Walmart-Self-Case-Study-FINAL

September 28, 2019

1 Problem Statement and Dataset from:https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/overview

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.metrics import mean_absolute_error
import numpy as np
import pylab
import scipy.stats as stats
from tqdm import tqdm_notebook as tqdm
from sklearn.ensemble import RandomForestRegressor
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
#define error metric weighted mean absolute error
def wmae(y_true,y_pred,weights):
    return mean_absolute_error(y_true, y_pred, sample_weight=weights)
```

[3]:
```
#read train dataset and merge with store and features dataset in such a way␣
 ↪that all the data in train is preserved
train_df = pd.read_csv('train.csv')
store_df = pd.read_csv('stores.csv')
features_df = pd.read_csv('features.csv')
train_df = train_df.merge(store_df,how='left').merge(features_df,how='left')
train_df.head()
```

[3]:
```
   Store  Dept        Date  Weekly_Sales  IsHoliday Type    Size  Temperature  \
0      1     1  2010-02-05      24924.50      False    A  151315        42.31
1      1     1  2010-02-12      46039.49       True    A  151315        38.51
2      1     1  2010-02-19      41595.55      False    A  151315        39.93
3      1     1  2010-02-26      19403.54      False    A  151315        46.63
4      1     1  2010-03-05      21827.90      False    A  151315        46.50

   Fuel_Price  MarkDown1  MarkDown2  MarkDown3  MarkDown4  MarkDown5  \
0       2.572        NaN        NaN        NaN        NaN        NaN
```

```
1       2.548        NaN        NaN        NaN        NaN        NaN
2       2.514        NaN        NaN        NaN        NaN        NaN
3       2.561        NaN        NaN        NaN        NaN        NaN
4       2.625        NaN        NaN        NaN        NaN        NaN

           CPI  Unemployment
0   211.096358         8.106
1   211.242170         8.106
2   211.289143         8.106
3   211.319643         8.106
4   211.350143         8.106
```

1.1 Perform exploratory data analysis

[4]:
```
train_df.describe()
```

[4]:
```
                Store           Dept   Weekly_Sales           Size  \
count  421570.000000  421570.000000  421570.000000  421570.000000
mean       22.200546      44.260317   15981.258123  136727.915739
std        12.785297      30.492054   22711.183519   60980.583328
min         1.000000       1.000000   -4988.940000   34875.000000
25%        11.000000      18.000000    2079.650000   93638.000000
50%        22.000000      37.000000    7612.030000  140167.000000
75%        33.000000      74.000000   20205.852500  202505.000000
max        45.000000      99.000000  693099.360000  219622.000000

         Temperature     Fuel_Price      MarkDown1      MarkDown2  \
count  421570.000000  421570.000000  150681.000000  111248.000000
mean       60.090059       3.361027    7246.420196    3334.628621
std        18.447931       0.458515    8291.221345    9475.357325
min        -2.060000       2.472000       0.270000    -265.760000
25%        46.680000       2.933000    2240.270000      41.600000
50%        62.090000       3.452000    5347.450000     192.000000
75%        74.280000       3.738000    9210.900000    1926.940000
max       100.140000       4.468000   88646.760000  104519.540000

           MarkDown3      MarkDown4      MarkDown5            CPI  \
count  137091.000000  134967.000000  151432.000000  421570.000000
mean     1439.421384    3383.168256    4628.975079     171.201947
std      9623.078290    6292.384031    5962.887455      39.159276
min       -29.100000       0.220000     135.160000     126.064000
25%         5.080000     504.220000    1878.440000     132.022667
50%        24.600000    1481.310000    3359.450000     182.318780
75%       103.990000    3595.040000    5563.800000     212.416993
max    141630.610000   67474.850000  108519.280000     227.232807

        Unemployment
```

```
count  421570.000000
mean        7.960289
std         1.863296
min         3.879000
25%         6.891000
50%         7.866000
75%         8.572000
max        14.313000
```

[5]:
```
# Total records in train
train_df.shape
```

[5]:
```
(421570, 16)
```

[6]:
```
# count number of unique stores
train_df['Store'].unique().shape
```

[6]:
```
(45,)
```

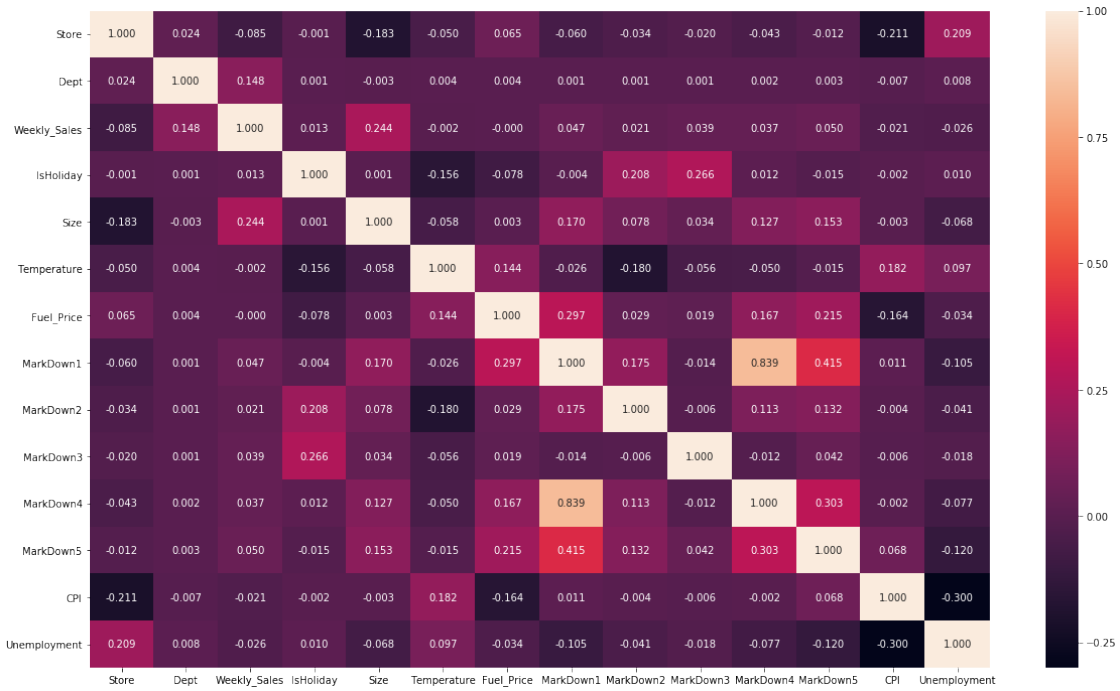
[7]:
```
# count number of unique department
train_df['Dept'].unique().shape
```

[7]:
```
(81,)
```

[8]:
```
#fill NA with zeros
train_df = train_df.fillna(0)
```

[9]:
```
#plot correlation matrix
fig = plt.figure(figsize=(20, 12))
sns.heatmap(train_df.corr(),annot=True,fmt=".3f")
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x288b5cb39e8>
```



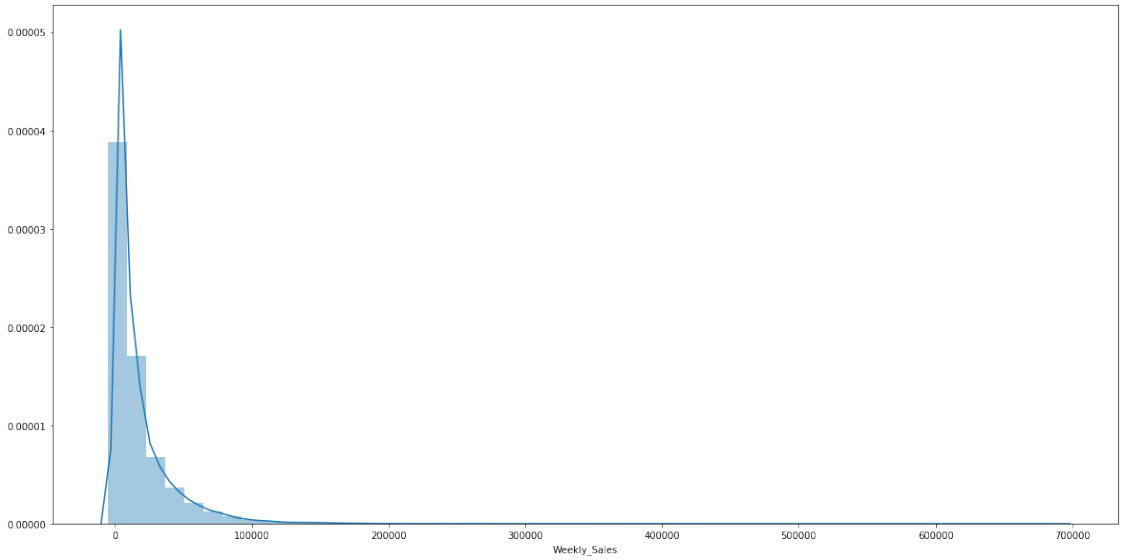
[10]:
```
train_df['Weekly_Sales'].value_counts()
```

[10]:
```
10.00       353
5.00        289
20.00       232
15.00       215
12.00       175
           ...
6835.41       1
10467.96      1
31889.20      1
6748.45       1
14543.76      1
Name: Weekly_Sales, Length: 359464, dtype: int64
```

1.1.1 Weekly Sales distribution plot

[11]:
```
plt.figure(figsize=(20,10))
sns.distplot(train_df['Weekly_Sales'])
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x288b593ffd0>
```



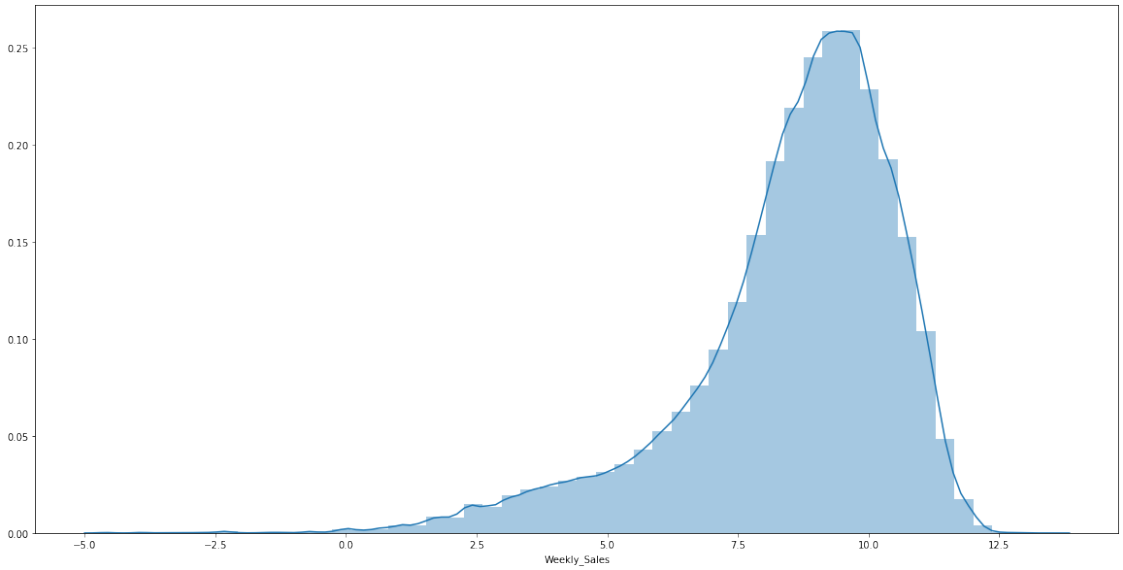
1.2 Inference

Not sure of type of distribution. Need to apply log transformation to see its log-normal distribution

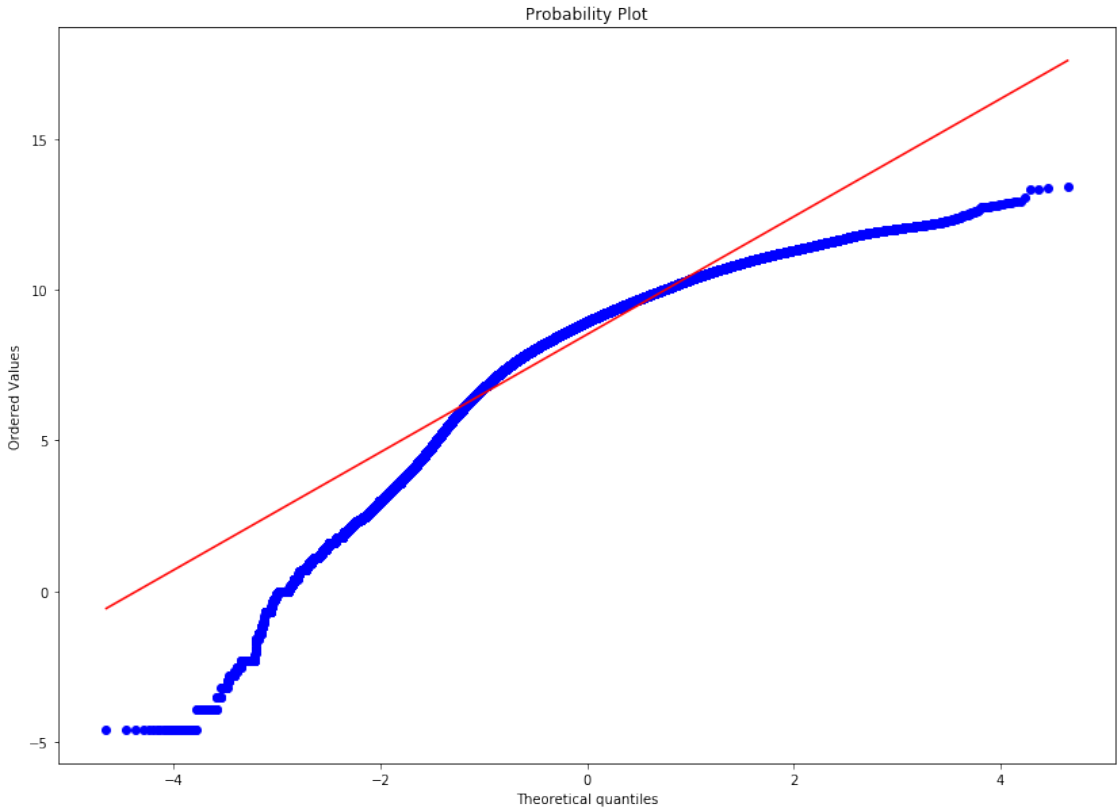
1.3 Log transformation on weekly sales

[45]:
```
import math
plt.figure(figsize=(20,10))
d = train_df[train_df['Weekly_Sales']>0]['Weekly_Sales']
sns.distplot(d.map(lambda x: np.log(x)))
```

[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x196725409b0>
```



[53]:
```
## check if wekly sales this is log-normal distribution using Q-Q plot
plt.figure(figsize=(14,10))
stats.probplot(d.map(lambda x: np.log(x)), dist="norm", plot=pylab)
pylab.show()
```

