Walmart - Store Sales Forecasting

Problem statement

We are provided with historical sales data for 45 Walmart stores located in different regions. Each store contains many departments, and participants must project the sales for each department in each store. To add to the challenge, selected holiday markdown events are included in the dataset. These markdowns are known to affect sales, but it is challenging to predict which departments are affected and the extent of the impact.

Downloading dataset

Before downloading the dataset, we have to install all libraries.

```
pip install numpy pandas matplotlib seaborn plotly sklearn opendatasets xgboost --quiet
```

```
  Building wheel for sklearn (setup.py) ... done
```

```
import os
import opendatasets as od
import pandas as pd
pd.set_option("display.max_columns", 120)
pd.set_option("display.max_rows", 120)
```

```
od.download('https://www.kaggle.com/competitions/walmart-recruiting-store-sales-forecas
```

```
Skipping, found downloaded files in "./walmart-recruiting-store-sales-forecasting" (use
force=True to force download)
```

The dataset is downloaded and extracted to the folder 'bosch-production-line-performance'

```
os.listdir('walmart-recruiting-store-sales-forecasting')
```

```
['stores.csv',
 'test.csv.zip',
 'features.csv.zip',
 'sampleSubmission.csv.zip',
 'train.csv.zip']
```

Reading the dataset

```
train_df = pd.read_csv('./walmart-recruiting-store-sales-forecasting/train.csv.zip')
train_df
```

	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

	Store	Dept	Date	Weekly_Sales	IsHoliday
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

```
features_df = pd.read_csv('./walmart-recruiting-store-sales-forecasting/features.csv.zi
features_df
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.0
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.2
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.2
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.3
4	1	2010-03-05	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.3
...	...	...	...	...	...	...	...	...	...	
8185	45	2013-06-28	76.05	3.639	4842.29	975.03	3.00	2449.97	3169.69	
8186	45	2013-07-05	77.50	3.614	9090.48	2268.58	582.74	5797.47	1514.93	
8187	45	2013-07-12	79.37	3.614	3789.94	1827.31	85.72	744.84	2150.36	
8188	45	2013-07-19	82.84	3.737	2961.49	1047.07	204.19	363.00	1059.46	
8189	45	2013-07-26	76.06	3.804	212.02	851.73	2.06	10.88	1864.57	

8190 rows × 12 columns

```
stores_df = pd.read_csv('./walmart-recruiting-store-sales-forecasting/stores.csv')
stores_df
```

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392

	Store	Type	Size
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

	Store	Type	Size
42	43	C	41062
43	44	C	39910
44	45	B	118221

```
test_df = pd.read_csv('./walmart-recruiting-store-sales-forecasting/test.csv.zip')
test_df
```

	Store	Dept	Date	IsHoliday
0	1	1	2012-11-02	False
1	1	1	2012-11-09	False
2	1	1	2012-11-16	False
3	1	1	2012-11-23	True
4	1	1	2012-11-30	False
...	...	...	...	...
115059	45	98	2013-06-28	False
115060	45	98	2013-07-05	False
115061	45	98	2013-07-12	False
115062	45	98	2013-07-19	False
115063	45	98	2013-07-26	False

115064 rows × 4 columns

```
submission_df = pd.read_csv('./walmart-recruiting-store-sales-forecasting/sampleSubmiss
submission_df
```

	Id	Weekly_Sales
0	1_1_2012-11-02	0
1	1_1_2012-11-09	0
2	1_1_2012-11-16	0
3	1_1_2012-11-23	0
4	1_1_2012-11-30	0
...	...	...
115059	45_98_2013-06-28	0
115060	45_98_2013-07-05	0
115061	45_98_2013-07-12	0
115062	45_98_2013-07-19	0
115063	45_98_2013-07-26	0

115064 rows × 2 columns

```
train_df.columns
```

```
Index(['Store', 'Dept', 'Date', 'Weekly_Sales', 'IsHoliday'], dtype='object')
```

```
features_df.columns
```

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

```
stores_df.columns
```

```
Index(['Store', 'Type', 'Size'], dtype='object')
```

Data field

stores.csv

This file contains anonymized information about the 45 stores, indicating the type and size of store.

train.csv

This is the historical training data, which covers to 2010-02-05 to 2012-11-01. Within this file you will find the following fields:

```
1. Store - the store number
2. Dept - the department number
3. Date - the week
4. Weekly_Sales -  sales for the given department in the given store
5. IsHoliday - whether the week is a special holiday week
```

test.csv

This file is identical to train.csv, except we have withheld the weekly sales. You must predict the sales for each triplet of store, department, and date in this file.

features.csv

This file contains additional data related to the store, department, and regional activity for the given dates. It contains the following fields:

```
1. Store - the store number
2. Date - the week
3. Temperature - average temperature in the region
4. Fuel_Price - cost of fuel in the region
5. MarkDown1-5 - anonymized data related to promotional markdowns that Walmart is rur
6. CPI - the consumer price index
7. Unemployment - the unemployment rate
8. IsHoliday - whether the week is a special holiday week
```

```
aux_df = features_df.merge(stores_df, how = 'left', on = "Store")
```

```
merged_df = train_df.merge(aux_df, how = 'left',on = ["Store","Date"])
```

```
merged_test_df = test_df.merge(aux_df, how = 'left', on = ["Store","Date"])
```

```
merged_df
```

	Store	Dept	Date	Weekly_Sales	IsHoliday_x	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDowr
0	1	1	2010-02-05	24924.50	False	42.31	2.572	NaN	NaN	Na
1	1	1	2010-02-12	46039.49	True	38.51	2.548	NaN	NaN	Na
2	1	1	2010-02-19	41595.55	False	39.93	2.514	NaN	NaN	Na
3	1	1	2010-02-26	19403.54	False	46.63	2.561	NaN	NaN	Na
4	1	1	2010-03-05	21827.90	False	46.50	2.625	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	
421565	45	98	2012-09-28	508.37	False	64.88	3.997	4556.61	20.64	1.5
421566	45	98	2012-10-05	628.10	False	64.89	3.985	5046.74	NaN	18.8
421567	45	98	2012-10-12	1061.02	False	54.47	4.000	1956.28	NaN	7.8
421568	45	98	2012-10-19	760.01	False	56.47	3.969	2004.02	NaN	3.1
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	4018.91	58.08	100.0

420285 rows × 17 columns

```
merged_test_df
```

	Store	Dept	Date	IsHoliday_x	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
0	1	1	2012-11-02	False	55.32	3.386	6766.44	5147.70	50.82	3639.90
1	1	1	2012-11-09	False	61.24	3.314	11421.32	3370.89	40.28	4646.79
2	1	1	2012-11-16	False	52.92	3.252	9696.28	292.10	103.78	1133.15
3	1	1	2012-11-23	True	56.23	3.211	883.59	4.17	74910.32	209.91
4	1	1	2012-11-30	False	52.34	3.207	2460.03	NaN	3838.35	150.57
...	...	...	...	...	...	...	...	...	...	..
115059	45	98	2013-06-28	False	76.05	3.639	4842.29	975.03	3.00	2449.97
115060	45	98	2013-07-05	False	77.50	3.614	9090.48	2268.58	582.74	5797.47

	Store	Dept	Date	IsHoliday_x	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
115061	45	98	2013-07-12	False	79.37	3.614	3789.94	1827.31	85.72	744.84
115062	45	98	2013-07-19	False	82.84	3.737	2961.49	1047.07	204.19	363.00
115063	45	98	2013-07-26	False	76.06	3.804	212.02	851.73	2.06	10.88

115064 rows × 16 columns

```
merged_df.isna().sum()
```

```
Store                0
Dept                 0
Date                 0
Weekly_Sales         0
IsHoliday_x          0
Temperature          0
Fuel_Price           0
MarkDown1       270889
MarkDown2       310322
MarkDown3       284479
MarkDown4       286603
MarkDown5       270138
CPI                  0
Unemployment         0
IsHoliday_y          0
Type                 0
Size                 0
dtype: int64
```

```
merged_test_df.isna().sum()
```

```
Store               0
Dept                0
Date                0
IsHoliday_x         0
Temperature         0
Fuel_Price          0
MarkDown1         149
MarkDown2       28627
MarkDown3        9829
MarkDown4       12888
MarkDown5           0
CPI             38162
Unemployment    38162
IsHoliday_y         0
Type                0
Size                0
dtype: int64
```

```python
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 17 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   Store         421570 non-null  int64  
 1   Dept          421570 non-null  int64  
 2   Date          421570 non-null  object 
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday_x   421570 non-null  bool   
 5   Temperature   421570 non-null  float64
 6   Fuel_Price    421570 non-null  float64
 7   MarkDown1     150681 non-null  float64
 8   MarkDown2     111248 non-null  float64
 9   MarkDown3     137091 non-null  float64
 10  MarkDown4     134967 non-null  float64
 11  MarkDown5     151432 non-null  float64
 12  CPI           421570 non-null  float64
 13  Unemployment  421570 non-null  float64
 14  IsHoliday_y   421570 non-null  bool   
 15  Type          421570 non-null  object 
 16  Size          421570 non-null  int64  
dtypes: bool(2), float64(10), int64(3), object(2)
memory usage: 52.3+ MB
```

