```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
import plotly.express as px
import numpy as np
import warnings
warnings = 'ignore'
%matplotlib inline
```

Dataset Description

link to dataset : https://www.kaggle.com/competitions/rossmann-store-sales/data

You are provided with historical sales data for 1,115 Rossmann stores. The task is to forecast the "Sales" column for the test set. Note that some stores in the dataset were temporarily closed for refurbishment.

Files train.csv - historical data including Sales

test.csv - historical data excluding Sales

sample_submission.csv - a sample submission file in the correct format

store.csv - supplemental information about the stores

Data fields

Most of the fields are self-explanatory. The following are descriptions for those that aren't.

Id - an Id that represents a (Store, Date) duple within the test set

Store - a unique Id for each store

Sales - the turnover for any given day (this is what you are predicting)

Customers - the number of customers on a given day

Open - an indicator for whether the store was open: 0 = closed, 1 = open

StateHoliday - indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = None

SchoolHoliday - indicates if the (Store, Date) was affected by the closure of public schools

StoreType - differentiates between 4 different store models: a, b, c, d

Assortment - describes an assortment level: a = basic, b = extra, c = extended

CompetitionDistance - distance in meters to the nearest competitor store

CompetitionOpenSince[Month/Year] - gives the approximate year and month of the time the nearest competitor was opened

Promo - indicates whether a store is running a promo on that day

Promo2 - Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not

participating, 1 = store is participating

Promo2Since[Year/Week] - describes the year and calendar week when the store started participating in Promo2

PromoInterval - describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store

```
sns.set_style('darkgrid')
plt.rcParams['font.size'] = 14
plt.rcParams['figure.figsize'] = (10, 6)
plt.rcParams['figure.facecolor'] = '#00000000'
import warnings
warnings.filterwarnings("ignore")
c_green = '#6DF10C'
c_yellow = '#F5DD0D'
c_cyan = '#0FFDEF'
c_blue = '#0141DE'
c_blue_light =  '#2775FD'
c_purple = '#FF0DE5'
c_green_dark = '#1BB200'
e = np.e

ross_df = pd.read_csv('csv\\train.csv')
store_df = pd.read_csv('csv\\store.csv')
test_df = pd.read_csv('csv\\test.csv')

ross_df
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo \
0	1	5	2015-07-31	5263	555	1	1
1	2	5	2015-07-31	6064	625	1	1
2	3	5	2015-07-31	8314	821	1	1
3	4	5	2015-07-31	13995	1498	1	1
4	5	5	2015-07-31	4822	559	1	1

```
...          ...         ...          ...    ...         ...   ...    ...
1017204   1111           2  2013-01-01      0           0     0      0
1017205   1112           2  2013-01-01      0           0     0      0
1017206   1113           2  2013-01-01      0           0     0      0
1017207   1114           2  2013-01-01      0           0     0      0
1017208   1115           2  2013-01-01      0           0     0      0

        StateHoliday  SchoolHoliday
0                  0              1
1                  0              1
2                  0              1
3                  0              1
4                  0              1
...              ...            ...
1017204            a              1
1017205            a              1
1017206            a              1
1017207            a              1
1017208            a              1

[1017209 rows x 9 columns]
```

store_df

```
      Store StoreType Assortment  CompetitionDistance  \
0         1         c          a               1270.0
1         2         a          a                570.0
2         3         a          a              14130.0
3         4         c          c                620.0
4         5         a          a              29910.0
...     ...       ...        ...                  ...
1110   1111         a          a               1900.0
1111   1112         c          c               1880.0
1112   1113         a          c               9260.0
1113   1114         a          c                870.0
1114   1115         d          c               5350.0

      CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                           9.0                    2008.0       0
1                          11.0                    2007.0       1
2                          12.0                    2006.0       1
3                           9.0                    2009.0       0
4                           4.0                    2015.0       0
...                         ...                       ...     ...
```

```
1110                            6.0                       2014.0        1
1111                            4.0                       2006.0        0
1112                            NaN                          NaN        0
1113                            NaN                          NaN        0
1114                            NaN                          NaN        1

      Promo2SinceWeek  Promo2SinceYear     PromoInterval
0                 NaN              NaN               NaN
1                13.0           2010.0   Jan,Apr,Jul,Oct
2                14.0           2011.0   Jan,Apr,Jul,Oct
3                 NaN              NaN               NaN
4                 NaN              NaN               NaN
...               ...              ...               ...
1110             31.0           2013.0   Jan,Apr,Jul,Oct
1111              NaN              NaN               NaN
1112              NaN              NaN               NaN
1113              NaN              NaN               NaN
1114             22.0           2012.0  Mar,Jun,Sept,Dec

[1115 rows x 10 columns]
```

```
merged_train_df = ross_df.merge(store_df,how='left' , on='Store')
merged_test_df = test_df.merge(store_df,how='left' , on='Store')

merged_train_df
```

```
         Store  DayOfWeek        Date  Sales  Customers  Open
Promo  \
0            1          5  2015-07-31   5263        555     1       1

1            2          5  2015-07-31   6064        625     1       1

2            3          5  2015-07-31   8314        821     1       1

3            4          5  2015-07-31  13995       1498     1       1

4            5          5  2015-07-31   4822        559     1       1

...        ...        ...         ...    ...        ...   ...     ...

1017204   1111          2  2013-01-01      0          0     0       0

1017205   1112          2  2013-01-01      0          0     0       0

1017206   1113          2  2013-01-01      0          0     0       0

1017207   1114          2  2013-01-01      0          0     0       0

1017208   1115          2  2013-01-01      0          0     0       0
```

```
        StateHoliday  SchoolHoliday StoreType Assortment
CompetitionDistance  \
0                  0              1         c          a
1270.0
1                  0              1         a          a
570.0
2                  0              1         a          a
14130.0
3                  0              1         c          c
620.0
4                  0              1         a          a
29910.0
...              ...            ...       ...        ...
...
1017204            a              1         a          a
1900.0
1017205            a              1         c          c
1880.0
1017206            a              1         a          c
9260.0
1017207            a              1         a          c
870.0
1017208            a              1         d          c
5350.0

         CompetitionOpenSinceMonth  CompetitionOpenSinceYear
Promo2  \
0                              9.0                    2008.0       0

1                             11.0                    2007.0       1

2                             12.0                    2006.0       1

3                              9.0                    2009.0       0

4                              4.0                    2015.0       0

...                            ...                       ...     ...

1017204                        6.0                    2014.0       1

1017205                        4.0                    2006.0       0

1017206                        NaN                       NaN       0

1017207                        NaN                       NaN       0

1017208                        NaN                       NaN       1
```

```
         Promo2SinceWeek  Promo2SinceYear     PromoInterval
0                    NaN              NaN               NaN
1                   13.0           2010.0   Jan,Apr,Jul,Oct
2                   14.0           2011.0   Jan,Apr,Jul,Oct
3                    NaN              NaN               NaN
4                    NaN              NaN               NaN
...                  ...              ...               ...
1017204             31.0           2013.0   Jan,Apr,Jul,Oct
1017205              NaN              NaN               NaN
1017206              NaN              NaN               NaN
1017207              NaN              NaN               NaN
1017208             22.0           2012.0  Mar,Jun,Sept,Dec

[1017209 rows x 18 columns]
```

merged_test_df

```
          Id  Store  DayOfWeek        Date  Open  Promo
StateHoliday  \
0          1      1          4  2015-09-17   1.0      1            0

1          2      3          4  2015-09-17   1.0      1            0

2          3      7          4  2015-09-17   1.0      1            0

3          4      8          4  2015-09-17   1.0      1            0

4          5      9          4  2015-09-17   1.0      1            0

...      ...    ...        ...         ...   ...    ...          ...

41083  41084   1111          6  2015-08-01   1.0      0            0

41084  41085   1112          6  2015-08-01   1.0      0            0

41085  41086   1113          6  2015-08-01   1.0      0            0

41086  41087   1114          6  2015-08-01   1.0      0            0

41087  41088   1115          6  2015-08-01   1.0      0            0


       SchoolHoliday StoreType Assortment  CompetitionDistance  \
0                  0         c          a               1270.0
1                  0         a          a              14130.0
2                  0         a          c              24000.0
3                  0         a          a               7520.0
4                  0         a          c               2030.0
...              ...       ...        ...                  ...
41083              0         a          a               1900.0
41084              0         c          c               1880.0
```

```
41085              0        a          c              9260.0
41086              0        a          c               870.0
41087              1        d          c              5350.0

       CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                            9.0                    2008.0       0
1                           12.0                    2006.0       1
2                            4.0                    2013.0       0
3                           10.0                    2014.0       0
4                            8.0                    2000.0       0
...                          ...                       ...     ...
41083                        6.0                    2014.0       1
41084                        4.0                    2006.0       0
41085                        NaN                       NaN       0
41086                        NaN                       NaN       0
41087                        NaN                       NaN       1

       Promo2SinceWeek  Promo2SinceYear     PromoInterval
0                  NaN              NaN               NaN
1                 14.0           2011.0   Jan,Apr,Jul,Oct
2                  NaN              NaN               NaN
3                  NaN              NaN               NaN
4                  NaN              NaN               NaN
...                ...              ...               ...
41083             31.0           2013.0   Jan,Apr,Jul,Oct
41084              NaN              NaN               NaN
41085              NaN              NaN               NaN
41086              NaN              NaN               NaN
41087             22.0           2012.0  Mar,Jun,Sept,Dec

[41088 rows x 17 columns]
```

Preprocessing and Feature Engineering

Let's take a look at the available columns, and figure out if we can create new columns or apply any useful transformations.

