Sales Forecasting

Load primary data

In [1]:

```
import pandas as pd
df_primary_data=pd.read_csv("train.csv")
df_primary_data
```

Out[1]:

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

421570 rows × 5 columns

In [2]:

```
df_primary_data.isna().sum()
```

Out[2]:

```
Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
dtype: int64
```

Load features

In [3]:

```
df_features=pd.read_csv("features.csv")
df_features
```

Out[3]:

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

8190 rows × 12 columns

In [4]:

```
df_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8190 entries, 0 to 8189
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         8190 non-null   int64  
 1   Date          8190 non-null   object 
 2   Temperature   8190 non-null   float64
 3   Fuel_Price    8190 non-null   float64
 4   MarkDown1     4032 non-null   float64
 5   MarkDown2     2921 non-null   float64
 6   MarkDown3     3613 non-null   float64
 7   MarkDown4     3464 non-null   float64
 8   MarkDown5     4050 non-null   float64
 9   CPI           7605 non-null   float64
 10  Unemployment  7605 non-null   float64
 11  IsHoliday     8190 non-null   bool   
dtypes: bool(1), float64(9), int64(1), object(1)
memory usage: 712.0+ KB
```

In [5]:

```
df_features.isna().sum()
```

Out[5]:

```
Store             0
Date              0
Temperature       0
Fuel_Price        0
MarkDown1      4158
MarkDown2      5269
MarkDown3      4577
MarkDown4      4726
MarkDown5      4140
CPI             585
Unemployment    585
IsHoliday         0
dtype: int64
```

In [6]:

```
df_features.describe()
```

Out[6]:

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	
count	8190.000000	8190.000000	8190.000000	4032.000000	2921.000000	3613.000000	3
mean	23.000000	59.356198	3.405992	7032.371786	3384.176594	1760.100180	3
std	12.987966	18.678607	0.431337	9262.747448	8793.583016	11276.462208	(
min	1.000000	-7.290000	2.472000	-2781.450000	-265.760000	-179.260000	
25%	12.000000	45.902500	3.041000	1577.532500	68.880000	6.600000	
50%	23.000000	60.710000	3.513000	4743.580000	364.570000	36.260000	
75%	34.000000	73.880000	3.743000	8923.310000	2153.350000	163.150000	3
max	45.000000	101.950000	4.468000	103184.980000	104519.540000	149483.310000	6

Fixed Missing Values

In [7]:

```
df_features[['MarkDown1',
             'MarkDown2',
             'MarkDown3',
             'MarkDown4',
             'MarkDown5']] = df_features[['MarkDown1',
                                          'MarkDown2',
                                          'MarkDown3',
                                          'MarkDown4',
                                          'MarkDown5']].fillna(-9999, axis = 1)
df_features
```

Out[7]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
0	1	2010-02-05	42.31	2.572	-9999.00	-9999.00	-9999.00	-9999.00
1	1	2010-02-12	38.51	2.548	-9999.00	-9999.00	-9999.00	-9999.00
2	1	2010-02-19	39.93	2.514	-9999.00	-9999.00	-9999.00	-9999.00
3	1	2010-02-26	46.63	2.561	-9999.00	-9999.00	-9999.00	-9999.00
4	1	2010-03-05	46.50	2.625	-9999.00	-9999.00	-9999.00	-9999.00
...	...	...	...	...	...	...	...	...
8185	45	2013-06-28	76.05	3.639	4842.29	975.03	3.00	2449.97
8186	45	2013-07-05	77.50	3.614	9090.48	2268.58	582.74	5797.47
8187	45	2013-07-12	79.37	3.614	3789.94	1827.31	85.72	744.84
8188	45	2013-07-19	82.84	3.737	2961.49	1047.07	204.19	363.00
8189	45	2013-07-26	76.06	3.804	212.02	851.73	2.06	10.88

8190 rows × 12 columns

In [8]:

```
df_features.isna().sum()
```

Out[8]:

```
Store             0
Date              0
Temperature       0
Fuel_Price        0
MarkDown1         0
MarkDown2         0
MarkDown3         0
MarkDown4         0
MarkDown5         0
CPI             585
Unemployment    585
IsHoliday         0
dtype: int64
```

In [9]:

```
df_features['CPI'].fillna(df_features['CPI'].mean(), inplace = True)
df_features['Unemployment'].fillna(df_features['Unemployment'].mean(), inplace = True)
df_features.isna().sum()
```

Out[9]:

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

Load stores data

In [10]:

```
df_stores=pd.read_csv("stores.csv")
df_stores
```

Out[10]:

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

	Store	Type	Size
34	35	B	103681
35	36	A	39910
36	37	C	39910
37	38	C	39690
38	39	A	184109
39	40	A	155083
40	41	A	196321
41	42	C	39690
42	43	C	41062
43	44	C	39910
44	45	B	118221

In [11]:

```
df_stores.isna().sum()
```

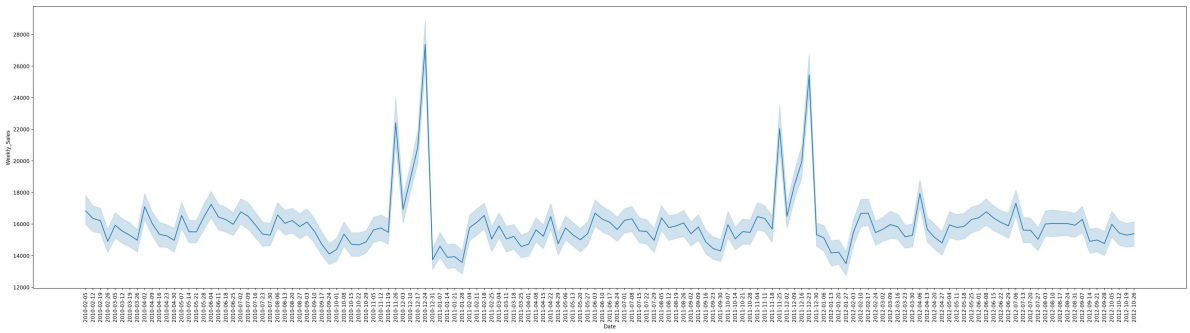
Out[11]:

```
Store    0
Type     0
Size     0
dtype: int64
```

Data Visualization for the preprocessed data

In [12]:

```
from matplotlib import pyplot as plt
import seaborn as sns
plt.figure(figsize = (40, 10), dpi = 200)
sns.lineplot(x = df_primary_data.Date, y = df_primary_data.Weekly_Sales)
plt.xticks(rotation = 90)
plt.show()
```

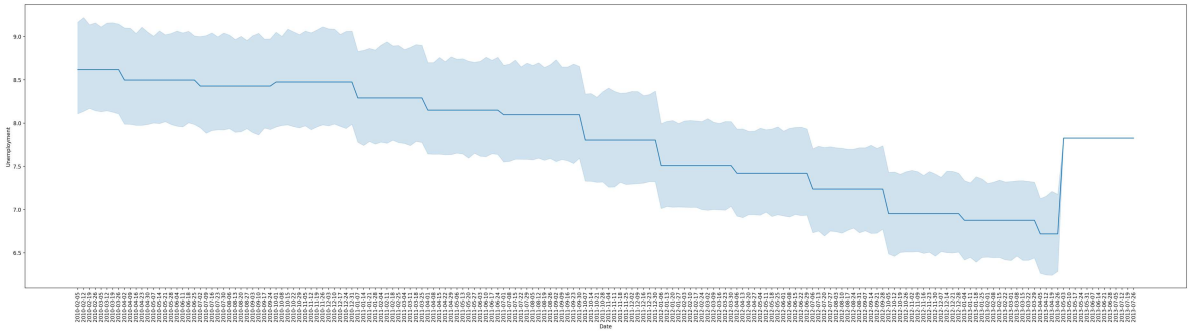


In [*]:

```
plt.figure(figsize = (40, 10), dpi = 200)
sns.lineplot(x = df_primary_data.Date, y = df_primary_data.Weekly_Sales, hue = df_primary_d
plt.xticks(rotation = 90)
plt.show()
```

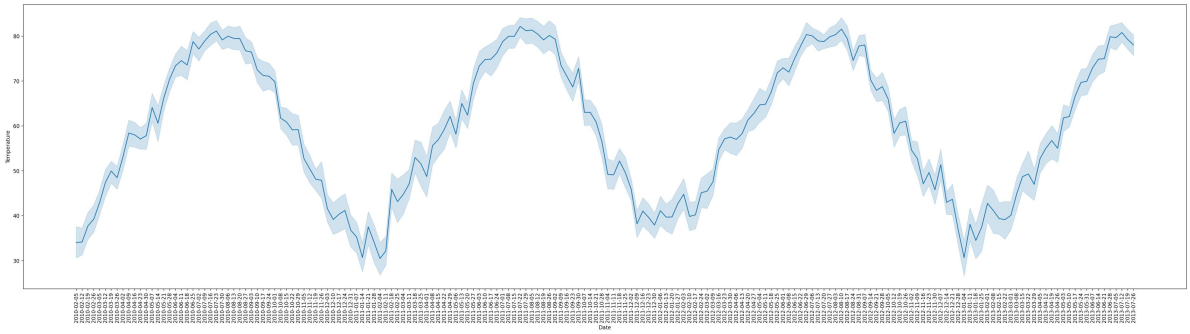
In [14]:

```
plt.figure(figsize = (40, 10), dpi = 200)
sns.lineplot(x = df_features.Date, y = df_features.Unemployment)
plt.xticks(rotation = 90)
plt.show()
```



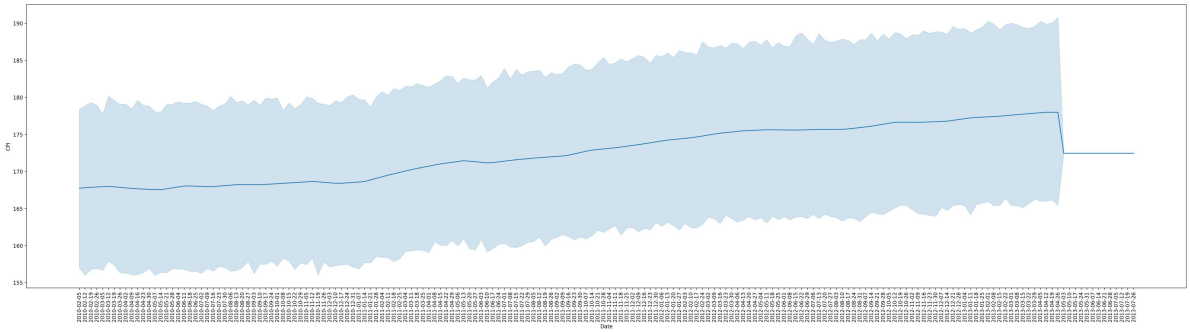
In [15]:

```
plt.figure(figsize = (40, 10), dpi = 200)
sns.lineplot(x = df_features.Date, y = df_features.Temperature)
plt.xticks(rotation = 90)
plt.show()
```



