In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [102]:
```
from datetime import datetime as dt
from sklearn import metrics
import seaborn as sns
import pandas_profiling
from sklearn.model_selection import cross_val_score
import statsmodels.formula.api as smf
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_predict

from sklearn.tree import plot_tree
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import classification_report,plot_confusion_matrix,C
from sklearn.tree import DecisionTreeClassifier
from sklearn.inspection import permutation_importance
```

In [103]:
```
train=pd.read_csv('train.csv',parse_dates=['Date'])
features=pd.read_csv('features.csv',parse_dates=['Date'])
stores=pd.read_csv('stores.csv')
```

In [13]:
```
df=pd.merge(features,stores,on=['Store'])
df.head()
```

Out[13]:

| | Store | Date | Temperature | Fuel_Price | MarkDown1 | MarkDown2 | MarkDown3 | MarkDown4 | M |
|---|---|---|---|---|---|---|---|---|---|
| **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 | |

In [14]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8190 entries, 0 to 8189
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         8190 non-null   int64
 1   Date          8190 non-null   datetime64[ns]
 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
 12  Type          8190 non-null   object
 13  Size          8190 non-null   int64
dtypes: bool(1), datetime64[ns](1), float64(9), int64(2), object(1)
memory usage: 903.8+ KB
```

In [15]:
```
1 df.drop(['MarkDown1','MarkDown2','MarkDown3','MarkDown4','MarkDown5'],axi
```

In [16]:
```
1 df.head()
```

Out[16]:

| | Store | Date | Temperature | Fuel_Price | CPI | Unemployment | IsHoliday | Type | Size |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-02-05 | 42.31 | 2.572 | 211.096358 | 8.106 | False | A | 151315 |
| **1** | 1 | 2010-02-12 | 38.51 | 2.548 | 211.242170 | 8.106 | True | A | 151315 |
| **2** | 1 | 2010-02-19 | 39.93 | 2.514 | 211.289143 | 8.106 | False | A | 151315 |
| **3** | 1 | 2010-02-26 | 46.63 | 2.561 | 211.319643 | 8.106 | False | A | 151315 |
| **4** | 1 | 2010-03-05 | 46.50 | 2.625 | 211.350143 | 8.106 | False | A | 151315 |

In [17]:
```
1 df.to_csv('all_features.csv',index=False)
```

In [19]:
```
data=pd.merge(train,df,on=['Date','Store','IsHoliday'])
data.head()
```

Out[19]:

| | Store | Dept | Date | Weekly_Sales | IsHoliday | Temperature | Fuel_Price | CPI | Unemploy |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 2010-02-05 | 24924.50 | False | 42.31 | 2.572 | 211.096358 | |
| **1** | 1 | 2 | 2010-02-05 | 50605.27 | False | 42.31 | 2.572 | 211.096358 | |
| **2** | 1 | 3 | 2010-02-05 | 13740.12 | False | 42.31 | 2.572 | 211.096358 | |
| **3** | 1 | 4 | 2010-02-05 | 39954.04 | False | 42.31 | 2.572 | 211.096358 | |
| **4** | 1 | 5 | 2010-02-05 | 32229.38 | False | 42.31 | 2.572 | 211.096358 | |

In [20]:
```
data.shape
```

Out[20]: (421570, 11)

In [105]:
```
# Checking Null values
data.isnull().sum()*100/data.shape[0]
```

Out[105]:
```
Store             0.0
Dept              0.0
Weekly_Sales      0.0
Temperature       0.0
Fuel_Price        0.0
CPI               0.0
Unemployment      0.0
Size              0.0
Year              0.0
Month             0.0
Day               0.0
IsHoliday_True    0.0
Type_C            0.0
dtype: float64
```

In [21]:
```
#splitting date into year, month and day
y=[]
m=[]
d=[]
for dt in data['Date']:
    y.append(dt.year)
    m.append(dt.month)
    d.append(dt.day)
```

In [22]:
```
data['Year'] = pd.Series(y)
data['Month'] = pd.Series(m)
data['Day'] = pd.Series(d)
```

In [23]:
```