```python
merged_df.drop(merged_df[merged_df.Weekly_Sales < 0].index, inplace=True)
```

```python
merged_df['Date'] = pd.to_datetime(merged_df['Date'])
merged_test_df['Date'] = pd.to_datetime(merged_test_df['Date'])
```

```python
merged_df['Year'] = pd.DatetimeIndex(merged_df.Date).year
merged_df['Month'] = pd.DatetimeIndex(merged_df.Date).month
merged_df['Day'] = pd.DatetimeIndex(merged_df.Date).day
```

```python
merged_test_df['Year'] = pd.DatetimeIndex(merged_test_df.Date).year
merged_test_df['Month'] = pd.DatetimeIndex(merged_test_df.Date).month
merged_test_df['Day'] = pd.DatetimeIndex(merged_test_df.Date).day
```

```python
merged_df
```

	Store	Dept	Date	Weekly_Sales	IsHoliday_x	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDowr

	Store	Dept	Date	Weekly_Sales	IsHoliday_x	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDowr
0	1	1	2010-02-05	24924.50	False	42.31	2.572	NaN	NaN	Na
1	1	1	2010-02-12	46039.49	True	38.51	2.548	NaN	NaN	Na
2	1	1	2010-02-19	41595.55	False	39.93	2.514	NaN	NaN	Na
3	1	1	2010-02-26	19403.54	False	46.63	2.561	NaN	NaN	Na
4	1	1	2010-03-05	21827.90	False	46.50	2.625	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	
421565	45	98	2012-09-28	508.37	False	64.88	3.997	4556.61	20.64	1.5
421566	45	98	2012-10-05	628.10	False	64.89	3.985	5046.74	NaN	18.8
421567	45	98	2012-10-12	1061.02	False	54.47	4.000	1956.28	NaN	7.8
421568	45	98	2012-10-19	760.01	False	56.47	3.969	2004.02	NaN	3.1
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	4018.91	58.08	100.0

420285 rows × 20 columns

```
merged_test_df
```

	Store	Dept	Date	IsHoliday_x	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
0	1	1	2012-11-02	False	55.32	3.386	6766.44	5147.70	50.82	3639.90
1	1	1	2012-11-09	False	61.24	3.314	11421.32	3370.89	40.28	4646.79
2	1	1	2012-11-16	False	52.92	3.252	9696.28	292.10	103.78	1133.15
3	1	1	2012-11-23	True	56.23	3.211	883.59	4.17	74910.32	209.91
4	1	1	2012-11-30	False	52.34	3.207	2460.03	NaN	3838.35	150.57
...	...	...	...	...	...	...	...	...	...	..
115059	45	98	2013-06-28	False	76.05	3.639	4842.29	975.03	3.00	2449.97
115060	45	98	2013-07-05	False	77.50	3.614	9090.48	2268.58	582.74	5797.47
115061	45	98	2013-07-12	False	79.37	3.614	3789.94	1827.31	85.72	744.84
115062	45	98	2013-07-19	False	82.84	3.737	2961.49	1047.07	204.19	363.00
115063	45	98	2013-07-26	False	76.06	3.804	212.02	851.73	2.06	10.88

115064 rows × 19 columns

```
(merged_df.IsHoliday_x == merged_df.IsHoliday_y).sum()
```

```
420285
```

```
(merged_test_df.IsHoliday_x == merged_test_df.IsHoliday_y).sum()
```

```
115064
```

Columns 'IsHoliday_x' and 'IsHoliday_y' are same. So we can drop IsHoliday_y from our merged_df.

```
merged_df.drop(columns = 'IsHoliday_y', inplace = True)
merged_test_df.drop(columns = 'IsHoliday_y', inplace = True)
```

```
merged_df['IsHoliday_x'] = merged_df.IsHoliday_x.astype(int)
merged_test_df['IsHoliday_x'] = merged_test_df.IsHoliday_x.astype(int)
# merged_df['IsHoliday_x'] = merged_df['IsHoliday_x'].apply(lambda x: 1 if x == True el
```

```
jovian.commit
```

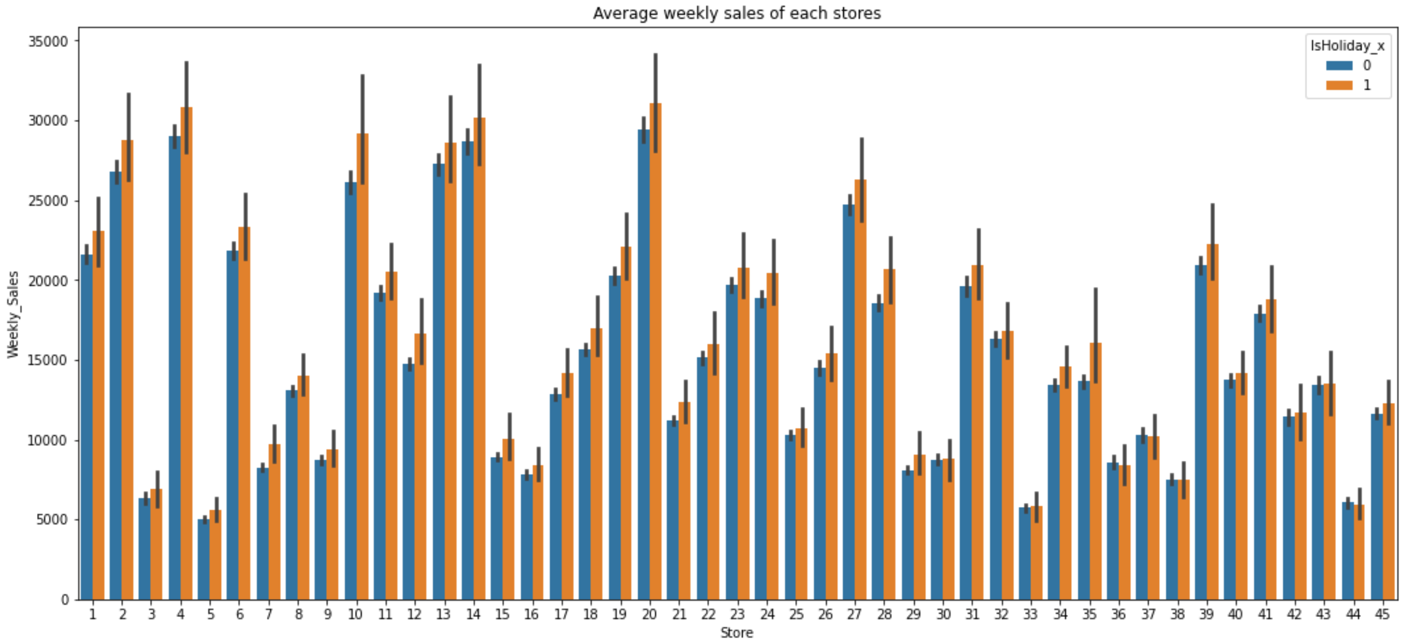
Exploratory data analysis

Let's import some libraries and explore the data.

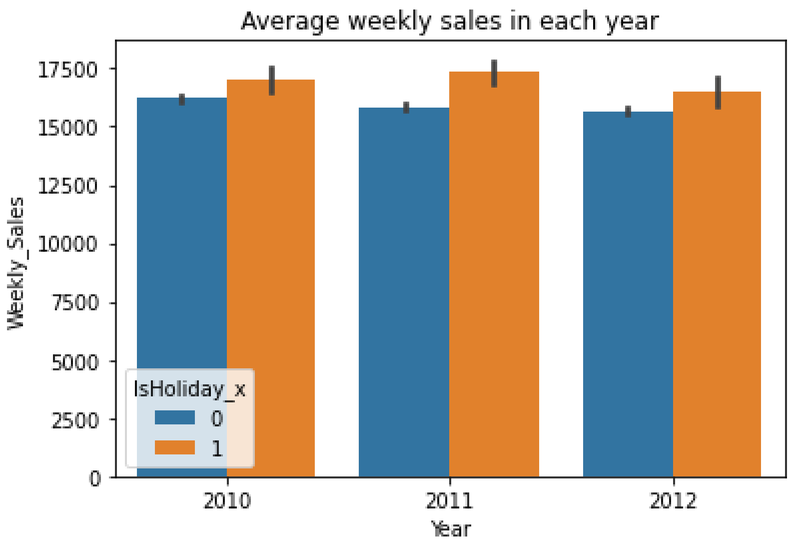
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline

pd.set_option("display.max_columns", 120)
pd.set_option("display.max_rows", 120)
```

