Walmart Sales Forecasting

1.Data Understanding

1.1 Data source : The main description and data of the case study can be found in this link:(https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting)

1.2 Problem Statement: The main problem that I am assigned with is that I have to predict the sales given the data-set. As I can understand from the problem itself is that it is a regression problem. That we have to use regression models in-order to predict the sales from the data-set.

setup configuration

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [70]:
```
stores=pd.read_csv(r'C:\Users\Dell\OneDrive\Desktop\walmart\stores.csv') #storeset
stores
```

Out[70]:

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875
5	6	A	202505
6	7	B	70713
7	8	A	155078
8	9	B	125833
9	10	B	126512
10	11	A	207499
11	12	B	112238
12	13	A	219622
13	14	A	200898
14	15	B	123737
15	16	B	57197
16	17	B	93188
17	18	B	120653
18	19	A	203819
19	20	A	203742
20	21	B	140167
21	22	B	119557
22	23	B	114533
23	24	A	203819
24	25	B	128107
25	26	A	152513
26	27	A	204184
27	28	A	206302
28	29	B	93638
29	30	C	42988
30	31	A	203750
31	32	A	203007
32	33	A	39690
33	34	A	158114
34	35	B	103681
35	36	A	39910
36	37	C	39910
37	38	C	39690
38	39	A	184109
39	40	A	155083
40	41	A	196321
41	42	C	39690
42	43	C	41062
43	44	C	39910
44	45	B	118221

In [41]:

```
train=pd.read_csv(r'C:\Users\Dell\OneDrive\Desktop\walmart\train.csv') #test dataset
train
```

Out[41]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False
...	...	...	...	...	...
421565	45	98	2012-09-28	508.37	False
421566	45	98	2012-10-05	628.10	False
421567	45	98	2012-10-12	1061.02	False
421568	45	98	2012-10-19	760.01	False
421569	45	98	2012-10-26	1076.80	False

421570 rows × 5 columns

In [43]:

```
feature=pd.read_csv(r'C:\Users\Dell\OneDrive\Desktop\walmart\features.csv') #External dataset
feature
```

Out[43]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsH
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	
4	1	2010-03-05	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	
...	...	...	...	...	...	...	...	...	...	...	...	
8185	45	2013-06-28	76.05	3.639	4842.29	975.03	3.00	2449.97	3169.69	NaN	NaN	
8186	45	2013-07-05	77.50	3.614	9090.48	2268.58	582.74	5797.47	1514.93	NaN	NaN	
8187	45	2013-07-12	79.37	3.614	3789.94	1827.31	85.72	744.84	2150.36	NaN	NaN	
8188	45	2013-07-19	82.84	3.737	2961.49	1047.07	204.19	363.00	1059.46	NaN	NaN	
8189	45	2013-07-26	76.06	3.804	212.02	851.73	2.06	10.88	1864.57	NaN	NaN	

8190 rows × 12 columns

Displaying first 5 rows

In [44]:

```
feature.head() #returns the First 5 rows of the dataset
```

Out[44]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHolic
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	Fa
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	T
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	Fa
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	Fa
4	1	2010-03-05	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	Fa

Displaying last 5 rows

In [75]:

```
stores.tail()#returns the last 5 rows of the store dataset
```

Out[75]:

	Store	Type	Size
40	41	A	196321
41	42	C	39690
42	43	C	41062
43	44	C	39910
44	45	B	118221

In [76]:

```
train.tail() #returns the last 5 rows of the  train dataset
```

Out[76]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
421565	45	98	2012-09-28	508.37	False
421566	45	98	2012-10-05	628.10	False
421567	45	98	2012-10-12	1061.02	False
421568	45	98	2012-10-19	760.01	False
421569	45	98	2012-10-26	1076.80	False

In [82]:

```
feature.tail() #returns the last 5 rows of the  feature dataset
```