data.drop('Date',axis=1,inplace=True)
```

In [24]:
```
data.head()
```

Out[24]:

| | Store | Dept | Weekly_Sales | IsHoliday | Temperature | Fuel_Price | CPI | Unemployment |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 24924.50 | False | 42.31 | 2.572 | 211.096358 | 8.106 |
| **1** | 1 | 2 | 50605.27 | False | 42.31 | 2.572 | 211.096358 | 8.106 |
| **2** | 1 | 3 | 13740.12 | False | 42.31 | 2.572 | 211.096358 | 8.106 |
| **3** | 1 | 4 | 39954.04 | False | 42.31 | 2.572 | 211.096358 | 8.106 |
| **4** | 1 | 5 | 32229.38 | False | 42.31 | 2.572 | 211.096358 | 8.106 |

In [25]:
```
numeric_var_names=[key for key in dict(data.dtypes) if dict(data.dtypes)[
cat_var_names=[key for key in dict(data.dtypes) if dict(data.dtypes)[key]
```

In [26]:
```
numeric_var_names
```

Out[26]:
```
['Store',
 'Dept',
 'Weekly_Sales',
 'Temperature',
 'Fuel_Price',
 'CPI',
 'Unemployment',
 'Size',
 'Year',
 'Month',
 'Day']
```

In [27]:
```
cat_var_names
```

Out[27]:
```
['IsHoliday', 'Type']
```

In [28]:
```
1 data_num=data[numeric_var_names]
2 data_num.head(5)
```

Out[28]:

| | Store | Dept | Weekly_Sales | Temperature | Fuel_Price | CPI | Unemployment | Size | Ye |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 24924.50 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |
| **1** | 1 | 2 | 50605.27 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |
| **2** | 1 | 3 | 13740.12 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |
| **3** | 1 | 4 | 39954.04 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |
| **4** | 1 | 5 | 32229.38 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |

In [67]:
```
1 df = df.fillna(0)
2 df.isna().sum()
```

Out[67]:
```
Store           0
Date            0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
IsHoliday       0
Type            0
Size            0
dtype: int64
```

In [29]:
```
1 # Replace negative and 0 as missing
2 pd.set_option('mode.chained_assignment', None)
3 data_num[data_num<=0]=np.NaN
```

In [30]:
```
1 data_cat=data[cat_var_names]
2 data_cat.head(5)
```

Out[30]:

| | IsHoliday | Type |
|---|---|---|
| **0** | False | A |
| **1** | False | A |
| **2** | False | A |
| **3** | False | A |
| **4** | False | A |

In [31]:
```
#Creating audit report
def var_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.m
                      index=['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VA

data_num.apply(var_summary).T
```

Out[31]:

| | N | NMISS | SUM | MEAN | MEDIAN | STD | |
|---|---|---|---|---|---|---|---|
| **Store** | 421570.0 | 0.0 | 9.359084e+06 | 22.200546 | 22.00000 | 12.785297 | 1.6 |
| **Dept** | 421570.0 | 0.0 | 1.865882e+07 | 44.260317 | 37.00000 | 30.492054 | 9.2 |
| **Weekly_Sales** | 420212.0 | 1358.0 | 6.737307e+09 | 16033.114591 | 7661.70000 | 22729.492116 | 5.1 |
| **Temperature** | 421501.0 | 69.0 | 2.533231e+07 | 60.100233 | 62.09000 | 18.432294 | 3.3 |
| **Fuel_Price** | 421570.0 | 0.0 | 1.416908e+06 | 3.361027 | 3.45200 | 0.458515 | 2.1 |
| **CPI** | 421570.0 | 0.0 | 7.217360e+07 | 171.201947 | 182.31878 | 39.159276 | 1.5 |
| **Unemployment** | 421570.0 | 0.0 | 3.355819e+06 | 7.960289 | 7.86600 | 1.863296 | 3.4 |
| **Size** | 421570.0 | 0.0 | 5.764039e+10 | 136727.915739 | 140167.00000 | 60980.583328 | 3.7 |
| **Year** | 421570.0 | 0.0 | 8.477640e+08 | 2010.968591 | 2011.00000 | 0.796876 | 6.3 |
| **Month** | 421570.0 | 0.0 | 2.718920e+06 | 6.449510 | 6.00000 | 3.243217 | 1.0 |
| **Day** | 421570.0 | 0.0 | 6.607322e+06 | 15.673131 | 16.00000 | 8.753549 | 7.6 |

In [32]:
```
num_summary=data_num.apply(var_summary).T
num_summary.to_csv('num_summary.csv')
```

In [33]:
```
def cat_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.value_counts()],
                      index=['N', 'NMISS', 'ColumnsNames'])