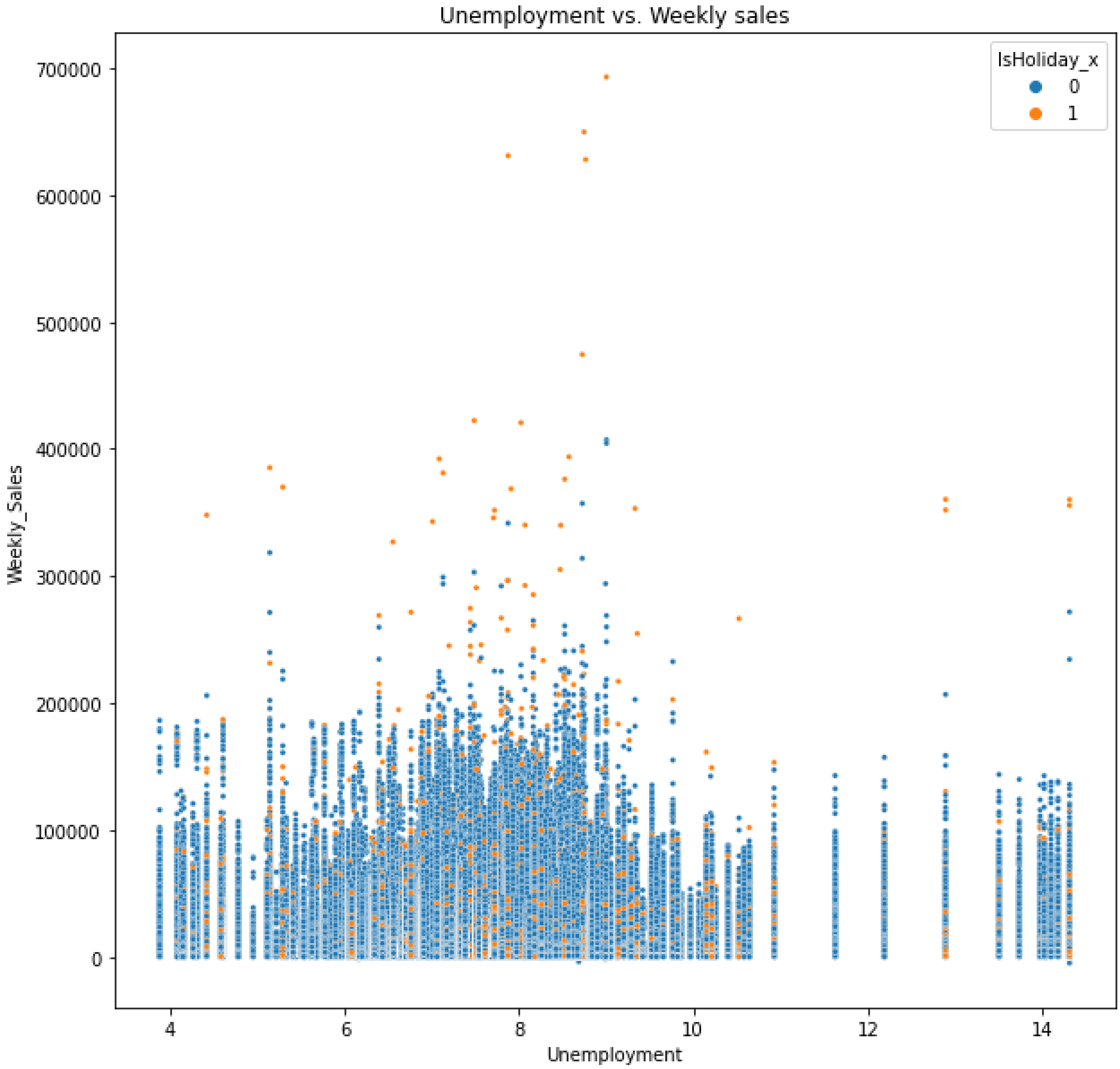
```
plt.figure(figsize = (18,8))
plt.title("Average weekly sales of each stores")
sns.barplot(data = merged_df, x = 'Store', y = 'Weekly_Sales', hue = 'IsHoliday_x');
```



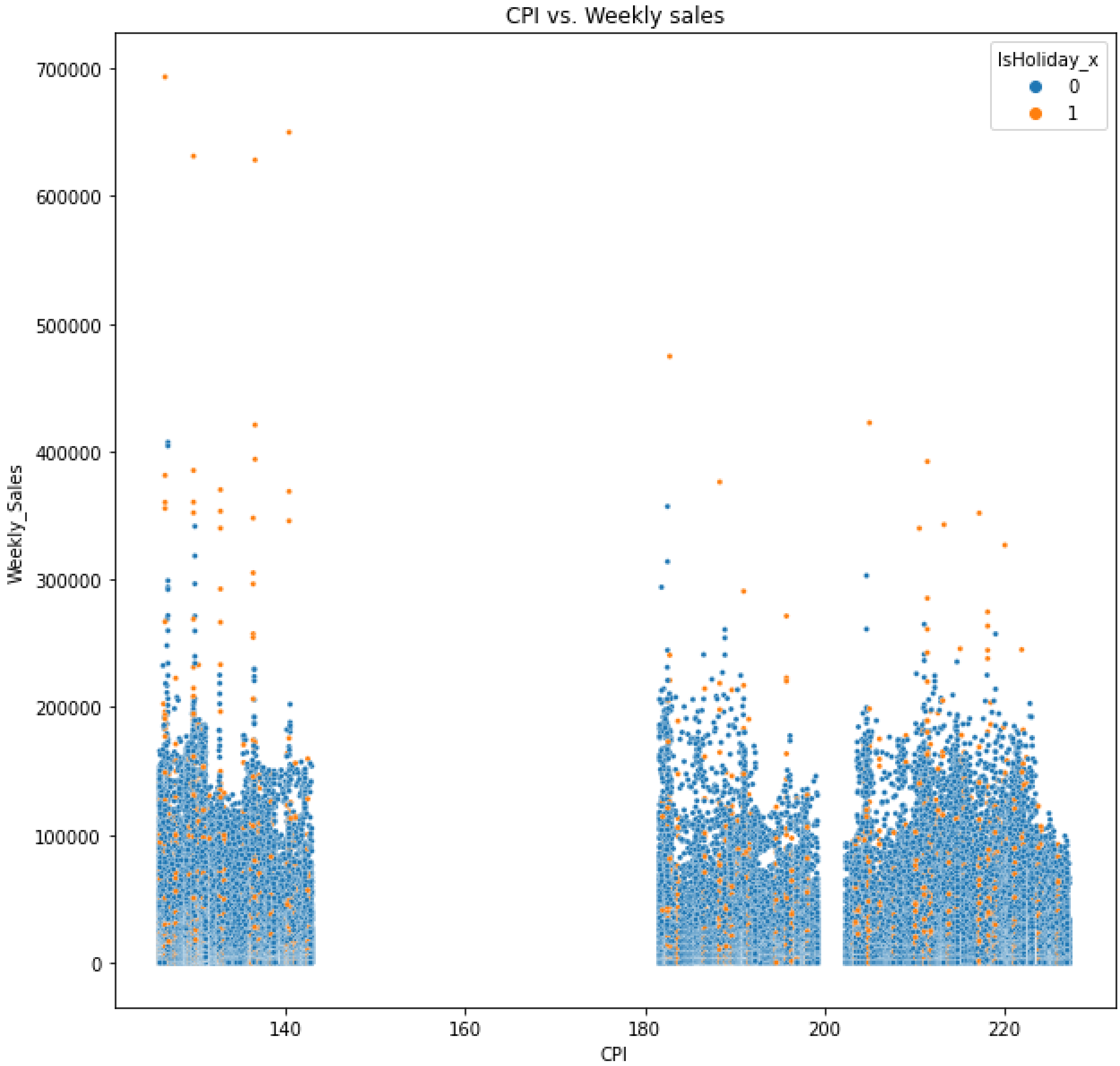

```
plt.title("Average weekly sales in each year")
sns.barplot(data = merged_df,
            x = 'Year', y = 'Weekly_Sales', hue = 'IsHoliday_x');
```




```
plt.figure(figsize = (10,10))
plt.title("Unemployment vs. Weekly sales")
sns.scatterplot(x = 'Unemployment', y = 'Weekly_Sales',  hue = 'IsHoliday_x', data = me
```




```
plt.figure(figsize = (10,10))
plt.title("CPI vs. Weekly sales")
sns.scatterplot(x = 'CPI', y = 'Weekly_Sales', hue = 'IsHoliday_x', data = merged_df, s
```