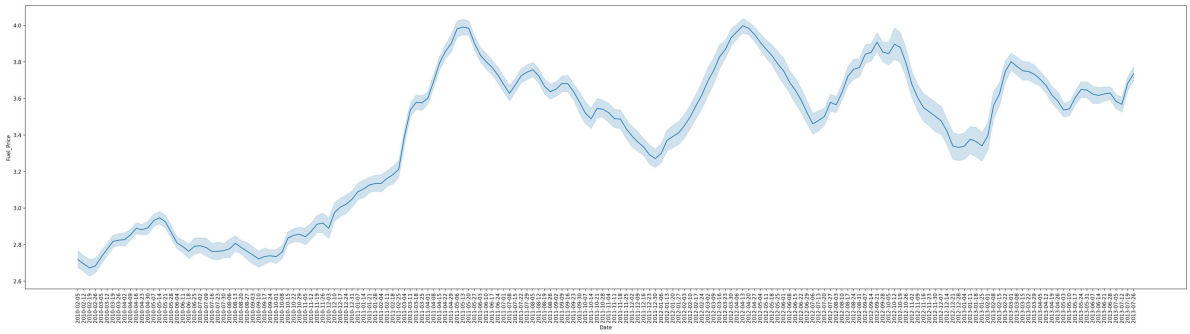
In [16]:

```
plt.figure(figsize = (40, 10), dpi = 200)
sns.lineplot(x = df_features.Date, y = df_features.CPI)
plt.xticks(rotation = 90)
plt.show()
```



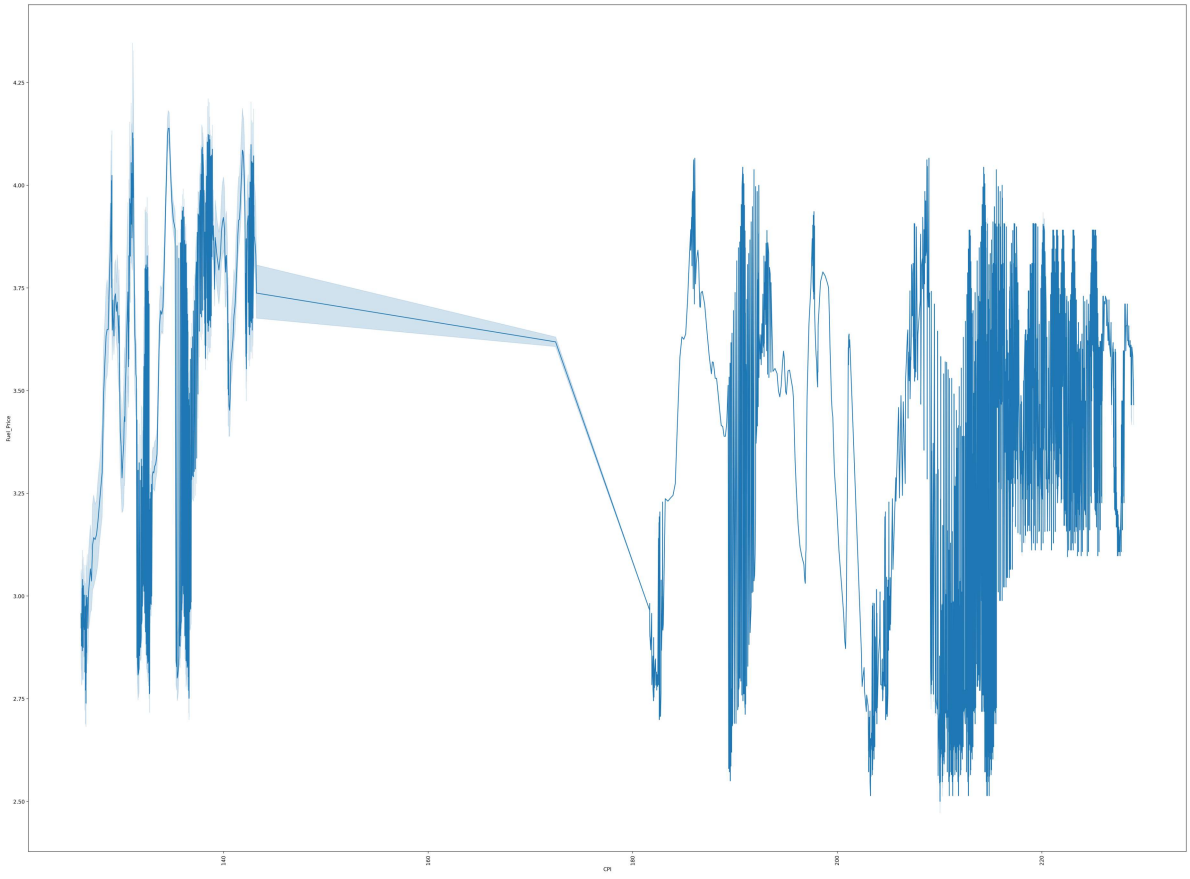
In [17]:

```
plt.figure(figsize = (40, 10), dpi = 200)
sns.lineplot(x = df_features.Date, y = df_features.Fuel_Price)
plt.xticks(rotation = 90)
plt.show()
```



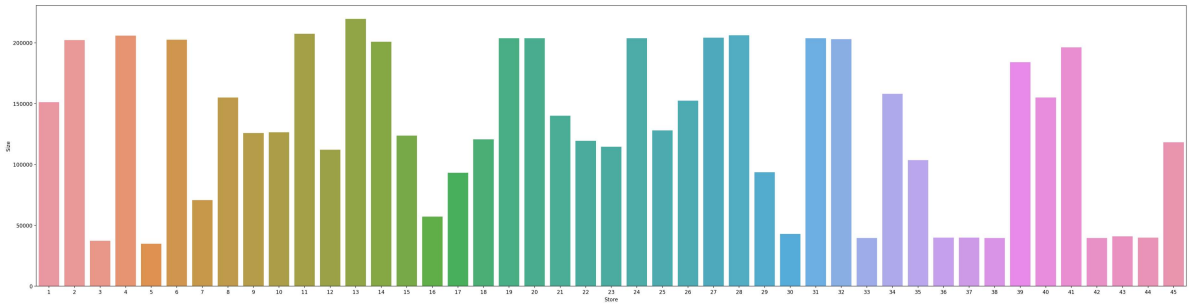
In [18]:

```
plt.figure(figsize = (40, 30), dpi = 200)
sns.lineplot(x = df_features.CPI, y = df_features.Fuel_Price)
plt.xticks(rotation = 90)
plt.show()
```



In [19]:

```
plt.figure(figsize = (40, 10), dpi = 200)
sns.barplot(x = df_stores.Store, y = df_stores.Size)
plt.show()
```



Merge the training data and features

In [20]:

```
df_stores_data = pd.merge(pd.merge(df_primary_data,
                                   df_features,
                                   on = ['Date', 'Store', 'IsHoliday'],
                                   how = 'inner'),
                                   df_stores,
                                   on = 'Store',
                                   how = 'inner')
df_stores_data
```

Out[20]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Ma
0	1	1	2010-02-05	24924.50	False	42.31	2.572	-9999.00	
1	1	2	2010-02-05	50605.27	False	42.31	2.572	-9999.00	
2	1	3	2010-02-05	13740.12	False	42.31	2.572	-9999.00	
3	1	4	2010-02-05	39954.04	False	42.31	2.572	-9999.00	
4	1	5	2010-02-05	32229.38	False	42.31	2.572	-9999.00	
...	...	...	...	...	...	...	...	...	
421565	45	93	2012-10-26	2487.80	False	58.85	3.882	4018.91	
421566	45	94	2012-10-26	5203.31	False	58.85	3.882	4018.91	
421567	45	95	2012-10-26	56017.47	False	58.85	3.882	4018.91	
421568	45	97	2012-10-26	6817.48	False	58.85	3.882	4018.91	
421569	45	98	2012-10-26	1076.80	False	58.85	3.882	4018.91	

421570 rows × 16 columns

In [21]:

```
df_stores_data_final = df_stores_data.groupby(['Store', 'Date']).mean()

df_stores_data_final
```

Out[21]:

Store	Date	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDc
1	2010-02-05	43.616438	22516.313699	0.0	42.31	2.572	-9999.00	-999
	2010-02-12	43.569444	22804.964444	1.0	38.51	2.548	-9999.00	-999
	2010-02-19	43.616438	22081.755753	0.0	39.93	2.514	-9999.00	-999
	2010-02-26	43.569444	19579.549861	0.0	46.63	2.561	-9999.00	-999
	2010-03-05	44.041096	21298.721644	0.0	46.50	2.625	-9999.00	-999
...	...	...	...	...	...	...	...	
45	2012-09-28	43.227273	10805.665909	0.0	64.88	3.997	4556.61	2
	2012-10-05	43.411765	10786.103971	0.0	64.89	3.985	5046.74	-999
	2012-10-12	43.227273	11128.247879	0.0	54.47	4.000	1956.28	-999
	2012-10-19	43.227273	10880.689848	0.0	56.47	3.969	2004.02	-999
	2012-10-26	43.388060	11347.484030	0.0	58.85	3.882	4018.91	5

6435 rows × 13 columns

In [22]:

```
2.75*52*45
```

Out[22]:

```
6435.0
```

In [23]:

```
df_agg_stores = df_stores_data.groupby('Date').mean()[['Weekly_Sales', 'IsHoliday', 'Temper
                                                       'Fuel_Price', 'MarkDown1', 'MarkDown2
                                                       'MarkDown4', 'MarkDown5', 'CPI', 'Une

df_agg_stores
```

