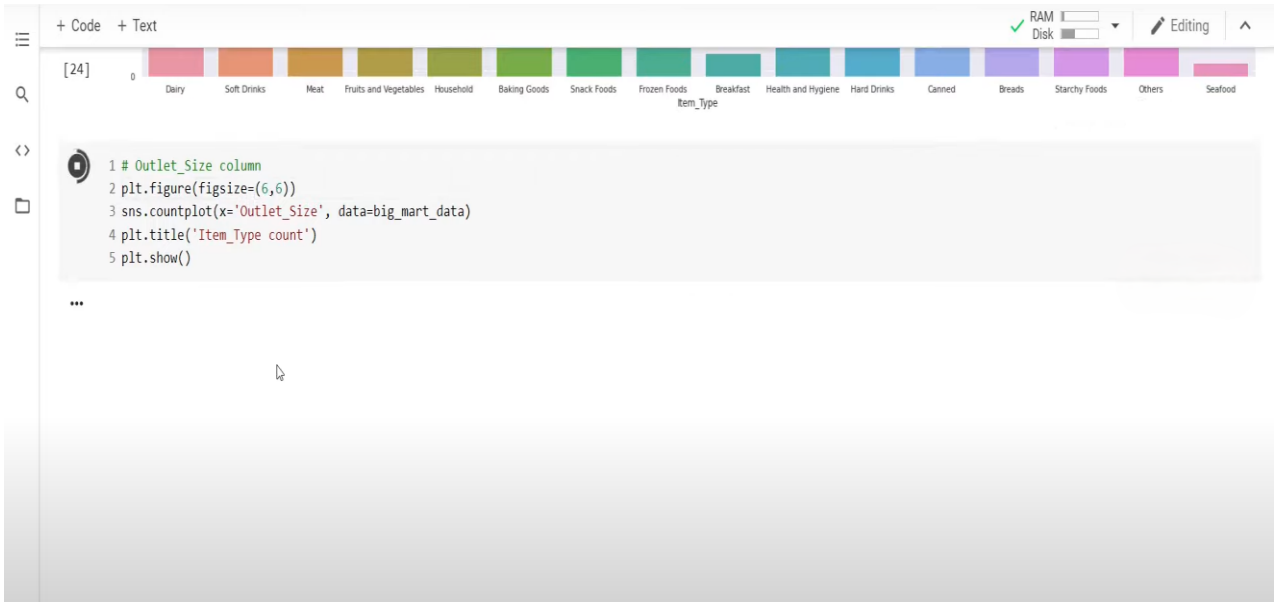

+ Code
+ Text
RAM
Disk
Editing
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Plea
warnings.warn(msg, FutureWarning)
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
Density
0
2000
4000
6000
8000
10000
12000
14000
Item_Outlet_Sales
1

