```
import pandas as pd   #Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error

train_data = pd.read_csv('/content/Sales_Train.csv')  #reading the
training and testing datasets
test_data = pd.read_csv('/content/Sales_Test.csv')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn as sns
from sklearn.metrics import mean_squared_error

print(train_data.head())   #This code prints the first few rows of the
'train_data' DataFrame
```

```
  Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
0           FDA15         9.30          Low Fat         0.016047
1           DRC01         5.92          Regular         0.019278
2           FDN15        17.50          Low Fat         0.016760
3           FDX07        19.20          Regular         0.000000
4           NCD19         8.93          Low Fat         0.000000

                Item_Type  Item_MRP Outlet_Identifier  \
0                   Dairy  249.8092            OUT049
1             Soft Drinks   48.2692            OUT018
2                    Meat  141.6180            OUT049
3  Fruits and Vegetables  182.0950            OUT010
4               Household   53.8614            OUT013

   Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
0                       1999      Medium               Tier 1
1                       2009      Medium               Tier 3
2                       1999      Medium               Tier 1
3                       1998         NaN               Tier 3
4                       1987        High               Tier 3

         Outlet_Type  Item_Outlet_Sales
0  Supermarket Type1          3735.1380
1  Supermarket Type2           443.4228
2  Supermarket Type1          2097.2700
3      Grocery Store           732.3800
4  Supermarket Type1           994.7052
```

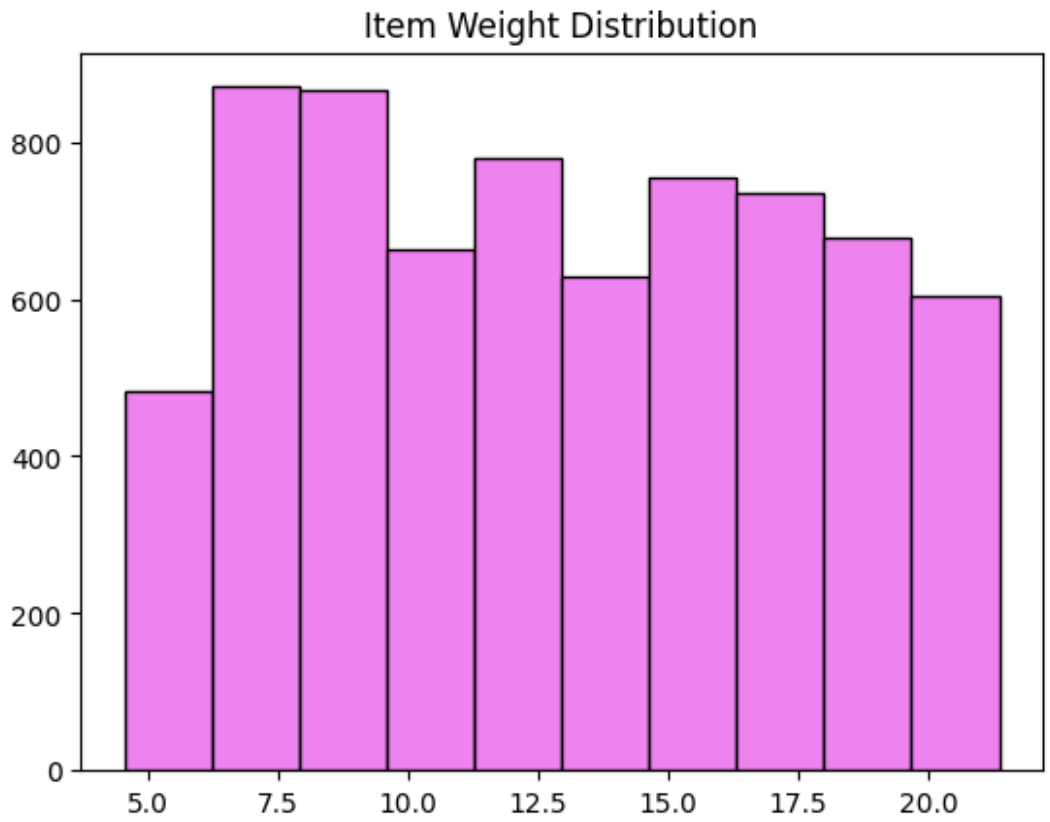
```
print(train_data.describe())
```

```
       Item_Weight  Item_Visibility     Item_MRP
Outlet_Establishment_Year  \
```

```
count  7060.000000      8523.000000  8523.000000
8523.000000
mean     12.857645         0.066132   140.992782
1997.831867
std       4.643456         0.051598    62.275067
8.371760
min       4.555000         0.000000    31.290000
1985.000000
25%       8.773750         0.026989    93.826500
1987.000000
50%      12.600000         0.053931   143.012800
1999.000000
75%      16.850000         0.094585   185.643700
2004.000000
max      21.350000         0.328391   266.888400
2009.000000
```

```
       Item_Outlet_Sales
count        8523.000000
mean         2181.288914
std          1706.499616
min            33.290000
25%           834.247400
50%          1794.331000
75%          3101.296400
max         13086.964800
```

```
plt.hist(train_data['Item_Weight'],color='violet',edgecolor='black')
#Histogram
plt.title('Item Weight Distribution')
plt.show()
```




```
train_data.isnull().sum()  #This code checks and prints the count of
missing values (null values) for each column
```

```
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
dtype: int64
```