```
merged_train_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1017209 entries, 0 to 1017208
Data columns (total 18 columns):
 #   Column                     Non-Null Count    Dtype
---  ------                     --------------    -----
 0   Store                      1017209 non-null  int64
 1   DayOfWeek                  1017209 non-null  int64
 2   Date                       1017209 non-null  object
 3   Sales                      1017209 non-null  int64
 4   Customers                  1017209 non-null  int64
```

```
 5   Open                       1017209 non-null  int64
 6   Promo                      1017209 non-null  int64
 7   StateHoliday               1017209 non-null  object
 8   SchoolHoliday              1017209 non-null  int64
 9   StoreType                  1017209 non-null  object
 10  Assortment                 1017209 non-null  object
 11  CompetitionDistance        1014567 non-null  float64
 12  CompetitionOpenSinceMonth  693861 non-null   float64
 13  CompetitionOpenSinceYear   693861 non-null   float64
 14  Promo2                     1017209 non-null  int64
 15  Promo2SinceWeek            509178 non-null   float64
 16  Promo2SinceYear            509178 non-null   float64
 17  PromoInterval              509178 non-null   object
dtypes: float64(5), int64(8), object(5)
memory usage: 139.7+ MB
```

```python
def split_date(df):
    df['Date'] = pd.to_datetime(df['Date'])
    df['Year'] = df.Date.dt.year
    df['Month'] = df.Date.dt.month
    df['Day'] = df.Date.dt.day
    df['WeekOfYear'] = df.Date.dt.isocalendar().week

split_date(merged_train_df)
split_date(merged_test_df)

merged_train_df
```

```
         Store  DayOfWeek       Date  Sales  Customers  Open  Promo  \
0            1          5 2015-07-31   5263        555     1      1
1            2          5 2015-07-31   6064        625     1      1
2            3          5 2015-07-31   8314        821     1      1
3            4          5 2015-07-31  13995       1498     1      1
4            5          5 2015-07-31   4822        559     1      1
...        ...        ...        ...    ...        ...   ...    ...
1017204   1111          2 2013-01-01      0          0     0      0
1017205   1112          2 2013-01-01      0          0     0      0
1017206   1113          2 2013-01-01      0          0     0      0
1017207   1114          2 2013-01-01      0          0     0      0
1017208   1115          2 2013-01-01      0          0     0      0

        StateHoliday  SchoolHoliday StoreType  ...
CompetitionOpenSinceMonth  \
0                  0              1         c  ...
9.0
1                  0              1         a  ...
11.0
2                  0              1         a  ...
12.0
3                  0              1         c  ...
```

```
9.0
4                    0                1          a  ...
4.0
...                ...              ...        ...  ...
...
1017204              a                1          a  ...
6.0
1017205              a                1          c  ...
4.0
1017206              a                1          a  ...
NaN
1017207              a                1          a  ...
NaN
1017208              a                1          d  ...
NaN

         CompetitionOpenSinceYear  Promo2  Promo2SinceWeek
Promo2SinceYear  \
0                          2008.0       0              NaN
NaN
1                          2007.0       1             13.0
2010.0
2                          2006.0       1             14.0
2011.0
3                          2009.0       0              NaN
NaN
4                          2015.0       0              NaN
NaN
...                           ...     ...              ...
...
1017204                    2014.0       1             31.0
2013.0
1017205                    2006.0       0              NaN
NaN
1017206                       NaN       0              NaN
NaN
1017207                       NaN       0              NaN
NaN
1017208                       NaN       1             22.0
2012.0

            PromoInterval  Year Month  Day  WeekOfYear
0                     NaN  2015     7   31          31
1         Jan,Apr,Jul,Oct  2015     7   31          31
2         Jan,Apr,Jul,Oct  2015     7   31          31
3                     NaN  2015     7   31          31
4                     NaN  2015     7   31          31
...                   ...   ...   ...  ...         ...
1017204   Jan,Apr,Jul,Oct  2013     1    1           1
```

```
1017205               NaN  2013     1    1           1
1017206               NaN  2013     1    1           1
1017207               NaN  2013     1    1           1
1017208  Mar,Jun,Sept,Dec  2013     1    1           1

[1017209 rows x 22 columns]
```

```
merged_train_df[merged_train_df.Open == 0 ].Sales.value_counts()
```

```
Sales
0    172817
Name: count, dtype: int64
```

```
merged_train_df = merged_train_df[merged_train_df.Open == 1].copy()
```

```
merged_train_df
```

```
         Store  DayOfWeek       Date  Sales  Customers  Open  Promo  \
0            1          5 2015-07-31   5263        555     1      1
1            2          5 2015-07-31   6064        625     1      1
2            3          5 2015-07-31   8314        821     1      1
3            4          5 2015-07-31  13995       1498     1      1
4            5          5 2015-07-31   4822        559     1      1
...        ...        ...        ...    ...        ...   ...    ...
1016776    682          2 2013-01-01   3375        566     1      0
1016827    733          2 2013-01-01  10765       2377     1      0
1016863    769          2 2013-01-01   5035       1248     1      0
1017042    948          2 2013-01-01   4491       1039     1      0
1017190   1097          2 2013-01-01   5961       1405     1      0

        StateHoliday  SchoolHoliday StoreType  ...
CompetitionOpenSinceMonth  \
0                  0              1         c  ...
9.0
1                  0              1         a  ...
11.0
2                  0              1         a  ...
12.0
3                  0              1         c  ...
9.0
4                  0              1         a  ...
4.0
...              ...            ...       ...  ...
...
1016776            a              1         b  ...
9.0
1016827            a              1         b  ...
10.0
1016863            a              1         b  ...
NaN
1017042            a              1         b  ...
```

```
NaN
1017190            a              1         b  ...
3.0

         CompetitionOpenSinceYear  Promo2  Promo2SinceWeek
Promo2SinceYear  \
0                          2008.0       0              NaN
NaN
1                          2007.0       1             13.0
2010.0
2                          2006.0       1             14.0
2011.0
3                          2009.0       0              NaN
NaN
4                          2015.0       0              NaN
NaN
...                           ...     ...              ...
...
1016776                    2006.0       0              NaN
NaN
1016827                    1999.0       0              NaN
NaN
1016863                       NaN       1             48.0
2012.0
1017042                       NaN       0              NaN
NaN
1017190                    2002.0       0              NaN
NaN

           PromoInterval  Year Month  Day  WeekOfYear
0                    NaN  2015     7   31          31
1        Jan,Apr,Jul,Oct  2015     7   31          31
2        Jan,Apr,Jul,Oct  2015     7   31          31
3                    NaN  2015     7   31          31
4                    NaN  2015     7   31          31
...                  ...   ...   ...  ...         ...
1016776              NaN  2013     1    1           1
1016827              NaN  2013     1    1           1
1016863  Jan,Apr,Jul,Oct  2013     1    1           1
1017042              NaN  2013     1    1           1
1017190              NaN  2013     1    1           1

[844392 rows x 22 columns]
```

Competition

Next, we can use the columns `CompetitionOpenSince[Month/Year]` columns from `store_df` to compute the number of months for which a competitor has been open near the store.

```
def comp_months(df):
    df['CompetitionOpen'] = 12 * (df.Year - 
df.CompetitionOpenSinceYear) + (df.Month - 
df.CompetitionOpenSinceMonth)
    df['CompetitionOpen'] = df['CompetitionOpen'].map(lambda x: 0 if x 
< 0 else x).fillna(0)

comp_months(merged_train_df)
comp_months(merged_test_df)

merged_train_df
```

```
         Store  DayOfWeek       Date  Sales  Customers  Open  Promo  \
0            1          5 2015-07-31   5263        555     1      1   
1            2          5 2015-07-31   6064        625     1      1   
2            3          5 2015-07-31   8314        821     1      1   
3            4          5 2015-07-31  13995       1498     1      1   
4            5          5 2015-07-31   4822        559     1      1   
...        ...        ...        ...    ...        ...   ...    ...   
1016776    682          2 2013-01-01   3375        566     1      0   
1016827    733          2 2013-01-01  10765       2377     1      0   
1016863    769          2 2013-01-01   5035       1248     1      0   
1017042    948          2 2013-01-01   4491       1039     1      0   
1017190   1097          2 2013-01-01   5961       1405     1      0   

        StateHoliday  SchoolHoliday StoreType  ...  
CompetitionOpenSinceYear  \
0                  0              1         c  ...  
2008.0   
1                  0              1         a  ...  
2007.0   
2                  0              1         a  ...  
2006.0   
3                  0              1         c  ...  
2009.0   
4                  0              1         a  ...  
2015.0   
...              ...            ...       ...  ...  
...   
1016776            a              1         b  ...  
2006.0   
1016827            a              1         b  ...  
1999.0   
1016863            a              1         b  ...  
NaN   
1017042            a              1         b  ...  
NaN   
1017190            a              1         b  ...  
2002.0   
```

```
         Promo2  Promo2SinceWeek  Promo2SinceYear    PromoInterval
Year  \
0             0              NaN              NaN              NaN
2015
1             1             13.0           2010.0  Jan,Apr,Jul,Oct
2015
2             1             14.0           2011.0  Jan,Apr,Jul,Oct
2015
3             0              NaN              NaN              NaN
2015
4             0              NaN              NaN              NaN
2015
...         ...              ...              ...              ...   .
..
1016776       0              NaN              NaN              NaN
2013
1016827       0              NaN              NaN              NaN
2013
1016863       1             48.0           2012.0  Jan,Apr,Jul,Oct
2013
1017042       0              NaN              NaN              NaN
2013
1017190       0              NaN              NaN              NaN
2013

         Month Day  WeekOfYear  CompetitionOpen
0            7  31          31             82.0
1            7  31          31             92.0
2            7  31          31            103.0
3            7  31          31             70.0
4            7  31          31              3.0
...        ...  ..         ...              ...
1016776      1   1           1             76.0
1016827      1   1           1            159.0
1016863      1   1           1              0.0
1017042      1   1           1              0.0
1017190      1   1           1            130.0

[844392 rows x 23 columns]
```

```
merged_train_df.columns
```

```
Index(['Store', 'DayOfWeek', 'Date', 'Sales', 'Customers', 'Open',
'Promo',
       'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment',
       'CompetitionDistance', 'CompetitionOpenSinceMonth',
       'CompetitionOpenSinceYear', 'Promo2', 'Promo2SinceWeek',
       'Promo2SinceYear', 'PromoInterval', 'Year', 'Month', 'Day',
       'WeekOfYear', 'CompetitionOpen'],
      dtype='object')
```

```
input_cols = ['Store' , 'DayOfWeek' , 'Promo' , 'StateHoliday' ,
'Assortment',
              'CompetitionDistance' , 'Day' , 'Month' , 'Year' ,
'WeekOfYear' ,
              'Promo2'  , 'CompetitionOpen' , 'SchoolHoliday' ,
'StoreType' ]
target_col = 'Sales'

train_inputs = merged_train_df[input_cols].copy()
train_targets = merged_train_df[target_col].copy()

test_inputs = merged_test_df[input_cols].copy()

train_inputs
```

```
         Store  DayOfWeek  Promo StateHoliday Assortment
CompetitionDistance  \
0            1          5      1            0          a
1270.0
1            2          5      1            0          a
570.0
2            3          5      1            0          a
14130.0
3            4          5      1            0          c
620.0
4            5          5      1            0          a
29910.0
...        ...        ...    ...          ...        ...
...
1016776    682          2      0            a          a
150.0
1016827    733          2      0            a          b
860.0
1016863    769          2      0            a          b
840.0
1017042    948          2      0            a          b
1430.0
1017190   1097          2      0            a          b
720.0