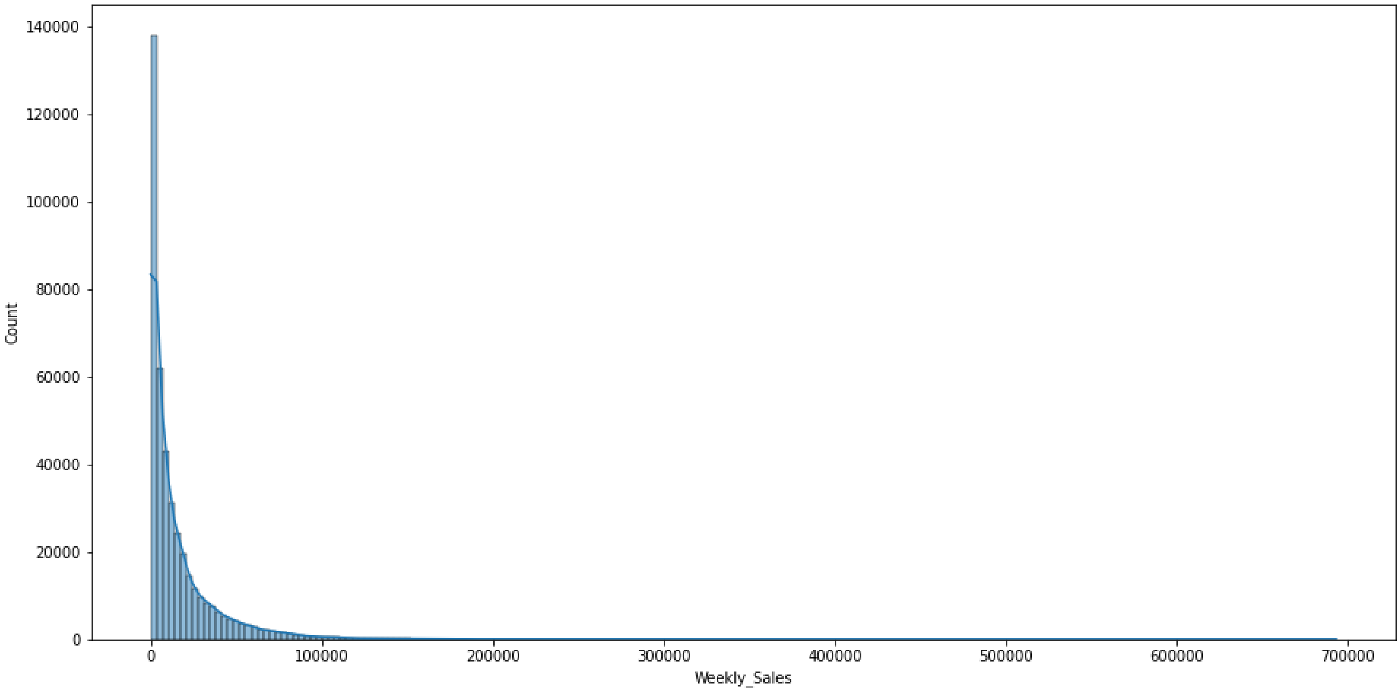

```
px.histogram(merged_df, x='Fuel_Price', y ='Weekly_Sales', color='IsHoliday_x', margina
```

Output hidden; open in https://colab.research.google.com to view.

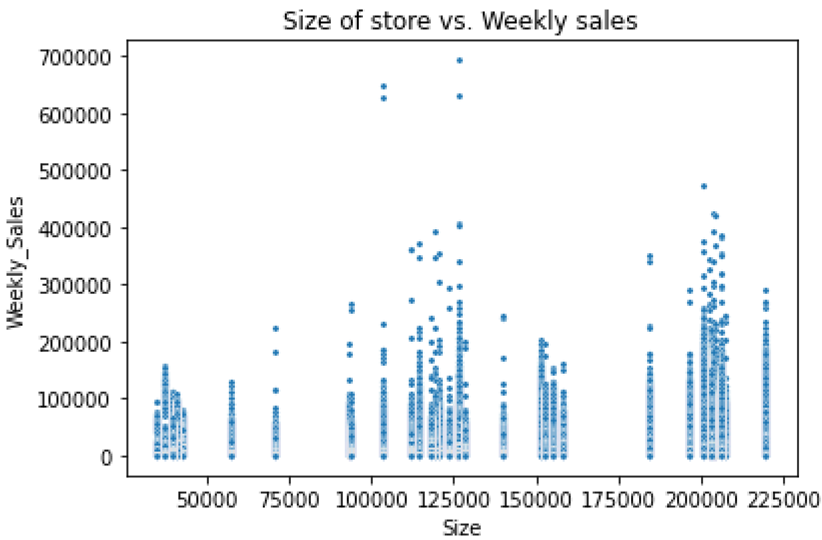
```
px.histogram(merged_df, x='Temperature', y ='Weekly_Sales', color='IsHoliday_x', margin
```

Output hidden; open in https://colab.research.google.com to view.

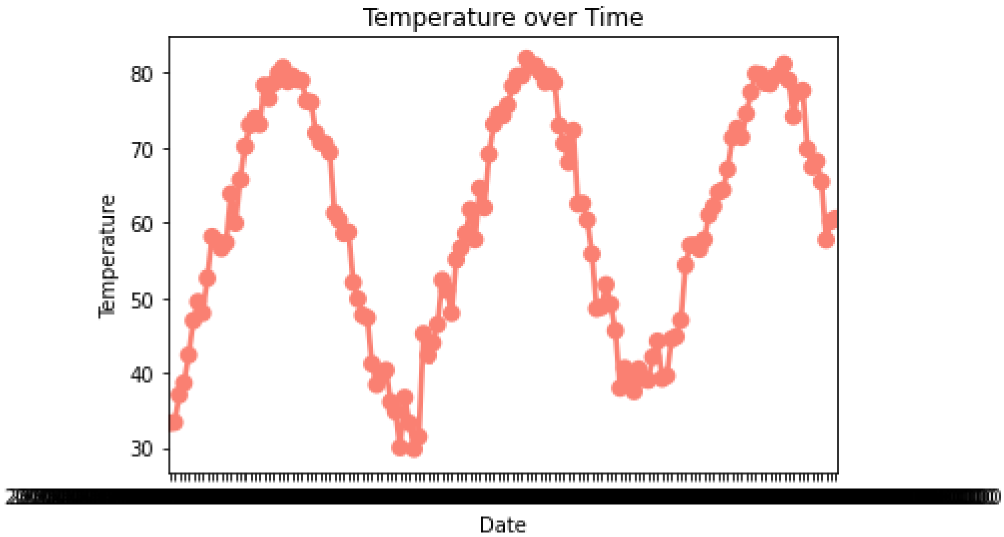
```
plt.figure(figsize = (16, 8))
sns.histplot(merged_df['Weekly_Sales'], bins=200,kde=True)
plt.show()
```



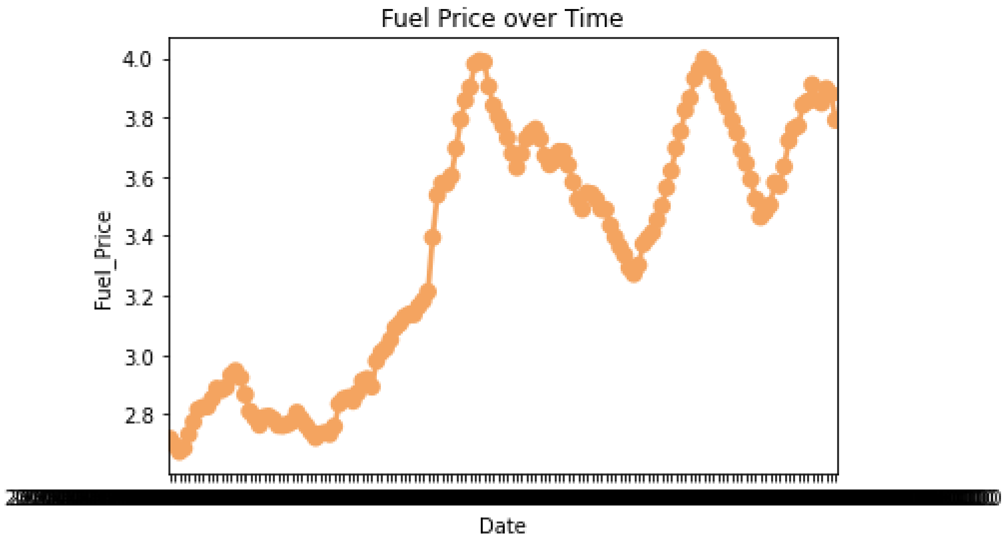

```
plt.title("Size of store vs. Weekly sales")
sns.scatterplot(x = 'Size', y = 'Weekly_Sales', data = merged_df, s = 10);
```