Out[23]:

Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkD
2010-02-05	16836.121997	0.0	33.277942	2.717869	-9999.000000	-9999.000000	-9999.0
2010-02-12	16352.056032	1.0	33.361810	2.696102	-9999.000000	-9999.000000	-9999.0
2010-02-19	16216.658979	0.0	37.038310	2.673666	-9999.000000	-9999.000000	-9999.0
2010-02-26	14899.549688	0.0	38.629563	2.685642	-9999.000000	-9999.000000	-9999.0
2010-03-05	15921.015727	0.0	42.373998	2.731816	-9999.000000	-9999.000000	-9999.0
...	...	...	...	...	...	...	
2012-09-28	14765.327279	0.0	68.151759	3.858245	4890.992718	-2235.927103	-2280.1
2012-10-05	15983.413747	0.0	65.456811	3.848435	4877.693905	-9999.000000	-544.5
2012-10-12	15427.596739	0.0	57.687284	3.897425	1955.152860	-9999.000000	-294.2
2012-10-19	15295.732397	0.0	60.152756	3.878413	1916.437102	-9999.000000	-498.6
2012-10-26	15391.725681	0.0	60.530277	3.791086	4761.817036	68.773298	-2255.7

143 rows × 12 columns

In [24]:

```
2.75*52
```

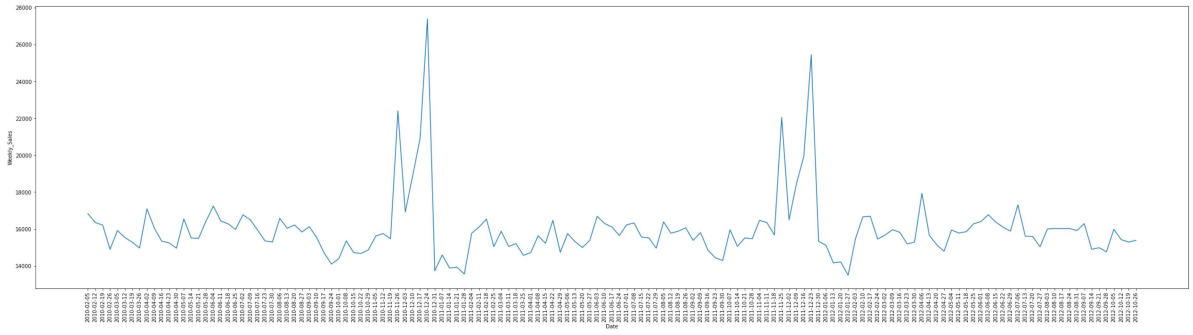
Out[24]:

```
143.0
```

Visualize the data for the final version

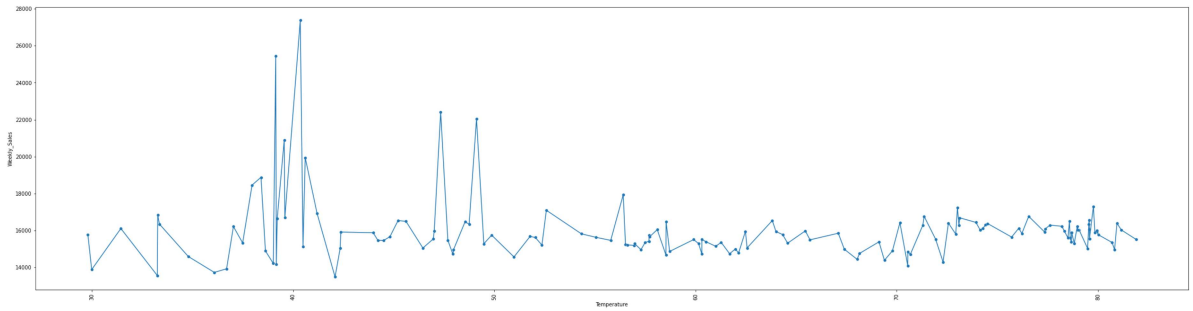
In [25]:

```
# Plot for the aggregate data
plt.figure(figsize = (40, 10))
sns.lineplot(x = df_agg_stores.index, y = df_agg_stores['Weekly_Sales'])
plt.xticks(rotation = 90)
plt.show()
```



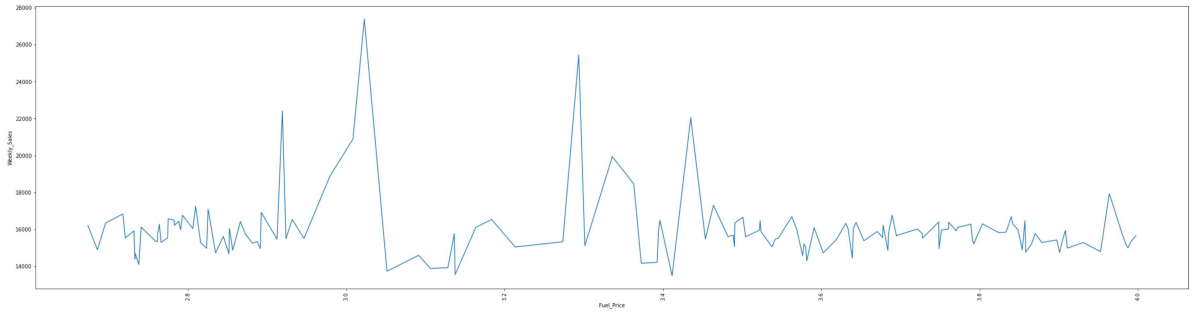
In [26]:

```
# Relationship of temperature with sales
plt.figure(figsize = (40, 10))
sns.lineplot(x = df_agg_stores.Temperature, y = df_agg_stores['Weekly_Sales'])
sns.scatterplot(x = df_agg_stores.Temperature, y = df_agg_stores['Weekly_Sales'])
plt.xticks(rotation = 90)
plt.show()
```



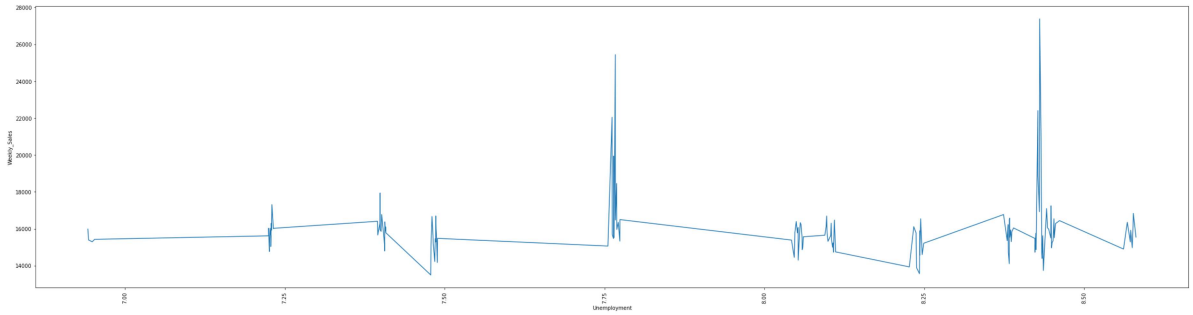
In [27]:

```
# Relationship of fuel price with sales
plt.figure(figsize = (40, 10))
sns.lineplot(x = df_agg_stores.Fuel_Price, y = df_agg_stores['Weekly_Sales'])
plt.xticks(rotation = 90)
plt.show()
```



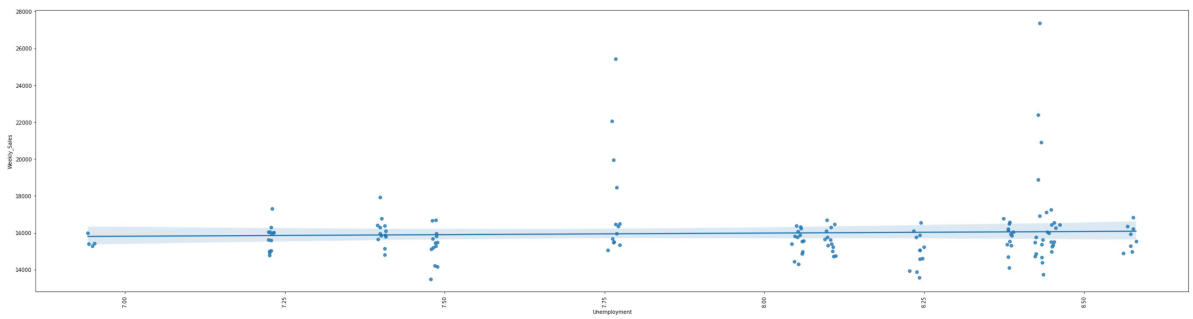
In [28]:

```
# Relationship of unemployment with sales
plt.figure(figsize = (40, 10))
sns.lineplot(x = df_agg_stores.Unemployment, y = df_agg_stores['Weekly_Sales'])
plt.xticks(rotation = 90)
plt.show()
```



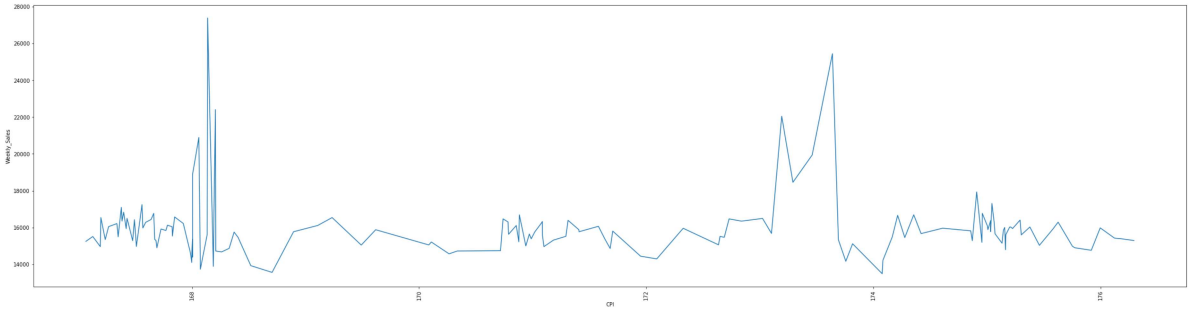
In [29]:

```
# Relationship of unemployment with sales
plt.figure(figsize = (40, 10))
sns.regplot(x = 'Unemployment', y = 'Weekly_Sales', data = df_agg_stores)
plt.xticks(rotation = 90)
plt.show()
```