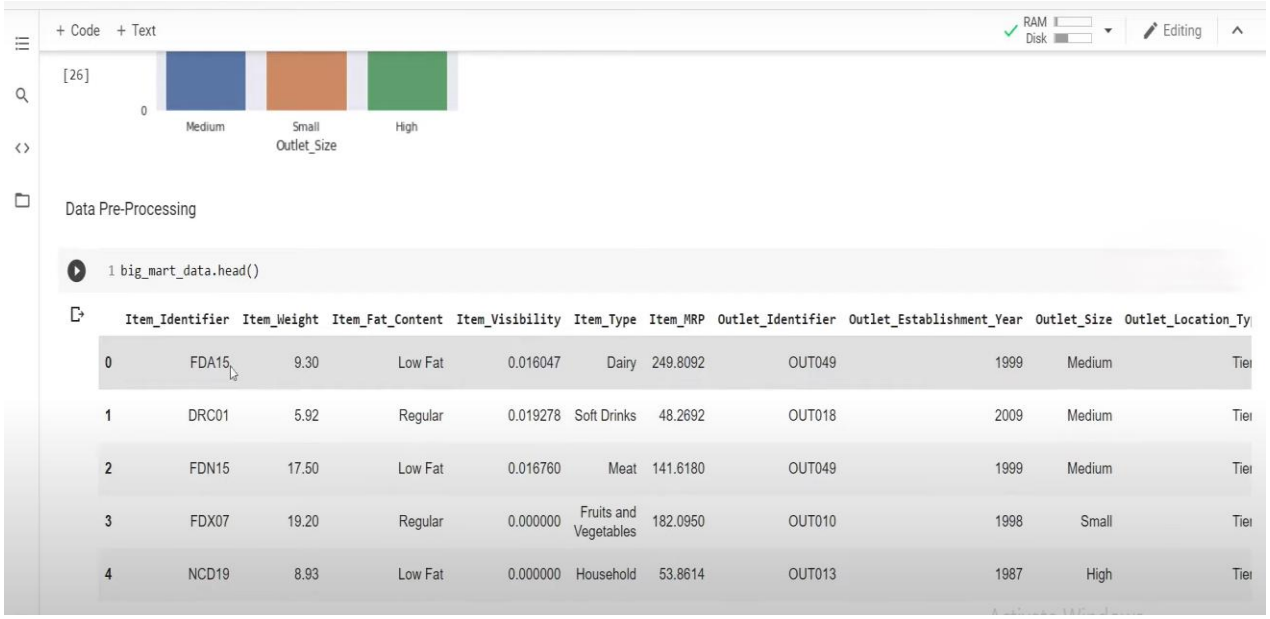

+ Code
+ Text
RAM
Disk
Editing
[24]
0.00000
0
2000
4000
6000
8000
10000
12000
14000
Item_Outlet_Sales
1 # Outlet_Establishment_Year column
2 plt.figure(figsize=(6,6))
3 sns.countplot(x='Outlet_Establishment_Year', data=big_mart_data)
4 plt.show()


+ Code
+ Text
RAM
Disk
Editing
4 plt.show()
1400
1200
1000
800
600
400
200
0
count
1985 1987 1997 1998 1999 2002 2004 2007 2009
Outlet_Establishment_Year
+ Code
+ Text
1


+ Code
+ Text
RAM
Disk
Editing
1 # Item_Fat_Content column
2 plt.figure(figsize=(6,6))
3 sns.countplot(x='Item_Fat_Content', data=big_mart_data)
4 plt.show()
5000
4000
3000
2000
1000
0
count
Low Fat
Regular
low fat
LF
reg
Item_Fat_Content
Activate Windows


+ Code
+ Text
RAM
Disk
Editing
1 # Item_Type column
2 plt.figure(figsize=(30,6))
3 sns.countplot(x='Item_Type', data=big_mart_data)
4 plt.show()
1200
1000
800
600
400
200
0
count
Dairy
Soft Drinks
Meat
Fruits and Vegetables
Household
Baking Goods
Snack Foods
Frozen Foods
Breakfast
Health and Hygiene
Hard Drinks
Canned
Breads
Starchy Foods
Others
Seafood
Item_Type