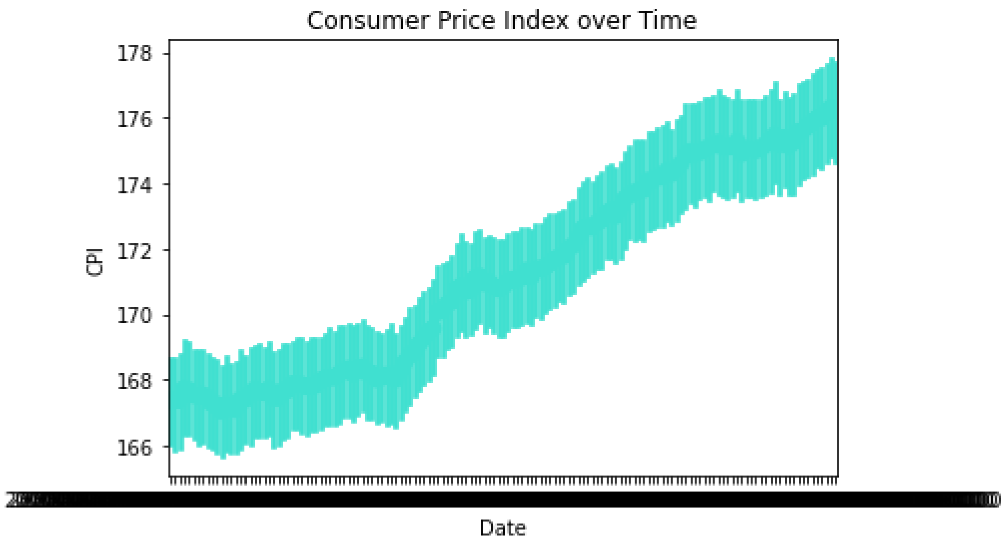

```
plt.title('Temperature over Time')
sns.pointplot(x="Date", y="Temperature", data=merged_df, color = 'salmon');
```



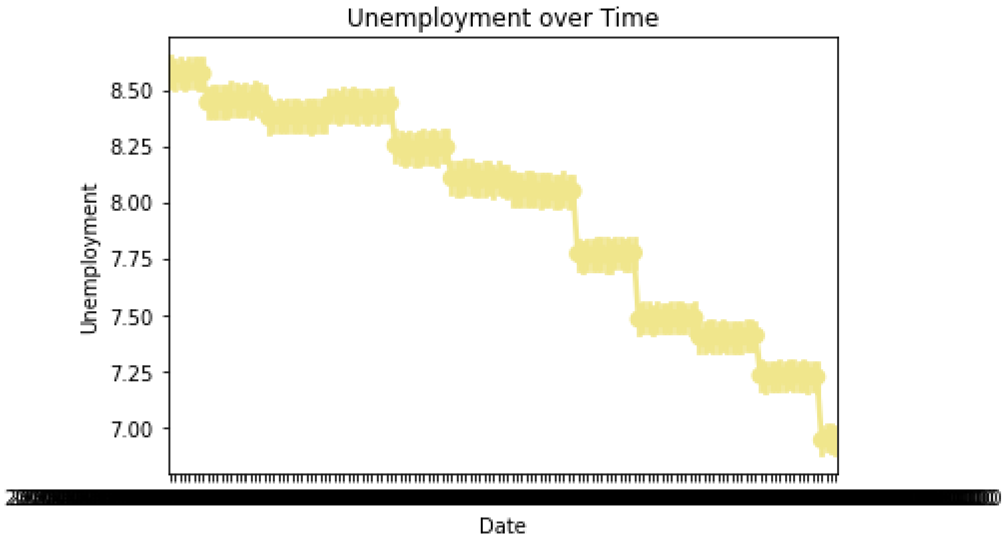

```
plt.title('Fuel Price over Time')
sns.pointplot(x="Date", y="Fuel_Price", data=merged_df, color = 'sandybrown');
```




```
plt.title('Consumer Price Index over Time')
sns.pointplot(x="Date", y="CPI", data=merged_df, color = 'turquoise');
```




```
plt.title('Unemployment over Time')
sns.pointplot(x="Date", y="Unemployment", data=merged_df, color='khaki');
```

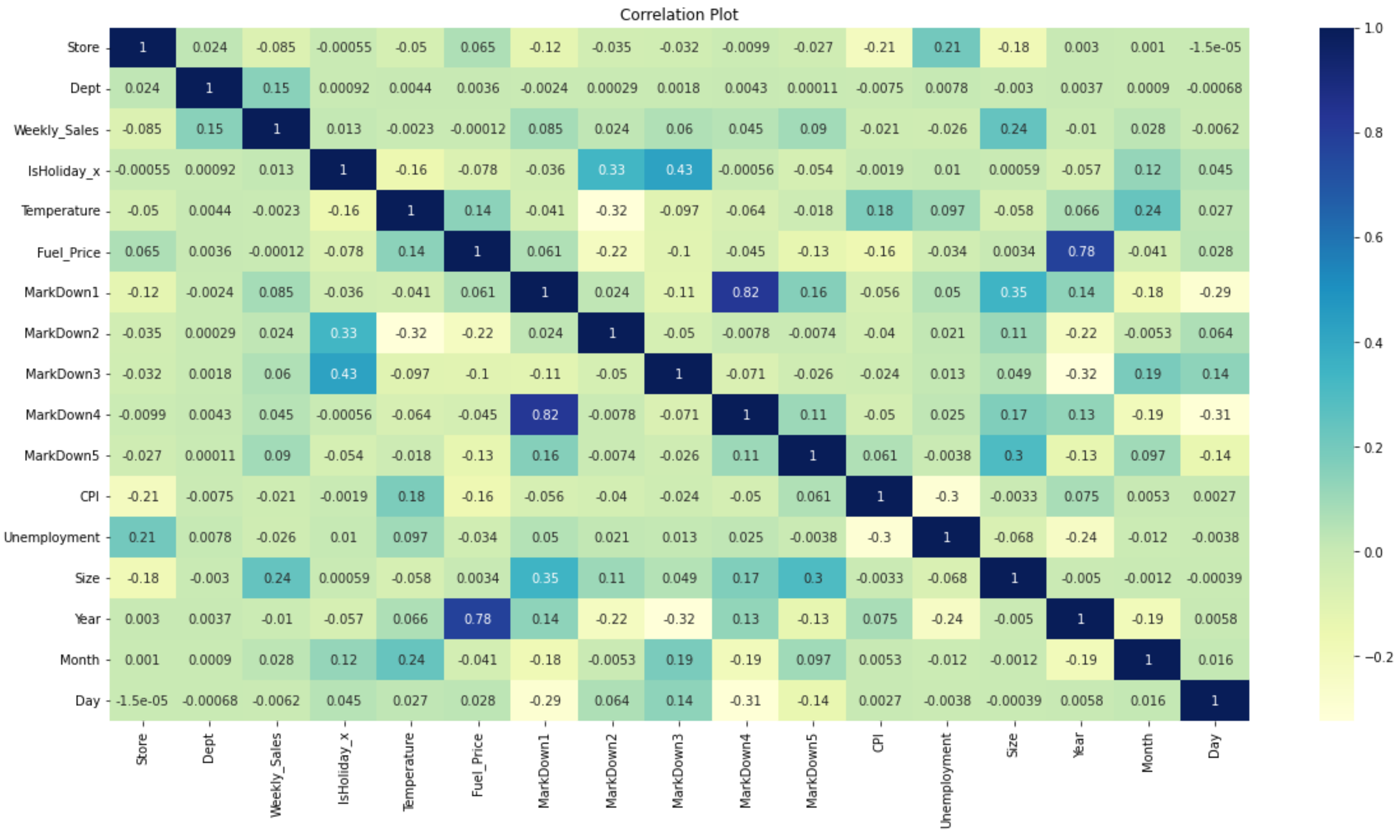



```
jovian.commit
```

Correlation between features

```
plt.figure(figsize = (20,10))

plt.title('Correlation Plot') # title
sns.heatmap(merged_df.corr(), annot = True, cmap = 'YlGnBu') # heatmap to visualize the
plt.show();
```



Feature Engineering

```
merged_df.columns
```

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

Splitting the dataset in training and validation sets

```
train_df = merged_df[(merged_df.Year < 2012) |  (merged_df.Month < 4)]
```

```
val_df = merged_df[(merged_df.Year == 2012) & (merged_df.Month >= 4)]
```

Identifying input and output columns

```
input_cols = ['Store','Dept','IsHoliday_x','Temperature','Fuel_Price','MarkDown1','Mark
              'MarkDown5','CPI','Unemployment','Type','Size', 'Year','Month']
target_col = 'Weekly_Sales'
```

Identifying numeric and categorical columns

```
numeric_cols = ['Store','Dept','IsHoliday_x','Temperature','Fuel_Price','MarkDown1','Ma
                'MarkDown4','MarkDown5','CPI','Unemployment','Size','Year','Month']
categorical_cols = ['Type']
```