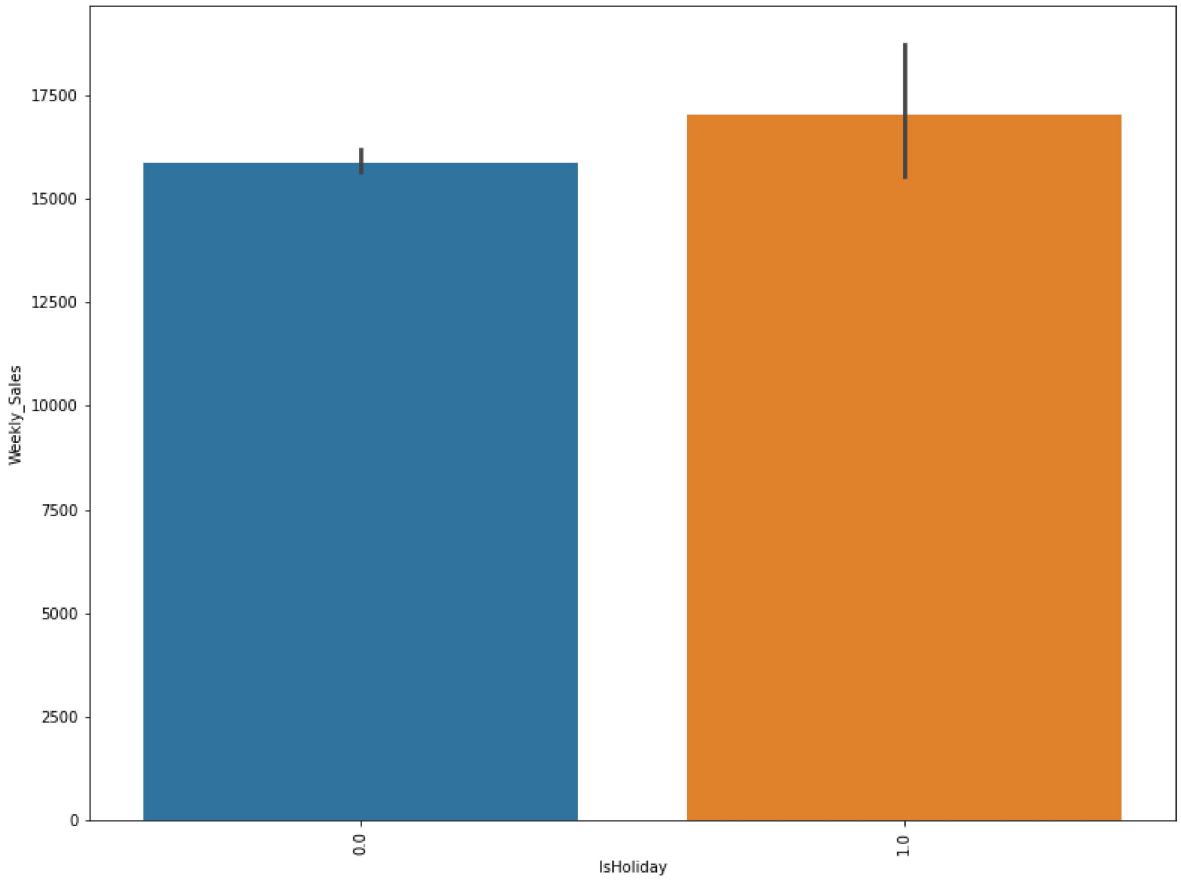
In [30]:

```
# Relationship of CPI with sales
plt.figure(figsize = (40, 10))
sns.lineplot(x = df_agg_stores.CPI, y = df_agg_stores['Weekly_Sales'])
plt.xticks(rotation = 90)
plt.show()
```



In [31]:

```
# Relationship of CPI with sales
plt.figure(figsize = (13, 10))
sns.barplot(x = df_agg_stores['IsHoliday'], y = df_agg_stores['Weekly_Sales'])
plt.xticks(rotation = 90)
plt.show()
```



created column to hold the sale difference in adjacent dates

In [32]:

```
# Add a shift of one for the Weekly_Sales column
df_agg_stores['diff_1'] = df_agg_stores['Weekly_Sales'].diff()
df_agg_stores
```

Out[32]:

Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDow
2010-02-05	16836.121997	0.0	33.277942	2.717869	-9999.000000	-9999.000000	-9999.000(
2010-02-12	16352.056032	1.0	33.361810	2.696102	-9999.000000	-9999.000000	-9999.000(
2010-02-19	16216.658979	0.0	37.038310	2.673666	-9999.000000	-9999.000000	-9999.000(
2010-02-26	14899.549688	0.0	38.629563	2.685642	-9999.000000	-9999.000000	-9999.000(
2010-03-05	15921.015727	0.0	42.373998	2.731816	-9999.000000	-9999.000000	-9999.000(
...	...	...	...	...	...	...	
2012-09-28	14765.327279	0.0	68.151759	3.858245	4890.992718	-2235.927103	-2280.125
2012-10-05	15983.413747	0.0	65.456811	3.848435	4877.693905	-9999.000000	-544.507
2012-10-12	15427.596739	0.0	57.687284	3.897425	1955.152860	-9999.000000	-294.237
2012-10-19	15295.732397	0.0	60.152756	3.878413	1916.437102	-9999.000000	-498.619
2012-10-26	15391.725681	0.0	60.530277	3.791086	4761.817036	68.773298	-2255.747

143 rows × 13 columns

In [33]:

```
df_agg_stores.dropna(inplace = True)
```

Train Test Split

In [34]:

```
X_train, y_train = df_agg_stores.values[:120, 1:], df_agg_stores.values[:120, 0]
```

In [35]:

```
X_test, y_test = df_agg_stores.values[120:, 1:], df_agg_stores.values[120:, 0]
```

In [36]:

```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[36]:

```
((120, 12), (120,), (22, 12), (22,))
```

In [37]:

```
y_train
```

Out[37]:

```
array([16352.0560318 , 16216.65897884, 14899.54968824, 15921.0157269 ,
       15546.85054502, 15286.77357798, 14975.89448592, 17098.62029841,
       16050.58977974, 15347.71300272, 15252.1147494 , 14967.50914726,
       16542.71607094, 15513.37446954, 15499.86535898, 16428.44945304,
       17246.92203436, 16440.88921279, 16281.04144615, 15978.41490915,
       16769.79242372, 16500.01011712, 15940.67548432, 15357.11567225,
       15300.09030511, 16576.68470426, 16044.34337362, 16222.32089624,
       15841.87040718, 16129.27464457, 15537.75888321, 14708.33953909,
       14105.90532401, 14391.78053492, 15362.04844346, 14726.78274744,
       14678.48313565, 14871.36116985, 15625.24979181, 15753.00574795,
       15475.16604252, 22403.33670524, 16924.0515022 , 18882.8936194 ,
       20892.46361947, 27378.69269283, 13738.53856609, 14599.2449727 ,
       13891.28348361, 13932.3673852 , 13566.23946214, 15773.44877649,
       16111.70619129, 16541.99121222, 15049.74755798, 15882.5570453 ,
       15056.45053981, 15216.11596134, 14573.82704963, 14726.86926127,
       15634.57131175, 15229.70814087, 16472.65382741, 14745.94606369,
       15757.2152959 , 15317.20418605, 15002.24727861, 15390.23344886,
       16691.30533196, 16302.91883037, 16111.22640747, 15649.41834243,
       16232.86233367, 16328.64680314, 15566.32350205, 15526.20418038,
       14965.15048989, 16393.12631273, 15774.06862551, 15888.02831697,
       16068.09503558, 15387.12216684, 15809.06948276, 14865.56689749,
       14446.43108894, 14298.8244019 , 15960.67895876, 15062.73601494,
       15521.32569106, 15470.9247807 , 16471.0711916 , 16348.81104553,
       15683.54628842, 22043.56347567, 16496.51185371, 18458.85305648,
       19942.14933289, 25437.14612157, 15332.15485847, 15121.23173562,
       14168.26651382, 14221.35740453, 13494.23261179, 15480.55360766,
       16664.2478907 , 16693.40105088, 15458.12447484, 15672.58694649,
       15965.18295562, 15823.71961538, 15195.47262749, 15289.7203715 ,
       17935.74115655, 15663.17145113, 15150.43017815, 14799.18716655,
       15947.27510321, 15784.01580558, 15856.39662039, 16284.41459028])
```

Making the linear regression model

In [38]:

```
from sklearn.linear_model import LinearRegression


lr = LinearRegression()

lr.fit(X_train, y_train)
```

Out[38]:

```
LinearRegression()
```

In [39]:

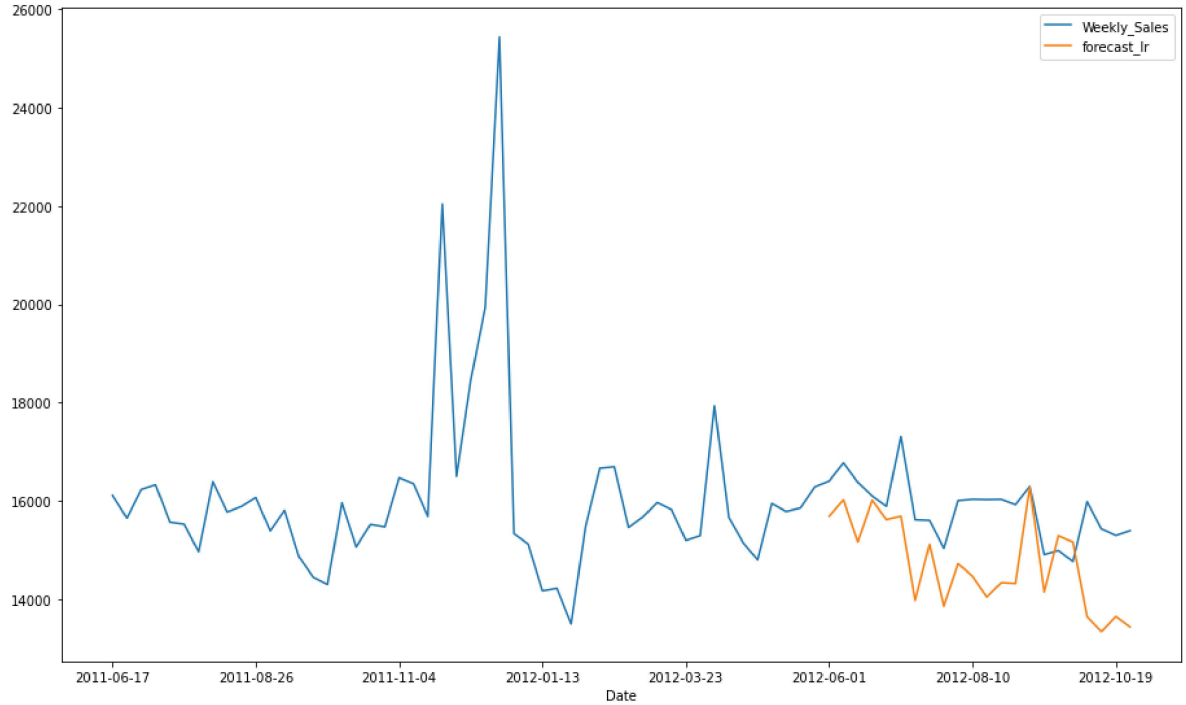
```
from sklearn.metrics import mean_absolute_error


pred_lr = lr.predict(X_test)

abs_error_lr = mean_absolute_error(y_test, pred_lr)
abs_error_lr
```

Out[39]:

```
1163.5872039677333
```

In [40]:

```
import numpy as np
pd.plotting.register_matplotlib_converters()
fig = plt.figure(figsize = (12, 7))
axes = fig.add_axes([0, 0, 1, 1])
df_agg_stores['forecast_lr'] = np.nan
df_agg_stores['forecast_lr'][120:] = pred_lr
df_agg_stores[70:][['Weekly_Sales', 'forecast_lr']].plot(ax = axes)
plt.show()
```



In [41]:

```
df_results = pd.DataFrame()
df_results["Weekly_Sales"] = df_agg_stores["Weekly_Sales"]
df_results["forecast_lr"] = df_agg_stores["forecast_lr"]
df_results[120:]
```

Out[41]:

Date	Weekly_Sales	forecast_lr
2012-06-01	16405.589439	15689.455276
2012-06-08	16774.044520	16021.916708
2012-06-15	16377.574662	15167.571166
2012-06-22	16098.711574	16013.641984
2012-06-29	15887.184494	15618.634635
2012-07-06	17309.362337	15689.856373
2012-07-13	15616.440413	13972.687205
2012-07-20	15602.826372	15116.641275
2012-07-27	15034.829516	13857.420652
2012-08-03	16004.684719	14722.392250
2012-08-10	16030.926967	14465.141113
2012-08-17	16025.195279	14041.652014
2012-08-24	16029.501216	14334.811865
2012-08-31	15921.552812	14318.705171
2012-09-07	16294.692957	16277.390703
2012-09-14	14905.978648	14145.973946
2012-09-21	14989.708385	15289.735934
2012-09-28	14765.327279	15161.005530
2012-10-05	15983.413747	13641.538223
2012-10-12	15427.596739	13337.774915
2012-10-19	15295.732397	13648.573587
2012-10-26	15391.725681	13432.572741

Making the Descision Tree regression model

In [42]:

```
from sklearn.tree import DecisionTreeRegressor

dtree = DecisionTreeRegressor()

dtree.fit(X_train, y_train)
```

Out[42]:

```
DecisionTreeRegressor()
```

In [43]:

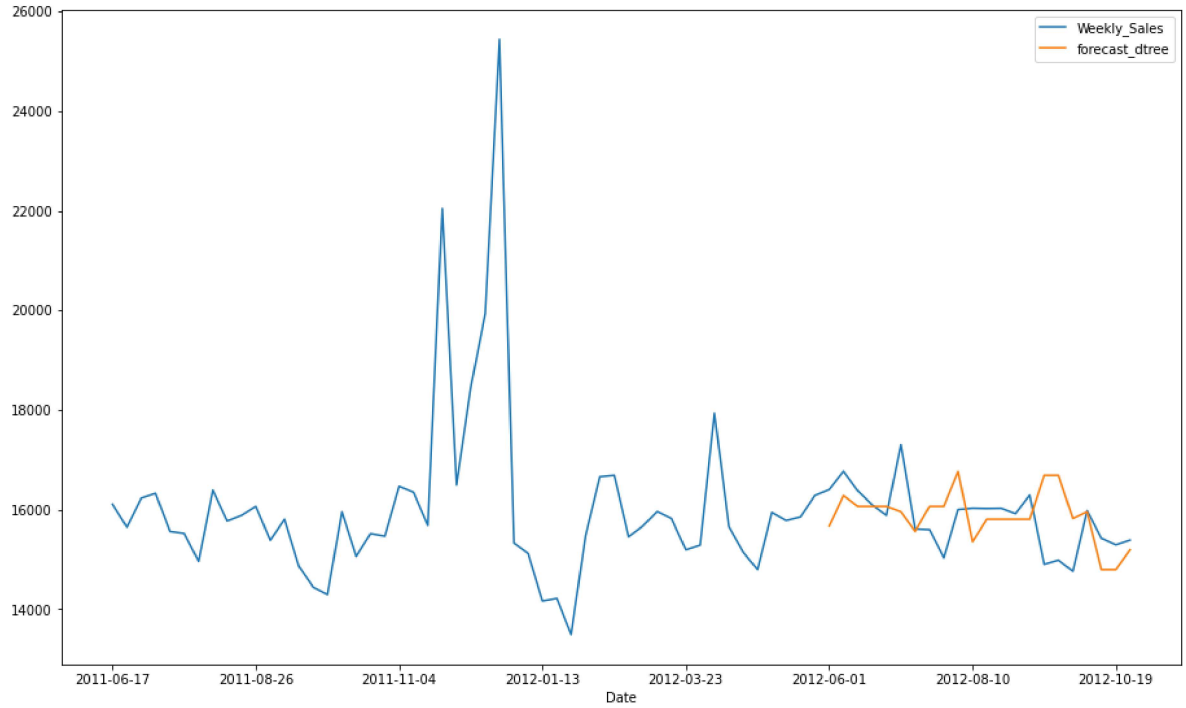
```
pred_dtree = dtree.predict(X_test)

abs_error_dtree = mean_absolute_error(y_test, pred_dtree)
abs_error_dtree
```

Out[43]:

```
591.2730447318302
```

In [44]:

```
pd.plotting.register_matplotlib_converters()
fig = plt.figure(figsize = (12, 7))
axes = fig.add_axes([0, 0, 1, 1])
df_agg_stores['forecast_dtree'] = np.nan
df_agg_stores['forecast_dtree'][120:] = pred_dtree#model.predict(X_test).reshape(-1)
df_agg_stores[70:][['Weekly_Sales', 'forecast_dtree']].plot(ax = axes)
plt.show()
```



In [45]:

```
df_results["forecast_dtree"] = df_agg_stores["forecast_dtree"]
df_results[120:]
```

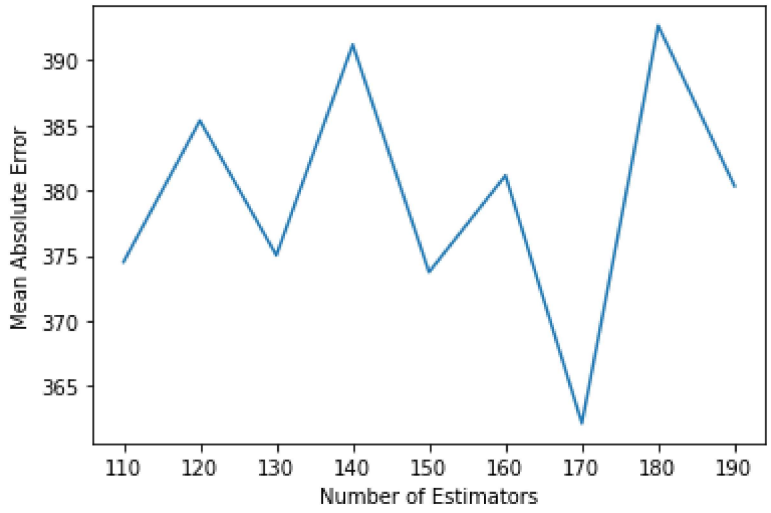
Out[45]:

Date	Weekly_Sales	forecast_lr	forecast_dtree
2012-06-01	16405.589439	15689.455276	15672.586946
2012-06-08	16774.044520	16021.916708	16284.414590
2012-06-15	16377.574662	15167.571166	16068.095036
2012-06-22	16098.711574	16013.641984	16068.095036
2012-06-29	15887.184494	15618.634635	16068.095036
2012-07-06	17309.362337	15689.856373	15960.678959
2012-07-13	15616.440413	13972.687205	15566.323502
2012-07-20	15602.826372	15116.641275	16068.095036
2012-07-27	15034.829516	13857.420652	16068.095036
2012-08-03	16004.684719	14722.392250	16769.792424
2012-08-10	16030.926967	14465.141113	15357.115672
2012-08-17	16025.195279	14041.652014	15809.069483
2012-08-24	16029.501216	14334.811865	15809.069483
2012-08-31	15921.552812	14318.705171	15809.069483
2012-09-07	16294.692957	16277.390703	15809.069483
2012-09-14	14905.978648	14145.973946	16693.401051
2012-09-21	14989.708385	15289.735934	16693.401051
2012-09-28	14765.327279	15161.005530	15823.719615
2012-10-05	15983.413747	13641.538223	15960.678959
2012-10-12	15427.596739	13337.774915	14799.187167
2012-10-19	15295.732397	13648.573587	14799.187167
2012-10-26	15391.725681	13432.572741	15195.472627

Making the Random forest model

In [46]:

```
from sklearn.ensemble import RandomForestRegressor
errors = []

for n in range(110, 200, 10):
  rfr = RandomForestRegressor(n_estimators=n)

  rfr.fit(X_train, y_train)

  pred_rfr = rfr.predict(X_test)
  errors.append(mean_absolute_error(y_test, pred_rfr))


plt.figure()
plt.plot(range(110, 200, 10), errors)
plt.xlabel("Number of Estimators")
plt.ylabel("Mean Absolute Error")
plt.show()

print(errors)
```




```
[374.51765008437593, 385.3386697430132, 375.0239961205274, 391.1609259751221
5, 373.7378019288057, 381.14083861812355, 362.15764313949785, 392.6143150481
801, 380.32764298789186]
```

In [47]:

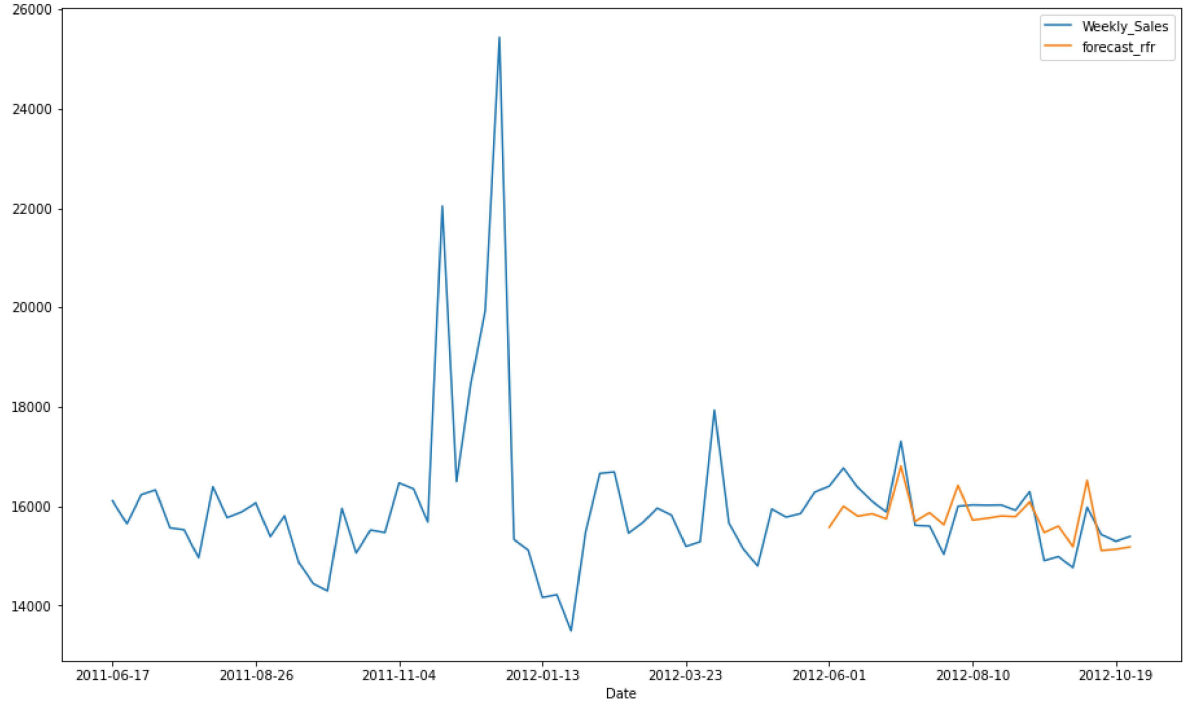
```
best_n = errors.index(min(errors))

best_n = 110 + 10*best_n

rfr = RandomForestRegressor(n_estimators=best_n)

rfr.fit(X_train, y_train)

pred_rfr = rfr.predict(X_test)
abs_error_rfr = mean_absolute_error(y_test, pred_rfr)
abs_error_rfr
```

Out[47]:

```
380.4729886517685
```

In [48]:

```
pd.plotting.register_matplotlib_converters()
fig = plt.figure(figsize = (12, 7))
axes = fig.add_axes([0, 0, 1, 1])
df_agg_stores['forecast_rfr'] = np.nan
df_agg_stores['forecast_rfr'][120:] = pred_rfr
df_agg_stores[70:][['Weekly_Sales', 'forecast_rfr']].plot(ax = axes)
plt.show()
```



In [49]:

```
df_results["forecast_rfr"] = df_agg_stores["forecast_rfr"]
df_results[120:]
```

Out[49]:

Date	Weekly_Sales	forecast_lr	forecast_dtree	forecast_rfr
2012-06-01	16405.589439	15689.455276	15672.586946	15577.832730
2012-06-08	16774.044520	16021.916708	16284.414590	16004.463766
2012-06-15	16377.574662	15167.571166	16068.095036	15803.330698
2012-06-22	16098.711574	16013.641984	16068.095036	15852.392785
2012-06-29	15887.184494	15618.634635	16068.095036	15747.915919
2012-07-06	17309.362337	15689.856373	15960.678959	16809.534432
2012-07-13	15616.440413	13972.687205	15566.323502	15696.578961
2012-07-20	15602.826372	15116.641275	16068.095036	15872.440597
2012-07-27	15034.829516	13857.420652	16068.095036	15628.058141
2012-08-03	16004.684719	14722.392250	16769.792424	16420.743500
2012-08-10	16030.926967	14465.141113	15357.115672	15722.793275
2012-08-17	16025.195279	14041.652014	15809.069483	15759.957449
2012-08-24	16029.501216	14334.811865	15809.069483	15806.277735
2012-08-31	15921.552812	14318.705171	15809.069483	15792.599231
2012-09-07	16294.692957	16277.390703	15809.069483	16087.874055
2012-09-14	14905.978648	14145.973946	16693.401051	15470.879980
2012-09-21	14989.708385	15289.735934	16693.401051	15601.981955
2012-09-28	14765.327279	15161.005530	15823.719615	15188.940404
2012-10-05	15983.413747	13641.538223	15960.678959	16523.003032
2012-10-12	15427.596739	13337.774915	14799.187167	15111.805673
2012-10-19	15295.732397	13648.573587	14799.187167	15137.399345
2012-10-26	15391.725681	13432.572741	15195.472627	15184.225723

Making the Support vector model

In [50]:

```
from sklearn.svm import SVR

svr = SVR()

svr.fit(X_train, y_train)
```

Out[50]:

```
SVR()
```

In [51]:

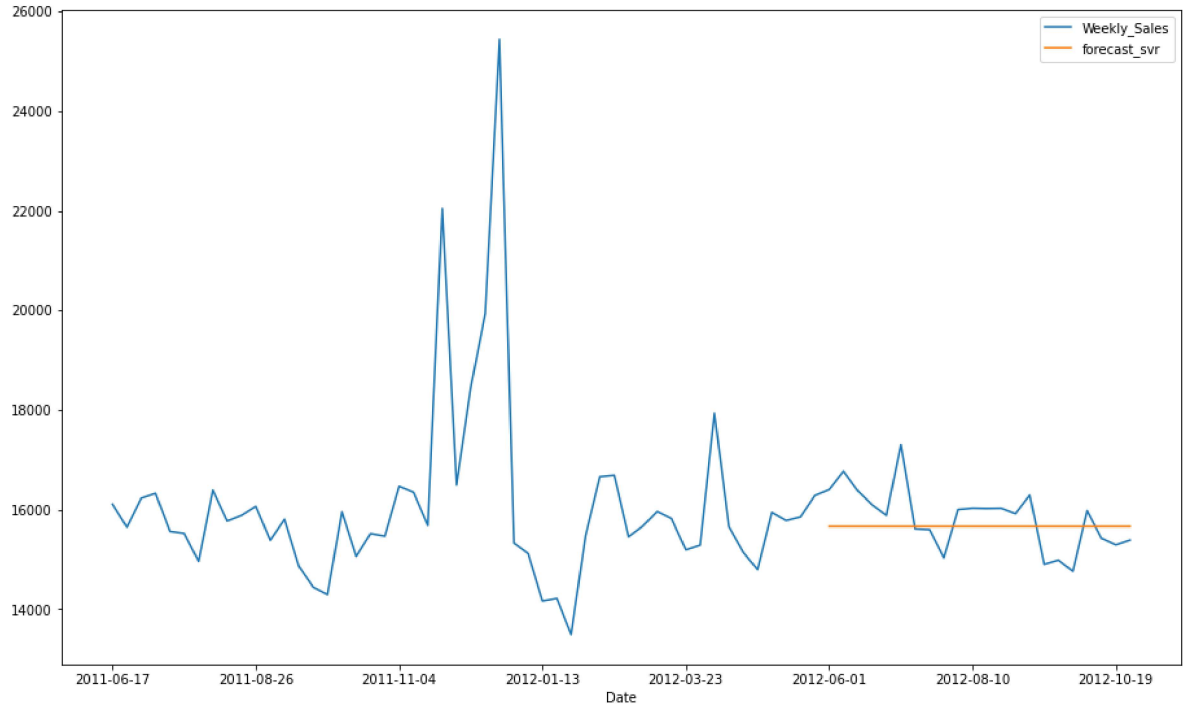
```
pred_svr = svr.predict(X_test)

abs_error_svr = mean_absolute_error(y_test, pred_svr)
abs_error_svr
```

Out[51]:

```
520.0152451944913
```

In [52]:

```
pd.plotting.register_matplotlib_converters()
fig = plt.figure(figsize = (12, 7))
axes = fig.add_axes([0, 0, 1, 1])
df_agg_stores['forecast_svr'] = np.nan
df_agg_stores['forecast_svr'][120:] = pred_svr
df_agg_stores[70:][['Weekly_Sales', 'forecast_svr']].plot(ax = axes)
plt.show()
```



In [53]:

```
df_results["forecast_svr"] = df_agg_stores["forecast_svr"]
df_results[120:]
```

Out[53]:

Date	Weekly_Sales	forecast_lr	forecast_dtree	forecast_rfr	forecast_svr
2012-06-01	16405.589439	15689.455276	15672.586946	15577.832730	15667.888324
2012-06-08	16774.044520	16021.916708	16284.414590	16004.463766	15667.905766
2012-06-15	16377.574662	15167.571166	16068.095036	15803.330698	15667.876777
2012-06-22	16098.711574	16013.641984	16068.095036	15852.392785	15667.880316
2012-06-29	15887.184494	15618.634635	16068.095036	15747.915919	15667.740458
2012-07-06	17309.362337	15689.856373	15960.678959	16809.534432	15667.932646
2012-07-13	15616.440413	13972.687205	15566.323502	15696.578961	15667.886032
2012-07-20	15602.826372	15116.641275	16068.095036	15872.440597	15667.865628
2012-07-27	15034.829516	13857.420652	16068.095036	15628.058141	15667.865941
2012-08-03	16004.684719	14722.392250	16769.792424	16420.743500	15668.062113
2012-08-10	16030.926967	14465.141113	15357.115672	15722.793275	15667.933851
2012-08-17	16025.195279	14041.652014	15809.069483	15759.957449	15667.834954
2012-08-24	16029.501216	14334.811865	15809.069483	15806.277735	15667.885074
2012-08-31	15921.552812	14318.705171	15809.069483	15792.599231	15667.994942
2012-09-07	16294.692957	16277.390703	15809.069483	16087.874055	15667.913049
2012-09-14	14905.978648	14145.973946	16693.401051	15470.879980	15667.939319
2012-09-21	14989.708385	15289.735934	16693.401051	15601.981955	15667.879543
2012-09-28	14765.327279	15161.005530	15823.719615	15188.940404	15667.843927
2012-10-05	15983.413747	13641.538223	15960.678959	16523.003032	15667.935943
2012-10-12	15427.596739	13337.774915	14799.187167	15111.805673	15667.900481
2012-10-19	15295.732397	13648.573587	14799.187167	15137.399345	15667.875859
2012-10-26	15391.725681	13432.572741	15195.472627	15184.225723	15667.793585

In [ ]:

```

```

In [ ]:

```

```

Generate sample data

In [25]:

```
X = np.sort(5 * np.random.rand(40, 1), axis=0)
y = np.sin(X).ravel()

# add noise to targets
y[::5] += 3 * (0.5 - np.random.rand(8))
```

Fit regression model

In [26]:

```
svr_rbf = SVR(kernel="rbf", C=100, gamma=0.1, epsilon=0.1)
svr_lin = SVR(kernel="linear", C=100, gamma="auto")
svr_poly = SVR(kernel="poly", C=100, gamma="auto", degree=3, epsilon=0.1, coef0=1)
```

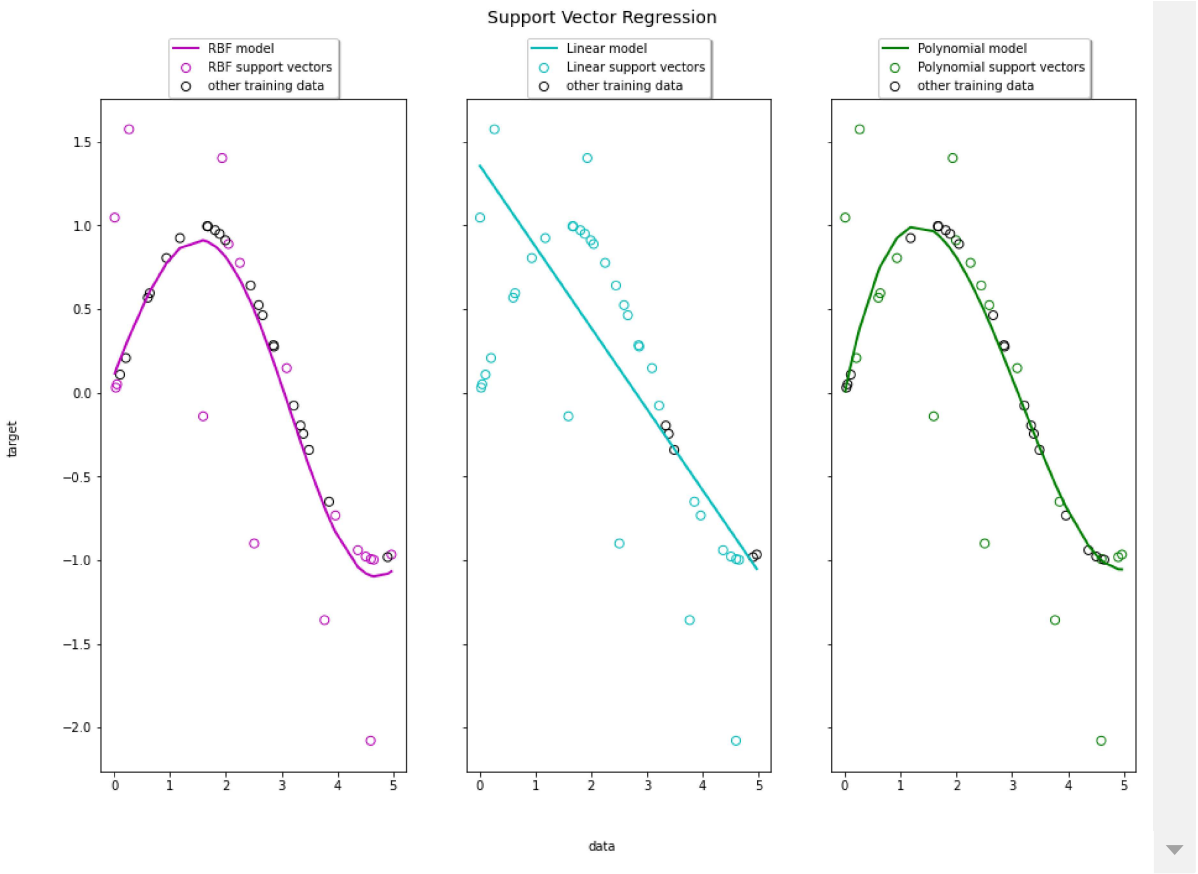
In [27]:

```
svr_rbf = SVR(kernel="rbf", C=100, gamma=0.1, epsilon=0.1)
svr_lin = SVR(kernel="linear", C=100, gamma="auto")
svr_poly = SVR(kernel="poly", C=100, gamma="auto", degree=3, epsilon=0.1, coef0=1)
```

Look at the results

In [28]:

```
lw = 2

svrs = [svr_rbf, svr_lin, svr_poly]
kernel_label = ["RBF", "Linear", "Polynomial"]
model_color = ["m", "c", "g"]

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 10), sharey=True)
for ix, svr in enumerate(svrs):
    axes[ix].plot(
        X,
        svr.fit(X, y).predict(X),
        color=model_color[ix],
        lw=lw,
        label="{} model".format(kernel_label[ix]),
    )
    axes[ix].scatter(
        X[svr.support_],
        y[svr.support_],
        facecolor="none",
        edgecolor=model_color[ix],
        s=50,
        label="{} support vectors".format(kernel_label[ix]),
    )
    axes[ix].scatter(
        X[np.setdiff1d(np.arange(len(X)), svr.support_)],
        y[np.setdiff1d(np.arange(len(X)), svr.support_)],
        facecolor="none",
        edgecolor="k",
        s=50,
        label="other training data",
    )
    axes[ix].legend(
        loc="upper center",
        bbox_to_anchor=(0.5, 1.1),
        ncol=1,
        fancybox=True,
        shadow=True,
    )

fig.text(0.5, 0.04, "data", ha="center", va="center")
fig.text(0.06, 0.5, "target", ha="center", va="center", rotation="vertical")
fig.suptitle("Support Vector Regression", fontsize=14)
plt.show()
```


Support Vector Regression
RBF model
RBF support vectors
other training data
Linear model
Linear support vectors
other training data
Polynomial model
Polynomial support vectors
other training data
target
data

Making the KNN model

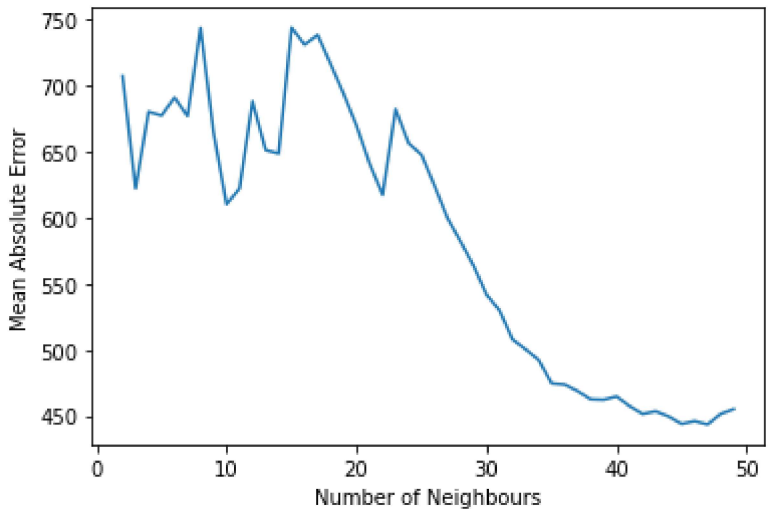
In [70]:
```
 from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor()
knr.fit(X_train, y_train)

```

Out[70]:
```
KNeighborsRegressor()
```

In [59]:
```
pred_knr = knr.predict(X_test)

abs_error_knr = mean_absolute_error(y_test, pred_knr)
abs_error_knr
```

Out[59]:
```
677.50190431271
```

In [60]:
```
errors = []

for k in range(2, 50):
  knn = KNeighborsRegressor(n_neighbors=k)

  knn.fit(X_train, y_train)

  pred_knn = knn.predict(X_test)
  errors.append(mean_absolute_error(y_test, pred_knn))