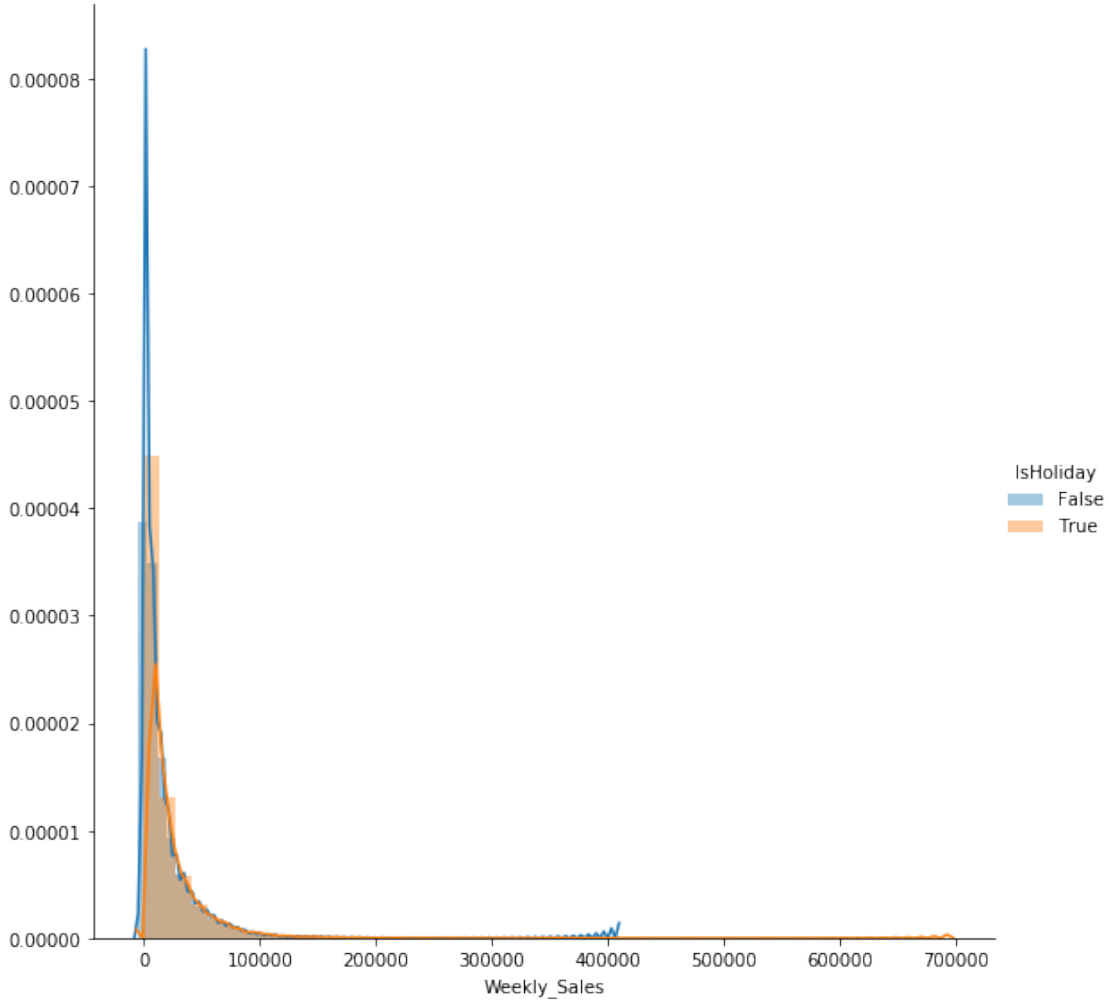


2 Inference

Weekly Sales do not follow log-normal distribution

2.0.1 Distribution of Weekly Sales in Holiday and Non-Holiday Weeks

[68]:
```
sns.FacetGrid(train_df, hue="IsHoliday", size=8) \
   .map(sns.distplot, "Weekly_Sales") \
   .add_legend();
plt.show();
```

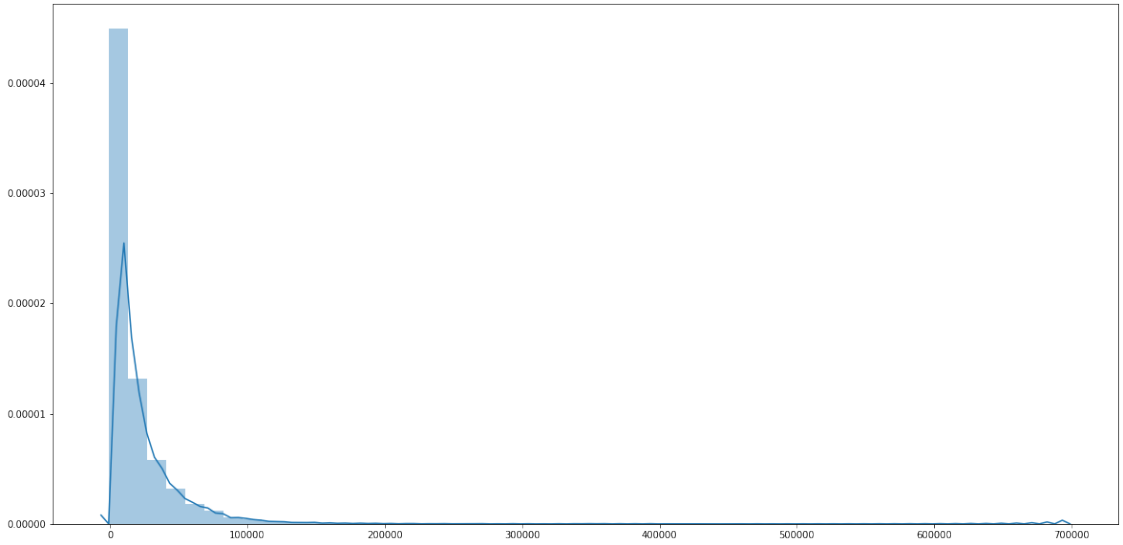


2.1 Inference

The PDF of weekly sales in holiday week and Non-holiday week are not separable using Holiday

[61]:
```
plt.figure(figsize=(20,10))
holiday_weekly_sales = train_df[train_df['IsHoliday']==True][['Weekly_Sales']]
non_holiday_weekly_sales =␣
 ↪train_df[train_df['IsHoliday']==False][['Weekly_Sales']]
print("Holiday weekly sales Distribution")
a =sns.distplot(holiday_weekly_sales)
```

```
Holiday weekly sales Distribution
```

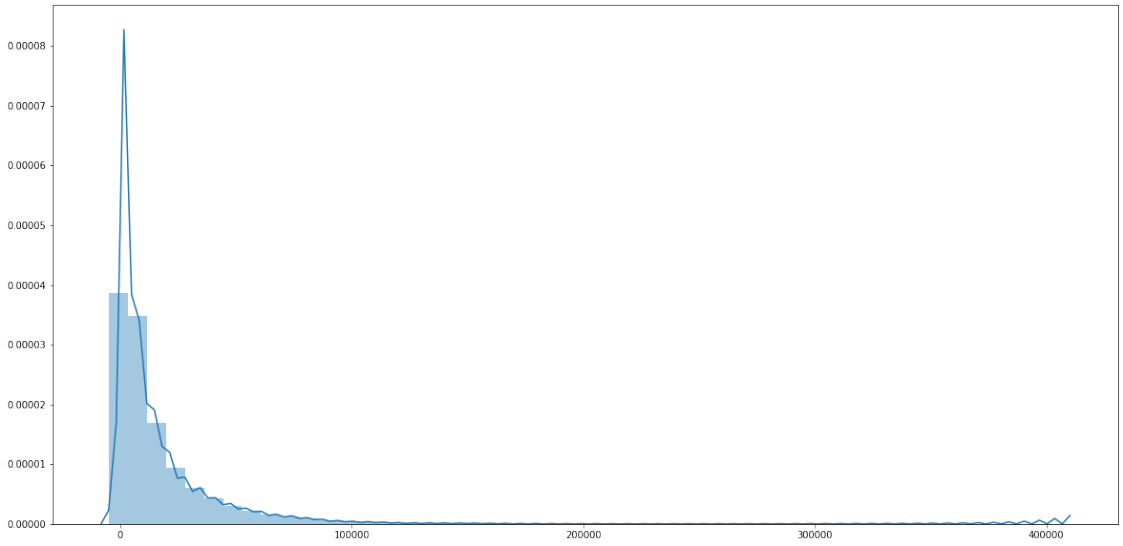


2.2 Inference

Not sure of type of distribution. Need to apply log transformation to see its log-normal distribution

[122]:
```
plt.figure(figsize=(20,10))
print("Non-Holiday weekly sales Distribution")
a = sns.distplot(non_holiday_weekly_sales)
```

```
Non-Holiday weekly sales Distribution
```



2.3 Inference

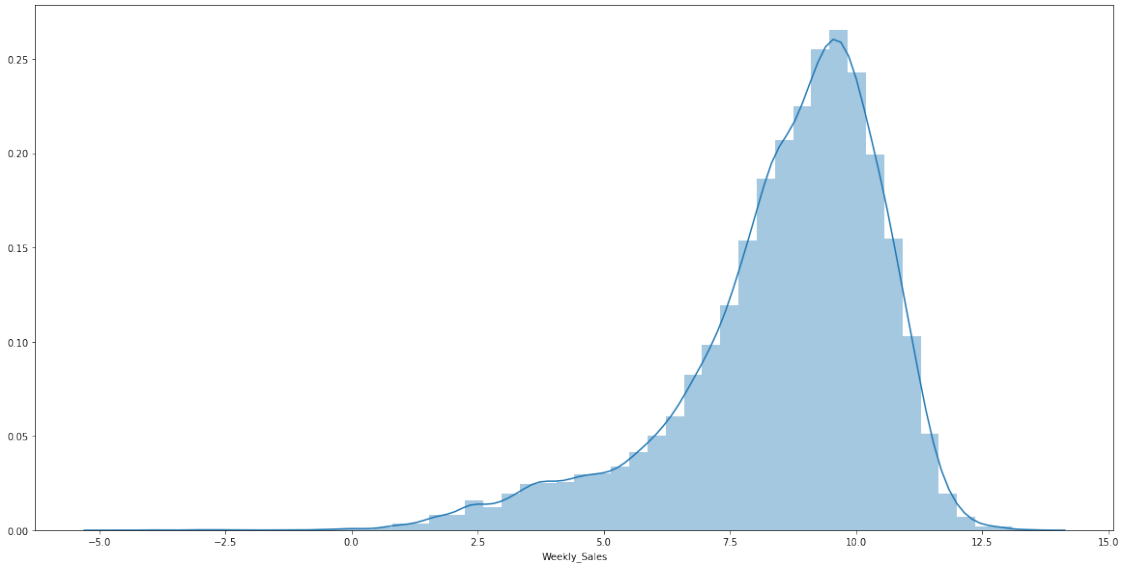
Not sure of type of distribution. Need to apply log transformation to see its log-normal distribution

3 log transformation of holiday weekly sales

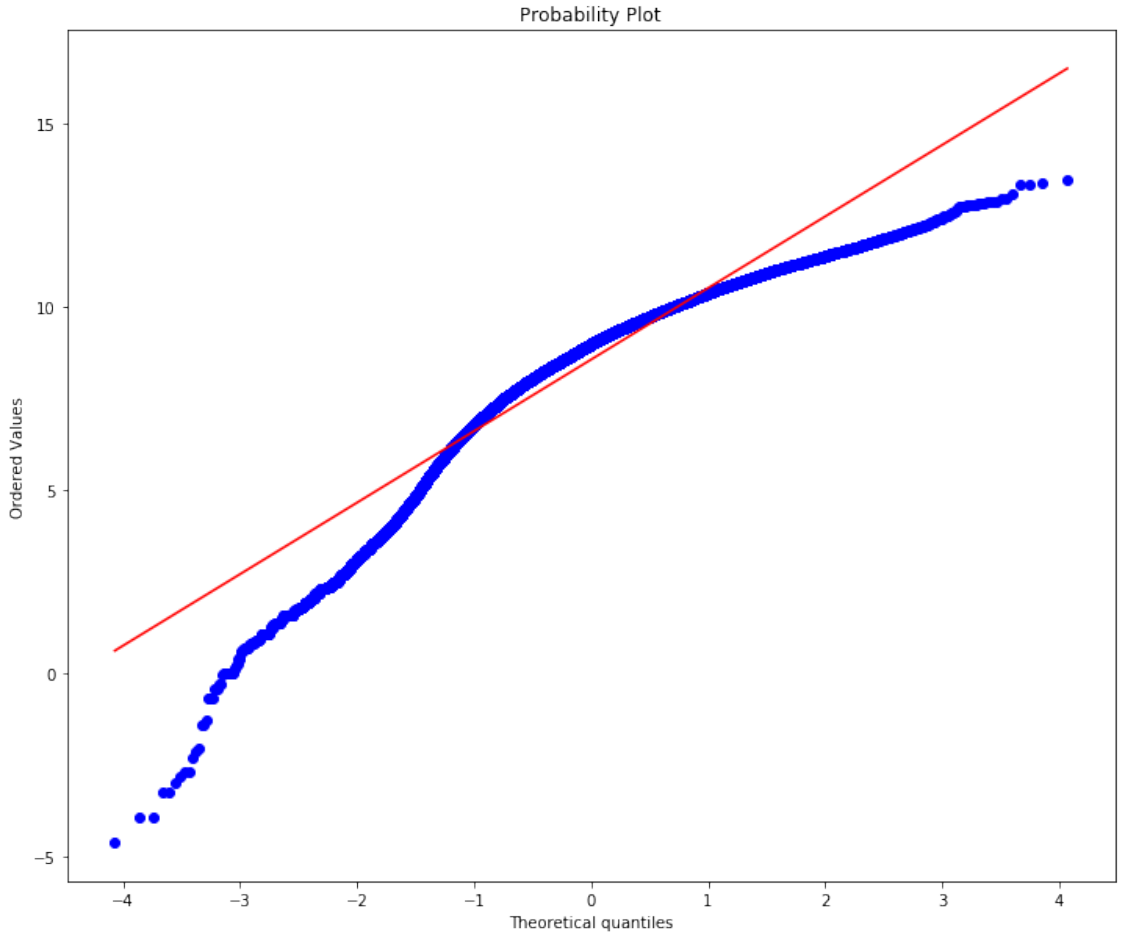
[56]:
```
plt.figure(figsize=(20,10))
holiday_weekly_sales = train_df[(train_df['IsHoliday']==True) &␣
 ↪(train_df['Weekly_Sales']>0)]
non_holiday_weekly_sales = train_df[(train_df['IsHoliday']==False) &␣
 ↪(train_df['Weekly_Sales']>0)]
print("Holiday weekly sales Distribution")
sns.distplot(holiday_weekly_sales['Weekly_Sales'].map(lambda x: np.log(x)))
```

```
Holiday weekly sales Distribution
```