         Day  Month  Year  WeekOfYear  Promo2  CompetitionOpen
SchoolHoliday  \
0         31      7  2015          31       0             82.0
1
1         31      7  2015          31       1             92.0
1
2         31      7  2015          31       1            103.0
1
3         31      7  2015          31       0             70.0
1
```

```
4          31      7  2015          31       0              3.0
1
...       ...    ...   ...         ...     ...              ...
...
1016776     1      1  2013           1       0             76.0
1
1016827     1      1  2013           1       0            159.0
1
1016863     1      1  2013           1       1              0.0
1
1017042     1      1  2013           1       0              0.0
1
1017190     1      1  2013           1       0            130.0
1

        StoreType
0               c
1               a
2               a
3               c
4               a
...           ...
1016776         b
1016827         b
1016863         b
1017042         b
1017190         b

[844392 rows x 14 columns]
```

```
train_inputs.columns
```

```
Index(['Store', 'DayOfWeek', 'Promo', 'StateHoliday', 'Assortment',
       'CompetitionDistance', 'Day', 'Month', 'Year', 'WeekOfYear',
'Promo2',
       'CompetitionOpen', 'SchoolHoliday', 'StoreType'],
      dtype='object')
```

```
numeric_cols = ['Store' , 'Promo' , 'SchoolHoliday' ,
            'CompetitionDistance' ,'CompetitionOpen' , 'Promo2' ,
            'Day' , 'Month' , 'Year'  , 'WeekOfYear']

categorical_cols = [ 'DayOfWeek' ,'StoreType' , 'Assortment']

train_inputs[numeric_cols].isna().sum().sort_values(ascending=False)
```

```
CompetitionDistance    2186
Store                     0
Promo                     0
SchoolHoliday             0
CompetitionOpen           0
```

```
Promo2                       0
Day                          0
Month                        0
Year                         0
WeekOfYear                   0
dtype: int64
```

```
test_inputs[numeric_cols].isna().sum().sort_values(ascending=False)
```

```
CompetitionDistance    96
Store                   0
Promo                   0
SchoolHoliday           0
CompetitionOpen         0
Promo2                  0
Day                     0
Month                   0
Year                    0
WeekOfYear              0
dtype: int64
```

Seems like competition distance is the only missing value, and we can simply fill it with the highest value (to indicate that competition is very far away).

```
max_distance = train_inputs.CompetitionDistance.max()

train_inputs['CompetitionDistance'].fillna(max_distance, inplace=True)
test_inputs['CompetitionDistance'].fillna(max_distance, inplace=True)

train_inputs[numeric_cols].isna().sum().sort_values(ascending=False)
```

```
Store                  0
Promo                  0
SchoolHoliday          0
CompetitionDistance    0
CompetitionOpen        0
Promo2                 0
Day                    0
Month                  0
Year                   0
WeekOfYear             0
dtype: int64
```

```
train_inputs
```

```
         Store  DayOfWeek  Promo StateHoliday Assortment
CompetitionDistance  \
0            1          5      1            0          a
1270.0
1            2          5      1            0          a
```

```
570.0
2              3           5       1             0           a
14130.0
3              4           5       1             0           c
620.0
4              5           5       1             0           a
29910.0
...          ...         ...     ...           ...         ...
...
1016776      682           2       0             a           a
150.0
1016827      733           2       0             a           b
860.0
1016863      769           2       0             a           b
840.0
1017042      948           2       0             a           b
1430.0
1017190     1097           2       0             a           b
720.0

         Day  Month  Year  WeekOfYear  Promo2  CompetitionOpen
SchoolHoliday  \
0         31      7  2015          31       0             82.0
1
1         31      7  2015          31       1             92.0
1
2         31      7  2015          31       1            103.0
1
3         31      7  2015          31       0             70.0
1
4         31      7  2015          31       0              3.0
1
...      ...    ...   ...         ...     ...              ...
...
1016776    1      1  2013           1       0             76.0
1
1016827    1      1  2013           1       0            159.0
1
1016863    1      1  2013           1       1              0.0
1
1017042    1      1  2013           1       0              0.0
1
1017190    1      1  2013           1       0            130.0
1

        StoreType
0               c
1               a
2               a
```

```
3                c
4                a
...            ...
1016776          b
1016827          b
1016863          b
1017042          b
1017190          b

[844392 rows x 14 columns]
```

```
from sklearn.preprocessing import MinMaxScaler , MaxAbsScaler ,
StandardScaler
from sklearn.preprocessing import OneHotEncoder , LabelEncoder ,
TargetEncoder
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from sklearn.metrics import accuracy_score,mean_squared_error
import joblib

def rmse(targets , predictions):
    return np.sqrt(np.mean(np.square(targets-predictions)))

def mse(targets , predictions):
    return np.mean(np.square(targets-predictions))

train_inputs
```

```
         Store  DayOfWeek  Promo StateHoliday Assortment
CompetitionDistance  \
0            1          5      1            0          a
1270.0
1            2          5      1            0          a
570.0
2            3          5      1            0          a
14130.0
3            4          5      1            0          c
620.0
4            5          5      1            0          a
29910.0
...        ...        ...    ...          ...        ...
...
1016776    682          2      0            a          a
150.0
```

```
1016827    733           2      0             a          b
860.0
1016863    769           2      0             a          b
840.0
1017042    948           2      0             a          b
1430.0
1017190   1097           2      0             a          b
720.0

         Day  Month  Year  WeekOfYear  Promo2  CompetitionOpen
SchoolHoliday  \
0         31      7  2015          31       0             82.0
1
1         31      7  2015          31       1             92.0
1
2         31      7  2015          31       1            103.0
1
3         31      7  2015          31       0             70.0
1
4         31      7  2015          31       0              3.0
1
...      ...    ...   ...         ...     ...              ...
...
1016776    1      1  2013           1       0             76.0
1
1016827    1      1  2013           1       0            159.0
1
1016863    1      1  2013           1       1              0.0
1
1017042    1      1  2013           1       0              0.0
1
1017190    1      1  2013           1       0            130.0
1

        StoreType
0               c
1               a
2               a
3               c
4               a
...           ...
1016776         b
1016827         b
1016863         b
1017042         b
1017190         b

[844392 rows x 14 columns]
```

```
class RossmanSalesPrediction:

    def __init__(self, train_inputs, train_target, test_inputs,
numeric_cols, categorical_cols):
        self.train_inputs = train_inputs
        self.train_target = train_target
        self.test_inputs = test_inputs
        self.numeric_cols = numeric_cols
        self.categorical_cols = categorical_cols

    def impute_missing_features(self,imputer,**params):
        try:
            self.imputer =
imputer(**params).fit(self.train_inputs[self.numeric_cols])
            self.imputer_stats = list(self.imputer.statistics_)
            self.train_inputs[self.numeric_cols] =
self.imputer.transform(self.train_inputs[self.numeric_cols])
        except:
            raise ValueError('No Missing Data in train_inputs')

    '''
    Takes Input train_inputs and test_inputs and scales them according
to the scaler is provided,
    Function returns a list which contains scaled train_inputs and
test_inputs [train_inputs , test_inputs]
    '''

    def scale_num_features(self,scaler):
        self.scaler =
scaler().fit(self.train_inputs[self.numeric_cols])
        self.train_inputs[self.numeric_cols] =
self.scaler.transform(self.train_inputs[self.numeric_cols])
        self.test_inputs[self.numeric_cols] =
self.scaler.transform(self.test_inputs[self.numeric_cols])
        return [self.train_inputs, self.test_inputs]

    '''
    Takes input: (train_inputs and test_inputs) then encodes them
according to the Encoder is provided,
    Function returns a list which contains encoded train_inputs,
test_inputs, [X_train, X_test]
    '''

    def encode_cat_features(self,encoder,**params):
        self.encoder =
encoder(sparse=False ,handle_unknown='ignore').fit(self.train_inputs[s
elf.categorical_cols])
        self.encoded_cols =
list(self.encoder.get_feature_names_out(self.categorical_cols))
        self.train_inputs[self.encoded_cols] =
```

```
self.encoder.transform(self.train_inputs[self.categorical_cols])
        self.test_inputs[self.encoded_cols] =
self.encoder.transform(self.test_inputs[self.categorical_cols])
        self.X_train = self.train_inputs[self.numeric_cols +
self.encoded_cols]
        self.X_test = self.test_inputs[self.numeric_cols +
self.encoded_cols]
        return [self.X_train, self.X_test , self.encoded_cols]

    '''
    Takes input: (split_size) then splits X_train according to the
split_size into X_val and val_target
    Function returns a list which contains
[self.X_train,self.X_val,self.train_target,self.val_target]
    '''

    def split_df(self,split_size):
        self.X_train, self.X_val, self.train_target, self.val_target =
train_test_split(self.X_train, self.train_target, test_size=split_size
, random_state=42)
        return
[self.X_train,self.X_val,self.train_target,self.val_target]

    def train_model(self,model,**params):
        self.model =
model(**params).fit(self.X_train,self.train_target)
        self.train_preds = self.model.predict(self.X_train)
        self.val_preds = self.model.predict(self.X_val)
        return [self.train_preds, self.val_preds, self.model]


    def get_accuracy_scores(self):
        self.train_rmse = rmse(self.train_target,self.train_preds)
        self.val_rmse = rmse(self.val_target,self.val_preds)
        self.r_2_score =
self.model.score(self.X_train,self.train_target)
        self.accuracy =
self.model.score(self.X_train,self.train_target)*100
        return {
            'Model': self.model,
            'train_rmse' : self.train_rmse,
            'val_rmse' : self.val_rmse,
            'r_2_score' : self.r_2_score,
            'Model Accuracy' : self.accuracy
        }

    def predict_input(self,new_input):
        self.input_df = pd.DataFrame([new_input])
        self.input_df[self.numeric_cols] =
self.imputer.transform(self.input_df[self.numeric_cols])
```