This output shows the count of missing values in each column of the 'train_data' DataFrame. The 'Item_Weight' column has 1463 missing values, and the 'Outlet_Size' column has 2410 missing values.

```
test_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
```

This output indicates the number of missing values in each column of the 'test_data' DataFrame. Specifically, the 'Item_Weight' column has 976 missing values, while the 'Outlet_Size' column has 1606 missing values.

Handling Missing Data

```
train_data.dropna(inplace=True)  #These lines of code remove rows with missing values (NaN)
test_data.dropna(inplace=True)
```

- removes rows in the train_data and test_data that have missing values*

```
train_data.isnull().sum() #After removing missing values, when you check for missing values, you find that there are no missing values (NaN) left

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

test_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

train_data.describe()

       Item_Weight  Item_Visibility     Item_MRP
Outlet_Establishment_Year  \
count  4650.000000      4650.000000  4650.000000
4650.000000
mean     12.898675         0.060700   141.716328
1999.190538
std       4.670973         0.044607    62.420534
7.388800
min       4.555000         0.000000    31.490000
1987.000000
25%       8.770000         0.025968    94.409400
1997.000000
50%      12.650000         0.049655   142.979900
1999.000000
75%      17.000000         0.088736   186.614150
2004.000000
max      21.350000         0.188323   266.888400
2009.000000

       Item_Outlet_Sales  Item_Fat_Content_LF  Item_Fat_Content_Low
Fat  \
count        4650.000000          4650.000000
4650.000000
mean         2272.037489             0.038065
0.596559
std          1497.964740             0.191373
0.490641
min            69.243200             0.000000
0.000000
25%          1125.202000             0.000000
0.000000
50%          1939.808300             0.000000
1.000000
75%          3111.616300             0.000000
1.000000
max         10256.649000             1.000000
1.000000

       Item_Fat_Content_Regular  Item_Fat_Content_low fat  \
count               4650.000000               4650.000000
```

```
mean                    0.338710                    0.011398
std                     0.473322                    0.106162
min                     0.000000                    0.000000
25%                     0.000000                    0.000000
50%                     0.000000                    0.000000
75%                     1.000000                    0.000000
max                     1.000000                    1.000000

       Item_Fat_Content_reg  ...  Item_Type_Soft Drinks  \
count           4650.000000  ...             4650.000000
mean               0.015269  ...                0.051828
std                0.122633  ...                0.221703
min                0.000000  ...                0.000000
25%                0.000000  ...                0.000000
50%                0.000000  ...                0.000000
75%                0.000000  ...                0.000000
max                1.000000  ...                1.000000

       Item_Type_Starchy Foods  Outlet_Size_High
Outlet_Size_Medium  \
count               4650.000000       4650.000000          4650.000000

mean                   0.018710          0.200430             0.399570

std                    0.135512          0.400365             0.489863

min                    0.000000          0.000000             0.000000

25%                    0.000000          0.000000             0.000000

50%                    0.000000          0.000000             0.000000

75%                    0.000000          0.000000             1.000000

max                    1.000000          1.000000             1.000000


       Outlet_Size_Small  Outlet_Location_Type_Tier 1  \
count        4650.000000                   4650.000000
mean            0.400000                      0.400000
std             0.489951                      0.489951
min             0.000000                      0.000000
25%             0.000000                      0.000000
50%             0.000000                      0.000000
75%             1.000000                      1.000000
max             1.000000                      1.000000

       Outlet_Location_Type_Tier 2  Outlet_Location_Type_Tier 3  \
count                  4650.000000                  4650.000000
mean                      0.200000                     0.400000
```

```
std                          0.400043                          0.489951
min                          0.000000                          0.000000
25%                          0.000000                          0.000000
50%                          0.000000                          0.000000
75%                          0.000000                          1.000000
max                          1.000000                          1.000000

       Outlet_Type_Supermarket Type1  Outlet_Type_Supermarket Type2
count                      4650.00000                     4650.00000
mean                          0.80043                        0.19957
std                           0.39972                        0.39972
min                           0.00000                        0.00000
25%                           1.00000                        0.00000
50%                           1.00000                        0.00000
75%                           1.00000                        0.00000
max                           1.00000                        1.00000

[8 rows x 34 columns]

train_data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 4650 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            4650 non-null   object
 1   Item_Weight                4650 non-null   float64
 2   Item_Fat_Content           4650 non-null   object
 3   Item_Visibility            4650 non-null   float64
 4   Item_Type                  4650 non-null   object
 5   Item_MRP                   4650 non-null   float64
 6   Outlet_Identifier          4650 non-null   object
 7   Outlet_Establishment_Year  4650 non-null   int64
 8   Outlet_Size                4650 non-null   object
 9   Outlet_Location_Type       4650 non-null   object
 10  Outlet_Type                4650 non-null   object
 11  Item_Outlet_Sales          4650 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 472.3+ KB

#One-hot encoding applied to categorical features in both train and
test datasets.
train_data = pd.get_dummies(train_data, columns=['Item_Fat_Content',
'Item_Type', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type'])
test_data = pd.get_dummies(test_data, columns=['Item_Fat_Content',
'Item_Type', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type'])
```

This process converts categorical variables into a binary (0 or 1) format, creating new binary columns for each category