[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19608cfa0b8>
```



[58]:
```
## check if this is log-normal distribution(Q-Q plot)
plt.figure(figsize=(12,10))
stats.probplot(holiday_weekly_sales['Weekly_Sales'].map(lambda x: np.log(x)),␣
 ↪dist="norm", plot=pylab)
pylab.show()
```



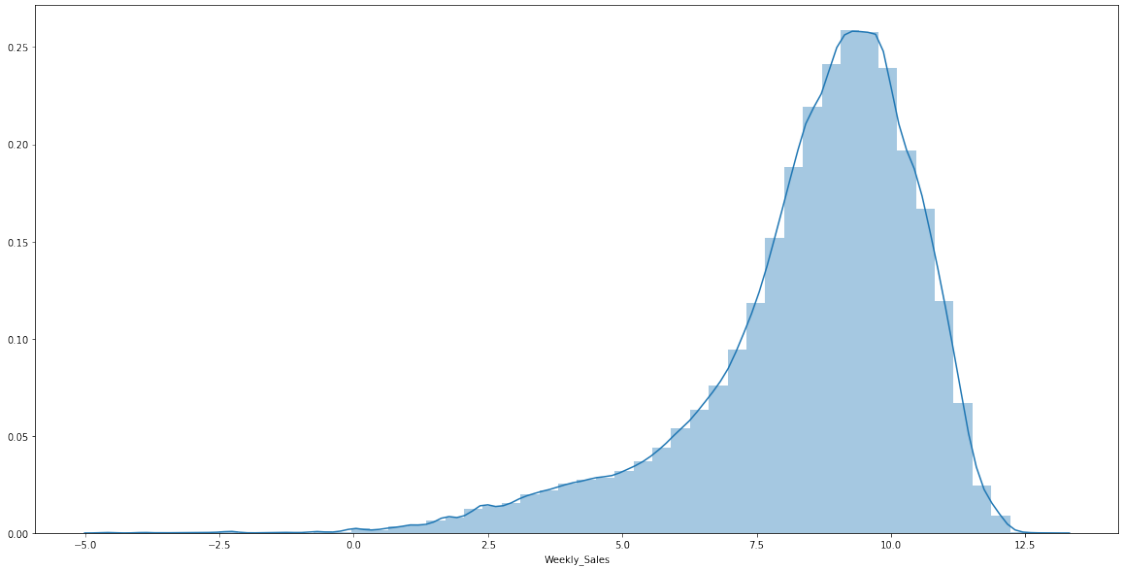
4 inference:

Holiday weekly sales is not log normally distributed

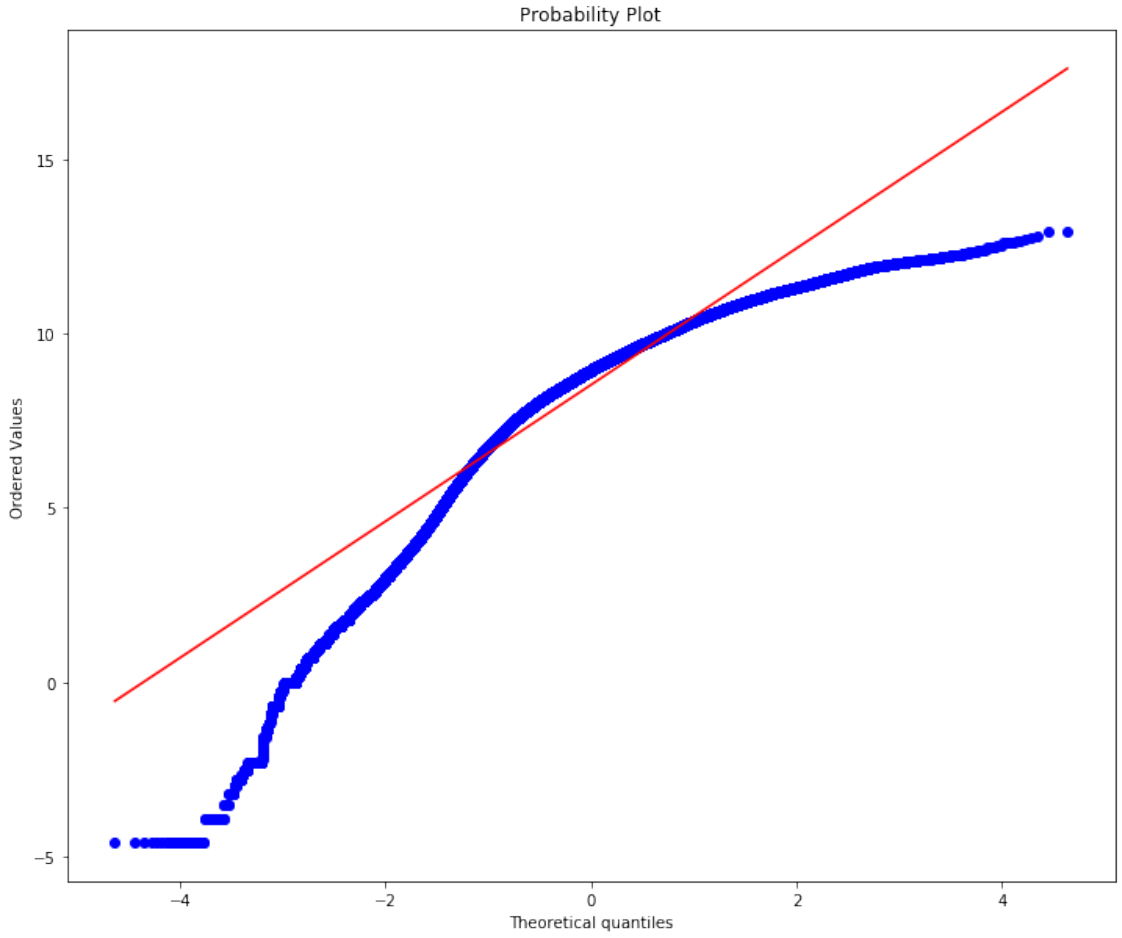
5 Log transformation of Non-holiday weekly sales

[59]:
```
plt.figure(figsize=(20,10))
sns.distplot(non_holiday_weekly_sales['Weekly_Sales'].map(lambda x: np.log(x)))
```

[59]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19608ea13c8>
```



[60]:
```
## check if this is log-normal distribution(Q-Q plot)
plt.figure(figsize=(12,10))
stats.probplot(non_holiday_weekly_sales['Weekly_Sales'].map(lambda x: np.
 ↪log(x)), dist="norm", plot=pylab)
pylab.show()
```



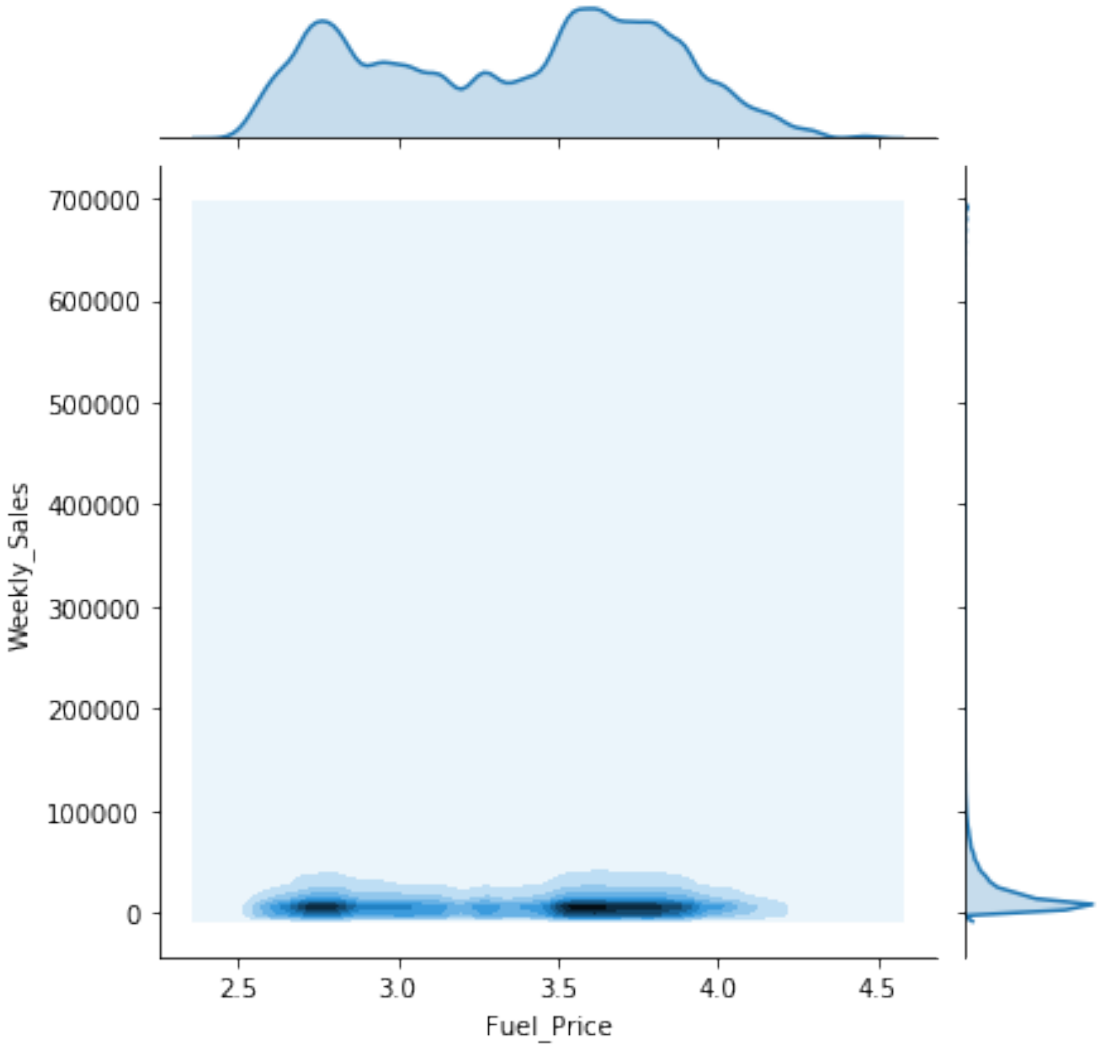
6 inference:

Non-Holiday weekly sales is not log normally distributed

6.0.1 Multivariate plot for Fuel Price

[27]:
```
#fig = plt.figure(figsize=(10, 8))
d = train_df[['Weekly_Sales','Fuel_Price','Temperature']]
sns.jointplot(x='Fuel_Price',y='Weekly_Sales',data=d,kind='kde')
```

[27]:
```
<seaborn.axisgrid.JointGrid at 0x19672497d30>
```



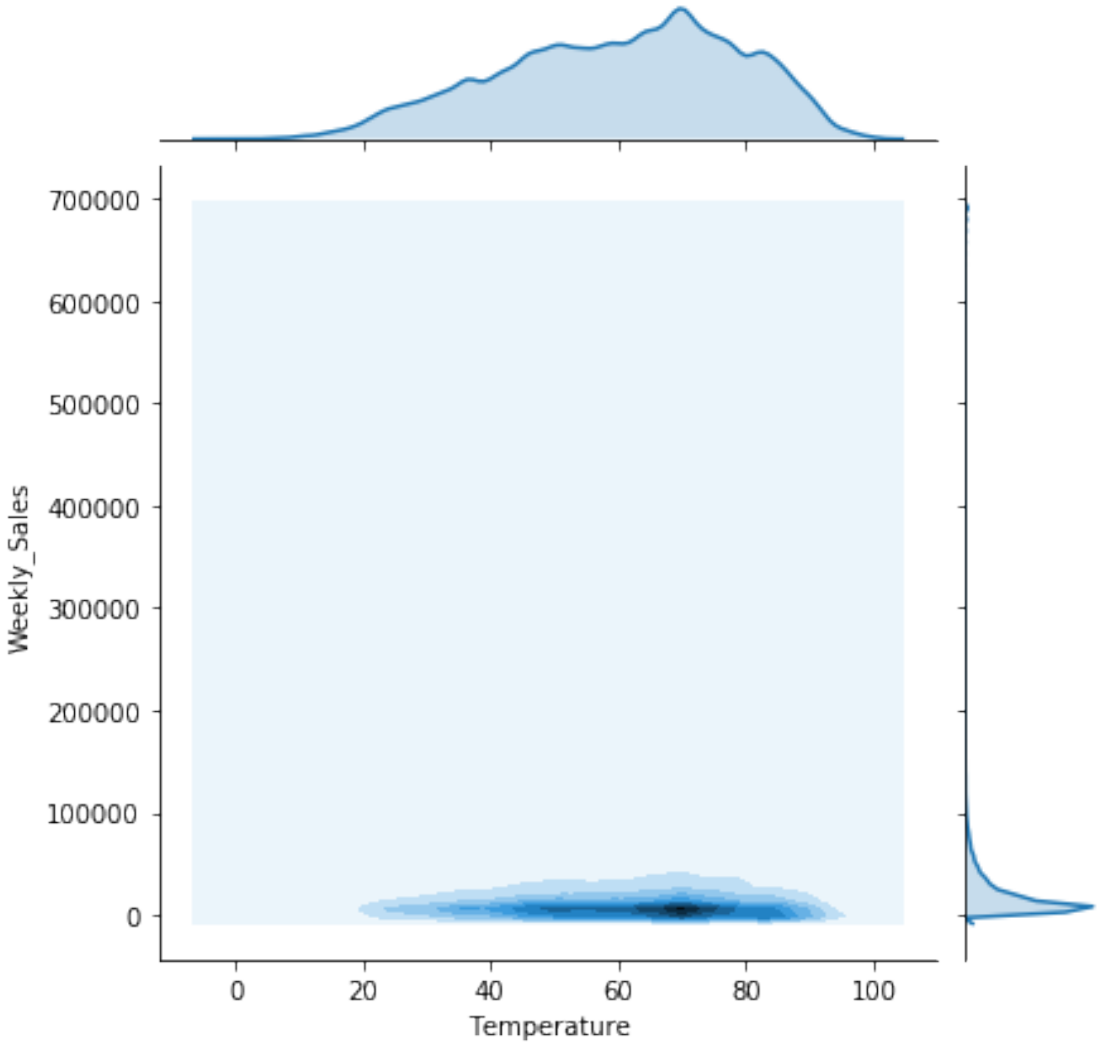
6.1 Inference

1. There is very low co-relation between lot of fuel price and Weekly Sales data.
2. Density of weekly sales is high between 2.5 and 3 and between 3.5 and 4.0 fuel price.

6.1.1 Multivariate plot for Temperature

[28]:
```
sns.jointplot(x='Temperature',y='Weekly_Sales',data=d,kind='kde')
```

[28]:
```
<seaborn.axisgrid.JointGrid at 0x196725b0a58>
```



6.2 Inference

1. There is very less co-relation between lot of fuel price and Weekly Sales data.
2. The high density of weekly sales for Temperature between 60 to 80.