cat_summary=data_cat.apply(cat_summary)
```

In [34]:
```
cat_summary
```

Out[34]:

| | IsHoliday | Type |
|---|---|---|
| **N** | 421570 | 421570 |
| **NMISS** | 0 | 0 |
| **ColumnsNames** | False 391909 True 29661 Name: IsHolida... | A 215478 B 163495 C 42597 Name: Type... |

In [35]:
```
#Handling Outliers
def outlier_capping(x):
    x = x.clip(upper=x.quantile(0.99))
    x = x.clip(lower=x.quantile(0.01))
    return x

data_num=data_num.apply(outlier_capping)
```

In [36]:
```
# An utility function to create dummy variable
def create_dummies( df, colname ):
    col_dummies = pd.get_dummies(df[colname], prefix=colname, drop_first=
    df = pd.concat([df, col_dummies], axis=1)
    df.drop( colname, axis = 1, inplace = True )
    return df
```

In [37]:
```
for c_feature in ['IsHoliday', 'Type']:
    data_cat.loc[:,c_feature] = data_cat[c_feature].astype('category')
    data_cat = create_dummies(data_cat , c_feature )

data_cat.head()
```

Out[37]:

| | IsHoliday_True | Type_B | Type_C |
|---|---|---|---|
| **0** | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 |

In [38]:
```
data = pd.concat([data_num, data_cat], axis=1)
data.head()
```

Out[38]:

| | Store | Dept | Weekly_Sales | Temperature | Fuel_Price | CPI | Unemployment | Size | Ye |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1.0 | 24924.50 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |
| **1** | 1 | 2.0 | 50605.27 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |
| **2** | 1 | 3.0 | 13740.12 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |
| **3** | 1 | 4.0 | 39954.04 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |
| **4** | 1 | 5.0 | 32229.38 | 42.31 | 2.572 | 211.096358 | 8.106 | 151315 | 20 |

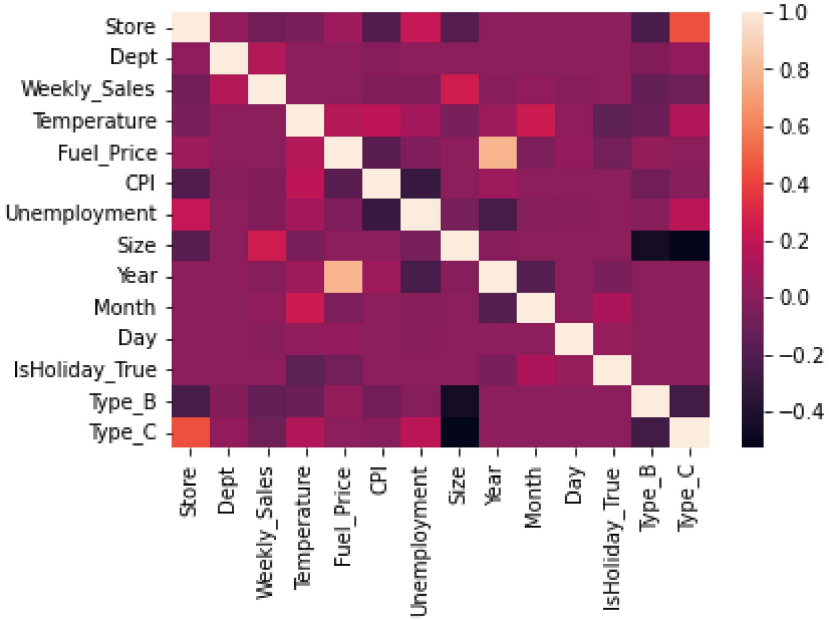
In [39]:
```
# correlation matrix (ranges from 1 to -1)
corrm=data.corr()
corrm.to_csv('corrm.csv')
corrm
```

Out[39]:

| | Store | Dept | Weekly_Sales | Temperature | Fuel_Price | CPI | Unempl |
|---|---|---|---|---|---|---|---|
| **Store** | 1.000000 | 0.024011 | -0.083726 | -0.052236 | 0.064984 | -0.210949 | 0 |
| **Dept** | 0.024011 | 1.000000 | 0.143747 | 0.004383 | 0.003514 | -0.007483 | 0 |
| **Weekly_Sales** | -0.083726 | 0.143747 | 1.000000 | 0.000578 | 0.001619 | -0.022952 | -0 |
| **Temperature** | -0.052236 | 0.004383 | 0.000578 | 1.000000 | 0.142032 | 0.184603 | 0 |
| **Fuel_Price** | 0.064984 | 0.003514 | 0.001619 | 0.142032 | 1.000000 | -0.162609 | -0 |
| **CPI** | -0.210949 | -0.007483 | -0.022952 | 0.184603 | -0.162609 | 1.000000 | -0 |
| **Unemployment** | 0.208671 | 0.007874 | -0.025302 | 0.095505 | -0.034938 | -0.300190 | 1 |
| **Size** | -0.182881 | -0.003022 | 0.252868 | -0.059273 | 0.004468 | -0.003307 | -0 |
| **Year** | 0.002997 | 0.003688 | -0.009238 | 0.066475 | 0.780315 | 0.074335 | -0 |
| **Month** | 0.001011 | 0.000894 | 0.025581 | 0.234560 | -0.042186 | 0.005130 | -0 |
| **Day** | -0.000015 | -0.000664 | -0.008031 | 0.025876 | 0.028013 | 0.002701 | -0 |
| **IsHoliday_True** | -0.000548 | 0.000906 | 0.007983 | -0.156726 | -0.077839 | -0.001934 | 0 |
| **Type_B** | -0.233461 | -0.029920 | -0.134640 | -0.107776 | 0.037372 | -0.081883 | -0 |
| **Type_C** | 0.439004 | 0.028344 | -0.099246 | 0.135717 | 0.001369 | -0.006557 | 0 |

In [40]:
```
# visualize correlation matrix in Seaborn using a heatmap
sns.heatmap(data.corr())
```

Out[40]: <AxesSubplot:>



In [41]:
```
1 data.columns
```

Out[41]:
```
Index(['Store', 'Dept', 'Weekly_Sales', 'Temperature', 'Fuel_Price', 'CPI',
       'Unemployment', 'Size', 'Year', 'Month', 'Day', 'IsHoliday_True',
       'Type_B', 'Type_C'],
      dtype='object')
```

In [42]:

```
#Model Building
lm=smf.ols('Weekly_Sales~Store+Dept+Temperature+Fuel_Price+CPI+Unemployme
print(lm.summary())
```

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | Weekly_Sales | R-squared: | 0.092 |
| Model: | OLS | Adj. R-squared: | 0.092 |
| Method: | Least Squares | F-statistic: | 3561. |
| Date: | Wed, 01 Feb 2023 | Prob (F-statistic): | 0.00 |
| Time: | 17:45:46 | Log-Likelihood: | -4.7532e+06 |
| No. Observations: | 420145 | AIC: | 9.506e+06 |
| Df Residuals: | 420132 | BIC: | 9.507e+06 |
| Df Model: | 12 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | 6.248e+05 | 1.41e+05 | 4.418 | 0.000 | 3.48e+05 | 9.02e+05 |
| Store | -123.5580 | 2.773 | -44.556 | 0.000 | -128.993 | -118.123 |
| Dept | 98.0683 | 1.003 | 97.787 | 0.000 | 96.103 | 100.034 |
| Temperature | 8.5709 | 1.879 | 4.560 | 0.000 | 4.887 | 12.255 |
| Fuel_Price | 310.7029 | 120.783 | 2.572 | 0.010 | 73.972 | 547.433 |
| CPI | -22.4452 | 0.905 | -24.811 | 0.000 | -24.218 | -20.672 |
| Unemployment | -239.5636 | 18.677 | -12.827 | 0.000 | -276.170 | -202.957 |
| Size | 0.0958 | 0.001 | 161.791 | 0.000 | 0.095 | 0.097 |
| Year | -308.7038 | 70.481 | -4.380 | 0.000 | -446.845 | -170.563 |
| Month | 136.7150 | 10.296 | 13.278 | 0.000 | 116.534 | 156.896 |
| Day | -20.7660 | 3.502 | -5.930 | 0.000 | -27.629 | -13.903 |
| IsHoliday_True | 541.7202 | 123.351 | 4.392 | 0.000 | 299.955 | 783.485 |
| Type_C | 5599.0401 | 134.163 | 41.733 | 0.000 | 5336.084 | 5861.996 |

| | | | |
|---|---|---|---|
| Omnibus: | 158881.021 | Durbin-Watson: | 1.276 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 576165.376 |