```
        self.input_df[self.numeric_cols] =
self.scaler.transform(self.input_df[self.numeric_cols])
        self.input_df[self.encoded_cols] =
self.encoder.transform(self.input_df[self.categorical_cols])
        X_input = self.input_df[self.numeric_cols + self.encoded_cols]
        prediction = self.model.predict(X_input)[0]
        return f'Prediction of {self.model} : {prediction} '

    def get_importance_df(self):
        self.importance_df = pd.DataFrame({
            'feature': self.X_train.columns,
            'importance': self.model.feature_importances_
                }).sort_values('importance', ascending=False)
        return self.importance_df

    def plot_importance(self):
        plt.figure(figsize=(10,10))
        plt.title('Feature Importance of model')
        sns.barplot(data=self.importance_df, x='importance',
y='feature')
        plt.show()

    def import_model(self):
        self.rossman_sales_model = {
            'model': self.model,
            'imputer': self.imputer,
            'scaler': self.scaler,
            'encoder': self.encoder,
            'numeric_cols': self.numeric_cols,
            'categorical_cols': self.categorical_cols,
            'encoded_cols': self.encoded_cols,
            'X_train' : self.X_train,
            'X_val' : self.X_val,
            'X_test' : self.X_test,
            'train_inputs' : self.train_inputs,
            'test_inputs': self.test_inputs,
            'train_target' : self.train_target,
            'val_target' : self.val_target,
            'imputer_statistics' : self.imputer_stats,
        }
        joblib.dump(self.rossman_sales_model, "files\\
rossman_sales_model.joblib")

    def list_params(self):
        return ['train_inputs',
            'test_inputs',
            'train_target' ,
            'val_target' ,
            'numeric_cols',
            'categorical_cols',
```

```
            'encoded_cols',
            'imputer_statistics',
            'X_train',
            'X_val',
            'X_test',
            'encoder',
            'scaler',
            'Model',
            'train_rmse',
            'val_rmse',
            'r_2_score',
            'Model Accuracy']

    def get_params(self):
        return {
            'train_inputs' : self.train_inputs,
            'test_inputs': self.test_inputs,
            'train_target' : self.train_target,
            'val_target' : self.val_target,
            'numeric_cols' : self.numeric_cols,
            'categorical_cols' : self.categorical_cols,
            'encoded_cols' : self.encoded_cols,
            'imputer_statistics' : self.imputer_stats,
            'X_train' : self.X_train,
            'X_val' : self.X_val,
            'X_test' : self.X_test,
            'encoder' : self.encoder,
            'scaler' : self.scaler,
            'Model' : self.model,
            'train_rmse' : self.train_rmse,
            'val_rmse' : self.val_rmse,
            'r_2_score' : self.r_2_score,
            'Model Accuracy' : self.accuracy

        }

model1 =
RossmanSalesPrediction(train_inputs,train_targets,test_inputs,numeric_
cols,categorical_cols)

imputer_params ={
    'strategy':'mean',
    'missing_values':np.nan
}

imputer =
model1.impute_missing_features(SimpleImputer,**imputer_params)
imputer

scaler = model1.scale_num_features(MinMaxScaler)
```

```
encoder = model1.encode_cat_features(OneHotEncoder)

splited = model1.split_df(0.1)

xgb_params = {
    'n_jobs':-1,
    'random_state': 42,
    'n_estimators':1000,
    'learning_rate':0.1,
    'max_depth':10,
    'subsample':0.9,
    'colsample_bytree':0.7
}

from xgboost import XGBRegressor

xgb = model1.train_model(XGBRegressor, **xgb_params)
xgb
```

```
[array([11869.245 ,  3592.7712,  8157.4316, ...,  6393.0723,  8789.75
,
        11786.23  ], dtype=float32),
 array([4679.055 , 8167.9214, 7206.459 , ..., 5404.9844, 4775.9844,
        7083.6616], dtype=float32),
 XGBRegressor(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.7, device=None,
early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None,
feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1,
max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=10, max_leaves=None,
              min_child_weight=None, missing=nan,
monotone_constraints=None,
              multi_strategy=None, n_estimators=1000, n_jobs=-1,
              num_parallel_tree=None, random_state=42, ...)]
```

```
xgb_scores = model1.get_accuracy_scores()
xgb_scores
```

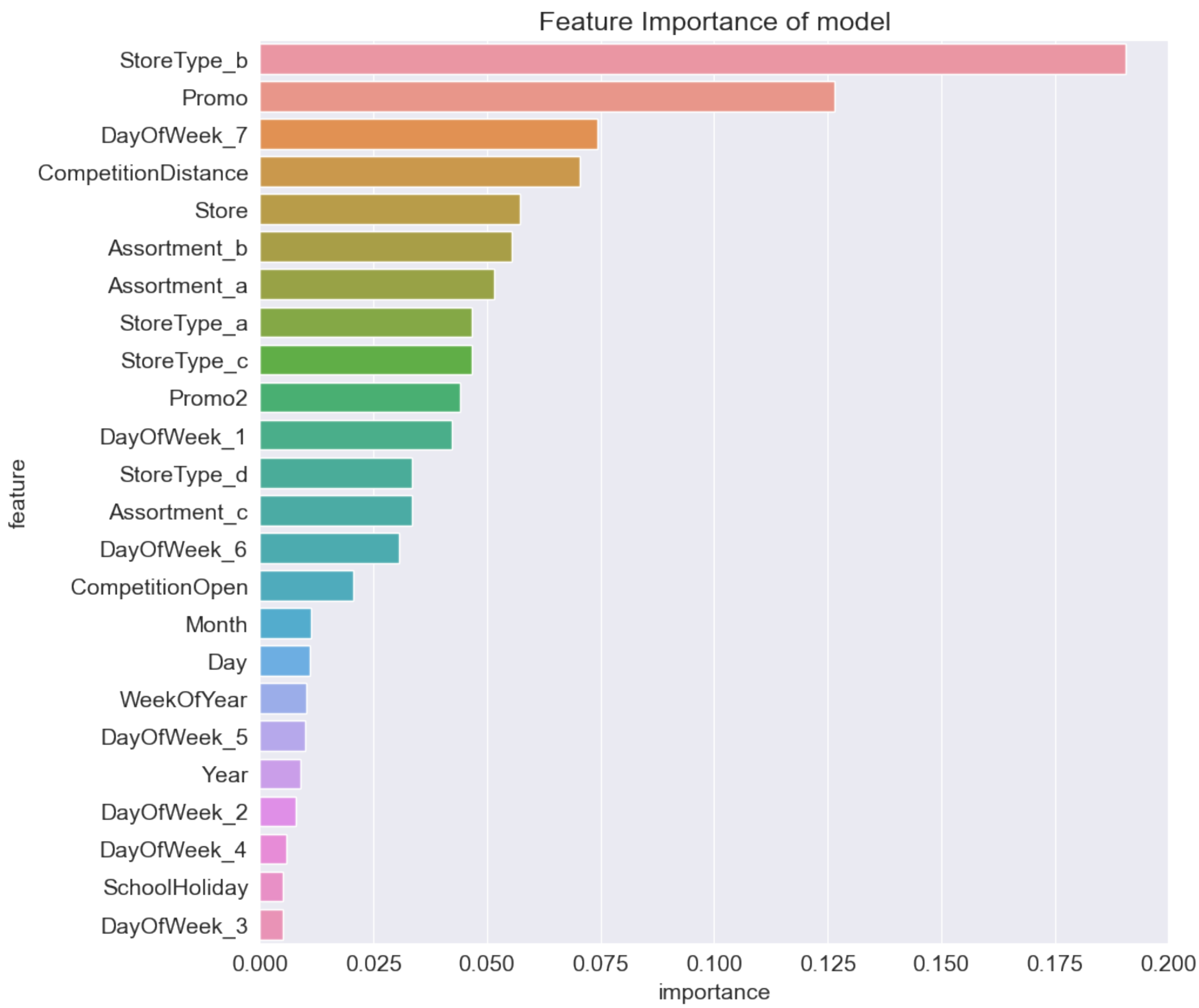
```
{'Model': XGBRegressor(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.7, device=None,
early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None,
feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1,
```

```
max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=10, max_leaves=None,
             min_child_weight=None, missing=nan,
monotone_constraints=None,
             multi_strategy=None, n_estimators=1000, n_jobs=-1,
             num_parallel_tree=None, random_state=42, ...),
 'train_rmse': 476.5882506580303,
 'val_rmse': 663.5009430476919,
 'r_2_score': 0.9764102211643134,
 'Model Accuracy': 97.64102211643134}
```

```
model1.get_importance_df()
```

	feature	importance
18	StoreType_b	0.190685
1	Promo	0.126486
16	DayOfWeek_7	0.074340
3	CompetitionDistance	0.070415
0	Store	0.057211
22	Assortment_b	0.055504
21	Assortment_a	0.051471
17	StoreType_a	0.046658
19	StoreType_c	0.046658
5	Promo2	0.044138
10	DayOfWeek_1	0.042414
20	StoreType_d	0.033610
23	Assortment_c	0.033391
15	DayOfWeek_6	0.030780
4	CompetitionOpen	0.020668
7	Month	0.011345
6	Day	0.010923
9	WeekOfYear	0.010193
14	DayOfWeek_5	0.009941
8	Year	0.008853
11	DayOfWeek_2	0.008038
13	DayOfWeek_4	0.005864
2	SchoolHoliday	0.005216
12	DayOfWeek_3	0.005199

```
model1.plot_importance()
```




```
new_input = {

}

model1.list_params()

['train_inputs',
 'test_inputs',
 'train_target',
 'val_target',
 'numeric_cols',
 'categorical_cols',
 'encoded_cols',
 'imputer_statistics',
 'X_train',
 'X_val',
 'X_test',
 'encoder',
```

```
 'scaler',
 'Model',
 'train_rmse',
 'val_rmse',
 'r_2_score',
 'Model Accuracy']

model_paarms= model1.get_params()
model_paarms.get('Model')

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.7, device=None,
early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1,
max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=10, max_leaves=None,
             min_child_weight=None, missing=nan,
monotone_constraints=None,
             multi_strategy=None, n_estimators=1000, n_jobs=-1,
             num_parallel_tree=None, random_state=42, ...)