Out[82]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsH
8168	45	2013-03-01	39.72	3.890	6614.32	147.82	5.60	27.55	1668.95	193.122173	8.625	
8169	45	2013-03-08	36.13	3.860	16382.54	88.67	34.62	3096.92	3486.91	193.211524	8.625	
8173	45	2013-04-05	43.94	3.763	16427.83	5341.41	182.59	1523.83	1743.09	193.516047	8.335	
8174	45	2013-04-12	57.39	3.724	8760.15	1713.11	21.08	1302.31	1380.74	193.589304	8.335	
8175	45	2013-04-19	56.27	3.676	1399.81	39.89	44.38	60.83	1445.05	193.589304	8.335	

Merging all three Datasets

In [81]:

```
new_data = pd.merge(feature, train, on=['Store','Date','IsHoliday'], how='inner')
# merging(adding) all stores info with new training data
final_data = pd.merge(new_data,stores,how='inner',on=['Store'])
final_data.head()
```

Out[81]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHolic
0	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa
1	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa
2	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa
3	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa
4	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa

Displaying column headings

In [83]:

```
final_data.columns #Displaying Columns names
```

Out[83]:

```
Index(['Store', 'Date', 'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2',
       'MarkDown3', 'MarkDown4', 'MarkDown5', 'CPI', 'Unemployment',
       'IsHoliday', 'Dept', 'Weekly_Sales', 'Type', 'Size'],
      dtype='object')
```

Displaying Statistical information

In [84]:

```
final_data.shape #returns the dimensions of the dataframe
```

Out[84]:

```
(97056, 16)
```

In [85]:

```
final_data.describe() #shows count, mean, std etc. for each column
```

Out[85]:

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	U
count	97056.000000	97056.000000	97056.000000	97056.000000	97056.000000	97056.000000	97056.000000	97056.000000	97056.000000	
mean	20.239408	57.348331	3.618946	8841.260245	3693.532392	1816.629491	4025.923108	5310.830581	174.766754	
std	12.037946	18.263734	0.280003	9258.091154	10058.901796	10989.284083	7173.060535	6535.397883	39.652638	
min	1.000000	7.460000	3.031000	32.500000	-265.760000	-29.100000	0.460000	170.640000	129.816710	
25%	10.000000	42.750000	3.413000	3600.790000	47.550000	5.400000	605.880000	2383.670000	136.856419	
50%	20.000000	57.950000	3.630000	6264.180000	192.000000	30.460000	1739.830000	3864.600000	189.194056	
75%	29.000000	72.660000	3.820000	10333.240000	2551.320000	123.420000	4082.990000	6197.530000	219.355063	
max	45.000000	95.910000	4.301000	88646.760000	104519.540000	141630.610000	67474.850000	108519.280000	227.036936	

In [86]:

```
final_data.max() #returns max value for all columns
```

Out[86]:

```
Store                    45
Date             2012-10-26
Temperature           95.91
Fuel_Price            4.301
MarkDown1          88646.76
MarkDown2         104519.54
MarkDown3         141630.61
MarkDown4          67474.85
MarkDown5         108519.28
CPI              227.036936
Unemployment          12.89
IsHoliday              True
Dept                     99
Weekly_Sales      630999.19
Type                      C
Size                 219622
dtype: object
```

In [87]:

```
final_data['Temperature'].max() #returns max value for that column
```

Out[87]:

```
95.91
```

Data Preparation

Duplicate data

In [88]:

```
final_data.duplicated() #Lets you remove identical rows.
```

Out[88]:

```
0        False
1        False
2        False
3        False
4        False
         ...  
97051    False
97052    False
97053    False
97054    False
97055    False
Length: 97056, dtype: bool
```

Missing Data

Data cleaning : The following isnull function will figure out if there are any missing values in the dataframe, and will then sum up the total for each column. **dropna() —** This function allows you to drop all(or some) of the rows that have missing values.</br> **fillna() —**This function allows you to replace the rows that have missing values with the value that you pass in.