[71]:
```
# we can also draw heat map correlation matrix
```

6.2.1 CPI and Unemployement Value Counts

[23]:
```
#train_df['CPI'].value_counts().plot(kind='bar')
train_df['CPI'].value_counts()
```

[23]:
```
129.855533    711
131.108333    708
```

```
129.845967    707
130.384903    706
130.683000    706
              ...
212.117421     45
207.620696     44
211.587991     44
213.173668     44
207.495309     44
Name: CPI, Length: 2145, dtype: int64
```

[24]:
```
train_df['Unemployment'].value_counts()
```

[24]:
```
8.099    5152
8.163    3636
7.852    3614
7.343    3416
7.057    3414
         ...
9.151     261
4.954     251
5.422     250
5.217     214
6.895     185
Name: Unemployment, Length: 349, dtype: int64
```

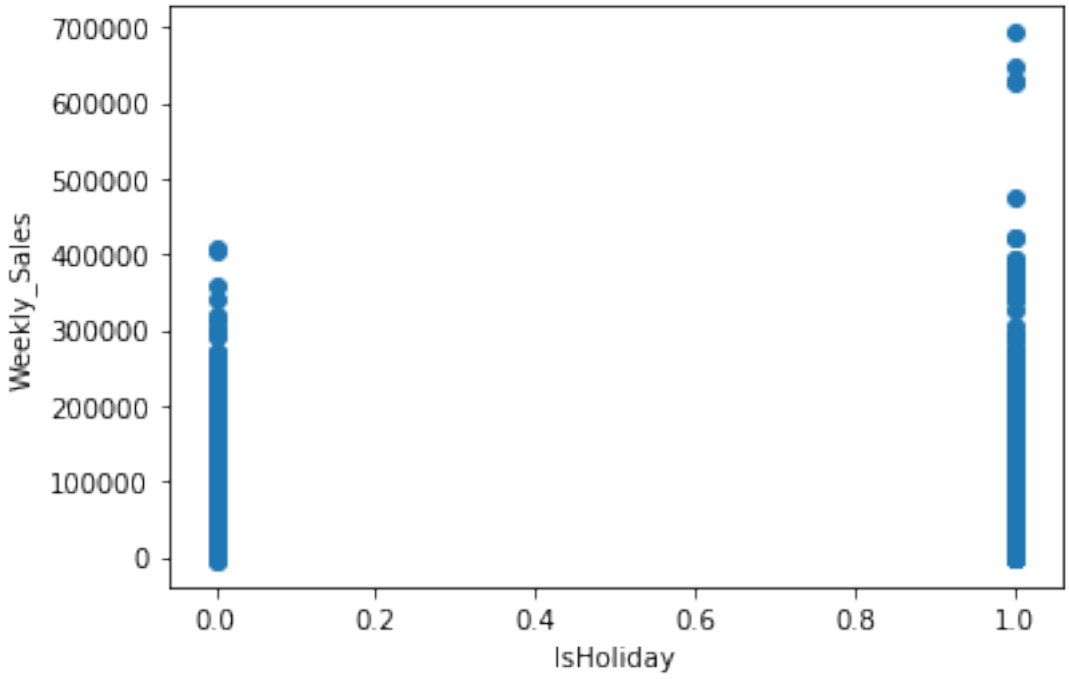
6.2.2 Number of records where more than 1 Markdown applied

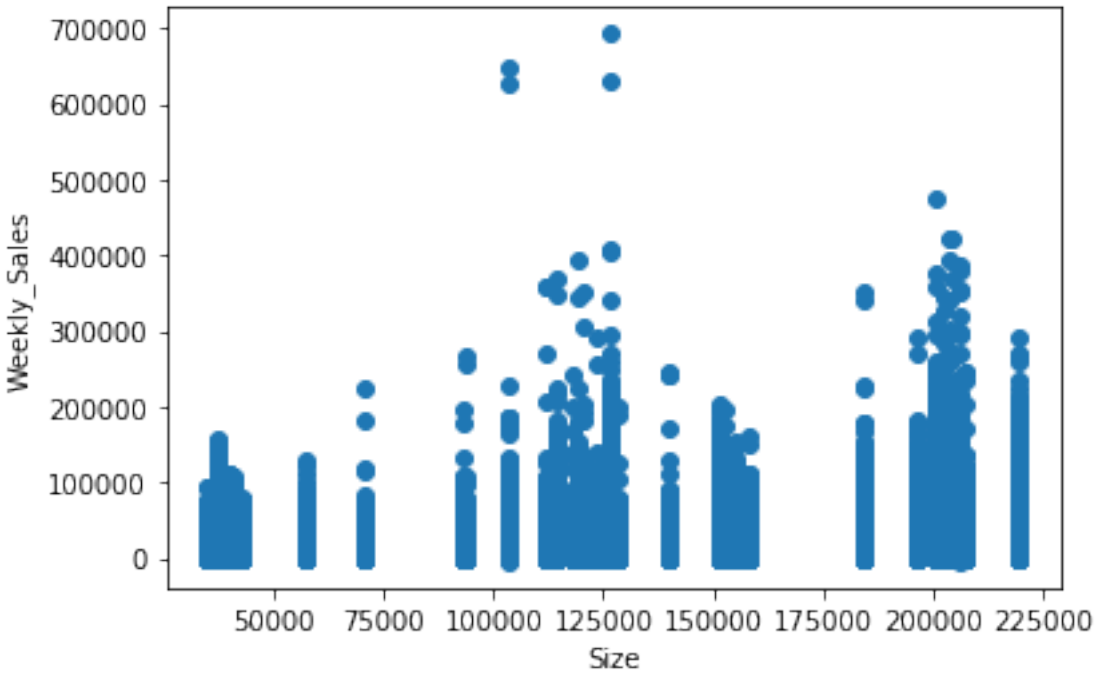
[38]:
```
#source:https://stackoverflow.com/questions/53939181/
 ↪select-rows-with-values-greater-than-v-in-more-than-c-columns
### Number of records where More than 1 Markdown applied
df = train_df[['MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']]
mask = (df.values !=0 ).sum(axis=1) > 1
train_df[mask].shape[0]
```

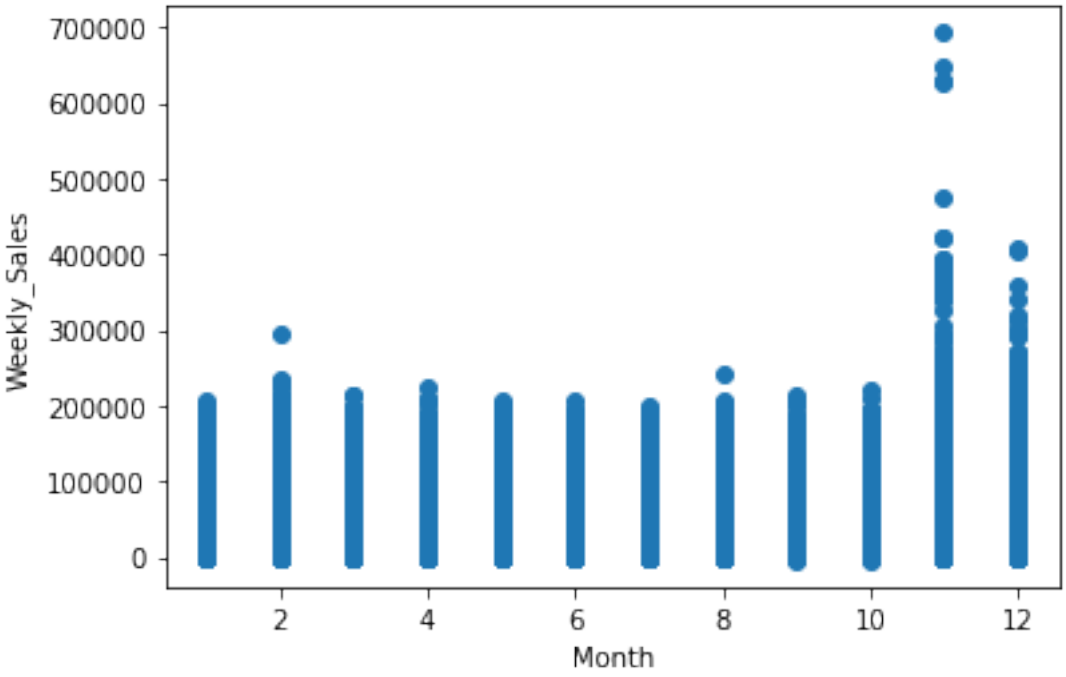
[38]:
```
151230
```

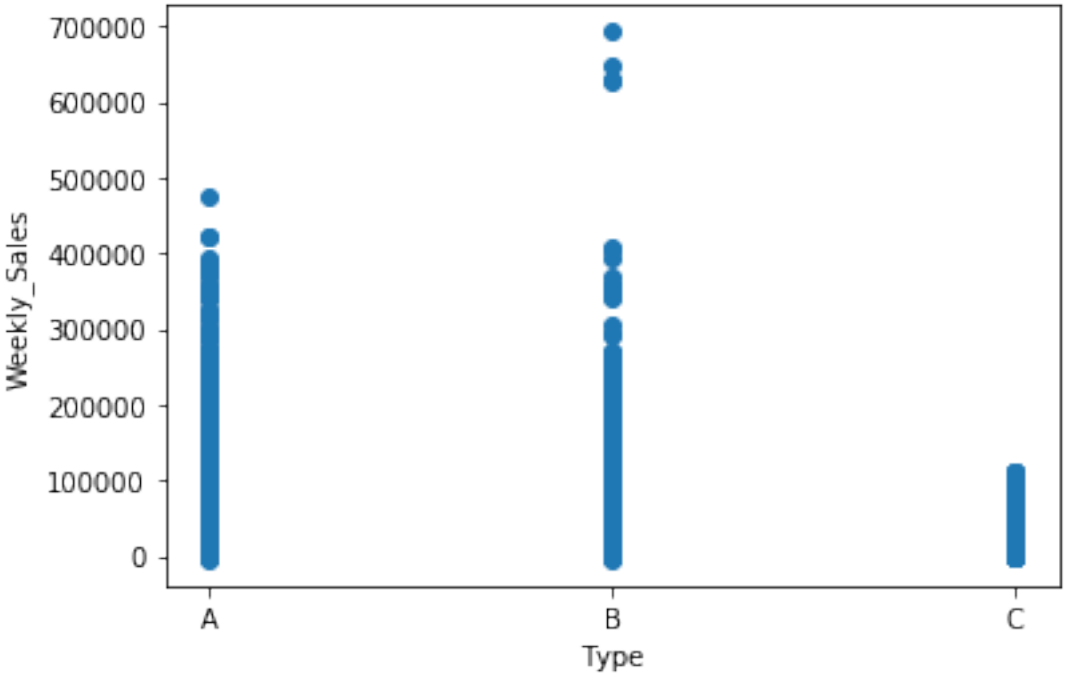
[35]:
```
def draw_scatter(df,column):
    plt.figure()
    plt.scatter(df[column] , df['Weekly_Sales'])
    plt.ylabel('Weekly_Sales')
    plt.xlabel(column)
```

[36]:
```
for column in ['IsHoliday','Size','Month','Type']:
    draw_scatter(train_df,column)
```


700000
600000
500000
400000
300000
200000
100000
0
Weekly_Sales
0.0
0.2
0.4
0.6
0.8
1.0
IsHoliday


700000
600000
500000
400000
300000
200000
100000
0
Weekly_Sales
50000
75000
100000
125000
150000
175000
200000
225000
Size



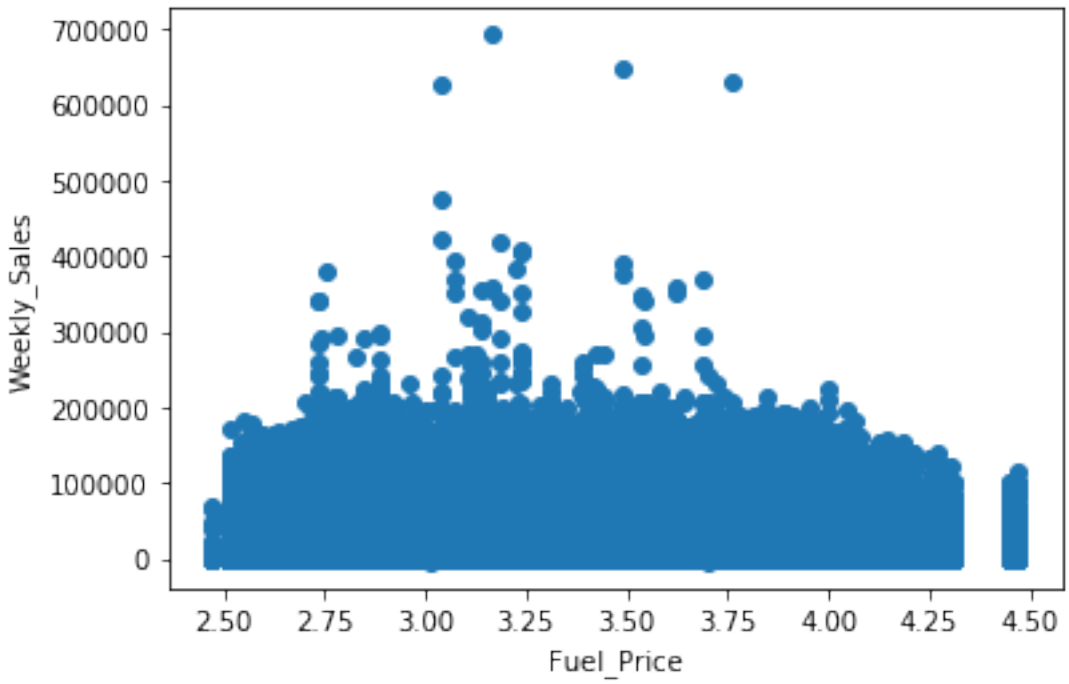


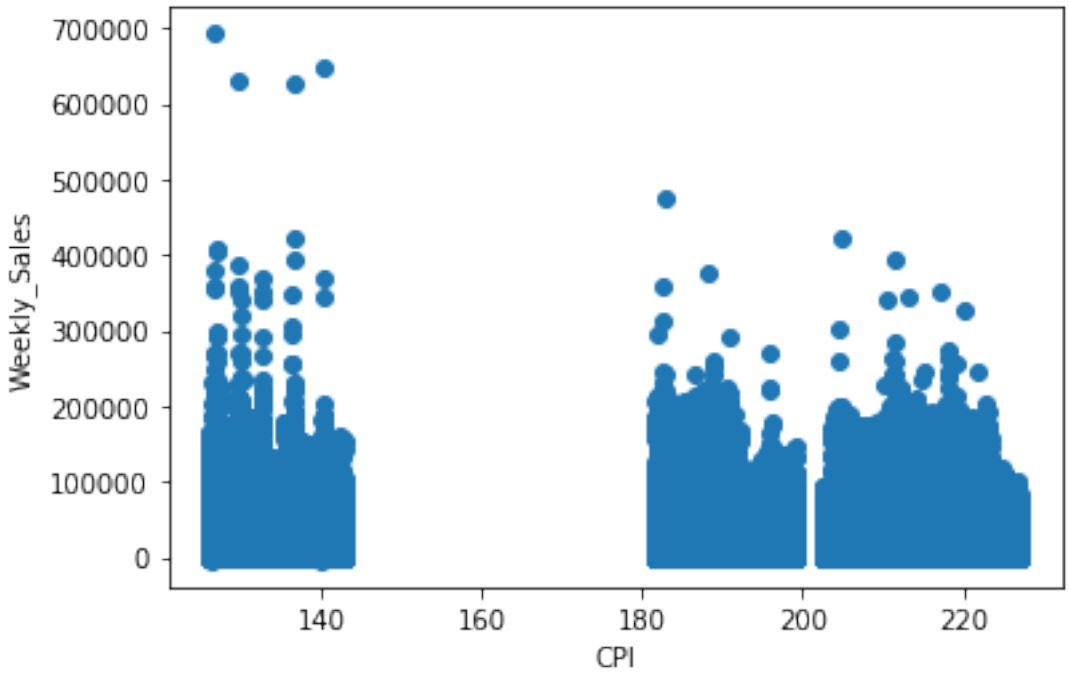
6.3 Inferences :

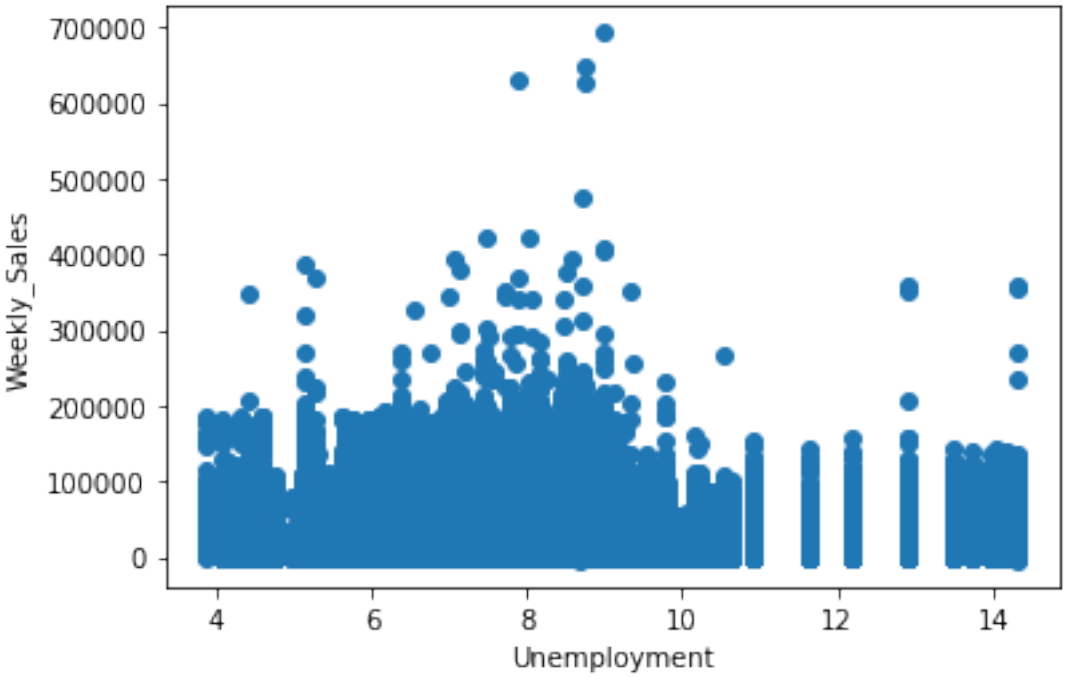
1. Weekly sales vary a lot in Month of november compared to other months.