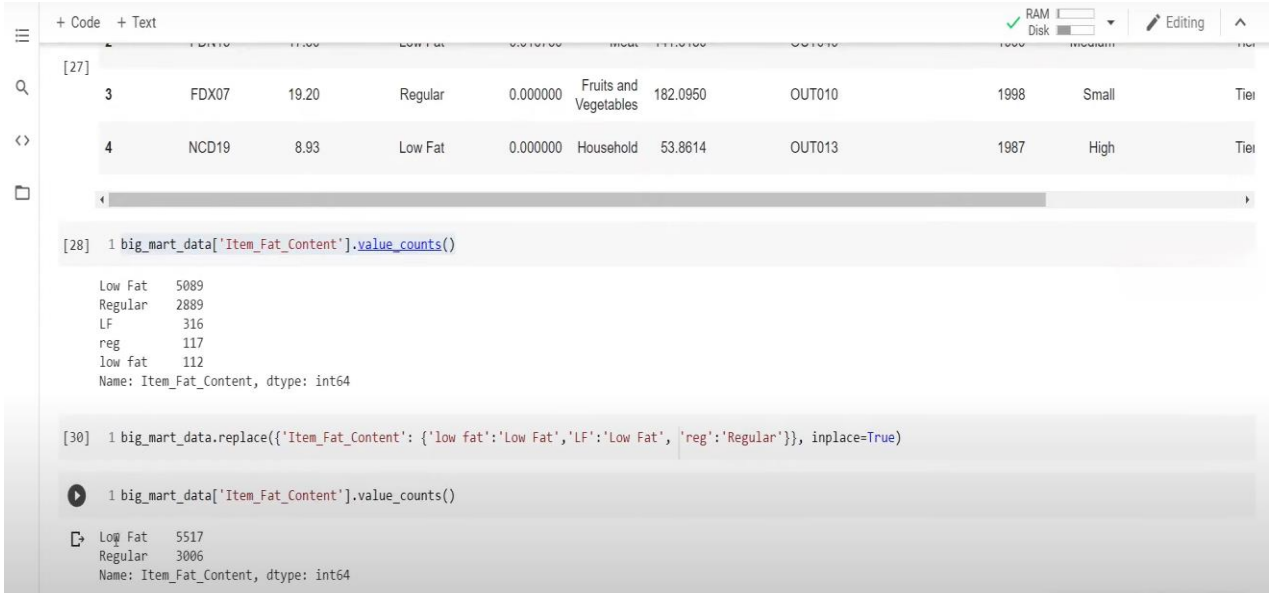




Data Pre-Processing

```
1 big_mart_data.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Ty
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tie
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tie
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tie
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Small	Tie
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tie



[27]

3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Small	Tier
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier

[28]
```
big_mart_data['Item_Fat_Content'].value_counts()
```

```
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64
```

[30]
```
big_mart_data.replace({'Item_Fat_Content': {'low fat':'Low Fat','LF':'Low Fat', 'reg':'Regular'}}, inplace=True)
```

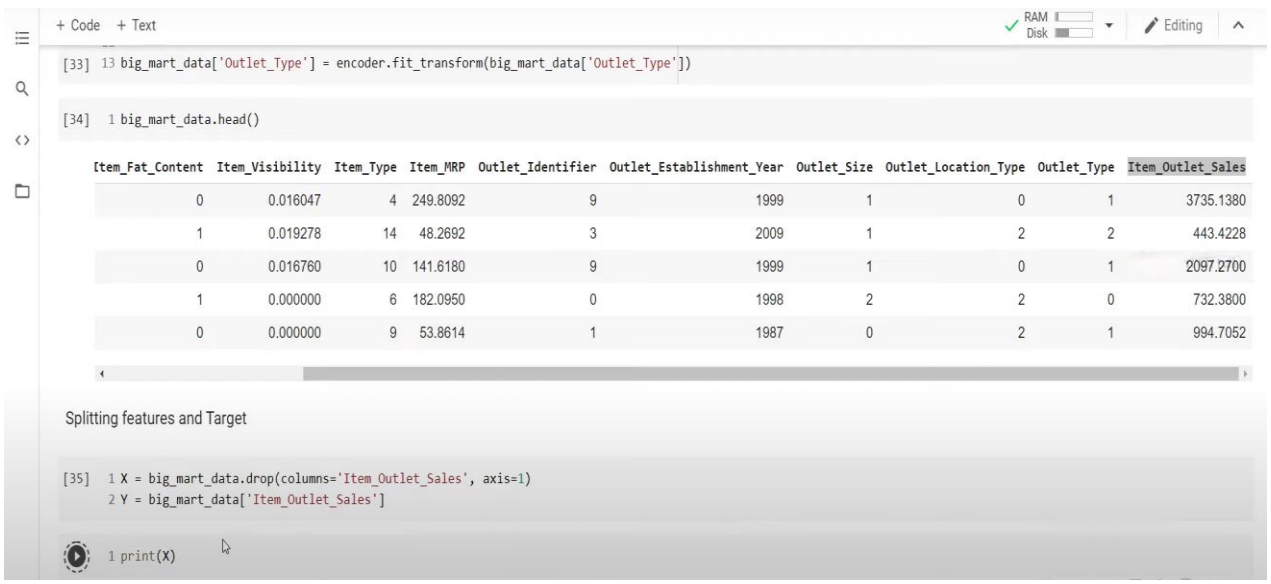
```
big_mart_data['Item_Fat_Content'].value_counts()
```

```
Low Fat    5517
Regular    3006
Name: Item_Fat_Content, dtype: int64
```

Label Encoding

[32]
```
encoder = LabelEncoder()
```

```
big_mart_data['Item_Identifier'] = encoder.fit_transform(big_mart_data['Item_Identifier'])

big_mart_data['Item_Fat_Content'] = encoder.fit_transform(big_mart_data['Item_Fat_Content'])

big_mart_data['Item_Type'] = encoder.fit_transform(big_mart_data['Item_Type'])

big_mart_data['Outlet_Identifier'] = encoder.fit_transform(big_mart_data['Outlet_Identifier'])

big_mart_data['Outlet_Size'] = encoder.fit_transform(big_mart_data['Outlet_Size'])

big_mart_data['Outlet_Location_Type'] = encoder.fit_transform(big_mart_data['Item_Identifier'])