model1.import_model()
```

Scaling Numeric Features

Another good practice is to scale numeric features to a small range of values e.g. $(0,1)$ or $(-1,1)$. Scaling numeric features ensures that no particular feature has a disproportionate impact on the model's loss. Optimization algorithms also work better in practice with smaller numbers.

The numeric columns in our dataset have varying ranges.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(train_inputs[numeric_cols])

MinMaxScaler()

train_inputs[numeric_cols] =
scaler.transform(train_inputs[numeric_cols])
test_inputs[numeric_cols] =
scaler.transform(test_inputs[numeric_cols])

train_inputs[numeric_cols].describe()
```

```
                  Store          Promo  SchoolHoliday
CompetitionDistance  \
count  844392.000000  844392.000000  844392.000000
844392.000000
mean        0.500380       0.446352       0.193580
0.074107
std         0.288808       0.497114       0.395103
0.113142
min         0.000000       0.000000       0.000000
0.000000
25%         0.250449       0.000000       0.000000
0.009098
50%         0.500000       0.000000       0.000000
0.030459
75%         0.750449       1.000000       0.000000
0.090849
max         1.000000       1.000000       1.000000
1.000000

       CompetitionOpen         Promo2            Day          Month  \
count    844392.000000  844392.000000  844392.000000  844392.000000
mean          0.030270       0.498684       0.494523       0.440522
std           0.047034       0.499999       0.289449       0.302176
min           0.000000       0.000000       0.000000       0.000000
25%           0.000000       0.000000       0.233333       0.181818
50%           0.011544       0.000000       0.500000       0.454545
75%           0.052670       1.000000       0.733333       0.636364
max           1.000000       1.000000       1.000000       1.000000

                   Year     WeekOfYear
count  844392.000000  844392.000000
mean        0.415969       0.444055
std         0.388630       0.282153
min         0.000000       0.000000
25%         0.000000       0.196078
50%         0.500000       0.431373
75%         0.500000       0.666667
max         1.000000       1.000000
```

```
test_inputs[numeric_cols]
```

```
          Store  Promo  SchoolHoliday  CompetitionDistance
CompetitionOpen  \
0      0.000000    1.0            0.0             0.016482
0.060606
1      0.001795    1.0            0.0             0.186050
0.075758
2      0.005386    1.0            0.0             0.316192
0.020924
3      0.006284    1.0            0.0             0.098892
```

```
0.007937
4      0.007181    1.0            0.0              0.026503
0.130592
...         ...    ...            ...                   ...
...
41083  0.996409    0.0            0.0              0.024789
0.010101
41084  0.997307    0.0            0.0              0.024525
0.080808
41085  0.998205    0.0            0.0              0.121835
0.000000
41086  0.999102    0.0            0.0              0.011208
0.000000
41087  1.000000    0.0            1.0              0.070280
0.000000

       Promo2       Day     Month  Year  WeekOfYear
0         0.0  0.533333  0.727273   1.0    0.725490
1         1.0  0.533333  0.727273   1.0    0.725490
2         0.0  0.533333  0.727273   1.0    0.725490
3         0.0  0.533333  0.727273   1.0    0.725490
4         0.0  0.533333  0.727273   1.0    0.725490
...       ...       ...       ...   ...         ...
41083     1.0  0.000000  0.636364   1.0    0.588235
41084     0.0  0.000000  0.636364   1.0    0.588235
41085     0.0  0.000000  0.636364   1.0    0.588235
41086     0.0  0.000000  0.636364   1.0    0.588235
41087     1.0  0.000000  0.636364   1.0    0.588235

[41088 rows x 10 columns]
```

Encoding Categorical Data

Since machine learning models can only be trained with numeric data, we need to convert categorical data to numbers. A common technique is to use one-hot encoding for categorical columns.

One hot encoding involves adding a new binary (0/1) column for each unique category of a categorical column.

```python
from sklearn.preprocessing import OneHotEncoder

train_inputs['StateHoliday'].unique()
```

```
array(['0', 'a', 'b', 'c', 0], dtype=object)
```

```python
train_inputs[categorical_cols]
```

```
         DayOfWeek StoreType Assortment
0                5         c          a
1                5         a          a
2                5         a          a
3                5         c          c
4                5         a          a
...            ...       ...        ...
1016776          2         b          a
1016827          2         b          b
1016863          2         b          b
1017042          2         b          b
1017190          2         b          b

[844392 rows x 3 columns]
```

```
encoder = OneHotEncoder(sparse=False,
handle_unknown='ignore').fit(train_inputs[categorical_cols])
encoded_cols = list(encoder.get_feature_names_out(categorical_cols))

train_inputs[encoded_cols] =
encoder.transform(train_inputs[categorical_cols])
test_inputs[encoded_cols] =
encoder.transform(test_inputs[categorical_cols])

X_train = train_inputs[numeric_cols + encoded_cols]
X_test = test_inputs[numeric_cols + encoded_cols]

X_train
```

```
            Store  Promo  SchoolHoliday  CompetitionDistance
CompetitionOpen  \
0        0.000000    1.0            1.0             0.016482
0.059163
1        0.000898    1.0            1.0             0.007252
0.066378
2        0.001795    1.0            1.0             0.186050
0.074315
3        0.002693    1.0            1.0             0.007911
0.050505
4        0.003591    1.0            1.0             0.394119
0.002165
...           ...    ...            ...                  ...
...
1016776  0.611311    0.0            1.0             0.001714
0.054834
1016827  0.657092    0.0            1.0             0.011076
0.114719
1016863  0.689408    0.0            1.0             0.010812
0.000000
1017042  0.850090    0.0            1.0             0.018592
```

```
0.000000
1017190  0.983842    0.0            1.0             0.009230
0.093795

         Promo2  Day     Month  Year  WeekOfYear  ...  DayOfWeek_5  \
0           0.0  1.0  0.545455   1.0    0.588235  ...          1.0
1           1.0  1.0  0.545455   1.0    0.588235  ...          1.0
2           1.0  1.0  0.545455   1.0    0.588235  ...          1.0
3           0.0  1.0  0.545455   1.0    0.588235  ...          1.0
4           0.0  1.0  0.545455   1.0    0.588235  ...          1.0
...         ...  ...       ...   ...         ...  ...          ...
1016776     0.0  0.0  0.000000   0.0    0.000000  ...          0.0
1016827     0.0  0.0  0.000000   0.0    0.000000  ...          0.0
1016863     1.0  0.0  0.000000   0.0    0.000000  ...          0.0
1017042     0.0  0.0  0.000000   0.0    0.000000  ...          0.0
1017190     0.0  0.0  0.000000   0.0    0.000000  ...          0.0

         DayOfWeek_6  DayOfWeek_7  StoreType_a  StoreType_b
StoreType_c  \
0                0.0          0.0          0.0          0.0
1.0
1                0.0          0.0          1.0          0.0
0.0
2                0.0          0.0          1.0          0.0
0.0
3                0.0          0.0          0.0          0.0
1.0
4                0.0          0.0          1.0          0.0
0.0
...              ...          ...          ...          ...          .
..
1016776          0.0          0.0          0.0          1.0
0.0
1016827          0.0          0.0          0.0          1.0
0.0
1016863          0.0          0.0          0.0          1.0
0.0
1017042          0.0          0.0          0.0          1.0
0.0
1017190          0.0          0.0          0.0          1.0
0.0

         StoreType_d  Assortment_a  Assortment_b  Assortment_c
0                0.0           1.0           0.0           0.0
1                0.0           1.0           0.0           0.0
2                0.0           1.0           0.0           0.0
3                0.0           0.0           0.0           1.0
4                0.0           1.0           0.0           0.0
...              ...           ...           ...           ...
1016776          0.0           1.0           0.0           0.0
```

```
1016827          0.0           0.0           1.0           0.0
1016863          0.0           0.0           1.0           0.0
1017042          0.0           0.0           1.0           0.0
1017190          0.0           0.0           1.0           0.0

[844392 rows x 24 columns]
```

```
from sklearn.model_selection import train_test_split

X_train, X_val, train_targets, val_targets = train_test_split(X_train,
train_targets, test_size=0.1)

X_train
```

```
           Store  Promo  SchoolHoliday  CompetitionDistance
CompetitionOpen  \
461690  0.776481    0.0            0.0             0.127373
0.000000
58084   0.093357    0.0            0.0             0.081355
0.000000
936213  0.357271    0.0            0.0             0.070411
0.004329
535814  0.254937    0.0            0.0             0.031514
0.000000
386157  0.197487    0.0            1.0             0.178138
0.007215
...          ...    ...            ...                  ...
...
210968  0.209156    0.0            0.0             0.057358
0.000000
418832  0.338420    0.0            0.0             0.027954
0.015873
538095  0.300718    1.0            0.0             0.002242
0.000000
791195  0.296230    0.0            1.0             0.008571
0.000000
768538  0.976661    1.0            1.0             0.068565
0.036797

        Promo2       Day     Month  Year  WeekOfYear  ...  DayOfWeek_5
\
461690     1.0  0.433333  0.363636   0.5    0.372549  ...          0.0

58084      1.0  0.266667  0.454545   1.0    0.450980  ...          0.0

936213     1.0  0.433333  0.181818   0.0    0.196078  ...          0.0

535814     0.0  0.233333  0.181818   0.5    0.176471  ...          0.0

386157     0.0  0.766667  0.545455   0.5    0.568627  ...          0.0
```

```
...         ...       ...       ...   ...         ...  ...           ...

210968      0.0  0.733333  0.000000   1.0    0.058824  ...           1.0

418832      0.0  0.666667  0.454545   0.5    0.470588  ...           0.0

538095      0.0  0.166667  0.181818   0.5    0.176471  ...           0.0

791195      1.0  0.700000  0.545455   0.0    0.568627  ...           0.0

768538      0.0  0.366667  0.636364   0.0    0.627451  ...           0.0


        DayOfWeek_6  DayOfWeek_7  StoreType_a  StoreType_b
StoreType_c  \
461690          0.0          0.0          0.0          0.0
0.0
58084           0.0          0.0          1.0          0.0
0.0
936213          0.0          0.0          1.0          0.0
0.0
535814          1.0          0.0          1.0          0.0
0.0
386157          0.0          0.0          0.0          0.0
0.0
...             ...          ...          ...          ...          ..
.
210968          0.0          0.0          0.0          0.0
0.0
418832          1.0          0.0          1.0          0.0
0.0
538095          0.0          0.0          1.0          0.0
0.0
791195          0.0          0.0          1.0          0.0
0.0
768538          0.0          0.0          0.0          0.0
0.0

        StoreType_d  Assortment_a  Assortment_b  Assortment_c
461690          1.0           1.0           0.0           0.0
58084           0.0           0.0           0.0           1.0
936213          0.0           1.0           0.0           0.0
535814          0.0           1.0           0.0           0.0
386157          1.0           0.0           0.0           1.0
...             ...           ...           ...           ...
210968          1.0           1.0           0.0           0.0
418832          0.0           0.0           0.0           1.0
538095          0.0           1.0           0.0           0.0
791195          0.0           0.0           0.0           1.0
768538          1.0           1.0           0.0           0.0
```

```
[759952 rows x 24 columns]

X_val

            Store  Promo  SchoolHoliday  CompetitionDistance
CompetitionOpen  \
167186  0.943447    1.0            0.0             0.066719
0.000000
896616  0.844704    0.0            0.0             0.090190
0.000000
460211  0.449731    0.0            0.0             0.002637
0.106061
430237  0.567325    0.0            1.0             0.153217
0.079365
552682  0.383303    1.0            0.0             0.038766
0.000000
...          ...    ...            ...                  ...
...
2628    0.357271    1.0            0.0             0.070411
0.024531
952982  0.396768    0.0            0.0             0.150053
0.062049
973111  0.449731    0.0            0.0             0.002637
0.095238
661273  0.774686    0.0            0.0             0.013186
0.009380
327890  0.874327    0.0            0.0             0.126714
0.000000

        Promo2       Day     Month  Year  WeekOfYear  ...  DayOfWeek_5
\
167186     1.0  0.100000  0.181818   1.0    0.176471  ...          0.0

896616     1.0  0.600000  0.272727   0.0    0.294118  ...          1.0

460211     1.0  0.466667  0.363636   0.5    0.372549  ...          0.0

430237     1.0  0.333333  0.454545   0.5    0.450980  ...          0.0

552682     1.0  0.666667  0.090909   0.5    0.137255  ...          1.0

...        ...       ...       ...   ...         ...  ...          ...

2628       1.0  0.933333  0.545455   1.0    0.588235  ...          0.0

952982     0.0  0.866667  0.090909   0.0    0.156863  ...          0.0

973111     1.0  0.266667  0.090909   0.0    0.098039  ...          0.0

661273     1.0  0.500000  0.909091   0.0    0.882353  ...          0.0
```

```
327890     1.0  0.800000  0.727273   0.5    0.745098  ...          0.0


        DayOfWeek_6  DayOfWeek_7  StoreType_a  StoreType_b
StoreType_c  \
167186          0.0          0.0          1.0          0.0
0.0
896616          0.0          0.0          0.0          0.0
0.0
460211          0.0          0.0          1.0          0.0
0.0
430237          0.0          0.0          0.0          0.0
0.0
552682          0.0          0.0          0.0          0.0
0.0
...             ...          ...          ...          ...          ..
.
2628            0.0          0.0          1.0          0.0
0.0
952982          0.0          0.0          0.0          0.0
0.0
973111          1.0          0.0          1.0          0.0
0.0
661273          1.0          0.0          1.0          0.0
0.0
327890          0.0          0.0          1.0          0.0
0.0

        StoreType_d  Assortment_a  Assortment_b  Assortment_c
167186          0.0           0.0           0.0           1.0
896616          1.0           0.0           0.0           1.0
460211          0.0           1.0           0.0           0.0
430237          1.0           1.0           0.0           0.0
552682          1.0           1.0           0.0           0.0
...             ...           ...           ...           ...
2628            0.0           1.0           0.0           0.0
952982          1.0           1.0           0.0           0.0
973111          0.0           1.0           0.0           0.0
661273          0.0           1.0           0.0           0.0
327890          0.0           0.0           0.0           1.0

[84440 rows x 24 columns]

def rmse(targets , predictions):
    return np.sqrt(np.mean(np.square(targets-predictions)))

def mse(targets , predictions):
    return np.mean(np.square(targets-predictions))
```