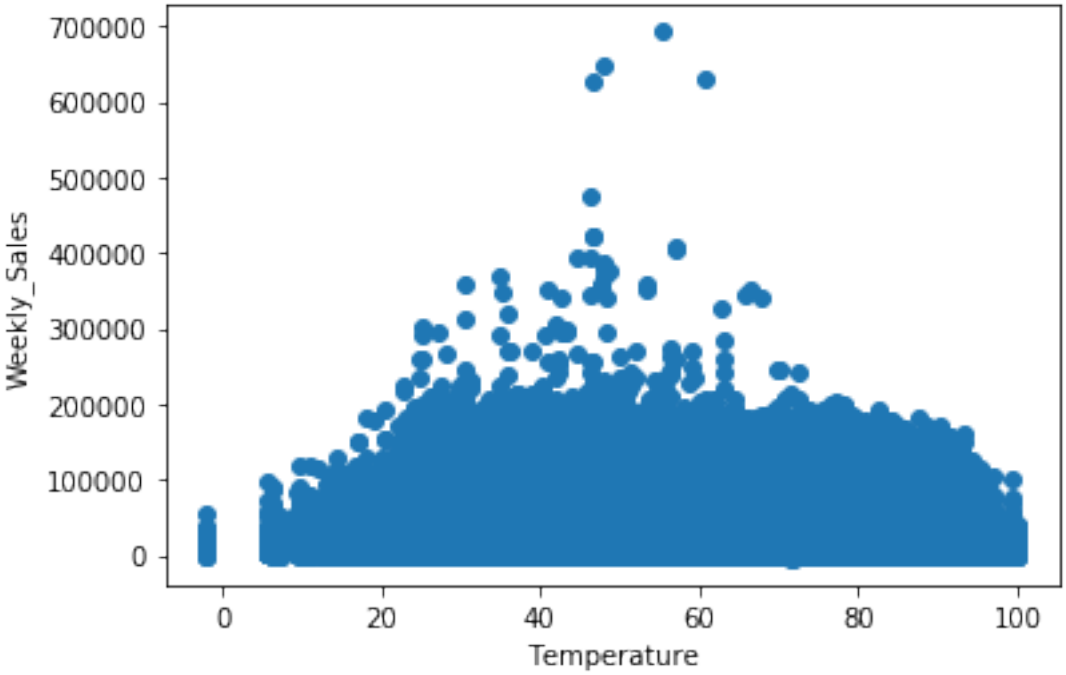
2. Weekly sales vary a lot on Holidays.
3. Weekly sales vary a lot for Type B stores.

[37]:
```
for column in ['Fuel_Price','CPI','Unemployment','Temperature']:
    draw_scatter(train_df,column)
```





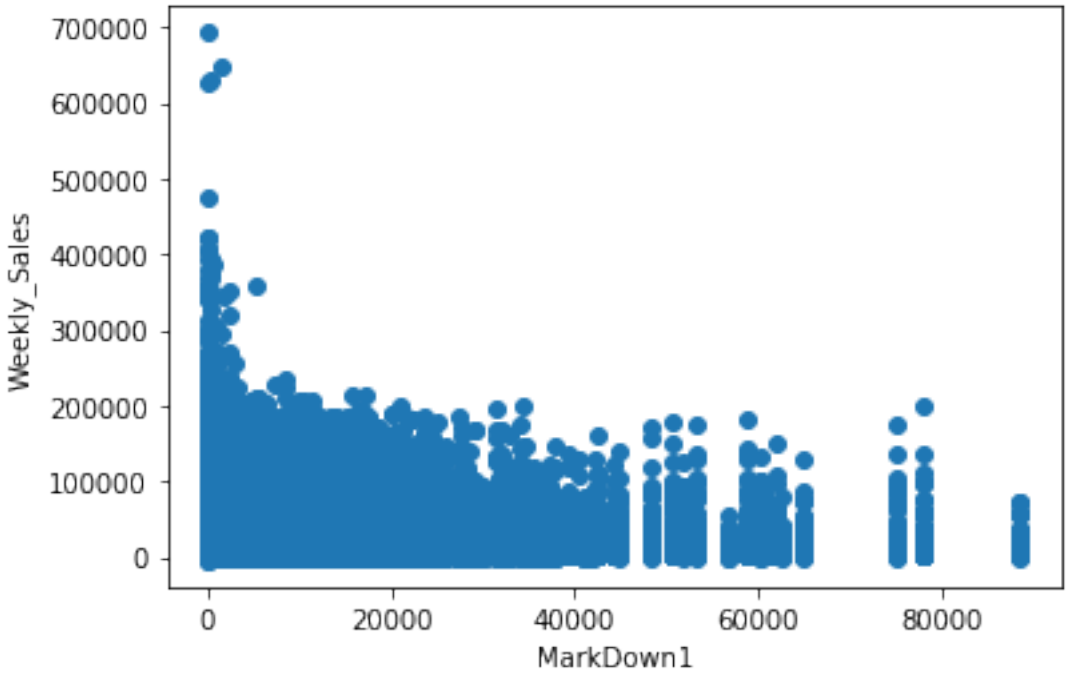

Weekly_Sales
700000
600000
500000
400000
300000
200000
100000
0
4
6
8
10
12
14
Unemployment

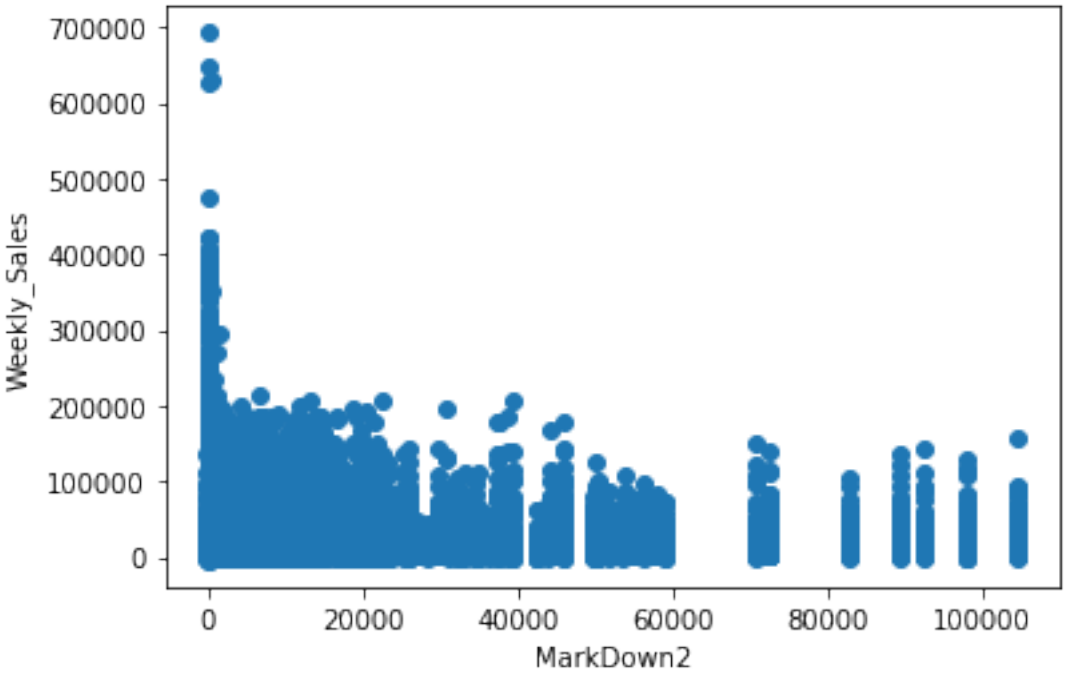

Weekly_Sales
700000
600000
500000
400000
300000
200000
100000
0
0
20
40
60
80
100
Temperature

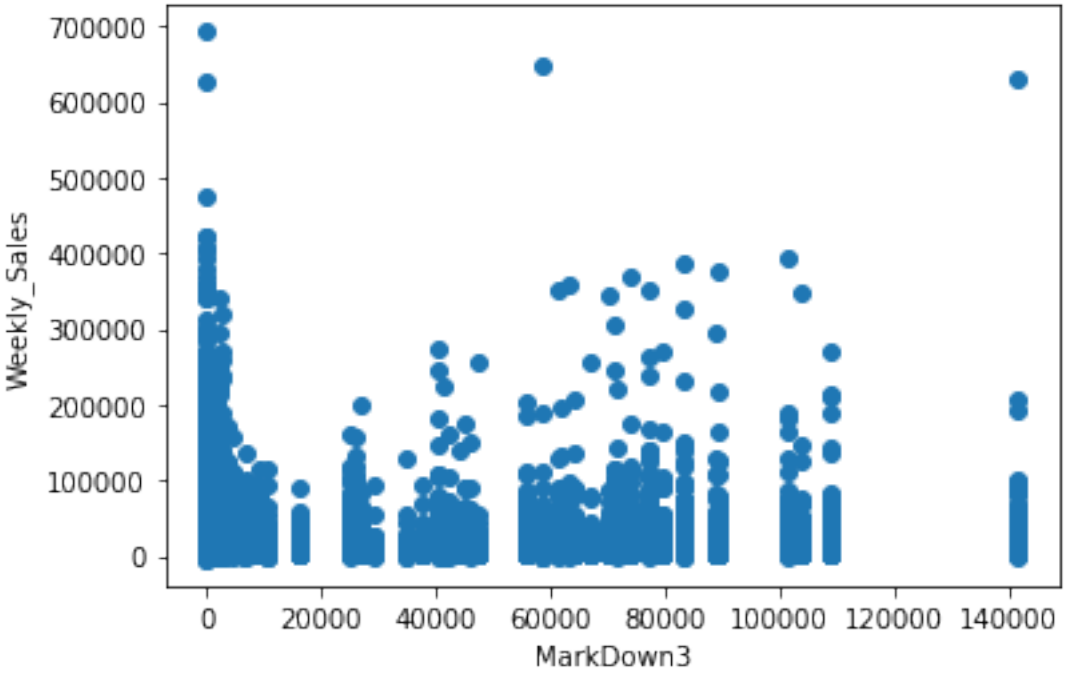
6.4 Inference

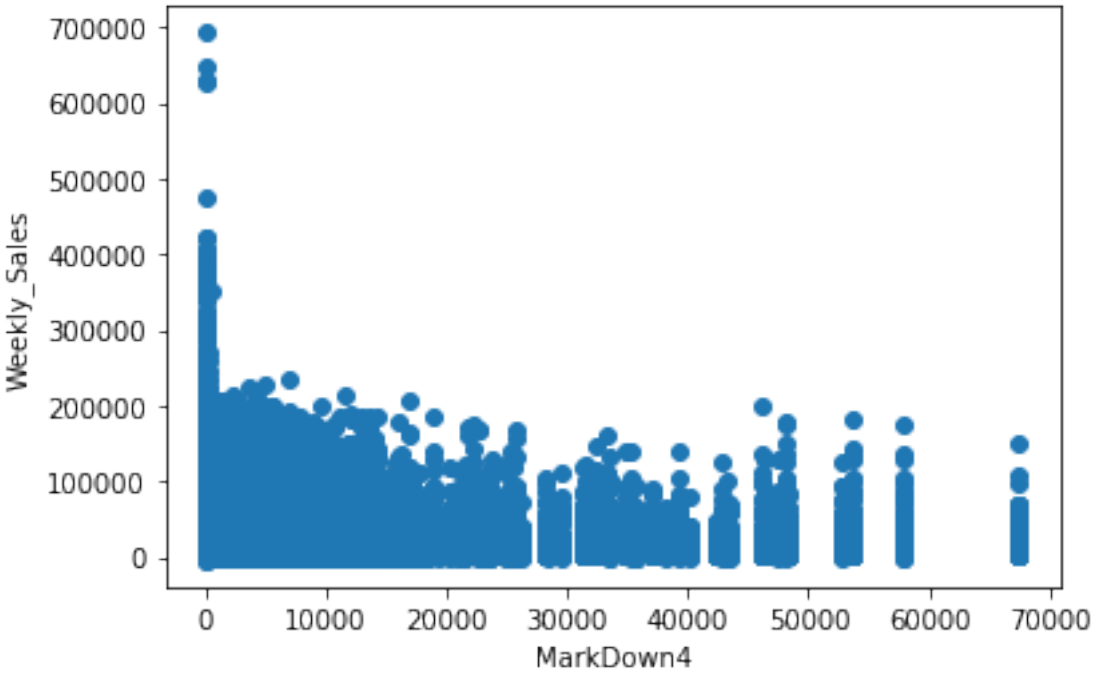
1. Weekly sales are weakly dependent on CPI and Fuel_price hence they can be eliminated from feature set.