In [89]:

```
final_data=final_data.dropna(how='any',axis=0) #removing all Nan values
final_data.head()
```

Out[89]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHolic
0	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa
1	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa
2	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa
3	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa
4	1	2011-11-11	59.11	3.297	10382.9	6115.67	215.07	2406.62	6551.42	217.998085	7.866	Fa

Checking for null values

In [90]:

```
print(final_data.isnull().sum())
```

```
Store           0
Date            0
Temperature     0
Fuel_Price      0
MarkDown1       0
MarkDown2       0
MarkDown3       0
MarkDown4       0
MarkDown5       0
CPI             0
Unemployment    0
IsHoliday       0
Dept            0
Weekly_Sales    0
Type            0
Size            0
dtype: int64
```

In [153…

```
grouped=stores.groupby('Type')
print(grouped.describe()['Size'].round(2))
```

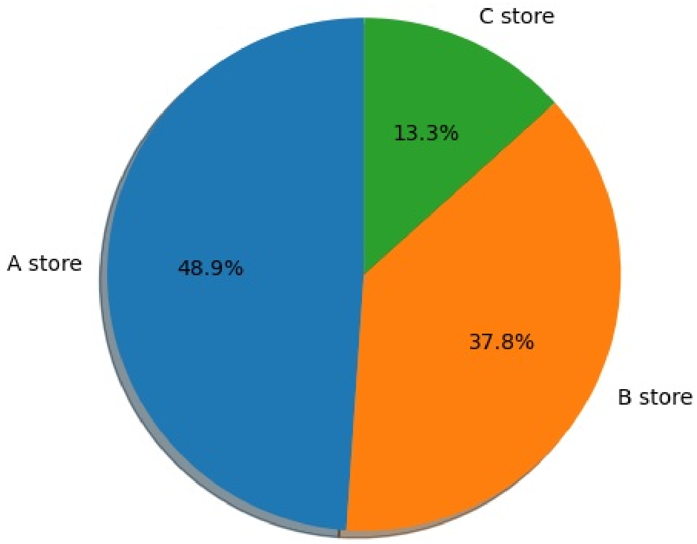
```
      count       mean       std      min        25%       50%       75%  \
Type
A      22.0  177247.73  49392.62  39690.0  155840.75  202406.0  203819.0
B      17.0  101190.71  32371.14  34875.0   93188.00  114533.0  123737.0
C       6.0   40541.67   1304.15  39690.0   39745.00   39910.0   40774.0

           max
Type
A     219622.0
B     140167.0
C      42988.0
```