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4650 entries, 0 to 8522
Data columns (total 36 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Item_Identifier                   4650 non-null   object 
 1   Item_Weight                       4650 non-null   float64
 2   Item_Visibility                   4650 non-null   float64
 3   Item_MRP                          4650 non-null   float64
 4   Outlet_Identifier                 4650 non-null   object 
 5   Outlet_Establishment_Year         4650 non-null   int64  
 6   Item_Outlet_Sales                 4650 non-null   float64
 7   Item_Fat_Content_LF               4650 non-null   uint8  
 8   Item_Fat_Content_Low Fat          4650 non-null   uint8  
 9   Item_Fat_Content_Regular          4650 non-null   uint8  
 10  Item_Fat_Content_low fat          4650 non-null   uint8  
 11  Item_Fat_Content_reg              4650 non-null   uint8  
 12  Item_Type_Baking Goods            4650 non-null   uint8  
 13  Item_Type_Breads                  4650 non-null   uint8  
 14  Item_Type_Breakfast               4650 non-null   uint8  
 15  Item_Type_Canned                  4650 non-null   uint8  
 16  Item_Type_Dairy                   4650 non-null   uint8  
 17  Item_Type_Frozen Foods            4650 non-null   uint8  
 18  Item_Type_Fruits and Vegetables   4650 non-null   uint8  
 19  Item_Type_Hard Drinks             4650 non-null   uint8  
 20  Item_Type_Health and Hygiene      4650 non-null   uint8  
 21  Item_Type_Household               4650 non-null   uint8  
 22  Item_Type_Meat                    4650 non-null   uint8  
 23  Item_Type_Others                  4650 non-null   uint8  
 24  Item_Type_Seafood                 4650 non-null   uint8  
 25  Item_Type_Snack Foods             4650 non-null   uint8  
 26  Item_Type_Soft Drinks             4650 non-null   uint8  
 27  Item_Type_Starchy Foods           4650 non-null   uint8  
 28  Outlet_Size_High                  4650 non-null   uint8  
 29  Outlet_Size_Medium                4650 non-null   uint8  
 30  Outlet_Size_Small                 4650 non-null   uint8  
 31  Outlet_Location_Type_Tier 1       4650 non-null   uint8  
 32  Outlet_Location_Type_Tier 2       4650 non-null   uint8  
 33  Outlet_Location_Type_Tier 3       4650 non-null   uint8  
 34  Outlet_Type_Supermarket Type1     4650 non-null   uint8  
 35  Outlet_Type_Supermarket Type2     4650 non-null   uint8  
dtypes: float64(4), int64(1), object(2), uint8(29)
memory usage: 422.3+ KB
```

```
#Splitting the train dataset into features (X) and target variable (y)
by excluding certain identifier columns.
from sklearn.model_selection import train_test_split
X = train_data.drop(columns=['Item_Identifier', 'Outlet_Identifier',
```

```
'Item_Outlet_Sales'])
y = train_data['Item_Outlet_Sales']

#Splitting the training dataset for model evaluation, using an 80-20
split ratio.
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

#Creating a K-Nearest Neighbors (KNN) regression model with `k=5`.
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error

k = 5
knn = KNeighborsRegressor(n_neighbors=k) # fitting it to the training
data.
knn.fit(X_train, y_train)
```

```
KNeighborsRegressor()
```

```
y_pred = knn.predict(X_val) #Predicting the target variable (`y_pred`)
for the validation set using the K-Nearest Neighbors (KNN) regression
model.

import numpy as np
rmse = np.sqrt(mean_squared_error(y_val, y_pred)) #Calculating and
printing the root mean squared error (RMSE) of the KNN regression
model for validation data.
print(f'RMSE: {rmse}')
```

```
RMSE: 1143.2309866577314
```

```
# Making predictions on the test data using the trained KNN regression
model.
test_predictions =
knn.predict(test_data.drop(columns=['Item_Identifier',
'Outlet_Identifier']))

# Creating a DataFrame with the predicted sales values for test items
and saving it as a CSV file for submission.
submission_data = pd.DataFrame({
    'Item_Identifier': test_data['Item_Identifier'],
    'Outlet_Identifier': test_data['Outlet_Identifier'],
    'Item_Outlet_Sales': test_predictions
})

submission_data.to_csv('Sales_Predictions.csv', index=False)

submission_data.to_csv('sales_predictions.csv', index=False) # Saving
the sales predictions to a CSV file for submission, ensuring there is
no index column.
```

```
df=pd.read_csv('sales_predictions.csv')
print(df) # Reading the sales predictions CSV file and printing its
contents
```

```
     Item_Identifier Outlet_Identifier  Item_Outlet_Sales
0              FDW58            OUT049         2009.11808
1              FDH56            OUT046         1978.75760
2              FDL48            OUT018          668.99584
3              FDU11            OUT049         2102.86272
4              DRL59            OUT013          749.15816
...              ...               ...                ...
3094           FDF46            OUT018         1956.91936
3095           DRL35            OUT046          547.02128
3096           FDW46            OUT049         1004.69220
3097           FDB58            OUT046         2103.92800
3098           FDD47            OUT018         2378.10444

[3099 rows x 3 columns]
```