Gradient Boosting

We're now ready to train our gradient boosting machine (GBM) model. Here's how a GBM model works:

1. The average value of the target column and uses as an initial prediction every input.
2. The residuals (difference) of the predictions with the targets are computed.
3. A decision tree of limited depth is trained to **predict just the residuals** for each input.
4. Predictions from the decision tree are scaled using a parameter called the learning rate (this prevents overfitting)
5. Scaled predictions for the tree are added to the previous predictions to obtain the new and improved predictions.
6. Steps 2 to 5 are repeated to create new decision trees, each of which is trained to predict just the residuals from the previous prediction.

The term "gradient" refers to the fact that each decision tree is trained with the purpose of reducing the loss from the previous iteration (similar to gradient descent). The term "boosting" refers the general technique of training new models to improve the results of an existing model.

> **EXERCISE**: Can you describe in your own words how a gradient boosting machine is different from a random forest?

For a mathematical explanation of gradient boosting, check out the following resources:

- XGBoost Documentation
- Video Tutorials on StatQuest

Here's a visual representation of gradient boosting:

```python
from sklearn.metrics import r2_score ,accuracy_score

from xgboost import XGBRegressor
```

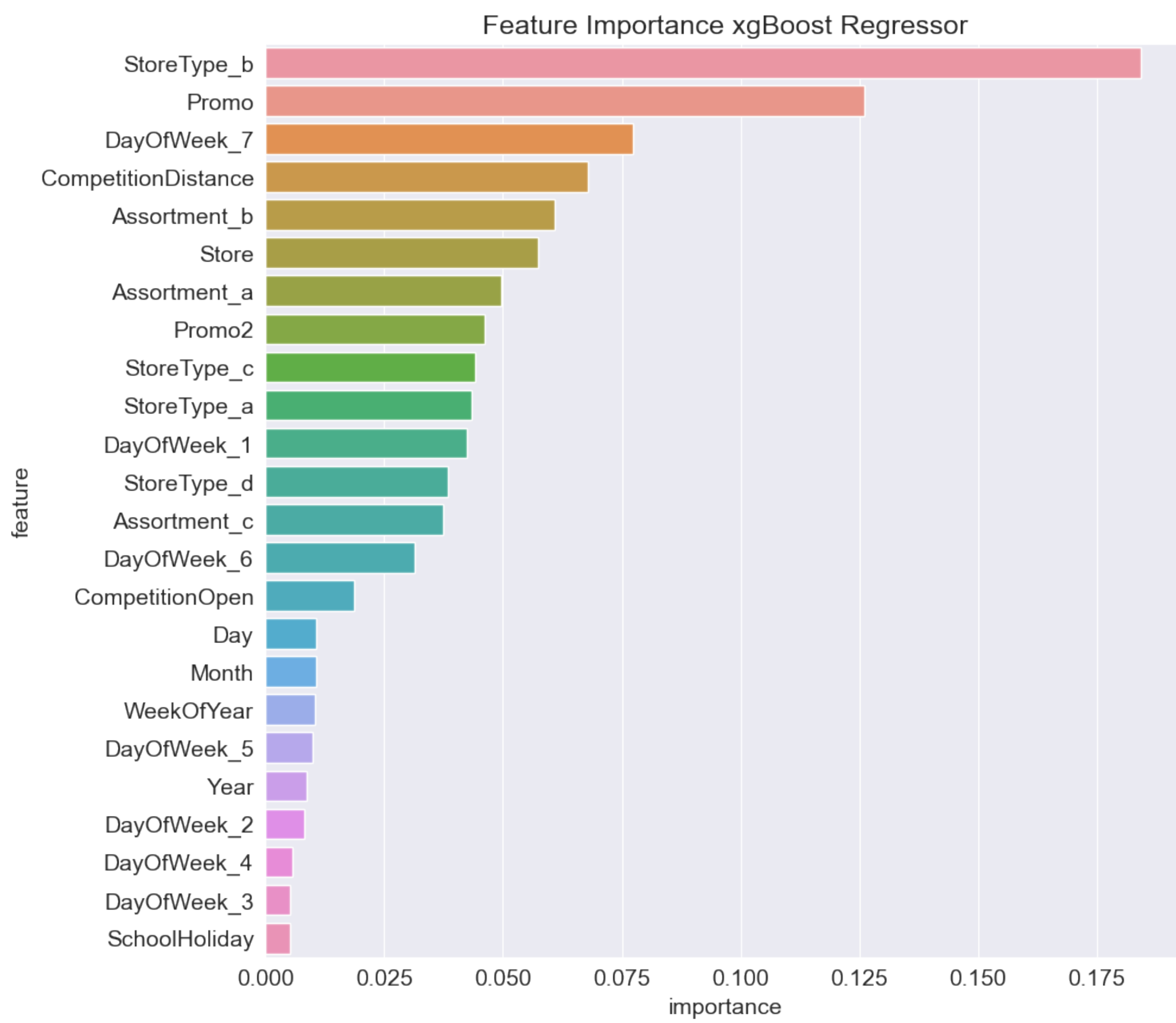
```
## Model Training and Hyper parameter Tuning
xgb_model = XGBRegressor(n_jobs=-1, random_state=42,
n_estimators=1000,
                     learning_rate=0.1, max_depth=10, subsample=0.9,
                     colsample_bytree=0.7)
# Fiting Model to Dataset -- X_train, train_targets
xgb_model.fit(X_train,train_targets)

# Making Prediction
train_preds = xgb_model.predict(X_train)
val_preds = xgb_model.predict(X_val)
r_2_score = xgb_model.score(X_train,train_targets)
accuracy = xgb_model.score(X_train,train_targets)*100

print(xgb_model)
print('Training RMSE : ',rmse(train_targets,train_preds))
print('Testing RMSE : ',rmse(val_targets,val_preds))
print('Model R_2 score : ',r_2_score)
print('Model Accuracy : ',accuracy)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.7, device=None,
early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1,
max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=10, max_leaves=None,
             min_child_weight=None, missing=nan,
monotone_constraints=None,
             multi_strategy=None, n_estimators=1000, n_jobs=-1,
             num_parallel_tree=None, random_state=42, ...)
Training RMSE :  478.61037819807126
Testing RMSE :  671.8424331144322
Model R_2 score :  0.9762329659464672
Model Accuracy :  97.62329659464672
```

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

importance_xgbregressor_df.head()
```

```
                feature  importance
18          StoreType_b    0.184327
1                 Promo    0.125972
```

```
16          DayOfWeek_7    0.077401
3   CompetitionDistance    0.067803
22         Assortment_b    0.060906

from sklearn.metrics import r2_score

plt.figure(figsize=(10,10))
plt.title('Feature Importance xgBoost Regressor')
sns.barplot(data=importance_xgbregressor_df, x='importance',
y='feature')
plt.show()
```



Decision Tree Regressor

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

Some advantages of decision trees are:

Simple to understand and to interpret. Trees can be visualized.

Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Some tree and algorithm combinations support missing values.

The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.

Able to handle both numerical and categorical data. However, the scikit-learn implementation does not support categorical variables for now. Other techniques are usually specialized in analyzing datasets that have only one type of variable. See algorithms for more information.

Able to handle multi-output problems.

Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.

Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.

Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.

Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.

Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.

The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.

There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.

Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree

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

```
?DecisionTreeRegressor
```

```
Init signature:
DecisionTreeRegressor(
    *,
    criterion='squared_error',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    ccp_alpha=0.0,
)
Docstring:
A decision tree regressor.

Read more in the :ref:`User Guide <tree>`.

Parameters
----------
criterion : {"squared_error", "friedman_mse", "absolute_error",
"poisson"}, default="squared_error"
    The function to measure the quality of a split. Supported criteria
    are "squared_error" for the mean squared error, which is equal to
    variance reduction as feature selection criterion and minimizes
the L2
    loss using the mean of each terminal node, "friedman_mse", which
uses
    mean squared error with Friedman's improvement score for potential
    splits, "absolute_error" for the mean absolute error, which
minimizes
    the L1 loss using the median of each terminal node, and "poisson"
which
    uses reduction in Poisson deviance to find splits.