big_mart_data['Item_Identifier'] = encoder.fit_transform(big_mart_data['Item_Identifier'])
```



[33]
```
big_mart_data['Outlet_Type'] = encoder.fit_transform(big_mart_data['Outlet_Type'])
```

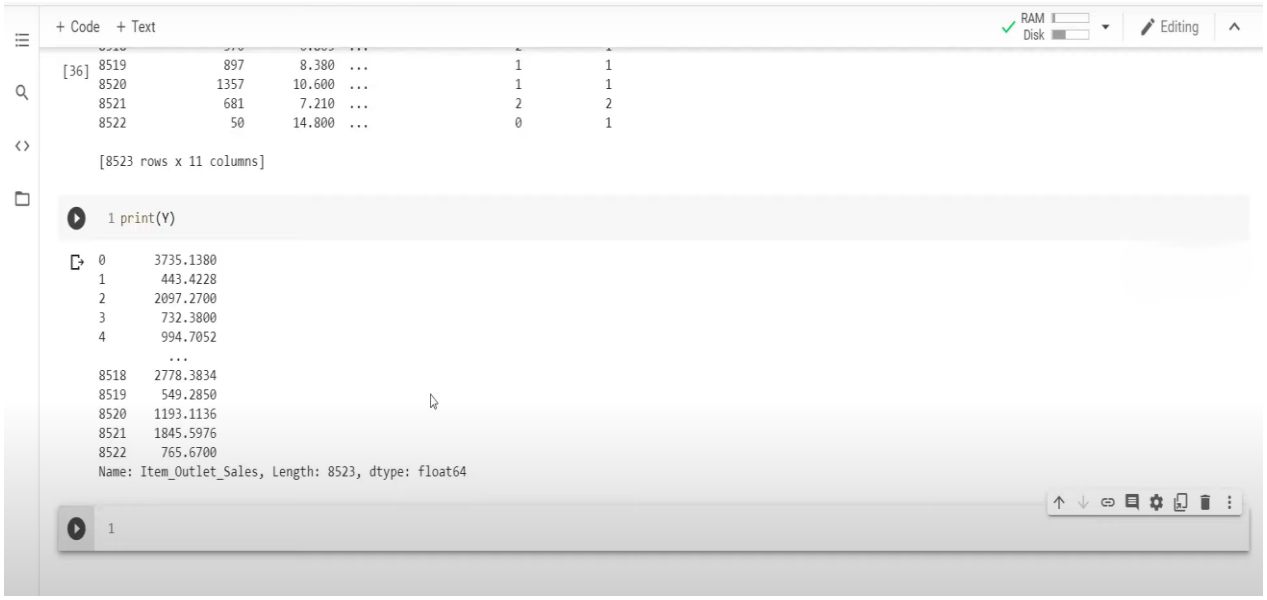
[34]
```
big_mart_data.head()
```

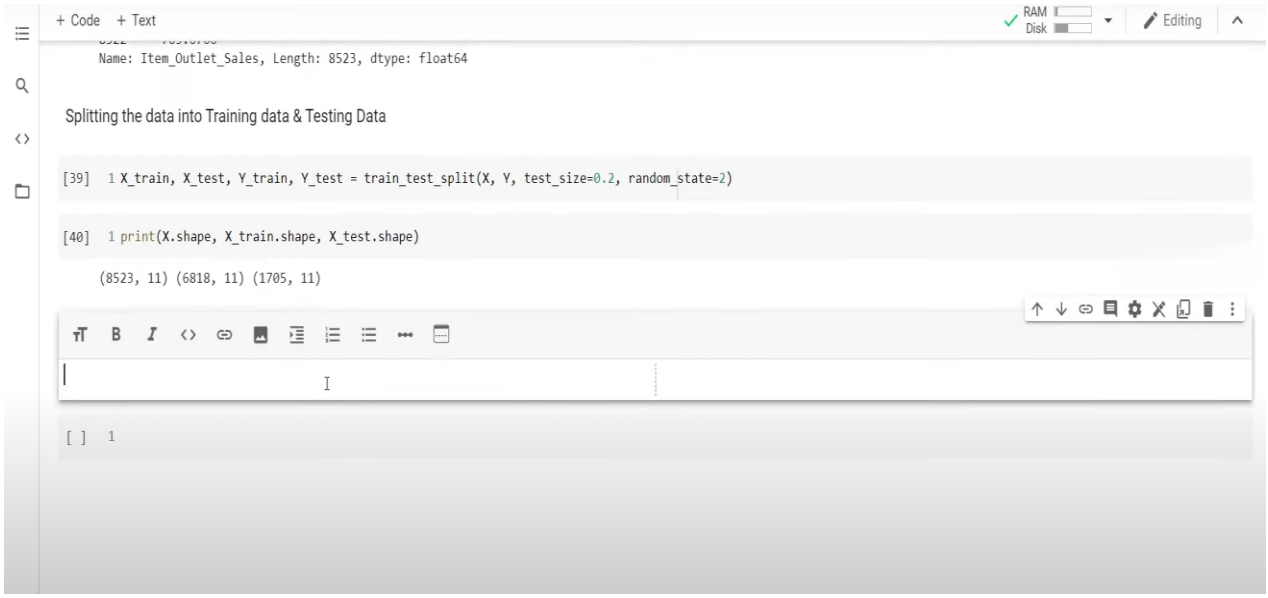
Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	0.016047	4	249.8092	9	1999	1	0	1	3735.1380