```
Skew:                          1.925   Prob(JB):                         0.
00
Kurtosis:                      7.253   Cond. No.                     6.93e+
08
================================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 6.93e+08. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

In [43]:
```
data.drop('Type_B',axis=1,inplace=True)
```

In [146]:
```
X=data[data.columns.difference(['Weekly_Sales'])]
y=data['Weekly_Sales']
```

In [148]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
model = DecisionTreeClassifier()
```

Decision Tree Model

In [150]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_predict
```

In [151]:
```
1 X_train
```

Out[151]:

| | CPI | Day | Dept | Fuel_Price | IsHoliday_True | Month | Size | Store | Temperature |
|---|---|---|---|---|---|---|---|---|---|
| **210547** | 136.887066 | 21 | 23.0 | 3.232 | 0 | 1 | 119557 | 22 | 26.80 |
| **149665** | 189.936850 | 30 | 46.0 | 2.690 | 0 | 7 | 57197 | 16 | 70.71 |
| **366935** | 214.793411 | 16 | 71.0 | 3.526 | 0 | 9 | 184109 | 39 | 83.11 |
| **138796** | 131.940807 | 9 | 58.0 | 2.992 | 0 | 4 | 123737 | 15 | 57.77 |
| **289858** | 215.612473 | 13 | 52.0 | 3.899 | 0 | 5 | 42988 | 30 | 75.04 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **105664** | 224.790910 | 23 | 32.0 | 3.787 | 0 | 3 | 207499 | 11 | 67.16 |
| **93622** | 129.430600 | 16 | 7.0 | 3.784 | 0 | 9 | 126512 | 10 | 77.49 |
| **356948** | 128.999867 | 29 | 40.0 | 4.151 | 0 | 4 | 39690 | 38 | 68.27 |
| **236738** | 138.653400 | 21 | 97.0 | 4.202 | 0 | 9 | 203819 | 24 | 61.25 |
| **345422** | 220.407558 | 11 | 17.0 | 3.764 | 0 | 5 | 39910 | 36 | 76.55 |

252900 rows × 12 columns

In [152]:
```
1 y_train
```

Out[152]:
```
210547    18971.65
149665    18388.06
366935     6737.93
138796      248.00
289858       60.67
             ...
105664     8562.70
93622     39469.62
356948    16063.01
236738    18618.99
345422      340.78
Name: Weekly_Sales, Length: 252900, dtype: float64
```

In [93]:
```
1 np.isnan(X)
```

Out[93]:

| | CPI | Day | Dept | Fuel_Price | IsHoliday_True | Month | Size | Store | Temperature | Type |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | False | Fa |
| **1** | False | False | False | False | False | False | False | False | False | Fa |
| **2** | False | False | False | False | False | False | False | False | False | Fa |
| **3** | False | False | False | False | False | False | False | False | False | Fa |
| **4** | False | False | False | False | False | False | False | False | False | Fa |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **421565** | False | False | False | False | False | False | False | False | False | Fa |
| **421566** | False | False | False | False | False | False | False | False | False | Fa |
| **421567** | False | False | False | False | False | False | False | False | False | Fa |
| **421568** | False | False | False | False | False | False | False | False | False | Fa |
| **421569** | False | False | False | False | False | False | False | False | False | Fa |