Imputing missing numerical values

A markdown is a reduction of the original price of goods to increase sales. Missing markdown values simply means no promotional markdowns run by Walmart at that period of time. So we can simply insert 0 in all missing MarkDown values.

```
train_df['MarkDown1'] = train_df['MarkDown1'].fillna(0)
train_df['MarkDown2'] = train_df['MarkDown2'].fillna(0)
train_df['MarkDown3'] = train_df['MarkDown3'].fillna(0)
train_df['MarkDown4'] = train_df['MarkDown4'].fillna(0)
train_df['MarkDown5'] = train_df['MarkDown5'].fillna(0)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

```
val_df['MarkDown1'] = val_df['MarkDown1'].fillna(0)
val_df['MarkDown2'] = val_df['MarkDown2'].fillna(0)
val_df['MarkDown3'] = val_df['MarkDown3'].fillna(0)
val_df['MarkDown4'] = val_df['MarkDown4'].fillna(0)
val_df['MarkDown5'] = val_df['MarkDown5'].fillna(0)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

```
merged_test_df['MarkDown1'] = merged_test_df['MarkDown1'].fillna(0)
merged_test_df['MarkDown2'] = merged_test_df['MarkDown2'].fillna(0)
merged_test_df['MarkDown3'] = merged_test_df['MarkDown3'].fillna(0)
merged_test_df['MarkDown4'] = merged_test_df['MarkDown4'].fillna(0)
merged_test_df['MarkDown5'] = merged_test_df['MarkDown5'].fillna(0)
```

```
train_inputs = train_df[input_cols].copy()
train_targets = train_df[target_col].copy()
```

```
val_inputs = val_df[input_cols].copy()
val_targets = val_df[target_col].copy()
```

```
test_inputs = merged_test_df[input_cols].copy()
```

Scaling numeric features

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
```

```
scaler.fit(merged_df[numeric_cols])
```

```
MinMaxScaler()
```

```
train_inputs[numeric_cols] = scaler.transform(train_inputs[numeric_cols])
val_inputs[numeric_cols] = scaler.transform(val_inputs[numeric_cols])
```

Encoding categorical data

```
from sklearn.preprocessing import OneHotEncoder
```

```
encoder = OneHotEncoder(sparse = False, handle_unknown = 'ignore')
```

```
encoder.fit(merged_df[categorical_cols])
```

```
OneHotEncoder(handle_unknown='ignore', sparse=False)
```

```
encoded_cols = list(encoder.get_feature_names_out(categorical_cols))
```

```
train_inputs[encoded_cols] = encoder.transform(train_inputs[categorical_cols])
val_inputs[encoded_cols] = encoder.transform(val_inputs[categorical_cols])
```

```
X_train = train_inputs[numeric_cols + encoded_cols]
X_val = val_inputs[numeric_cols + encoded_cols]
```

Training different machine learning model

1. LinearRegressor

```
!pip install scikit-learn --quiet
```

```
from sklearn.linear_model import SGDRegressor
```

```
model_sgdr = SGDRegressor().fit(X_train, train_targets)
```

```
sgdr_train_preds = model_sgdr.predict(X_train)
```

```
sgdr_val_preds = model_sgdr.predict(X_val)
```

```
sgdr_val_preds
```

```
array([14463.26845217, 13996.65998638, 14201.64611765, ...,
```

```
      16222.7172854 , 16006.24622685, 15998.98133345])
```

```
from sklearn.metrics import mean_squared_error
```

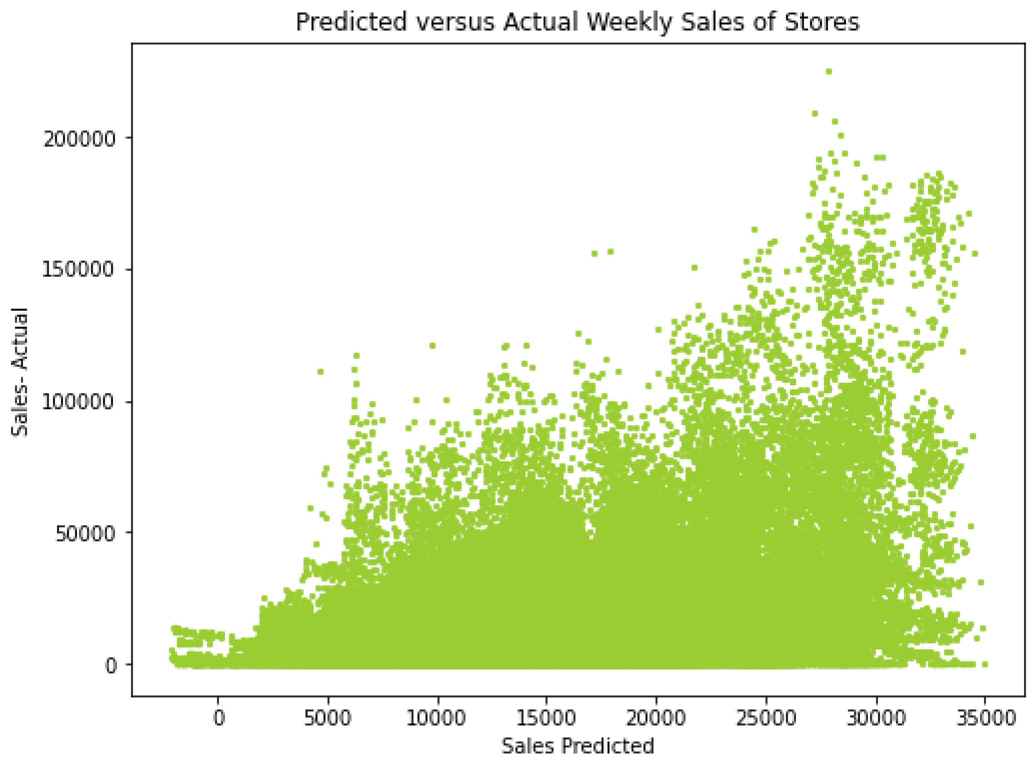
```
sgdr_train_rmse = mean_squared_error(train_targets, sgdr_train_preds, squared = False)
```

```
sgdr_val_rmse = mean_squared_error(val_targets, sgdr_val_preds, squared = False)
```

```
sgdr_rmse = sgdr_train_rmse, sgdr_val_rmse
sgdr_rmse
```

```
(21836.141564355934, 21004.055720234177)
```

```
fig = plt.subplots(figsize=(8,6))
plt.scatter(sgdr_val_preds,val_targets, c='yellowgreen', s = 5)
plt.xlabel('Sales Predicted')
plt.ylabel('Sales- Actual')
plt.title('Predicted versus Actual Weekly Sales of Stores')
plt.show()
```



2. DecisionTreeRegressor

```
from sklearn.tree import DecisionTreeRegressor
```

```
#Create the model
model_dt = DecisionTreeRegressor().fit(X_train, train_targets)
```

```
dt_train_preds = model_dt.predict(X_train)
dt_val_preds = model_dt.predict(X_val)
```

```
dt_train_rmse = mean_squared_error(train_targets, dt_train_preds, squared = False)
```

```
dt_val_rmse = mean_squared_error(val_targets, dt_val_preds, squared = False)
dt_rmse = dt_train_rmse, dt_val_rmse
```

```
dt_rmse
```

```
(4.517123693939653e-17, 5683.249793268674)
```

```
fig = plt.subplots(figsize=(8,6))
plt.scatter(dt_val_preds,val_targets, c='yellowgreen', s = 5)
plt.xlabel('Sales Predicted')
plt.ylabel('Sales- Actual')
plt.title('Predicted versus Actual Weekly Sales of Stores')
plt.show()
```

We can reduce losses even further by introducing hyperparameters because model is overfitted on training dataset.

3. RandomForestRegressor

```
from sklearn.ensemble import RandomForestRegressor
```

```
model_rf = RandomForestRegressor(n_jobs = -1, random_state = 42).fit(X_train, train_tar
```

```
rf_train_preds = model_rf.predict(X_train)
rf_val_preds = model_rf.predict(X_val)
```