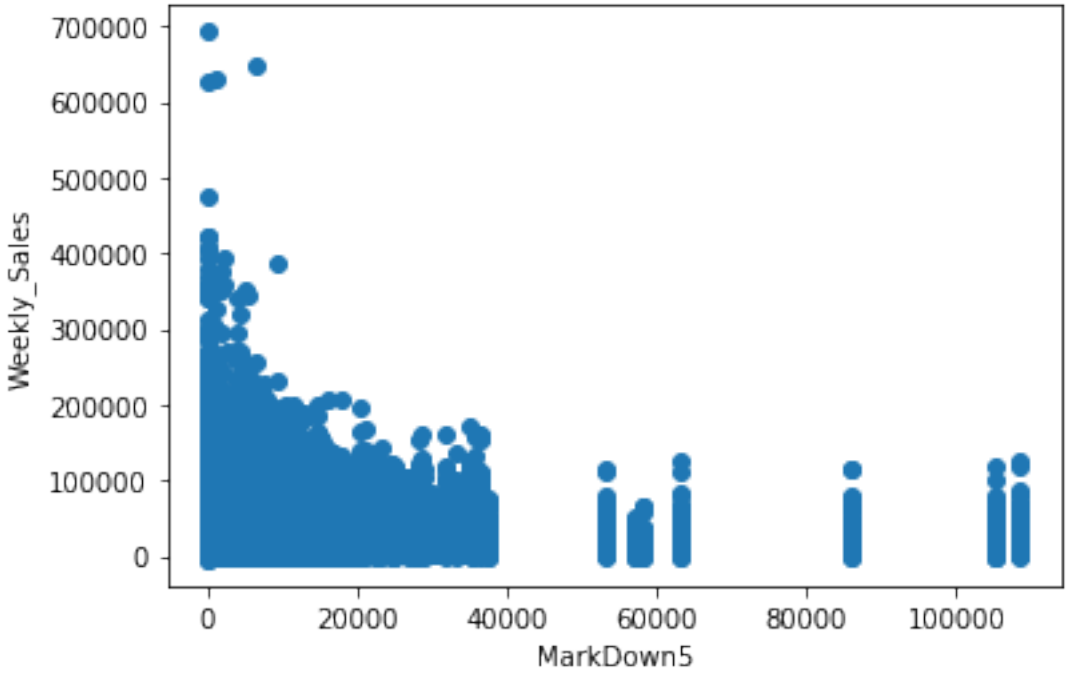
[38]:
```
for column in ['MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']:
    draw_scatter(train_df,column)
```




700000
600000
500000
400000
300000
200000
100000
0
Weekly_Sales
0
20000
40000
60000
80000
100000
MarkDown2


700000
600000
500000
400000
300000
200000
100000
0
Weekly_Sales
0
20000
40000
60000
80000
100000
120000
140000
MarkDown3





6.5 Inference

1. Weekly sales variation is more with MarkDown3, hence can be eliminated.

[40]:
```
# final feature set to work with
train_df.columns
```

[40]:
```
Index(['Store', 'Dept', 'Date', 'Weekly_Sales', 'IsHoliday', 'Type', 'Size',
       'Temperature', 'MarkDown1', 'MarkDown2', 'MarkDown4', 'MarkDown5',
       'Unemployment'],
      dtype='object')
```

7 Test_DF

[3]:
```
# create test_df similarly
#read test dataset and merge with store and features dataset in such a way that␣
 ↪all the data in train is preserved
test_df = pd.read_csv('test.csv')
store_df2 = pd.read_csv('stores.csv')
features_df2 = pd.read_csv('features.csv')
test_df = test_df.merge(store_df2,how='left').merge(features_df2,how='left')
test_df['Month'] = pd.to_datetime(test_df['Date']).dt.month
import datetime
test_df['Date'] =test_df['Date'].map(lambda x: datetime.datetime.strptime(x,␣
 ↪'%Y-%m-%d'))
test_df['Month'] = test_df['Date'].map(lambda x: x.month)
test_df['year_week_number'] = test_df['Date'].map(lambda x: datetime.date(x.
 ↪year, x.month, x.day).isocalendar()[1])
test_df = test_df.fillna(0)
# train_df1 = train_df1.drop(columns=['Fuel_Price','CPI','MarkDown3'])
test_df.head()
```

[3]:

	Store	Dept	Date	IsHoliday	Type	Size	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Month	year_week_number
0	1	1	2012-11-02	False	A	151315	55.32	3.386	6766.44	5147.70	50.82	3639.90	2737.42	223.462779	6.573	11	44
1	1	1	2012-11-09	False	A	151315	61.24	3.314	11421.32	3370.89	40.28	4646.79	6154.16	223.481307	6.573	11	45
2	1	1	2012-11-16	False	A	151315	52.92	3.252	9696.28	292.10	103.78	1133.15	6612.69	223.512911	6.573	11	46
3	1	1	2012-11-23	True	A	151315	56.23	3.211	883.59	4.17	74910.32	209.91	303.32	223.561947	6.573	11	47
4	1	1	2012-11-30	False	A	151315	52.34	3.207	2460.03	0.00	3838.35	150.57	6966.34	223.610984			

```
4         6.573     11                48
```

8 Train DF

[4]:
```
#read train dataset and merge with store and features dataset in such a way␣
 ↪that all the data in train is preserved
train_df1 = pd.read_csv('train.csv')
store_df1 = pd.read_csv('stores.csv')
features_df1 = pd.read_csv('features.csv')
# to draw pair-plots and heat map we replace date with month
train_df1 = train_df1.merge(store_df1,how='left').merge(features_df1,how='left')
train_df1['Month'] = pd.to_datetime(train_df1['Date']).dt.month
import datetime
train_df1['Date'] =train_df1['Date'].map(lambda x: datetime.datetime.
 ↪strptime(x, '%Y-%m-%d'))
train_df1['Month'] = train_df1['Date'].map(lambda x: x.month)
train_df1['year_week_number'] = train_df1['Date'].map(lambda x: datetime.date(x.
 ↪year, x.month, x.day).isocalendar()[1])
train_df1 = train_df1.fillna(0)
# train_df1 = train_df1.drop(columns=['Fuel_Price','CPI','MarkDown3'])
train_df1.head()
```

[4]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Type	Size	Temperature	\
0	1	1	2010-02-05	24924.50	False	A	151315	42.31	
1	1	1	2010-02-12	46039.49	True	A	151315	38.51	
2	1	1	2010-02-19	41595.55	False	A	151315	39.93	
3	1	1	2010-02-26	19403.54	False	A	151315	46.63	
4	1	1	2010-03-05	21827.90	False	A	151315	46.50	

	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	\
0	2.572	0.0	0.0	0.0	0.0	0.0	
1	2.548	0.0	0.0	0.0	0.0	0.0	
2	2.514	0.0	0.0	0.0	0.0	0.0	
3	2.561	0.0	0.0	0.0	0.0	0.0	
4	2.625	0.0	0.0	0.0	0.0	0.0	

	CPI	Unemployment	Month	year_week_number
0	211.096358	8.106	2	5
1	211.242170	8.106	2	6
2	211.289143	8.106	2	7
3	211.319643	8.106	2	8
4	211.350143	8.106	3	9

[5]:
```
# create month_week_number feature from dataset such that start week is from␣
 ↪train dataset and end week from test
# with 7 days a week. This feature will be used by baseline model EMA
# given date find week number to search in history for corresponding month
```

```
import datetime
train_df1['Date'] = pd.to_datetime(train_df1.Date)
train_df1 = train_df1.sort_values(by='Date')
start_date = train_df1.iloc[0]['Date']
print("Start date:",start_date)

test_df['Date'] = pd.to_datetime(test_df.Date)
end_date = test_df.sort_values(by='Date').iloc[-1]['Date']
print("End date:",end_date)
date_to_week_number = dict()
week_number = 1
date_to_week_number[start_date] = week_number
curr_month = start_date.month
next_date = start_date
while next_date != end_date:
    next_date = next_date + datetime.timedelta(days=7)
    week_number += 1
    if curr_month != next_date.month:
        week_number = 1
        curr_month =  next_date.month
    date_to_week_number[next_date] = week_number
```

```
Start date: 2010-02-05 00:00:00
End date: 2013-07-26 00:00:00
```

[6]:
```
from datetime import datetime
datetime_object = datetime.strptime('2010-02-12', '%Y-%m-%d')
print(datetime_object)
print("Week number:",date_to_week_number[datetime_object])
```

```
2010-02-12 00:00:00
Week number: 2
```

[7]:
```
train_df1['month_week_number'] = train_df1['Date'].map(lambda date:␣
 ↪date_to_week_number[date])
test_df['month_week_number'] = test_df['Date'].map(lambda date:␣
 ↪date_to_week_number[date])
```

[8]:
```
formater = '%Y-%m-%d'
train_df1['date_week_number'] = train_df1['Date'].map(lambda date: f"{date.
 ↪strftime(formater)},WN:{date_to_week_number[date]}")
```

[9]:
```
train_df1.head()
```

[9]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Type	Size	\
0	1	1	2010-02-05	24924.50	False	A	151315	
277665	29	5	2010-02-05	15552.08	False	B	93638	
277808	29	6	2010-02-05	3200.22	False	B	93638	

```
277951     29      7 2010-02-05       10820.05      False    B   93638
278094     29      8 2010-02-05       20055.64      False    B   93638

        Temperature  Fuel_Price  MarkDown1  MarkDown2  MarkDown3  MarkDown4  \
0             42.31       2.572        0.0        0.0        0.0        0.0
277665        24.36       2.788        0.0        0.0        0.0        0.0
277808        24.36       2.788        0.0        0.0        0.0        0.0
277951        24.36       2.788        0.0        0.0        0.0        0.0
278094        24.36       2.788        0.0        0.0        0.0        0.0

        MarkDown5         CPI  Unemployment  Month  year_week_number  \
0             0.0  211.096358         8.106      2                 5
277665        0.0  131.527903        10.064      2                 5
277808        0.0  131.527903        10.064      2                 5
277951        0.0  131.527903        10.064      2                 5
278094        0.0  131.527903        10.064      2                 5

        month_week_number date_week_number
0                       1  2010-02-05,WN:1
277665                  1  2010-02-05,WN:1
277808                  1  2010-02-05,WN:1
277951                  1  2010-02-05,WN:1
278094                  1  2010-02-05,WN:1
```

[10]:
```
train_df1['month_week_number'].isnull().values.any()
```

[10]:
```
False
```

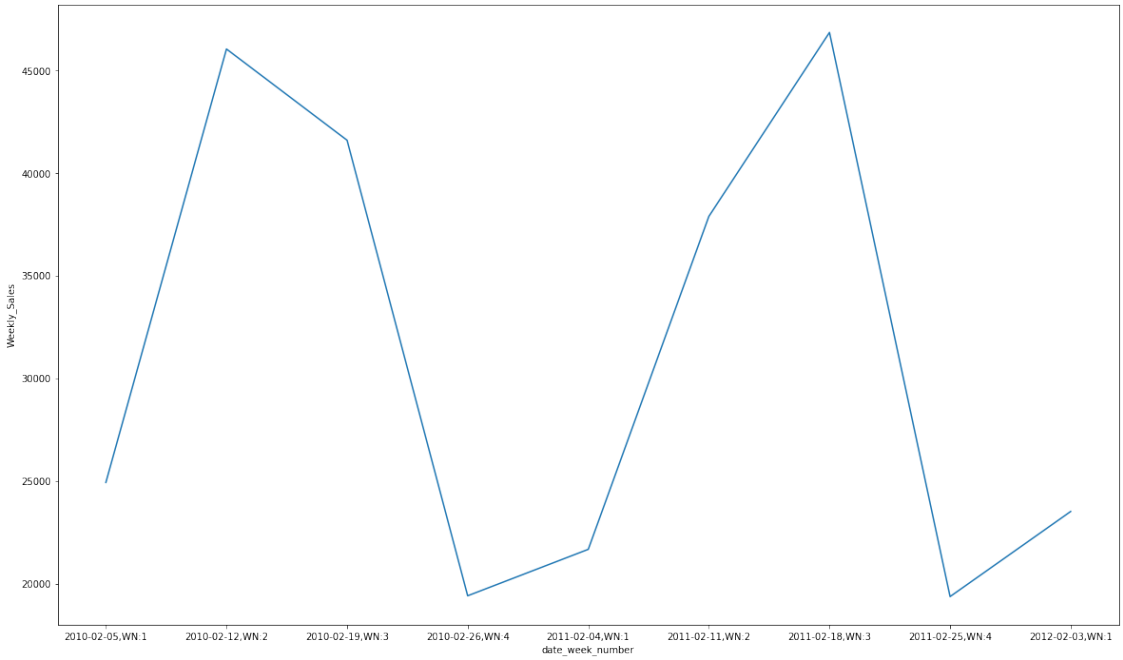
8.1 Finding if there is continuity between sales for corresponding weeks of a month between different years

[85]:
```
for name,group in train_df1.groupby(['Store','Dept']):
    #print("store and department:",name)
    if(name[0]==1 and name[1]==1):
        print("store and department:",name)
        #print(group[['Month','Weekly_Sales','Date']].head(10))
        ␣
↪print(group[['Month','Weekly_Sales','date_week_number','IsHoliday','MarkDown1','MarkDown2',
↪head(9))
        fig = plt.figure(figsize=(20, 12))
        d =␣
↪group[['Month','Weekly_Sales','date_week_number']][group['Month']==2][:9]
        #d['Str_Date'] = d['Date'].map(lambda x: x.strftime('%Y-%m-%d'))
        sns.lineplot(y='Weekly_Sales',x='date_week_number',data=d)
        break
```

```
store and department: (1, 1)
     Month  Weekly_Sales date_week_number  IsHoliday  MarkDown1  MarkDown2  \
```

0	2	24924.50	2010-02-05,WN:1	False	0.00	0.00
1	2	46039.49	2010-02-12,WN:2	True	0.00	0.00
2	2	41595.55	2010-02-19,WN:3	False	0.00	0.00
3	2	19403.54	2010-02-26,WN:4	False	0.00	0.00
52	2	21665.76	2011-02-04,WN:1	False	0.00	0.00
53	2	37887.17	2011-02-11,WN:2	True	0.00	0.00
54	2	46845.87	2011-02-18,WN:3	False	0.00	0.00
55	2	19363.83	2011-02-25,WN:4	False	0.00	0.00
104	2	23510.49	2012-02-03,WN:1	False	34577.06	3579.21

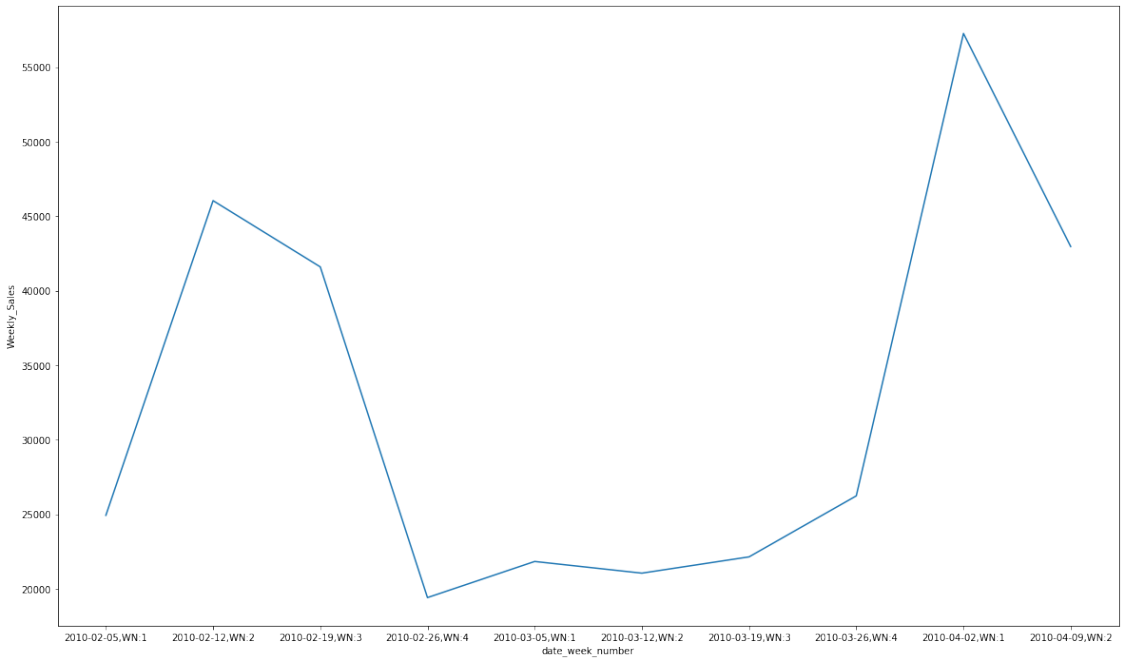
	MarkDown3	MarkDown4	MarkDown5
0	0.00	0.00	0.0
1	0.00	0.00	0.0
2	0.00	0.00	0.0
3	0.00	0.00	0.0
52	0.00	0.00	0.0
53	0.00	0.00	0.0
54	0.00	0.00	0.0
55	0.00	0.00	0.0
104	160.53	32403.87	5630.4