    .. versionadded:: 0.18
```

```
       Mean Absolute Error (MAE) criterion.

    .. versionadded:: 0.24
        Poisson deviance criterion.

splitter : {"best", "random"}, default="best"
    The strategy used to choose the split at each node. Supported
    strategies are "best" to choose the best split and "random" to
choose
    the best random split.

max_depth : int, default=None
    The maximum depth of the tree. If None, then nodes are expanded
until
    all leaves are pure or until all leaves contain less than
    min_samples_split samples.

min_samples_split : int or float, default=2
    The minimum number of samples required to split an internal node:

    - If int, then consider `min_samples_split` as the minimum number.
    - If float, then `min_samples_split` is a fraction and
      `ceil(min_samples_split * n_samples)` are the minimum
      number of samples for each split.

    .. versionchanged:: 0.18
       Added float values for fractions.

min_samples_leaf : int or float, default=1
    The minimum number of samples required to be at a leaf node.
    A split point at any depth will only be considered if it leaves at
    least ``min_samples_leaf`` training samples in each of the left
and
    right branches.  This may have the effect of smoothing the model,
    especially in regression.

    - If int, then consider `min_samples_leaf` as the minimum number.
    - If float, then `min_samples_leaf` is a fraction and
      `ceil(min_samples_leaf * n_samples)` are the minimum
      number of samples for each node.

    .. versionchanged:: 0.18
       Added float values for fractions.

min_weight_fraction_leaf : float, default=0.0
    The minimum weighted fraction of the sum total of weights (of all
    the input samples) required to be at a leaf node. Samples have
    equal weight when sample_weight is not provided.

max_features : int, float or {"auto", "sqrt", "log2"}, default=None
```

```
    The number of features to consider when looking for the best
split:

    - If int, then consider `max_features` features at each split.
    - If float, then `max_features` is a fraction and
      `max(1, int(max_features * n_features_in_))` features are
considered at each
      split.
    - If "sqrt", then `max_features=sqrt(n_features)`.
    - If "log2", then `max_features=log2(n_features)`.
    - If None, then `max_features=n_features`.

    Note: the search for a split does not stop until at least one
    valid partition of the node samples is found, even if it requires
to
    effectively inspect more than ``max_features`` features.

random_state : int, RandomState instance or None, default=None
    Controls the randomness of the estimator. The features are always
    randomly permuted at each split, even if ``splitter`` is set to
    ``"best"``. When ``max_features < n_features``, the algorithm will
    select ``max_features`` at random at each split before finding the
best
    split among them. But the best found split may vary across
different
    runs, even if ``max_features=n_features``. That is the case, if
the
    improvement of the criterion is identical for several splits and
one
    split has to be selected at random. To obtain a deterministic
behaviour
    during fitting, ``random_state`` has to be fixed to an integer.
    See :term:`Glossary <random_state>` for details.

max_leaf_nodes : int, default=None
    Grow a tree with ``max_leaf_nodes`` in best-first fashion.
    Best nodes are defined as relative reduction in impurity.
    If None then unlimited number of leaf nodes.

min_impurity_decrease : float, default=0.0
    A node will be split if this split induces a decrease of the
impurity
    greater than or equal to this value.

    The weighted impurity decrease equation is the following::

        N_t / N * (impurity - N_t_R / N_t * right_impurity
                            - N_t_L / N_t * left_impurity)

    where ``N`` is the total number of samples, ``N_t`` is the number
```

```
of
    samples at the current node, ``N_t_L`` is the number of samples in
the
    left child, and ``N_t_R`` is the number of samples in the right
child.

    ``N``, ``N_t``, ``N_t_R`` and ``N_t_L`` all refer to the weighted
sum,
    if ``sample_weight`` is passed.

    .. versionadded:: 0.19

ccp_alpha : non-negative float, default=0.0
    Complexity parameter used for Minimal Cost-Complexity Pruning. The
    subtree with the largest cost complexity that is smaller than
    ``ccp_alpha`` will be chosen. By default, no pruning is performed.
See
    :ref:`minimal_cost_complexity_pruning` for details.

    .. versionadded:: 0.22

Attributes
----------
feature_importances_ : ndarray of shape (n_features,)
    The feature importances.
    The higher, the more important the feature.
    The importance of a feature is computed as the
    (normalized) total reduction of the criterion brought
    by that feature. It is also known as the Gini importance [4]_.

    Warning: impurity-based feature importances can be misleading for
    high cardinality features (many unique values). See
    :func:`sklearn.inspection.permutation_importance` as an
alternative.

max_features_ : int
    The inferred value of max_features.

n_features_in_ : int
    Number of features seen during :term:`fit`.

    .. versionadded:: 0.24

feature_names_in_ : ndarray of shape (`n_features_in_`,)
    Names of features seen during :term:`fit`. Defined only when `X`
    has feature names that are all strings.

    .. versionadded:: 1.0

n_outputs_ : int
```

```
    The number of outputs when ``fit`` is performed.

tree_ : Tree instance
    The underlying Tree object. Please refer to
    ``help(sklearn.tree._tree.Tree)`` for attributes of Tree object
and
    :ref:`sphx_glr_auto_examples_tree_plot_unveil_tree_structure.py`
    for basic usage of these attributes.

See Also
--------
DecisionTreeClassifier : A decision tree classifier.

Notes
-----
The default values for the parameters controlling the size of the
trees
(e.g. ``max_depth``, ``min_samples_leaf``, etc.) lead to fully grown
and
unpruned trees which can potentially be very large on some data sets.
To
reduce memory consumption, the complexity and size of the trees should
be
controlled by setting those parameter values.

References
----------

.. [1] https://en.wikipedia.org/wiki/Decision_tree_learning

.. [2] L. Breiman, J. Friedman, R. Olshen, and C. Stone,
"Classification
       and Regression Trees", Wadsworth, Belmont, CA, 1984.

.. [3] T. Hastie, R. Tibshirani and J. Friedman. "Elements of
Statistical
       Learning", Springer, 2009.

.. [4] L. Breiman, and A. Cutler, "Random Forests",

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Examples
--------
>>> from sklearn.datasets import load_diabetes
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.tree import DecisionTreeRegressor
>>> X, y = load_diabetes(return_X_y=True)
>>> regressor = DecisionTreeRegressor(random_state=0)
>>> cross_val_score(regressor, X, y, cv=10)
```

```
...                    # doctest: +SKIP
...
array([-0.39..., -0.46...,  0.02...,  0.06..., -0.50...,
       0.16...,  0.11..., -0.73..., -0.30..., -0.00...])
File:           c:\users\saket\appdata\local\programs\python\
python311\lib\site-packages\sklearn\tree\_classes.py
Type:           ABCMeta
Subclasses:     ExtraTreeRegressor
```

```python
## Model Training and Hyper parameter Tuning
decision_tree_model = DecisionTreeRegressor(random_state=12 ,
max_depth=12 , max_leaf_nodes=2**20 ,min_samples_split=15 )
# Fiting Model to Dataset -- X_train, train_targets
decision_tree_model.fit(X_train,train_targets)

# Making Prediction
train_preds = decision_tree_model.predict(X_train)
val_preds = decision_tree_model.predict(X_val)
r_2_score = decision_tree_model.score(X_train,train_targets)
accuracy = decision_tree_model.score(X_train,train_targets)*100

print(decision_tree_model)
print('Training RMSE : ',rmse(train_targets,train_preds))
print('Testing RMSE : ',rmse(val_targets,val_preds))
print('Model r_2 score : ',r_2_score)
print('Model Accuracy : ',accuracy)
```

```
DecisionTreeRegressor(max_depth=12, max_leaf_nodes=1048576,
                      min_samples_split=15, random_state=12)
Training RMSE :  2132.7798691120297
Testing RMSE :  2173.2814436015874
Model r_2 score :  0.5280425663493122
Model Accuracy :  52.80425663493122
```

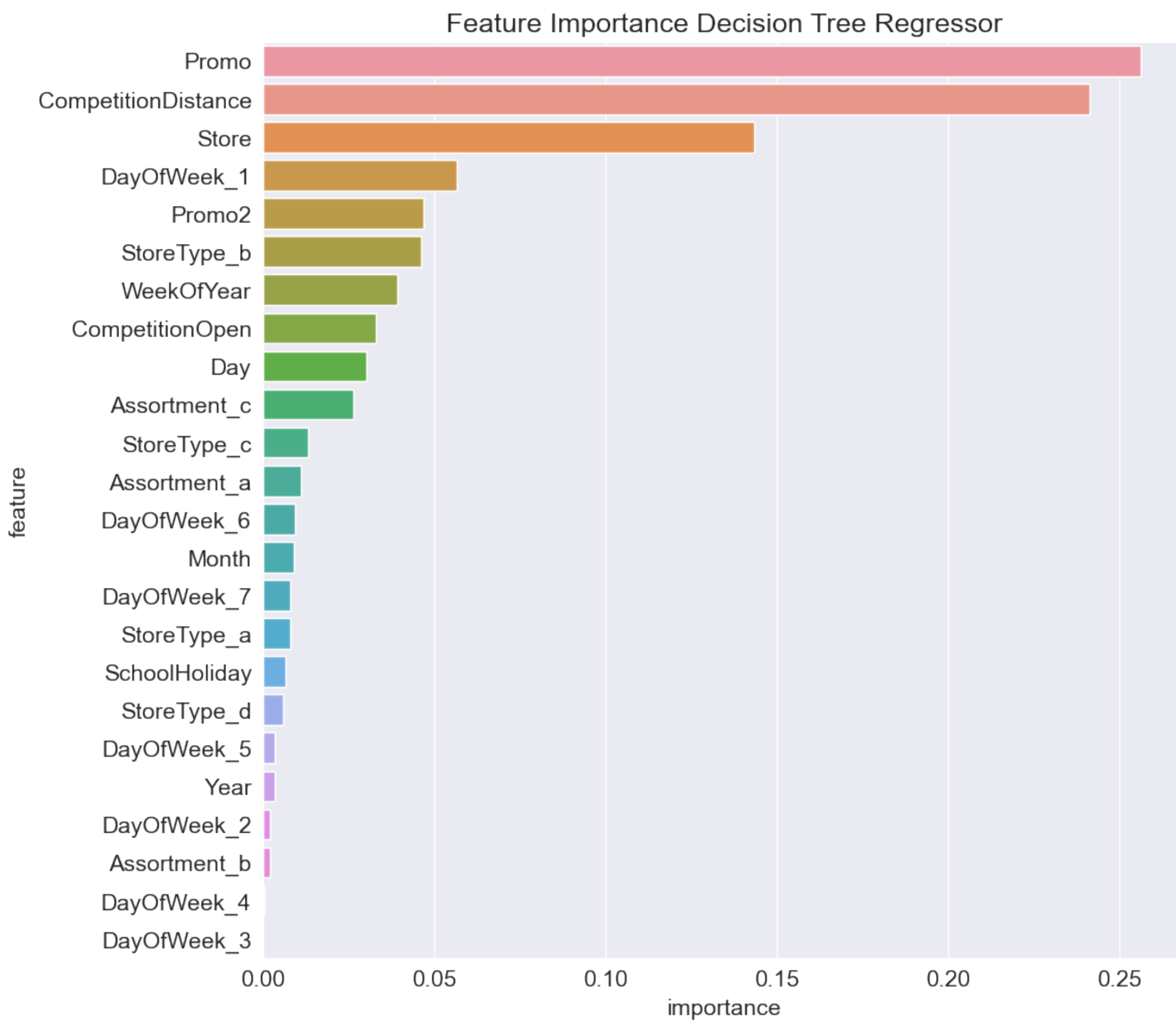
```python
decision_tree_model.feature_importances_
```