```
rf_val_preds
```

```
array([41147.9443, 39966.0182, 30145.6312, ...,   808.1058,   831.0594,
         703.0313])
```

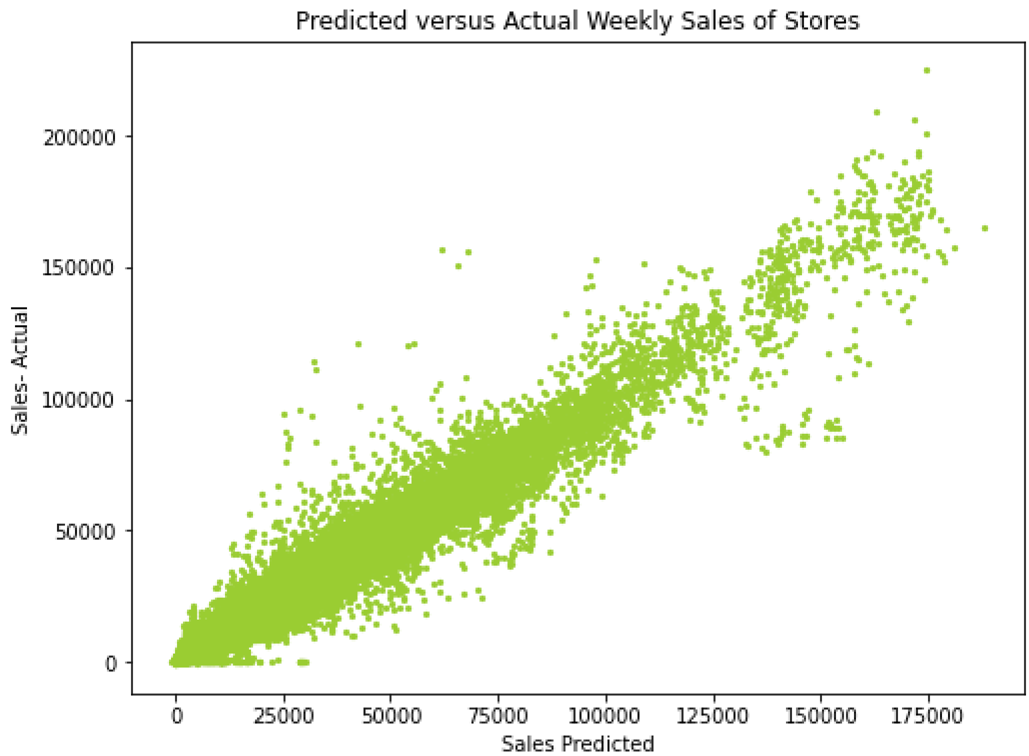
```
rf_train_rmse = mean_squared_error(train_targets, rf_train_preds, squared = False)
```

```
rf_val_rmse = mean_squared_error(val_targets, rf_val_preds, squared = False)
rf_rmse = rf_train_rmse, rf_val_rmse
```

```
rf_rmse
```

```
(1595.072899234597, 4120.911170819646)
```

```
fig = plt.subplots(figsize=(8,6))
plt.scatter(rf_val_preds,val_targets, c='yellowgreen', s = 5)
plt.xlabel('Sales Predicted')
plt.ylabel('Sales- Actual')
plt.title('Predicted versus Actual Weekly Sales of Stores')
plt.show()
```



4. XGBoostRegressor

```
from xgboost import XGBRegressor
```

```
model_xgb = XGBRegressor(random_state = 42, n_jobs = -1).fit(X_train, train_targets)
```

```
[16:26:43] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
```

```
xgb_train_preds = model_xgb.predict(X_train)
xgb_val_preds = model_xgb.predict(X_val)
```

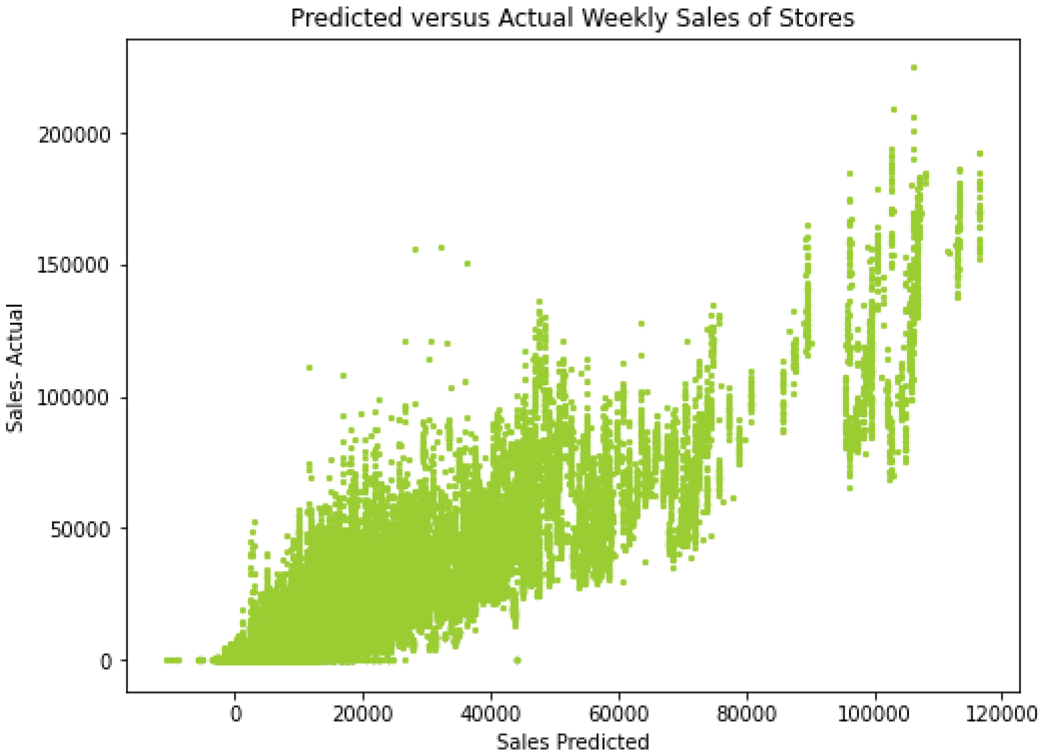
```
xgb_train_rmse = mean_squared_error(train_targets, xgb_train_preds, squared = False)
```

```
xgb_val_rmse = mean_squared_error(val_targets, xgb_val_preds, squared = False)
xgb_rmse = xgb_train_rmse, xgb_val_rmse
```

```
xgb_rmse
```

```
(11772.816046692174, 10512.510122336356)
```

```
fig = plt.subplots(figsize=(8,6))
plt.scatter(xgb_val_preds,val_targets, c='yellowgreen', s = 5)
plt.xlabel('Sales Predicted')
plt.ylabel('Sales- Actual')
plt.title('Predicted versus Actual Weekly Sales of Stores')
plt.show()
```



Linear regressor validation loss = 20997.43397354532

Decision tree validation loss = 5581.382957057745

Random forest validation loss = 4120.911170819646

Gradient boost validation loss = 10512.51012233635

RandomForestRegressor model has the lowest losses.

```
```

Hyperparameter tuning with Random forest

Let's define a helper function **test_params** to test hyperparameters and **test_params_and_plot** to test and plot various values of a single hyperparameter.

```
def test_params(**params):
    model = RandomForestRegressor(random_state=42, n_jobs=-1, **params).fit(X_train, tr
    train_rmse = mean_squared_error(model.predict(X_train), train_targets, squared=Fals
    val_rmse = mean_squared_error(model.predict(X_val), val_targets, squared=False)
    return train_rmse, val_rmse
```

```
def test_param_and_plot(param_name, param_values):
    train_errors, val_errors = [], []
    for value in param_values:
        params = {param_name: value}
        train_rmse, val_rmse = test_params(**params)
        train_errors.append(train_rmse)
        val_errors.append(val_rmse)
    plt.figure(figsize=(10,6))
    plt.title('Overfitting curve: ' + param_name)
    plt.plot(param_values, train_errors, 'b-o')
    plt.plot(param_values, val_errors, 'r-o')
    plt.xlabel(param_name)
    plt.ylabel('RMSE')
    plt.legend(['Training', 'Validation'])
```

```
test_params(max_depth=10)
```

```
(8015.151881231166, 7220.911172374859)
```

```
test_params(max_depth=20)
```

```
(2073.9312013510175, 4140.3591041882255)
```

```
test_params(n_estimators =200)
```

```
(1569.9731968073665, 4114.928346362817)
```

```
test_params(n_estimators =300)
```

```
(1566.723415485546, 4114.2505627530845)
```

```
test_params(max_features = 0.7)
```

```
(1654.8661464088773, 3899.8848674909023)
```

```
test_params(max_features = 0.6)
```

```
(1797.234395624805, 4056.173583217264)
```

```
test_params(max_features = 0.8)
```

```
(1603.8511092250692, 3899.45360938875)
```

```
test_params(max_features = 0.7, n_estimators =200)
```

```
(1625.7527195981554, 3900.923190958408)
```

```
test_params(min_samples_split = 4)
```

```
(1920.011188201959, 4125.877590484456)
```

Training the best model

```
model = RandomForestRegressor(max_depth=48, max_features=0.8, n_estimators=300, n_jobs=
```

```
model.fit(X_train, train_targets)
```

```
RandomForestRegressor(max_depth=48, max_features=0.8, n_estimators=300,
                      n_jobs=-1, random_state=42)
```

```
train_preds = model.predict(X_train)
val_preds = model.predict(X_val)
```

```
train_rmse = mean_squared_error(train_targets, train_preds, squared = False)
val_rmse = mean_squared_error(val_targets, val_preds, squared = False)
```

```
# max_depth=48, max_features=0.8, n_estimators=200, n_jobs=-1, random_state=42
model_rmse = train_rmse, val_rmse
model_rmse
```