421570 rows × 12 columns

In [94]:
```
1 np.where(np.isnan(X))
```

Out[94]:
```
(array([], dtype=int64), array([], dtype=int64))
```

In [95]:
```
1 np.nan_to_num(X)
```

Out[95]:
```
array([[2.11096358e+02, 5.00000000e+00, 1.00000000e+00, ...,
        0.00000000e+00, 8.10600000e+00, 2.01000000e+03],
       [2.11096358e+02, 5.00000000e+00, 2.00000000e+00, ...,
        0.00000000e+00, 8.10600000e+00, 2.01000000e+03],
       [2.11096358e+02, 5.00000000e+00, 3.00000000e+00, ...,
        0.00000000e+00, 8.10600000e+00, 2.01000000e+03],
       ...,
       [1.92308899e+02, 2.60000000e+01, 9.50000000e+01, ...,
        0.00000000e+00, 8.66700000e+00, 2.01200000e+03],
       [1.92308899e+02, 2.60000000e+01, 9.70000000e+01, ...,
        0.00000000e+00, 8.66700000e+00, 2.01200000e+03],
       [1.92308899e+02, 2.60000000e+01, 9.80000000e+01, ...,
        0.00000000e+00, 8.66700000e+00, 2.01200000e+03]])
```

In [97]:
```
1 np.isnan(y)
```

Out[97]:
```
0         False
1         False
2         False
3         False
4         False
          ...
421565    False
421566    False
421567    False
421568    False
421569    False
Name: Weekly_Sales, Length: 421570, dtype: bool
```

In [98]:
```
1 np.where(np.isnan(y))
```

Out[98]:
```
(array([], dtype=int64),)
```

In [99]:
```
1 np.where(np.isnan(X))
```

Out[99]:
```
(array([], dtype=int64), array([], dtype=int64))
```

In [126]:
```
1 #converting flot64 into flot32 to fit into the model
2 X= np.array([1.37097033e+002, 0.00000000e+000, -1.82710826e+296,
3                          1.22703799e+002, 1.37097033e+002, -2.56391552e+0
4                          1.11457878e+002, 1.37097033e+002, -2.56391552e+0
5                          9.81898928e+001, 1.22703799e+002, -2.45139066e+0
6
7 print(X.dtype)
```

```
float64
```

In [127]:
```
1 print(X.astype(np.float32))
```

```
[[137.09703    0.             -inf 122.7038  ]
 [137.09703  -25.639154 111.45788  137.09703 ]
 [-25.639154  98.189896 122.7038   -24.513906]]
```

In [153]:
```
1 #DecisionTreeRegressor
2 regressor = DecisionTreeRegressor(max_depth=5,random_state=0)
3 regressor.fit(X_train, y_train)
```

Out[153]:
```
DecisionTreeRegressor(max_depth=5, random_state=0)
```

In [154]:
```
1 pred=regressor.predict(X_train)
2 y_pred=regressor.predict(X_test)
```

In [130]:
```
print('Train : ')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_train, pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_train, pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_tr
```

```
Train :
Mean Absolute Error: 9184.635673321476
Mean Squared Error: 193157750.80695537
Root Mean Squared Error: 13898.120405542448
```

In [131]:
```
print('Test : ')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_te
```

```
Test :
Mean Absolute Error: 9195.727322025185
Mean Squared Error: 194808218.42431915
Root Mean Squared Error: 13957.37147260612
```

In [132]:
```
from sklearn.metrics import r2_score

print("R-squared for Train:",r2_score(y_train, pred))
print("R-squared for Test:",r2_score(y_test, y_pred))
```

```
R-squared for Train: 0.5518236575556659
R-squared for Test: 0.5507072489745466
```