8.2 Finding if there is continuity for sales between different months of same year

[86]:
```
for name,group in train_df1.groupby(['Store','Dept']):
    print("store and dept",name)
    #print(group[['Month','Weekly_Sales','Date']].head(10))
    print(group[['Month','Weekly_Sales','date_week_number']].head(10))
    fig = plt.figure(figsize=(20, 12))
    sns.
↪lineplot(y='Weekly_Sales',x='date_week_number',data=group[['Month','Weekly_Sales','date_wee
↪head(10))
    break
```

```
store and dept (1, 1)
   Month  Weekly_Sales date_week_number
0      2      24924.50  2010-02-05,WN:1
1      2      46039.49  2010-02-12,WN:2
2      2      41595.55  2010-02-19,WN:3
3      2      19403.54  2010-02-26,WN:4
4      3      21827.90  2010-03-05,WN:1
5      3      21043.39  2010-03-12,WN:2
6      3      22136.64  2010-03-19,WN:3
7      3      26229.21  2010-03-26,WN:4
8      4      57258.43  2010-04-02,WN:1
9      4      42960.91  2010-04-09,WN:2
```



9 We clearly see that continuity exists between corresponding months of different years

[14]:
```
train_df1 = train_df1.sort_values(by='Date')
train_df1 = train_df1.reset_index(drop = True)
train_df1['year'] = train_df1['Date'].map(lambda d: d.year)
```

[15]:
```
test_df['year'] = test_df['Date'].map(lambda d: d.year)
```

[16]:
```
#Split data such that cv has similar data set in terms of months compared to␣
 ↪test
import datetime
def dateconverter(date):
    date_object = datetime.datetime.strptime(date, '%Y-%m-%d').date()
    return date_object
def datecompare(date1,date2):
    if date1>date2:
        return True
    return False
partition_date = '2011-11-01'
inde=-1
for index,row in train_df1.iterrows():
    if datecompare(row['Date'],dateconverter(partition_date)):
        inde=index
        break
print("*"*50,inde)
```

********** 267184

[17]:
```
train_df = train_df1.iloc[:inde]
cv_df = train_df1.iloc[inde:]
```

[18]:
```
train_df.tail(10)
```

[18]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Type	Size	\
267174	11	3	2011-10-28	11220.12	False	A	207499	
267175	21	79	2011-10-28	21996.16	False	B	140167	
267176	6	79	2011-10-28	20005.18	False	A	202505	
267177	42	42	2011-10-28	204.07	False	C	39690	
267178	18	32	2011-10-28	7490.64	False	B	120653	
267179	25	30	2011-10-28	2274.38	False	B	128107	
267180	13	14	2011-10-28	21242.32	False	A	219622	
267181	21	80	2011-10-28	4.98	False	B	140167	
267182	11	4	2011-10-28	26975.64	False	A	207499	
267183	29	10	2011-10-28	8723.09	False	B	93638	

	Temperature	Fuel_Price	MarkDown1	...	MarkDown3	MarkDown4	\
267174	72.66	3.372	0.0	...	0.0	0.0	
267175	65.46	3.372	0.0	...	0.0	0.0	

```
267176        69.51       3.372        0.0  ...        0.0        0.0
267177        72.79       3.843        0.0  ...        0.0        0.0
267178        45.61       3.604        0.0  ...        0.0        0.0
267179        46.28       3.569        0.0  ...        0.0        0.0
267180        47.41       3.567        0.0  ...        0.0        0.0
267181        65.46       3.372        0.0  ...        0.0        0.0
267182        72.66       3.372        0.0  ...        0.0        0.0
267183        49.31       3.604        0.0  ...        0.0        0.0

        MarkDown5         CPI  Unemployment  Month  year_week_number  \
267174        0.0  221.080184         7.197     10                43
267175        0.0  217.325182         7.441     10                43
267176        0.0  219.237049         6.551     10                43
267177        0.0  129.793677         7.874     10                43
267178        0.0  136.488452         8.471     10                43
267179        0.0  210.691890         7.082     10                43
267180        0.0  129.793677         6.392     10                43
267181        0.0  217.325182         7.441     10                43
267182        0.0  221.080184         7.197     10                43
267183        0.0  136.488452         9.357     10                43

        month_week_number  date_week_number  year
267174                  4   2011-10-28,WN:4  2011
267175                  4   2011-10-28,WN:4  2011
267176                  4   2011-10-28,WN:4  2011
267177                  4   2011-10-28,WN:4  2011
267178                  4   2011-10-28,WN:4  2011
267179                  4   2011-10-28,WN:4  2011
267180                  4   2011-10-28,WN:4  2011
267181                  4   2011-10-28,WN:4  2011
267182                  4   2011-10-28,WN:4  2011
267183                  4   2011-10-28,WN:4  2011

[10 rows x 21 columns]
```

[19]:
```
cv_df.head()
```

[19]:
```
        Store  Dept       Date  Weekly_Sales  IsHoliday Type    Size  \
267184      4    87 2011-11-04      12931.91      False    A  205863
267185     28    81 2011-11-04      27492.47      False    A  206302
267186     27    83 2011-11-04       6348.32      False    A  204184
267187     18    11 2011-11-04      32225.10      False    B  120653
267188      2    90 2011-11-04      97130.54      False    A  202307

        Temperature  Fuel_Price  MarkDown1  ...  MarkDown3  MarkDown4  \
267184        49.86       3.322        0.0  ...        0.0        0.0
267185        59.77       3.828        0.0  ...        0.0        0.0
267186        44.46       3.738        0.0  ...        0.0        0.0
```

```
267187        38.29       3.586         0.0  ...         0.0         0.0
267188        55.53       3.332         0.0  ...         0.0         0.0

        MarkDown5         CPI  Unemployment  Month  year_week_number  \
267184        0.0  129.805194         5.143     11                44
267185        0.0  129.805194        12.890     11                44
267186        0.0  140.427976         7.906     11                44
267187        0.0  136.475129         8.471     11                44
267188        0.0  217.485360         7.441     11                44

        month_week_number  date_week_number  year
267184                  1   2011-11-04,WN:1  2011
267185                  1   2011-11-04,WN:1  2011
267186                  1   2011-11-04,WN:1  2011
267187                  1   2011-11-04,WN:1  2011
267188                  1   2011-11-04,WN:1  2011

[5 rows x 21 columns]
```

[20]:
```
train_df.shape
```

[20]:
```
(267184, 21)
```

[21]:
```
cv_df.shape
```

[21]:
```
(154386, 21)
```

[22]:
```
#train_df.shape
```

[24]:
```
store_dept_month_week_group_all = dict()
key_sep="$"
for name,group in train_df1.
 ↪groupby(['Store','Dept','Month','month_week_number']):
    key =␣
 ↪str(name[0])+key_sep+str(name[1])+key_sep+str(name[2])+key_sep+str(name[3])
    if key not in store_dept_month_week_group_all:
        store_dept_month_week_group_all[key] = group
```

[25]:
```
# Reason for getting 52 weeks before sale not possible using just month week␣
 ↪number and month
# as we can see below different year_week_number possible for same (month week␣
 ↪number and month)
# Hence we use year_week number to obtain 52 weeeks before sale
for key,groups in store_dept_month_week_group_all.items():
    if len(groups['year_week_number'].map(lambda x: int(x)).unique())>1:
        print(groups)
        break
```

```
        Store  Dept       Date  Weekly_Sales  IsHoliday Type    Size  \
24716       1     1 2010-04-02      57258.43      False    A  151315
```

```
177086      1     1 2011-04-01      20398.09      False    A  151315
334998      1     1 2012-04-06      57592.12      False    A  151315

        Temperature  Fuel_Price  MarkDown1  ...  MarkDown3  MarkDown4  \
24716         62.27       2.719       0.00  ...       0.00       0.00
177086        59.17       3.524       0.00  ...       0.00       0.00
334998        70.43       3.891   10121.97  ...      77.98    3750.59

        MarkDown5         CPI  Unemployment  Month  year_week_number  \
24716        0.00  210.820450         7.808      4                13
177086       0.00  214.837166         7.682      4                13
334998    4510.72  221.435611         7.143      4                14

        month_week_number  date_week_number  year
24716                   1   2010-04-02,WN:1  2010
177086                  1   2011-04-01,WN:1  2011
334998                  1   2012-04-06,WN:1  2012

[3 rows x 21 columns]
```

[23]:
```
store_dept_year_week_group_all = dict()
key_sep="$"
for name,group in train_df1.groupby(['Store','Dept','year_week_number']):
    key = str(name[0])+key_sep+str(name[1])+key_sep+str(name[2])
    if key not in store_dept_year_week_group_all:
        store_dept_year_week_group_all[key] = group
```

[117]:
```
train_df_2011 = train_df[(train_df['Date'] >= datetime.datetime.
 ↪strptime('2011-01-01', '%Y-%m-%d'))]
#past_df_2010 = train_df1[train_df1['Date'] < datetime.datetime.
 ↪strptime('2011-01-01', '%Y-%m-%d')]
```

[22]:
```
def update_data(row,pred,truth_lis,pred_lis,w_lis):
    if row['IsHoliday'] == False:
        w = 1
    else:
        w=5
    pred =pred
    pred_lis.append(pred)
    actual = row['Weekly_Sales']
    truth_lis.append(actual)
    w_lis.append(w)
```

10 Baseline Model: Exponential Moving Average

[21]:
```
def ema(history,alpha):
    #convert history to ratios
    predict  = history[0]
    for sales in history:
        sale_next_pred =  alpha *  predict + (1-alpha) * sales
        predict = sale_next_pred
    return predict
```

10.1 Train SET

[175]:
```
from tqdm import tqdm_notebook as tqdm
best_alpha = -1
min_error = 1e10
alphas = [float(a/10) for a in range(1,4,1)]
for alpha in tqdm(alphas):
    w_lis = []
    pred_lis = []
    truth_lis = []
    print("using alpha:",alpha)
    for index,row in tqdm(train_df_2011.iterrows()):
        key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(row['Month'])+key_sep+str(row['month
        if key not in store_dept_month_week_group_all:
            if train_df[(train_df['Store']==row['Store']) &␣
↪(train_df['Dept']==row['Dept']) & (train_df['year']<row['year'])].shape[0]␣
↪== 0:
                    #print('No info of this combination in history',('store:
↪',row['Store'],'dept:',row['Dept']))
                    #set random value as 0
                    update_data(row,0,truth_lis,pred_lis,w_lis)
                    continue
            #search_nearby week_numbers
            initial = row['month_week_number']
            m_initial = row['Month']
            while key not in store_dept_month_week_group_all:
                initial=(initial+1)%6
                if row['month_week_number'] == initial:
                    # All week_numbers searched, change month and search
                    m_initial = (m_initial+1)%13
                    initial = row['month_week_number']
                    #cnt=1
                key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(m_initial)+key_sep+str(initial)
        group = store_dept_month_week_group_all[key]
```

```
        history = group[group['year']<row['year']]['Weekly_Sales'].values
        if len(history) !=0:
                pred = ema(history,alpha)
        else:
                pred = 0
        update_data(row,pred,truth_lis,pred_lis,w_lis)
    print("CV Error:",wmae(np.array(truth_lis),np.array(pred_lis),np.
 ↪array(w_lis)))
    if wmae(np.array(truth_lis),np.array(pred_lis),np.array(w_lis)) <=␣
 ↪min_error:
        best_alpha = alpha
        min_error = wmae(np.array(truth_lis),np.array(pred_lis),np.array(w_lis))
print(f"Summary:Best_alpha:{best_alpha} and minimum error is {min_error}")
```

```
HBox(children=(IntProgress(value=0, max=3), HTML(value='')))


using alpha: 0.1

HBox(children=(IntProgress(value=1, bar_style='info', max=1), HTML(value='')))


CV Error: 3122.319351318218
using alpha: 0.2

HBox(children=(IntProgress(value=1, bar_style='info', max=1), HTML(value='')))


CV Error: 3122.319351318218
using alpha: 0.3

HBox(children=(IntProgress(value=1, bar_style='info', max=1), HTML(value='')))


CV Error: 3122.319351318218
Summary:Best_alpha:0.3 and minimum error is 3122.319351318218
```

[176]:
```
alphas = [best_alpha]
for alpha in tqdm(alphas):
    w_lis = []
    pred_lis = []
    truth_lis = []
    print("using alpha:",alpha)
    for index,row in train_df_2011.iterrows():
        key =␣
 ↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(row['Month'])+key_sep+str(row['month
        if key not in store_dept_month_week_group_all:
```

```
            if train_df[(train_df['Store']==row['Store']) &␣
↪(train_df['Dept']==row['Dept']) & (train_df['year']<row['year'])].shape[0]␣
↪== 0:
                    #print('No info of this combination in history',('store:
↪',row['Store'],'dept:',row['Dept']))
                    #set random value as 0
                    update_data(row,0,truth_lis,pred_lis,w_lis)
                    continue
            #search_nearby week_numbers
            initial = row['month_week_number']
            m_initial = row['Month']
            while key not in store_dept_month_week_group_all:
                initial=(initial+1)%6
                if row['month_week_number'] == initial:
                    # All week_numbers searched, change month and search
                    m_initial = (m_initial+1)%13
                    initial = row['month_week_number']
                    #cnt=1
                key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(m_initial)+key_sep+str(initial)
        group = store_dept_month_week_group_all[key]
        history = group[group['year']<row['year']]['Weekly_Sales'].values
        if len(history) !=0:
                pred = ema(history,alpha)
        else:
                pred = 0
        update_data(row,pred,truth_lis,pred_lis,w_lis)
    print("CV Error:",wmae(np.array(truth_lis),np.array(pred_lis),np.
↪array(w_lis)))
```

```
HBox(children=(IntProgress(value=0, max=1), HTML(value='')))
```

```
using alpha: 0.3
CV Error: 3122.319351318218
```

[177]:
```
#52 weeks before
prev_year = []
for index,row in tqdm(train_df_2011.iterrows()):
    key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(row['year_week_number'])
    initial_key = key
    if key not in store_dept_year_week_group_all:
        prev_year.append(0)
        continue
    group = store_dept_year_week_group_all[key]
    vals = group[group['year']==(row['year']-1)]['Weekly_Sales'].values
```

```
    if len(vals) > 0:
        prev_year.append(vals[0])
    else:
        prev_year.append(0)
```

```
HBox(children=(IntProgress(value=1, bar_style='info', max=1), HTML(value='')))
```

[178]:
```
train_df_2011['Baseline_Pred'] = pred_lis
train_df_2011['Prev_Year'] = prev_year
```

10.2 CV Set

[179]:
```
from tqdm import tqdm_notebook as tqdm
best_alpha = -1
min_error = 1e10
alphas = [float(a/10) for a in range(1,4,1)]
for alpha in tqdm(alphas):
    w_lis = []
    pred_lis = []
    truth_lis = []
    print("using alpha:",alpha)
    for index,row in cv_df.iterrows():
        key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(row['Month'])+key_sep+str(row['month
        if key not in store_dept_month_week_group_all:
            if train_df[(train_df['Store']==row['Store']) &␣
↪(train_df['Dept']==row['Dept']) & (train_df['year']<row['year'])].shape[0]␣
↪== 0:
                    #print('No info of this combination in history',('store:
↪',row['Store'],'dept:',row['Dept']))
                    #set random value as 0
                    update_data(row,0,truth_lis,pred_lis,w_lis)
                    continue
            #search nearby week_numbers
            initial = row['month_week_number']
            m_initial = row['Month']
            while key not in store_dept_month_week_group_all:
                initial=(initial+1)%6
                if row['month_week_number'] == initial:
                    # All week_numbers searched, change month and search
                    m_initial = (m_initial+1)%13
                    initial = row['month_week_number']
                    #cnt=1
                key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(m_initial)+key_sep+str(initial)
        group = store_dept_month_week_group_all[key]
```

```
        history = group[group['year']<row['year']]['Weekly_Sales'].values
        if len(history) !=0:
                pred = ema(history,alpha)
        else:
                pred = 0
        update_data(row,pred,truth_lis,pred_lis,w_lis)
    print("CV Error:",wmae(np.array(truth_lis),np.array(pred_lis),np.
↪array(w_lis)))
    if wmae(np.array(truth_lis),np.array(pred_lis),np.array(w_lis)) <=␣
↪min_error:
        best_alpha = alpha
        min_error = wmae(np.array(truth_lis),np.array(pred_lis),np.array(w_lis))
print(f"Summary:Best_alpha:{best_alpha} and minimum error is {min_error}")
```

```
HBox(children=(IntProgress(value=0, max=3), HTML(value='')))


using alpha: 0.1
CV Error: 2601.690260252039
using alpha: 0.2
CV Error: 2595.869080158141
using alpha: 0.3
CV Error: 2603.201617840376
Summary:Best_alpha:0.2 and minimum error is 2595.869080158141
```