```
(1568.505497055511, 3861.584249110508)
```

```
# max_depth=30, max_features=15, n_estimators=80, n_jobs=-1, random_state=42
model_rmse = train_rmse, val_rmse
model_rmse
```

```
(1576.1726215358765, 3910.8721500193733)
```

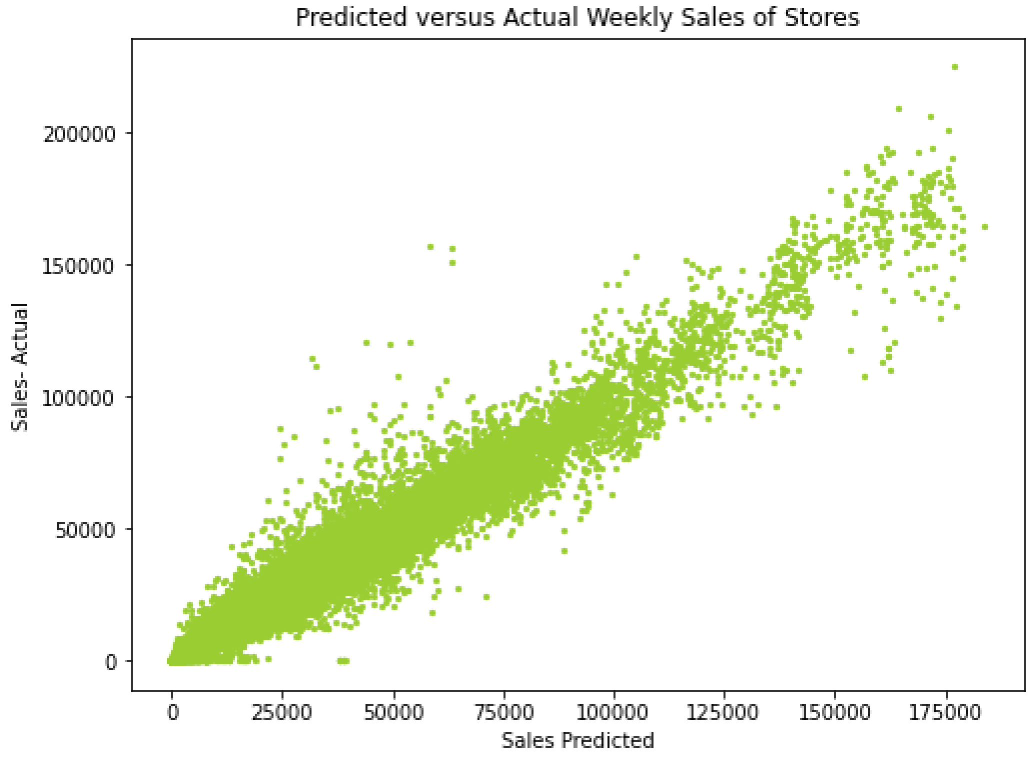
```
# max_depth=30, random_state=42
model_rmse = train_rmse, val_rmse
model_rmse
```

(1598.5462507147522, 4122.5634000674945)

```
# max_depth=48, max_features=0.8, n_estimators=300, n_jobs=-1, random_state=42
model_rmse = train_rmse, val_rmse
model_rmse
```

(1560.0817273117455, 3848.188031102481)

```
fig = plt.subplots(figsize=(8,6))
plt.scatter(val_preds, val_targets, c='yellowgreen', s = 5)
plt.xlabel('Sales Predicted')
plt.ylabel('Sales- Actual')
plt.title('Predicted versus Actual Weekly Sales of Stores')
plt.show()
```




```
train_score = model.score(X_train, train_targets)
val_score = model.score(X_val, val_targets)
```

```
model_score = train_score, val_score
```

```
model_score
```

(0.9953573018547699, 0.9696471954732058)

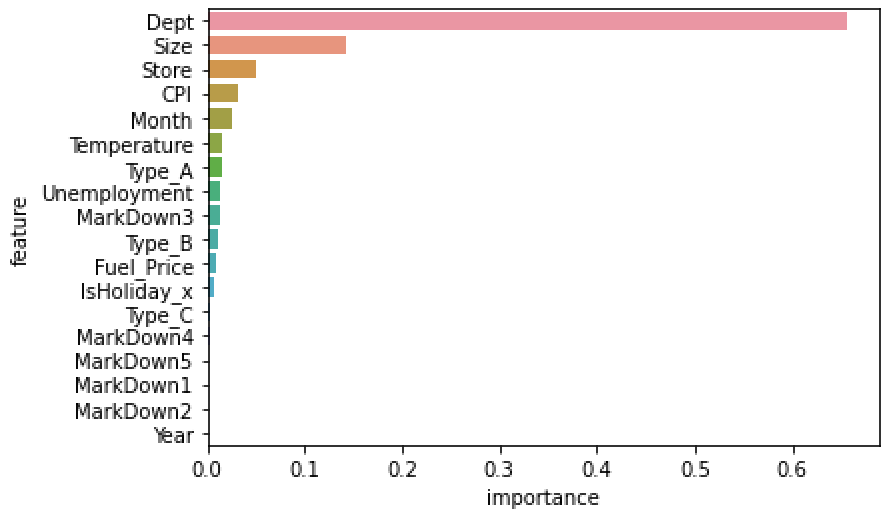
Let's also view and plot the feature importances.

```python
model_importance_df = pd.DataFrame({
    'feature': X_train.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```python
model_importance_df
```

	feature	importance
1	Dept	0.653642
12	Size	0.142494
0	Store	0.050783
10	CPI	0.030690
14	Month	0.027017
15	Type_A	0.016567
3	Temperature	0.015757
11	Unemployment	0.013457
7	MarkDown3	0.012650
16	Type_B	0.010404
4	Fuel_Price	0.009615
2	IsHoliday_x	0.006722
17	Type_C	0.003315
8	MarkDown4	0.002153
9	MarkDown5	0.001579
5	MarkDown1	0.001424
6	MarkDown2	0.001309
13	Year	0.000422

```python
sns.barplot(data=model_importance_df, x='importance', y='feature');
```



Predicting the test set

```
merged_test_df.Unemployment.fillna(merged_test_df.Unemployment.mean(), inplace = True)
```

```
merged_test_df.CPI.fillna(merged_test_df.CPI.mean(), inplace = True)
```

```
# Scaling
scaler.fit(merged_test_df[numeric_cols])
test_inputs[numeric_cols] = scaler.transform(test_inputs[numeric_cols])
```

```
# Encoding
test_inputs[encoded_cols] = encoder.transform(test_inputs[categorical_cols])
```

```
X_test = test_inputs[numeric_cols + encoded_cols]
```

```
test_preds = model.predict(X_test)
```

```
test_preds
```

```
array([25446.03016667, 21089.5386    , 25150.6523    , ...,
         765.5697    ,   761.91546667,   858.90906667])
```

```
submission_df['Weekly_Sales'] = test_preds
```

```
submission_df
```

	Id	Weekly_Sales
0	1_1_2012-11-02	25446.030167
1	1_1_2012-11-09	21089.538600
2	1_1_2012-11-16	25150.652300
3	1_1_2012-11-23	27204.663967
4	1_1_2012-11-30	27189.618333
...	...	...
115059	45_98_2013-06-28	771.683467
115060	45_98_2013-07-05	780.419000
115061	45_98_2013-07-12	765.569700
115062	45_98_2013-07-19	761.915467
115063	45_98_2013-07-26	858.909067

115064 rows × 2 columns

Downloading the submission.csv file.

```
submission_df.to_csv('submission.csv', index=None)
```

```
!pip install jovian --upgrade --quiet
```

```
import jovian
```

```
# Execute this to save new versions of the notebook
jovian.commit(project="walmart-store-sales-forecasting")
```

```
[jovian] Detected Colab notebook...
[jovian] Please enter your API key ( from https://jovian.ai/ ):
API KEY: ··········
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/sharma289/walmart-store-sales-forecasting
```

```
'https://jovian.ai/sharma289/walmart-store-sales-forecasting'
```

Result -

Accuracy score on Training set = 99.53%

Accuracy score on Validation set = 96.96%

Conclusion -

1. Size of the store is the highest contributing predictor in the model out of all.
2. Each store has a unique prediction power. They can be separately analyzed to get prediction for each individual store.
3. The Sales are very high during November and December and go down in January. So its better to employee more staff as casual employee in November and December and encourage permanent staff to take leaves during January.
4. The predicted sales data can be used to analyse the sales pattern and accordingly adjust the staff in the store.
5. When we implement the project to department level it helps to plan the inventory and staff from a centralised station to every store, which will further help in better planning and cost cutting for inventory management, supply chain management and human resource.
6. The low selling stores should look forward to increasing their size and capacity to store more items and consumer products.
7. Special discount coupons can be distributed during low selling periods to attract more customers.
8. Sales are likely to fluctuate during holidays. Special offers can be given during festive season accompanied with suitable marketing to keep the sales high during holidays as well.