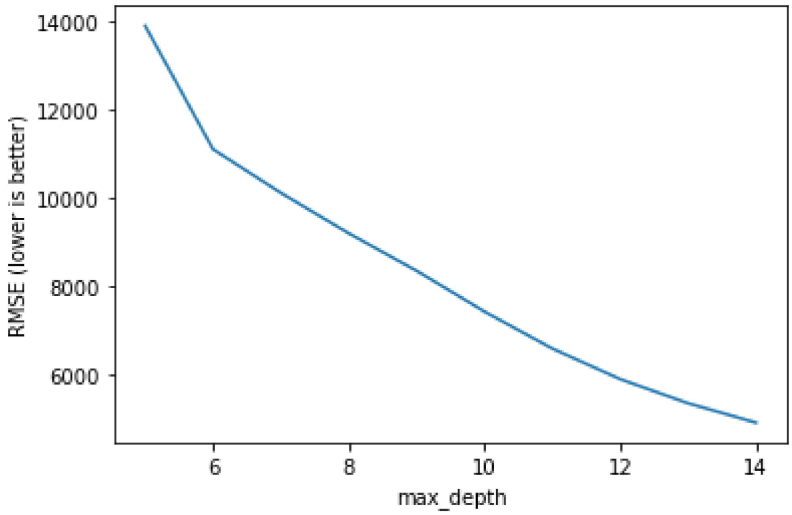
In [155]:
```
#values to try
max_depth_range = range(5, 15)

# list to store the average RMSE for each value of max_depth
RMSE_Scores = []
MSE_Scores = []

# using leave one out cross validation with each value of max_depth
for depth in max_depth_range:
    treereg = DecisionTreeRegressor(max_depth=depth, random_state=345)

    MSE_scores = cross_val_score(treereg, X_train, y_train, scoring='neg_

    RMSE_Scores.append(np.mean(np.sqrt(-MSE_scores)))
    MSE_Scores.append(MSE_scores)
```

In [156]:
```
print (RMSE_Scores)
```

```
[13897.551399637468, 11107.790293011207, 10121.03512206471, 9208.46690220469
3, 8359.170443777039, 7437.370716535212, 6602.0313022511455, 5910.5418048931
06, 5365.25587523505, 4929.164528024818]
```

In [157]:
```
# plot max_depth (x-axis) versus RMSE (y-axis)
plt.plot(max_depth_range, RMSE_Scores)
plt.xlabel('max_depth')
plt.ylabel('RMSE (lower is better)')
```

Out[157]: Text(0, 0.5, 'RMSE (lower is better)')




## Building Final Decision Tree Model

In [158]:
```
# max_depth=14 was best, so fitting a tree using that parameter
treereg = DecisionTreeRegressor(max_depth=14, random_state=345)
treereg.fit(X_train, y_train)
```

Out[158]: DecisionTreeRegressor(max_depth=14, random_state=345)

In [159]:
```
treereg.feature_importances_
```

Out[159]:
```
array([1.91149632e-02, 6.27894185e-03, 6.84155083e-01, 2.02643911e-03,
       1.24807035e-03, 1.62133882e-02, 1.86476467e-01, 5.57612488e-02,
       5.61057685e-03, 1.22374662e-02, 1.06651832e-02, 2.12172657e-04])
```

In [160]:
```
# Gini importance of each feature - total reduction of error brought by t
pd.DataFrame({'feature':data.columns.difference(['Weekly_Sales']), 'impor
```

Out[160]:

| | feature | importance |
|---|---|---|
| 0 | CPI | 0.019115 |
| 1 | Day | 0.006279 |
| 2 | Dept | 0.684155 |
| 3 | Fuel_Price | 0.002026 |
| 4 | IsHoliday_True | 0.001248 |
| 5 | Month | 0.016213 |
| 6 | Size | 0.186476 |
| 7 | Store | 0.055761 |
| 8 | Temperature | 0.005611 |
| 9 | Type_C | 0.012237 |
| 10 | Unemployment | 0.010665 |
| 11 | Year | 0.000212 |

KNN Model

In [161]:
```
from sklearn import neighbors
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [162]:
```
knn = neighbors.KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)
```

Out[162]:
```
KNeighborsRegressor()
```

In [163]:
```
pred=knn.predict(X_train)
y_pred=knn.predict(X_test)
```

In [164]:
```
print('Train : ')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_train, pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_train, pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_tr
```

```
Train :
Mean Absolute Error: 8631.206873331277
Mean Squared Error: 175977277.05015382
Root Mean Squared Error: 13265.642730382642
```

In [165]:
```
print('Test : ')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_te
```

```
Test :
Mean Absolute Error: 10523.184206715263
Mean Squared Error: 260515292.56450468
Root Mean Squared Error: 16140.486131604112
```

In [166]:
```
from sklearn.metrics import r2_score

print("R-squared for Train:",r2_score(y_train, pred))
print("R-squared for Test:",r2_score(y_test, y_pred))
```