1	0.019278	14	48.2692	3	2009	1	2	2	443.4228
0	0.016760	10	141.6180	9	1999	1	0	1	2097.2700
1	0.000000	6	182.0950	0	1998	2	2	0	732.3800
0	0.000000	9	53.8614	1	1987	0	2	1	994.7052

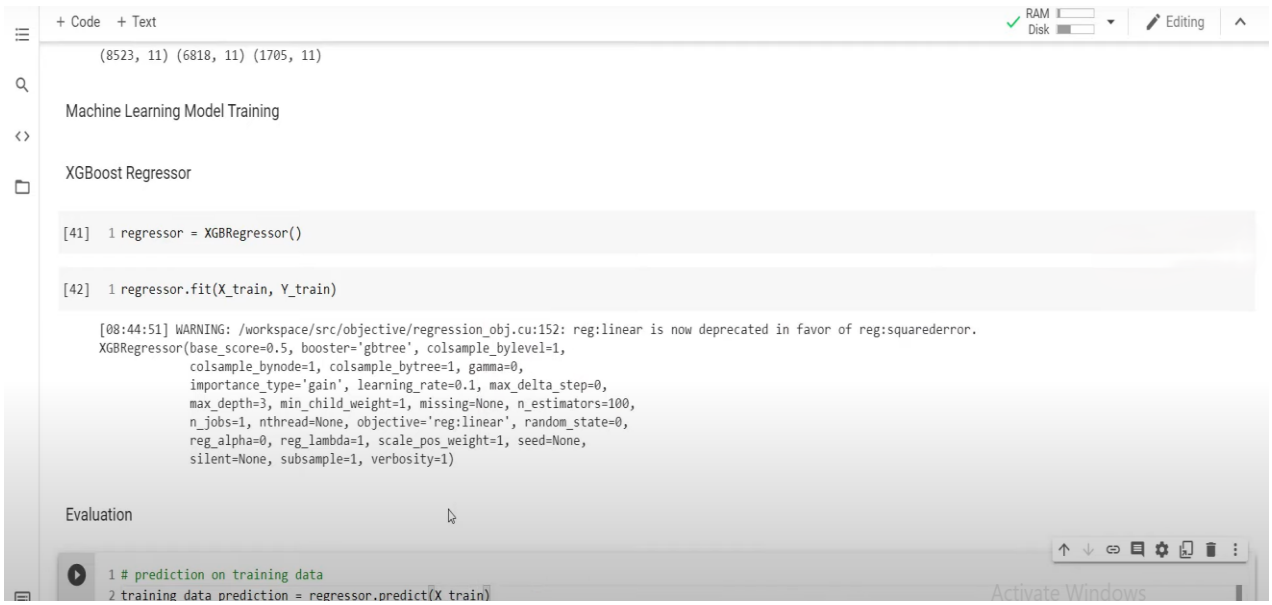
Splitting features and Target

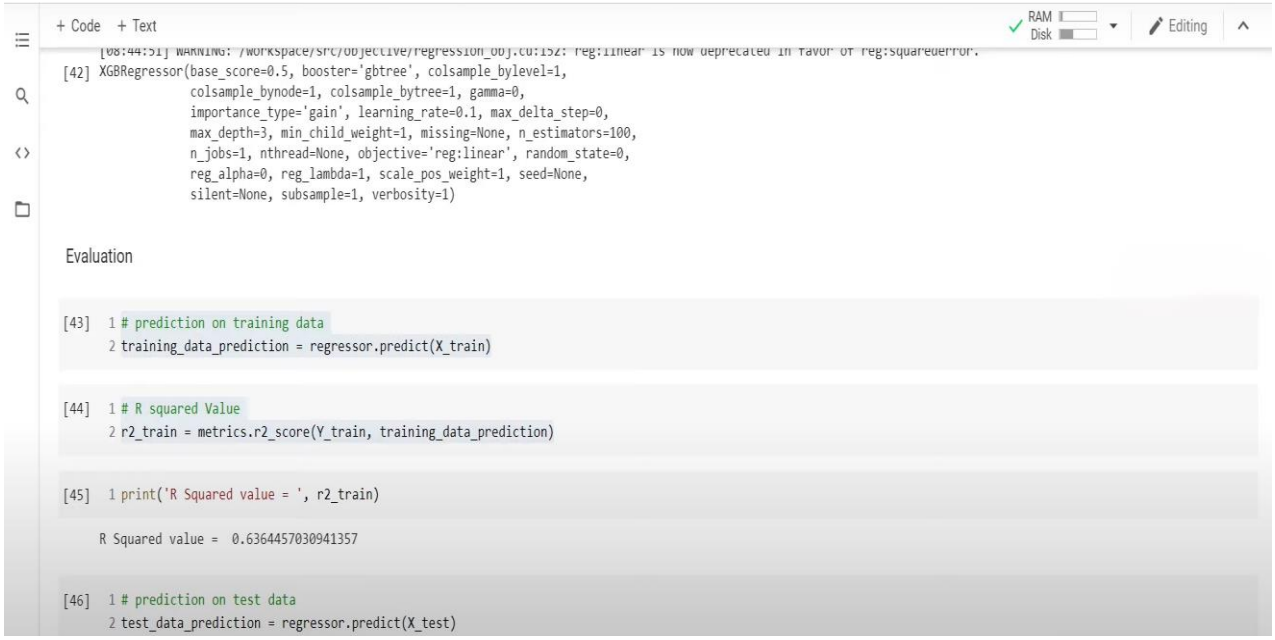
[35]
```
X = big_mart_data.drop(columns='Item_Outlet_Sales', axis=1)
Y = big_mart_data['Item_Outlet_Sales']
```