[180]:
```
alphas = [best_alpha]
for alpha in tqdm(alphas):
    w_lis = []
    pred_lis = []
    truth_lis = []
    print("using alpha:",alpha)
    for index,row in cv_df.iterrows():
        key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(row['Month'])+key_sep+str(row['month
        if key not in store_dept_month_week_group_all:
            if train_df[(train_df['Store']==row['Store']) &␣
↪(train_df['Dept']==row['Dept']) & (train_df['year']<row['year'])].shape[0]␣
↪== 0:
                    #print('No info of this combination in history',('store:
↪',row['Store'],'dept:',row['Dept']))
                    #set random value as 0
                    update_data(row,0,truth_lis,pred_lis,w_lis)
                    continue
            #search_nearby week_numbers
            initial = row['month_week_number']
            m_initial = row['Month']
            while key not in store_dept_month_week_group_all:
```

```
                initial=(initial+1)%6
                if row['month_week_number'] == initial:
                    # All week_numbers searched, change month and search
                    m_initial = (m_initial+1)%13
                    initial = row['month_week_number']
                    #cnt=1
                key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(m_initial)+key_sep+str(initial)
        group = store_dept_month_week_group_all[key]
        history = group[group['year']<row['year']]['Weekly_Sales'].values
        vals = group[group['year']==(row['year']-1)]['Weekly_Sales'].values
        if len(history) !=0:
                pred = ema(history,alpha)
        else:
                pred = 0
        update_data(row,pred,truth_lis,pred_lis,w_lis)
    print("CV Error:",wmae(np.array(truth_lis),np.array(pred_lis),np.
↪array(w_lis)))
```

```
HBox(children=(IntProgress(value=0, max=1), HTML(value='')))
```

```
using alpha: 0.2
CV Error: 2595.869080158141
```

[181]:
```
prev_year = []
for index,row in tqdm(cv_df.iterrows()):
    key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(row['year_week_number'])
    initial_key = key
    if key not in store_dept_year_week_group_all:
        prev_year.append(0)
        continue
    group = store_dept_year_week_group_all[key]
    vals = group[group['year']==(row['year']-1)]['Weekly_Sales'].values
    if len(vals) > 0:
        prev_year.append(vals[0])
    else:
        prev_year.append(0)
```

```
HBox(children=(IntProgress(value=1, bar_style='info', max=1), HTML(value='')))
```

[182]:
```
cv_df['Baseline_Pred'] = pred_lis
cv_df['Prev_Year'] = prev_year
```

10.3 Test Set

[183]:
```
alphas = [best_alpha]
for alpha in tqdm(alphas):
    pred_lis = []
    print("using alpha:",alpha)
    for index,row in tqdm(test_df.iterrows()):
        key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(row['Month'])+key_sep+str(row['month
        if key not in store_dept_month_week_group_all:
            if train_df1[(train_df1['Store']==row['Store']) &␣
↪(train_df1['Dept']==row['Dept']) & (train_df1['year']<row['year'])].shape[0]␣
↪== 0:
                    #print('No info of this combination in history',('store:
↪',row['Store'],'dept:',row['Dept']))
                    #set random value as 0
                    pred_lis.append(0)
                    continue
            #search_nearby week_numbers
            initial = row['month_week_number']
            m_initial = row['Month']
            while key not in store_dept_month_week_group_all:
                initial=(initial+1)%6
                if row['month_week_number'] == initial:
                    # All week_numbers searched, change month and search
                    m_initial = (m_initial+1)%13
                    initial = row['month_week_number']
                    #cnt=1
                key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(m_initial)+key_sep+str(initial)
        group = store_dept_month_week_group_all[key]
        history = group[group['year']<row['year']]['Weekly_Sales'].values
        if len(history) !=0:
                pred = ema(history,alpha)
        else:
                pred = 0
        pred_lis.append(pred)
```

```
HBox(children=(IntProgress(value=0, max=1), HTML(value='')))


using alpha: 0.2

HBox(children=(IntProgress(value=1, bar_style='info', max=1), HTML(value='')))
```

[184]:
```
prev_year = []
for index,row in tqdm(test_df.iterrows()):
```

```
    key =␣
↪str(row['Store'])+key_sep+str(row['Dept'])+key_sep+str(row['year_week_number'])
    initial_key = key
    if key not in store_dept_year_week_group_all:
        prev_year.append(0)
        continue
    group = store_dept_year_week_group_all[key]
    vals = group[group['year']==(row['year']-1)]['Weekly_Sales'].values
    if len(vals) > 0:
        prev_year.append(vals[0])
    else:
        prev_year.append(0)
```

```
HBox(children=(IntProgress(value=1, bar_style='info', max=1), HTML(value='')))
```

[185]:
```
test_df['Baseline_Pred'] = pred_lis
test_df['Prev_Year'] = prev_year
```

[186]:
```
test_df['IsHoliday'] = test_df['IsHoliday'].map(lambda x: 1 if x==True else 0)
```

[187]:
```
train_df_2011['IsHoliday'] = train_df_2011['IsHoliday'].map(lambda x: 1 if␣
↪x==True else 0)
cv_df['IsHoliday'] = cv_df['IsHoliday'].map(lambda x: 1 if x==True else 0)
```

[189]:
```
def get_weights(holiday_type_lis):
    w_lis = []
    for h in holiday_type_lis:
        if h == 1:
            w_lis.append(5)
        else:
            w_lis.append(1)
    return w_lis
```

[247]:
```
import matplotlib.pyplot as plt
import xgboost as xgb
import warnings
warnings.filterwarnings("ignore")
#from sklearn.calibration import CalibratedClassifierCV
import math
max_depth = [2,5,10,15,20]
n_estimators = [40,80,100,200,300,400]
# max_depth = [2,5,10,15,20]
# n_estimators = [2,20,40,80,100]
max_depth_plot = []
n_estimators_plot = []
train_wmae = []
cv_wmae = []
min_wmae = 1e7
```

```
best_d = -1
best_s = -1
w_lis_train = get_weights(train_df_2011['IsHoliday'].values)
w_lis_cv = get_weights(cv_df['IsHoliday'].values)
X_train =␣
 ↪train_df_2011[['Store','Dept','Prev_Year','IsHoliday','year_week_number','Month','month_wee
X_cv =␣
 ↪cv_df[['Store','Dept','Prev_Year','IsHoliday','year_week_number','Month','month_week_number
Y_train = train_df_2011[['Weekly_Sales']]
Y_cv = cv_df[['Weekly_Sales']]
actual_sales_train = train_df_2011['Weekly_Sales'].values
actual_sales_cv = cv_df['Weekly_Sales'].values
for d in max_depth:
    for s in n_estimators:
        clf = xgb.XGBRegressor(max_depth=d,n_estimators=s,objective='reg:
 ↪squarederror')
        clf.fit(X_train,Y_train)
        y_train_pred =clf.predict(X_train)
        y_cv_pred = clf.predict(X_cv)
        max_depth_plot.append(d)
        n_estimators_plot.append(s)
        if wmae(np.array(actual_sales_cv),y_cv_pred,np.array(w_lis_cv)) <␣
 ↪min_wmae:
            min_wmae = wmae(np.array(actual_sales_cv),y_cv_pred,np.
 ↪array(w_lis_cv))
            best_d = d
            best_s = s
        train_wmae.append(wmae(np.array(actual_sales_train),y_train_pred,np.
 ↪array(w_lis_train)))
        cv_wmae.append(wmae(np.array(actual_sales_cv),y_cv_pred,np.
 ↪array(w_lis_cv)))
        print("Done for d={},s={}, wmae={}".format(d,s,cv_wmae[-1]))
        print("="*50)


import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
train_e=[]
train_d=[]
train_a=[]
cv_e=[]
cv_d=[]
cv_a=[]
for index in np.argsort(train_wmae):
```

```
    train_e.append(n_estimators_plot[index])
    train_d.append(max_depth_plot[index])
    train_a.append(train_wmae[index])
for index in np.argsort(cv_wmae):
    cv_e.append(n_estimators_plot[index])
    cv_d.append(max_depth_plot[index])
    cv_a.append(cv_wmae[index])
trace1 = go.Scatter3d(x=train_e,y=train_d,z=train_a, name = 'train')
trace2 = go.Scatter3d(x=cv_e,y=cv_d,z=cv_a, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='WMAE'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

```
Done for d=2,s=40, wmae=3028.2643190433314
==================================================
Done for d=2,s=80, wmae=2739.606506394686
==================================================
Done for d=2,s=100, wmae=2719.3276247844497
==================================================
Done for d=2,s=200, wmae=2706.6305443709175
==================================================
Done for d=2,s=300, wmae=2703.4544669886614
==================================================
Done for d=2,s=400, wmae=2718.628418007371
==================================================
Done for d=5,s=40, wmae=2354.29580664956
==================================================
Done for d=5,s=80, wmae=2239.8128558049825
==================================================
Done for d=5,s=100, wmae=2226.01677634129
==================================================
Done for d=5,s=200, wmae=2213.4808066501505
==================================================
Done for d=5,s=300, wmae=2224.791993540526
==================================================
Done for d=5,s=400, wmae=2226.4022965102613
==================================================
Done for d=10,s=40, wmae=2171.06596356144
==================================================
Done for d=10,s=80, wmae=2126.1333106665775
==================================================
```

```
Done for d=10,s=100, wmae=2131.8522140819323
==================================================
Done for d=10,s=200, wmae=2150.1294086102007
==================================================
Done for d=10,s=300, wmae=2155.8097195682867
==================================================
Done for d=10,s=400, wmae=2159.4371434449536
==================================================
Done for d=15,s=40, wmae=2145.690895508366
==================================================
Done for d=15,s=80, wmae=2098.345796262128
==================================================
Done for d=15,s=100, wmae=2105.1132683720416
==================================================
Done for d=15,s=200, wmae=2121.1830537765095
==================================================
Done for d=15,s=300, wmae=2126.8220074252454
==================================================
Done for d=15,s=400, wmae=2129.6388367539016
==================================================
Done for d=20,s=40, wmae=2182.5397415963716
==================================================
Done for d=20,s=80, wmae=2132.0662629522562
==================================================
Done for d=20,s=100, wmae=2133.804428234077
==================================================
Done for d=20,s=200, wmae=2141.112969340172
==================================================
Done for d=20,s=300, wmae=2142.0886832544766
==================================================
Done for d=20,s=400, wmae=2142.536013078998
==================================================
```

[248]:
```
print(f'min wmae:{min_wmae}, best_d:{best_d},best_s:{best_s}')
```

```
min wmae:2098.345796262128, best_d:15,best_s:80
```

[249]:
```
X_train =␣
 ↪cv_df[['Store','Dept','Prev_Year','IsHoliday','year_week_number','Month','month_week_number
X_test =␣
 ↪test_df[['Store','Dept','Prev_Year','IsHoliday','year_week_number','Month','month_week_numb
Y_train = cv_df[['Weekly_Sales']]
clf = xgb.XGBRegressor(max_depth=best_d,n_estimators=best_s,objective='reg:
 ↪squarederror')
clf.fit(X_train,Y_train)
y_test_pred = clf.predict(X_test)
test_df['XGBOOST_PRED'] = y_test_pred
```

[250]:
```
test_df.head(6)
```

[250]:
```
   Store  Dept       Date  IsHoliday Type    Size  Temperature  Fuel_Price  \
0      1     1 2012-11-02          0    A  151315        55.32       3.386
1      1     1 2012-11-09          0    A  151315        61.24       3.314
2      1     1 2012-11-16          0    A  151315        52.92       3.252
3      1     1 2012-11-23          1    A  151315        56.23       3.211
4      1     1 2012-11-30          0    A  151315        52.34       3.207
5      1     1 2012-12-07          0    A  151315        64.12       3.198

   MarkDown1  MarkDown2  ...  MarkDown5         CPI  Unemployment  Month  \
0    6766.44    5147.70  ...    2737.42  223.462779         6.573     11
1   11421.32    3370.89  ...    6154.16  223.481307         6.573     11
2    9696.28     292.10  ...    6612.69  223.512911         6.573     11
3     883.59       4.17  ...     303.32  223.561947         6.573     11
4    2460.03       0.00  ...    6966.34  223.610984         6.573     11
5    6343.16       0.00  ...   10147.90  223.660021         6.573     12

   year_week_number  month_week_number  year  Baseline_Pred  Prev_Year  \
0                44                  1  2012      38756.624   39886.06
1                45                  2  2012      18861.510   18689.54
2                46                  3  2012      19151.096   19050.66
3                47                  4  2012      20493.058   20911.25
4                48                  5  2012      38756.624   25293.49
5                49                  1  2012      24738.304   33305.92

   XGBOOST_PRED
0  48447.210938
1  21106.427734
2  20437.935547
3  21421.406250
4  18348.978516
5  31121.078125

[6 rows x 22 columns]
```

[254]:
```
ids  = []
value_prev = []
val_xgboost = []
val_baseline = []
formater = '%Y-%m-%d'
for index,row in test_df.iterrows():
    key = str(row['Store'])+'_'+str(row['Dept'])+'_'+str(row['Date'].
 ↪strftime(formater))
    ids.append(key)
    value_prev.append(float(row['Prev_Year']))
    val_xgboost.append(float(row['XGBOOST_PRED']))
    val_baseline.append(float(row['Baseline_Pred']))
```

```
final_pred = []
t = tuple(zip(value_prev,val_baseline,val_xgboost))
for e in t:
    weight = 0
    final_pred.append((e[0]+e[1]+e[2])/3)
df_print = pd.DataFrame({'Id':ids,'Weekly_Sales':final_pred})
```

[255]:
```
df_print.head()
```

[255]:

	Id	Weekly_Sales
0	1_1_2012-11-02	42363.298312
1	1_1_2012-11-09	19552.492578
2	1_1_2012-11-16	19546.563849
3	1_1_2012-11-23	20941.904750
4	1_1_2012-11-30	27466.364172

[256]:
```
df_print.to_csv('final_submission(xgboost+prev+baseline).csv', index=False)
```

11 KAGGLE Private Score:2629.16

12 KAGGLE Public Score:2586.44

13 Model would be ranked at 20 on private leaderboard.

14 Private LeaderBoard:https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/leaderboard

15 Summary

1. There was high co-relation with previous year sales for same month and week for a given store and dept.
2. Hence previous year sales were very important
3. Overall model did well when
    a. Week number for given month was constructed from given dataset. Since week number started from FEB-05-2010,week number for given month was built from FEB-05-2010 until end of test_data date with 7 days a week. So FEB-05-2010 was 1st week as per this construction. This really helped for EMA.
    b. Week number for year was also taken into account and this helped to get previous year(52 weeks prior) sales.
    c. XGBOOST was used to build upon both EMA and Prev_Year sales along with other features.
4. Final prediction was taken by simple mean of EMA,PREV_YEAR and XGBOOST predictions