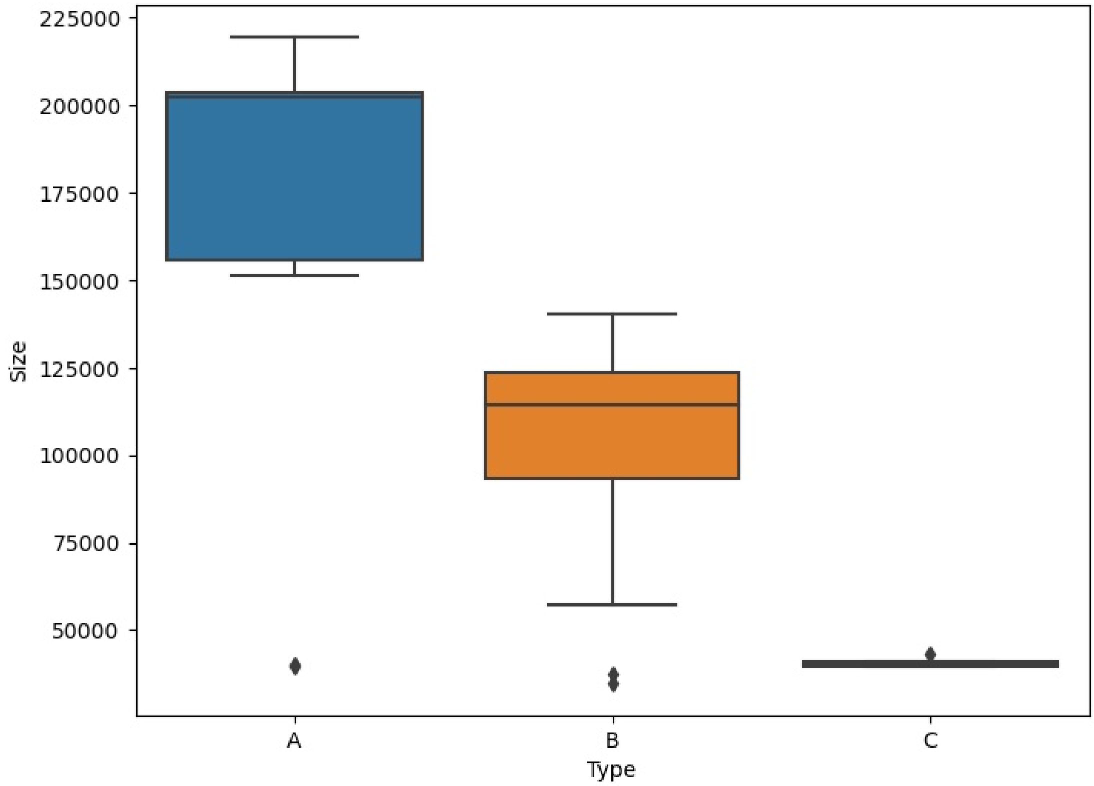
Data Exploration using various plots

In [104…

```
# Equal aspect ratio ensures that pie is drawn as a circle.
labels = 'A store','B store','C store'
sizes = [(22/(45))*100,(17/(45))*100,(6/(45))*100]
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.show()
```



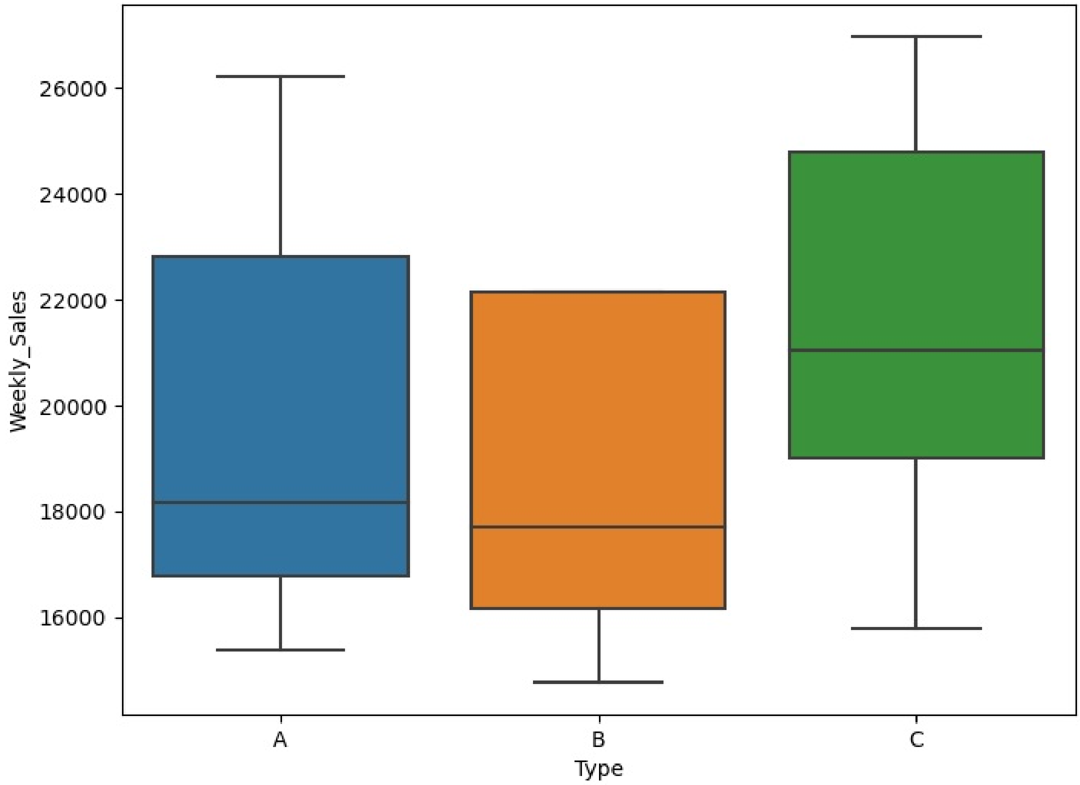
In [102…
```
# boxplot for sizes of types of stores
store_type = pd.concat([stores['Type'], stores['Size']], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x='Type', y='Size', data=store_type)
```