plt.figure()
plt.plot(range(2, 50), errors)
plt.xlabel('Number of Neighbours')
plt.ylabel('Mean Absolute Error')
plt.show()
```



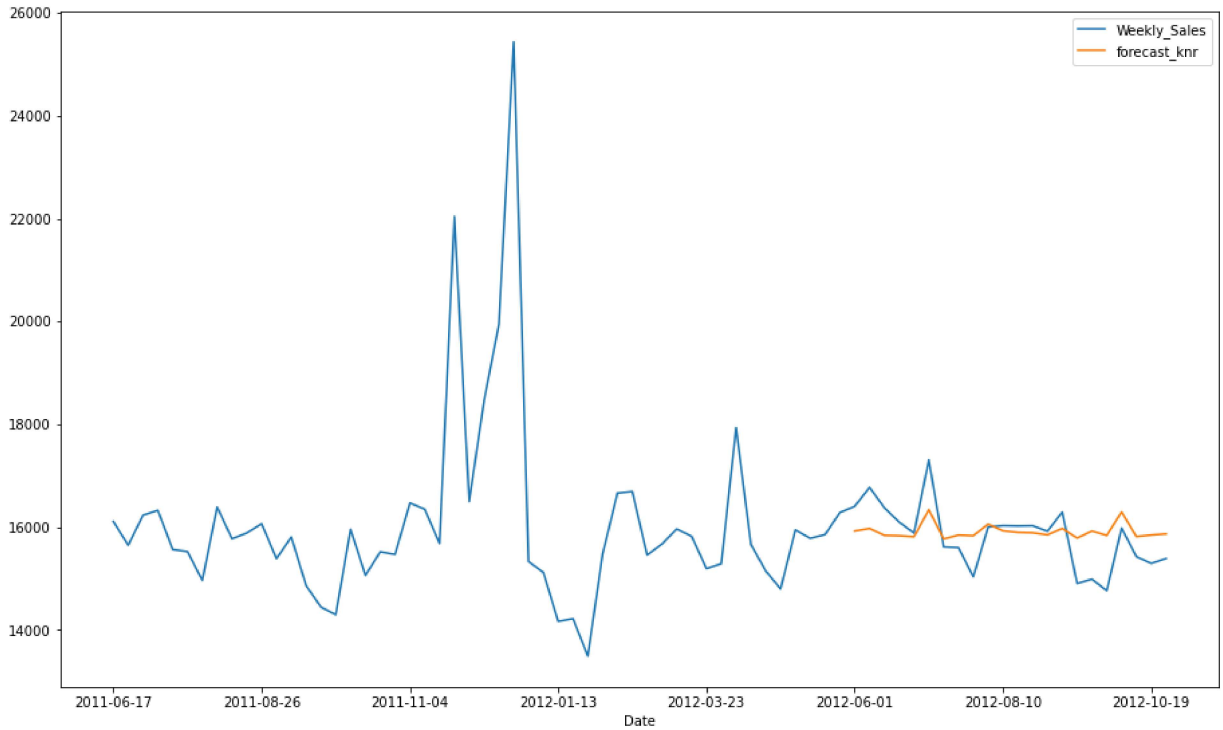
In [61]:
```
best_n = errors.index(min(errors))
best_n = 10 + 2*best_n
rfr = RandomForestRegressor(n_estimators=best_n)
rfr.fit(X_train, y_train)
pred_rfr = rfr.predict(X_test)
abs_error_rfr = mean_absolute_error(y_test, pred_rfr)
abs_error_rfr

```

Out[61]: 379.82064803881656

In [62]:
```
best_k = errors.index(min(errors)) + 2

knr = KNeighborsRegressor(n_neighbors = best_k)

knr.fit(X_train, y_train)

pred_knr = knr.predict(X_test)
abs_error_knr = mean_absolute_error(y_test, pred_knr)
abs_error_knr
```

Out[62]: 443.41919696363294

In [63]:
```
pd.plotting.register_matplotlib_converters()
fig = plt.figure(figsize = (12, 7))
axes = fig.add_axes([0, 0, 1, 1])
df_agg_stores['forecast_knr'] = np.nan
df_agg_stores['forecast_knr'][120:] = pred_knr
df_agg_stores[70:][['Weekly_Sales', 'forecast_knr']].plot(ax = axes)
plt.show()
```



In [64]:
```
df_results["forecast_knr"] = df_agg_stores["forecast_knr"]
df_results[120:]
```

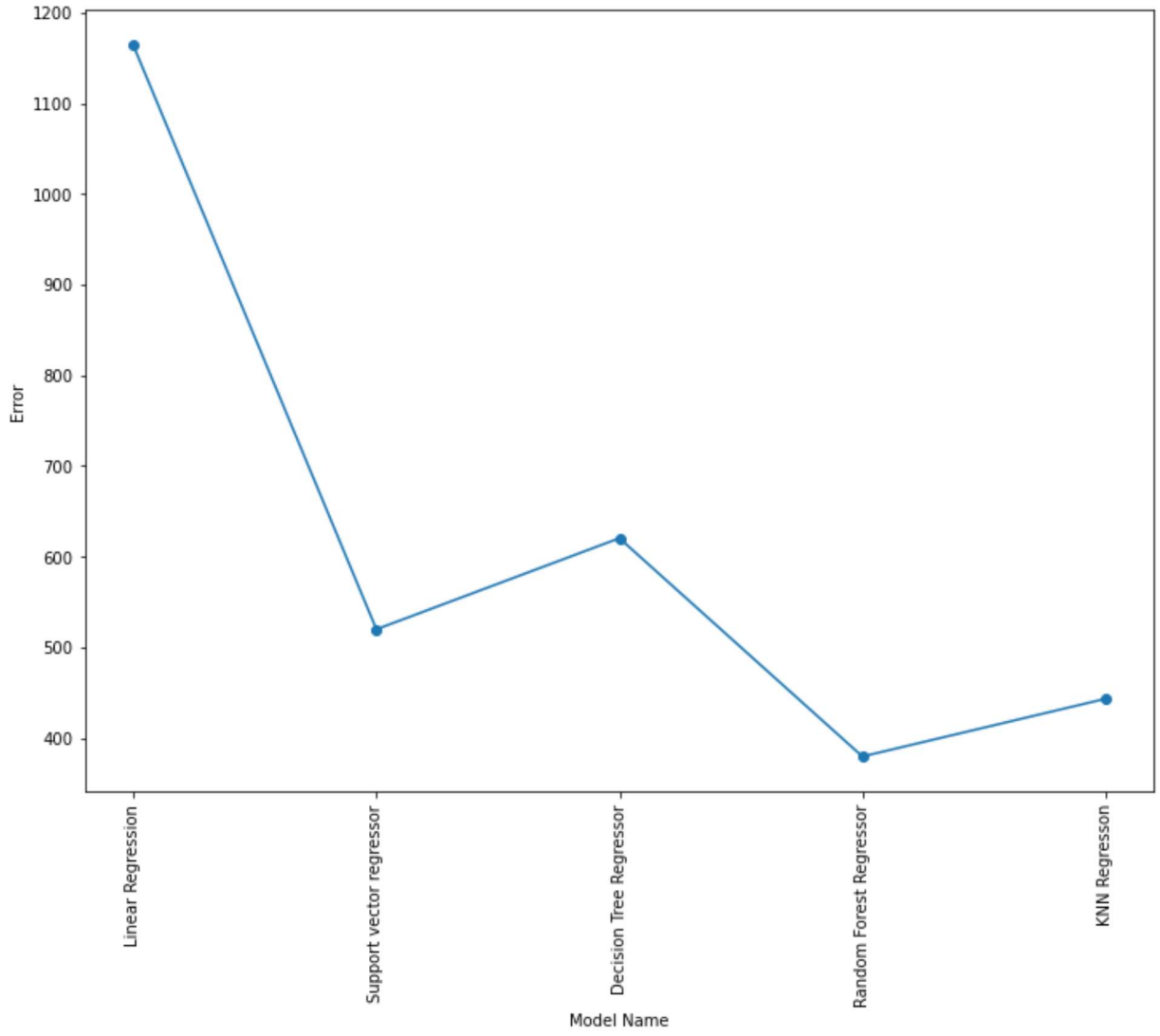
Out[64]:

Date	Weekly_Sales	forecast_lr	forecast_dtree	forecast_rfr	forecast_svr	forecast_knr
2012-06-01	16405.589439	15689.455276	15823.719615	15657.233734	15667.888324	15925.988594
2012-06-08	16774.044520	16021.916708	15965.182956	16065.923212	15667.905766	15976.189276
2012-06-15	16377.574662	15167.571166	16068.095036	15923.274719	15667.876777	15841.506154
2012-06-22	16098.711574	16013.641984	16068.095036	15933.910834	15667.880316	15833.818494
2012-06-29	15887.184494	15618.634635	16068.095036	15829.535297	15667.740458	15814.226451
2012-07-06	17309.362337	15689.856373	15480.553608	16776.393196	15667.932646	16340.562935
2012-07-13	15616.440413	13972.687205	15387.122167	15762.298375	15667.886032	15771.344312
2012-07-20	15602.826372	15116.641275	16068.095036	15974.011046	15667.865628	15846.930758
2012-07-27	15034.829516	13857.420652	16068.095036	15683.206189	15667.865941	15834.994002
2012-08-03	16004.684719	14722.392250	16769.792424	16415.287289	15668.062113	16059.829362
2012-08-10	16030.926967	14465.141113	15526.204180	15726.408615	15667.933851	15930.056948
2012-08-17	16025.195279	14041.652014	15809.069483	15835.497780	15667.834954	15899.849956
2012-08-24	16029.501216	14334.811865	15809.069483	15927.587192	15667.885074	15892.047772
2012-08-31	15921.552812	14318.705171	15774.068626	15932.342823	15667.994942	15851.374598
2012-09-07	16294.692957	16277.390703	15809.069483	16080.477288	15667.913049	15976.189276
2012-09-14	14905.978648	14145.973946	16496.511854	15536.620719	15667.939319	15788.779285
2012-09-21	14989.708385	15289.735934	16496.511854	15627.199011	15667.879543	15925.988594
2012-09-28	14765.327279	15161.005530	15289.720371	15193.413019	15667.843927	15839.900945
2012-10-05	15983.413747	13641.538223	15480.553608	16508.731072	15667.935943	16297.575050
2012-10-12	15427.596739	13337.774915	14799.187167	14936.647246	15667.900481	15817.573222
2012-10-19	15295.732397	13648.573587	14799.187167	15004.465538	15667.875859	15848.244039

Date	Weekly_Sales	forecast_lr	forecast_dtree	forecast_rfr	forecast_svr	forecast_knr
2012-10-26	15391.725681	13432.572741	14799.187167	15161.580896	15667.793585	15869.800856

Model Comparison

In [69]:
```
model_errors = {'Linear Regression': abs_error_lr,
 'Support vector regressor': abs_error_svr,
 'Decision Tree Regressor': abs_error_dtree,
 'Random Forest Regressor': abs_error_rfr,
 'KNN Regresson': abs_error_knr}
plt.figure(figsize = (12, 9))
plt.plot(list(model_errors.keys()), list(model_errors.values()), marker = 'o
plt.xlabel('Model Name')
plt.ylabel('Error')
plt.xticks(rotation = 90)
plt.show()
```



The best we have got so far is random forest regressor