```
R-squared for Train: 0.591596721582016
R-squared for Test: 0.39951131983086685
```

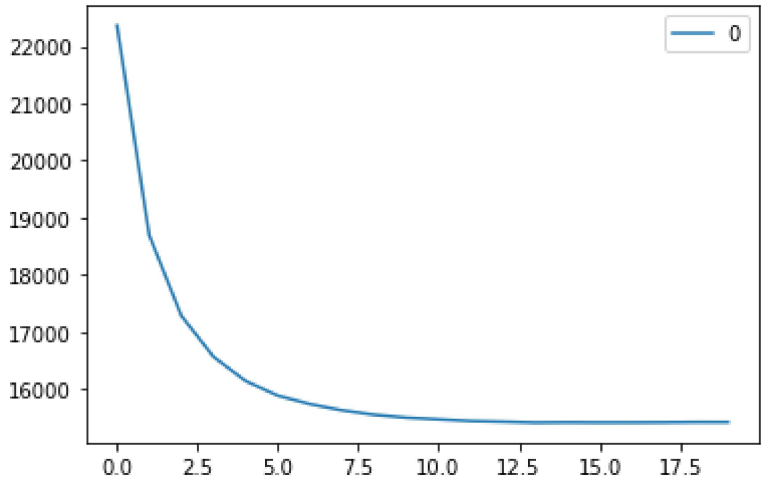
KNN tuning

In [167]:
```
rmse_val = [] #to store rmse values for different k
for K in range(20):
    K = K+1
    model = neighbors.KNeighborsRegressor(n_neighbors = K)

    model.fit(X_train, y_train)  #fit the model
    y_pred=model.predict(X_test) #make prediction on test set
    error = sqrt(mean_squared_error(y_test,y_pred)) #calculate rmse
    rmse_val.append(error) #store rmse values
    print('RMSE value for k= ' , K , 'is:', error)
```

```
RMSE value for k=  1 is: 22361.46308419623
RMSE value for k=  2 is: 18696.069184789707
RMSE value for k=  3 is: 17282.222534781533
RMSE value for k=  4 is: 16561.987595112718
RMSE value for k=  5 is: 16140.486131604112
RMSE value for k=  6 is: 15884.155200052954
RMSE value for k=  7 is: 15736.40209062235
RMSE value for k=  8 is: 15627.098853220215
RMSE value for k=  9 is: 15548.854579157736
RMSE value for k=  10 is: 15498.742058209797
RMSE value for k=  11 is: 15469.01256439094
RMSE value for k=  12 is: 15438.969555778402
RMSE value for k=  13 is: 15427.06231520774
RMSE value for k=  14 is: 15412.72972698155
RMSE value for k=  15 is: 15416.015345833122
RMSE value for k=  16 is: 15413.826219883227
RMSE value for k=  17 is: 15414.094465963073
RMSE value for k=  18 is: 15416.197321060637
RMSE value for k=  19 is: 15420.91267000322
RMSE value for k=  20 is: 15420.376716713094
```

In [168]:
```
#plotting the rmse values against k values
curve = pd.DataFrame(rmse_val)
curve.plot()
```

Out[168]: <AxesSubplot:>




KNN Final Model

In [169]:
```
knn = neighbors.KNeighborsRegressor(n_neighbors=11)
knn.fit(X_train, y_train)
```

Out[169]: KNeighborsRegressor(n_neighbors=11)

In [170]:
```
pred=knn.predict(X_train)
y_pred=knn.predict(X_test)
```

In [171]:
```
print('Train : ')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_train, pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_train, pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_tr
```

```
Train :
Mean Absolute Error: 9250.135192450627
Mean Squared Error: 197706712.91825768
Root Mean Squared Error: 14060.821914748003
```

In [172]:
```
print('Test : ')
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_te
```

```
Test :
Mean Absolute Error: 10174.286536555106
Mean Squared Error: 239290349.71728474
Root Mean Squared Error: 15469.01256439094
```

In [173]:
```
print("R-squared for Train:",r2_score(y_train, pred))
print("R-squared for Test:",r2_score(y_test, y_pred))
```

```
R-squared for Train: 0.5411676378078778
R-squared for Test: 0.448434888929355
```

In [ ]:
```

```