By boxplot and piechart, we can say that type A store is the largest store and C is the smallest There is no overlapped area in size among A, B, and C.

boxplot for weekly sales for different types of stores :

In [106…
```
store_sale = pd.concat([stores['Type'], train['Weekly_Sales']], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x='Type', y='Weekly_Sales', data=store_sale, showfliers=False)
```



The median of A is the highest and C is the lowest i.e stores with more sizes have higher sales

Sales on holiday is a little bit more than sales in not-holiday

In [144…

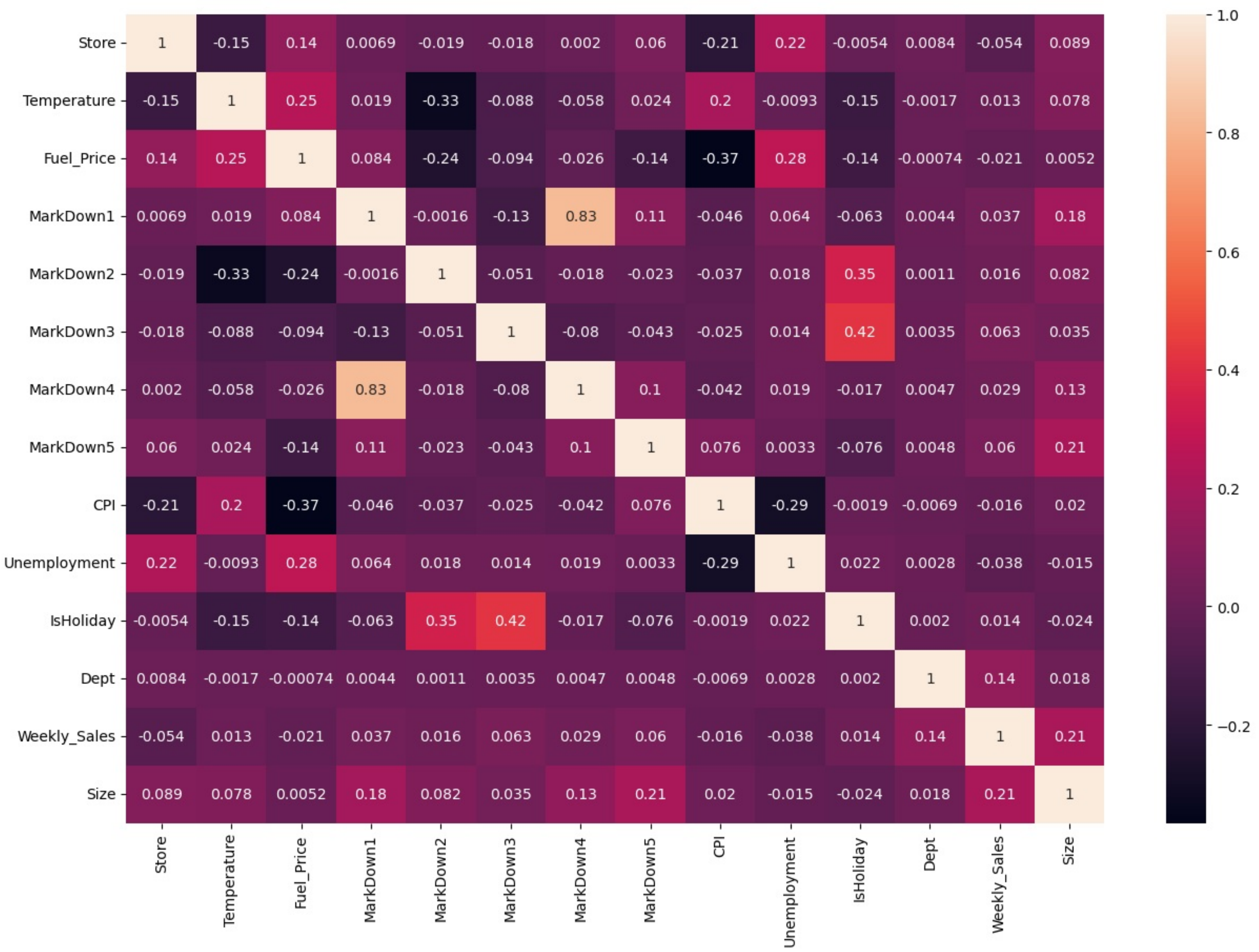
```
# total count of sales on holidays and non holidays
print('sales on non-holiday : ',train[train['IsHoliday']==False]['Weekly_Sales'].count().round(1))
print('sales on holiday : ',train[train['IsHoliday']==True]['Weekly_Sales'].count().round(1))
```

```
sales on non-holiday :  391909
sales on holiday :  29661
```