```
print(X)
```


+ Code + Text
RAM
Disk
Editing
[36]
8519 897 8.380 ... 1 1
8520 1357 10.600 ... 1 1
8521 681 7.210 ... 2 2
8522 50 14.800 ... 0 1
[8523 rows x 11 columns]
1 print(Y)
0 3735.1380
1 443.4228
2 2097.2700
3 732.3800
4 994.7052
...
8518 2778.3834
8519 549.2850
8520 1193.1136
8521 1845.5976
8522 765.6700
Name: Item_Outlet_Sales, Length: 8523, dtype: float64
1


+ Code + Text
RAM
Disk
Editing
Name: Item_Outlet_Sales, Length: 8523, dtype: float64
Splitting the data into Training data & Testing Data
[39] 1 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
[40] 1 print(X.shape, X_train.shape, X_test.shape)
(8523, 11) (6818, 11) (1705, 11)
[ ] 1


+ Code + Text
RAM
Disk
Editing
(8523, 11) (6818, 11) (1705, 11)
Machine Learning Model Training
XGBoost Regressor
[41] 1 regressor = XGBRegressor()
[42] 1 regressor.fit(X_train, Y_train)
[08:44:51] 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)
Evaluation
1 # prediction on training data
2 training data prediction = regressor.predict(X train)



```
[08:44:51] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
[42] 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)
```

Evaluation

```
[43] 1 # prediction on training data
     2 training_data_prediction = regressor.predict(X_train)
```

```
[44] 1 # R squared Value
     2 r2_train = metrics.r2_score(Y_train, training_data_prediction)
```

```
[45] 1 print('R Squared value = ', r2_train)

     R Squared value =  0.6364457030941357
```

```
[46] 1 # prediction on test data
     2 test_data_prediction = regressor.predict(X_test)
```

Evaluation

```
[43] 1 # prediction on training data
     2 training_data_prediction = regressor.predict(X_train)
```

```
[44] 1 # R squared Value
     2 r2_train = metrics.r2_score(Y_train, training_data_prediction)
```

```
[45] 1 print('R Squared value = ', r2_train)

     R Squared value =  0.6364457030941357
```

```
[46] 1 # prediction on test data
     2 test_data_prediction = regressor.predict(X_test)
```

```
     1 # R squared Value
     2 r2_test = metrics.r2_score(Y_test, test_data_prediction)
```

```
[48] 1 print('R Squared value = ', r2_test)

     R Squared value =  0.5867640914432671
```