In [109…

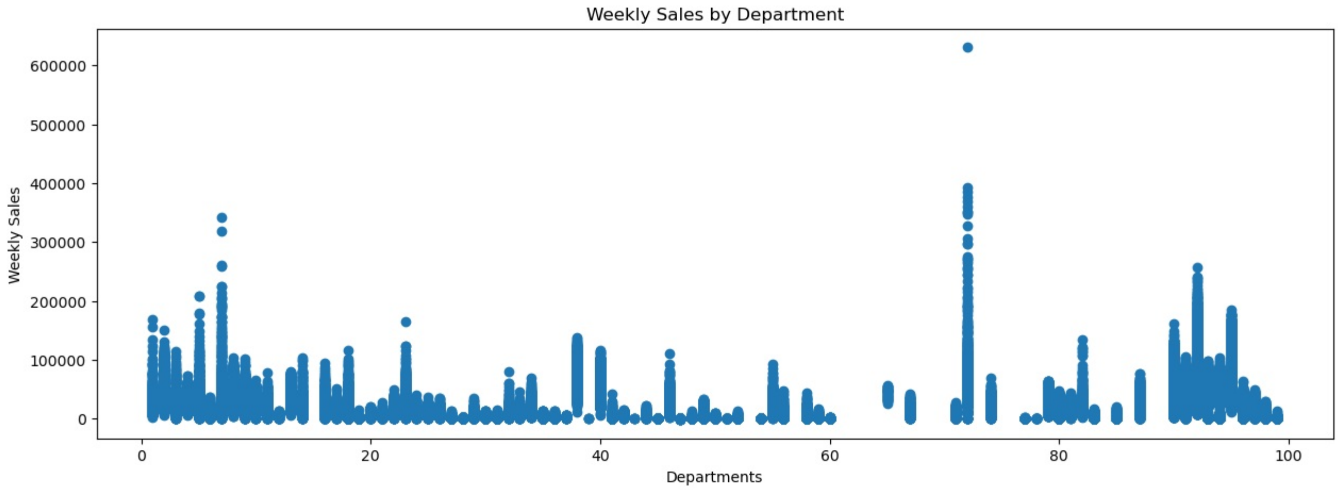
```
# Plotting correlation between all important features
corr = final_data.corr()
plt.figure(figsize=(15, 10))
sns.heatmap(corr, annot=True)
plt.plot()
```

Out[109]: []



In [148…
```
final_data["Year"] = pd.to_datetime(final_data["Date"], format="%Y-%m-%d").dt.year
```

In [146…
```
x = df['Dept']
y = df['Weekly_Sales']
plt.figure(figsize=(15,5))
plt.title('Weekly Sales by Department')
plt.xlabel('Departments')
plt.ylabel('Weekly Sales')
plt.scatter(x,y)
plt.show()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js