```
array([1.43192757e-01, 2.56351287e-01, 6.54033381e-03, 2.41440344e-01,
       3.27276444e-02, 4.66660024e-02, 3.02306257e-02, 8.82616074e-03,
       3.26795741e-03, 3.92584852e-02, 5.63993511e-02, 2.08982019e-03,
       4.11403487e-05, 1.47682764e-04, 3.47045767e-03, 9.28093906e-03,
       7.95077941e-03, 7.90502633e-03, 4.59897490e-02, 1.29668663e-02,
       5.79147029e-03, 1.09401279e-02, 2.06858053e-03, 2.64564109e-
02])
```

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

importance_decision_tree_df.head(5)
```

```
                feature  importance
1                 Promo    0.256351
3   CompetitionDistance    0.241440
0                 Store    0.143193
10          DayOfWeek_1    0.056399
5                Promo2    0.046666
```

```
plt.figure(figsize=(10,10))
plt.title('Feature Importance Decision Tree Regressor')
sns.barplot(data=importance_decision_tree_df, x='importance',
y='feature')
plt.show()
```



Saving and Loading Trained Models

We can save the parameters (weights and biases) of our trained model to disk, so that we needn't retrain the model from scratch each time we wish to use it. Along with the model, it's also important to save imputers, scalers, encoders and even column names. Anything that will be required while generating predictions using the model should be saved.

We can use the `joblib` module to save and load Python objects on the disk.

```
import joblib

rossman_xgboost_model = {
    'model': xgb_model,
    'scaler': scaler,
    'encoder': encoder,
    'input_cols': input_cols,
    'train_inputs'  : train_inputs,
    'train_targets' : train_targets,
    'test_inputs' : test_inputs,
    'X_train' : X_train,
    'X_test' : X_test,
    'target_col': target_col,
    'numeric_cols': numeric_cols,
    'categorical_cols': categorical_cols,
    'encoded_cols': encoded_cols
}

joblib.dump(rossman_xgboost_model, "files\\
rossman_xgboost_model.joblib")
```

```
['files\\rossman_xgboost_model.joblib']
```

```
joblib.load("files\\rossman_xgboost_model.joblib")
```

```
{'model': XGBRegressor(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.7, device=None,
early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None,
feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1,
max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=10, max_leaves=None,
              min_child_weight=None, missing=nan,
monotone_constraints=None,
              multi_strategy=None, n_estimators=1000, n_jobs=-1,
              num_parallel_tree=None, random_state=42, ...),
 'scaler': MinMaxScaler(),
 'encoder': OneHotEncoder(handle_unknown='ignore', sparse=False,
sparse_output=False),
 'input_cols': ['Store',
  'DayOfWeek',
  'Promo',
```

```
  'StateHoliday',
  'Assortment',
  'CompetitionDistance',
  'Day',
  'Month',
  'Year',
  'WeekOfYear',
  'Promo2',
  'CompetitionOpen',
  'SchoolHoliday',
  'StoreType'],
 'train_inputs':              Store  DayOfWeek  Promo StateHoliday
Assortment  \
 0        0.000000          5    1.0            0           a
 1        0.000898          5    1.0            0           a
 2        0.001795          5    1.0            0           a
 3        0.002693          5    1.0            0           c
 4        0.003591          5    1.0            0           a
 ...           ...        ...    ...          ...         ...
 1016776  0.611311          2    0.0            a           a
 1016827  0.657092          2    0.0            a           b
 1016863  0.689408          2    0.0            a           b
 1017042  0.850090          2    0.0            a           b
 1017190  0.983842          2    0.0            a           b

          CompetitionDistance  Day     Month  Year  WeekOfYear  ...  \
 0                   0.016482  1.0  0.545455   1.0    0.588235  ...
 1                   0.007252  1.0  0.545455   1.0    0.588235  ...
 2                   0.186050  1.0  0.545455   1.0    0.588235  ...
 3                   0.007911  1.0  0.545455   1.0    0.588235  ...
 4                   0.394119  1.0  0.545455   1.0    0.588235  ...
 ...                      ...  ...       ...   ...         ...  ...
 1016776             0.001714  0.0  0.000000   0.0    0.000000  ...
 1016827             0.011076  0.0  0.000000   0.0    0.000000  ...
 1016863             0.010812  0.0  0.000000   0.0    0.000000  ...
 1017042             0.018592  0.0  0.000000   0.0    0.000000  ...
 1017190             0.009230  0.0  0.000000   0.0    0.000000  ...

          DayOfWeek_5  DayOfWeek_6  DayOfWeek_7 StoreType_a
StoreType_b  \
 0                1.0          0.0          0.0         0.0
0.0
 1                1.0          0.0          0.0         1.0
0.0
 2                1.0          0.0          0.0         1.0
0.0
 3                1.0          0.0          0.0         0.0
0.0
 4                1.0          0.0          0.0         1.0
```

```
0.0
 ...               ...          ...          ...         ...           .
..
 1016776           0.0          0.0          0.0         0.0
1.0
 1016827           0.0          0.0          0.0         0.0
1.0
 1016863           0.0          0.0          0.0         0.0
1.0
 1017042           0.0          0.0          0.0         0.0
1.0
 1017190           0.0          0.0          0.0         0.0
1.0

          StoreType_c  StoreType_d  Assortment_a  Assortment_b
Assortment_c
 0                 1.0          0.0           1.0           0.0
0.0
 1                 0.0          0.0           1.0           0.0
0.0
 2                 0.0          0.0           1.0           0.0
0.0
 3                 1.0          0.0           0.0           0.0
1.0
 4                 0.0          0.0           1.0           0.0
0.0
 ...               ...          ...           ...           ...
...
 1016776           0.0          0.0           1.0           0.0
0.0
 1016827           0.0          0.0           0.0           1.0
0.0
 1016863           0.0          0.0           0.0           1.0
0.0
 1017042           0.0          0.0           0.0           1.0
0.0
 1017190           0.0          0.0           0.0           1.0
0.0

 [844392 rows x 28 columns],
 'train_targets': 461690     4360
 58084      5154
 936213     4369
 535814     5943
 386157     4837
           ...
 210968     8291
 418832     6057
 538095    15623
```

```
 791195     5429
 768538    13436
 Name: Sales, Length: 759952, dtype: int64,
 'test_inputs':           Store  DayOfWeek  Promo StateHoliday
Assortment  \
 0      0.000000          4    1.0            0          a
 1      0.001795          4    1.0            0          a
 2      0.005386          4    1.0            0          c
 3      0.006284          4    1.0            0          a
 4      0.007181          4    1.0            0          c
 ...         ...        ...    ...          ...        ...
 41083  0.996409          6    0.0            0          a
 41084  0.997307          6    0.0            0          c
 41085  0.998205          6    0.0            0          c
 41086  0.999102          6    0.0            0          c
 41087  1.000000          6    0.0            0          c

        CompetitionDistance       Day     Month  Year  WeekOfYear  ...
\
 0                 0.016482  0.533333  0.727273   1.0    0.725490  ...

 1                 0.186050  0.533333  0.727273   1.0    0.725490  ...

 2                 0.316192  0.533333  0.727273   1.0    0.725490  ...

 3                 0.098892  0.533333  0.727273   1.0    0.725490  ...

 4                 0.026503  0.533333  0.727273   1.0    0.725490  ...

 ...                    ...       ...       ...   ...         ...  ...

 41083             0.024789  0.000000  0.636364   1.0    0.588235  ...

 41084             0.024525  0.000000  0.636364   1.0    0.588235  ...

 41085             0.121835  0.000000  0.636364   1.0    0.588235  ...

 41086             0.011208  0.000000  0.636364   1.0    0.588235  ...

 41087             0.070280  0.000000  0.636364   1.0    0.588235  ...


        DayOfWeek_5  DayOfWeek_6  DayOfWeek_7 StoreType_a  StoreType_b
\
 0              0.0          0.0          0.0         0.0          0.0

 1              0.0          0.0          0.0         1.0          0.0

 2              0.0          0.0          0.0         1.0          0.0

 3              0.0          0.0          0.0         1.0          0.0
```

```
 4                0.0          0.0          0.0         1.0          0.0

 ...              ...          ...          ...         ...          ...

 41083            0.0          1.0          0.0         1.0          0.0

 41084            0.0          1.0          0.0         0.0          0.0

 41085            0.0          1.0          0.0         1.0          0.0

 41086            0.0          1.0          0.0         1.0          0.0

 41087            0.0          1.0          0.0         0.0          0.0


        StoreType_c  StoreType_d  Assortment_a  Assortment_b
Assortment_c
 0                1.0          0.0           1.0           0.0
0.0
 1                0.0          0.0           1.0           0.0
0.0
 2                0.0          0.0           0.0           0.0
1.0
 3                0.0          0.0           1.0           0.0
0.0
 4                0.0          0.0           0.0           0.0
1.0
 ...              ...          ...           ...           ...
...
 41083            0.0          0.0           1.0           0.0
0.0
 41084            1.0          0.0           0.0           0.0
1.0
 41085            0.0          0.0           0.0           0.0
1.0
 41086            0.0          0.0           0.0           0.0
1.0
 41087            0.0          1.0           0.0           0.0
1.0

 [41088 rows x 28 columns],
 'X_train':            Store  Promo  SchoolHoliday
CompetitionDistance  CompetitionOpen  \
 461690  0.776481    0.0            0.0             0.127373
0.000000
 58084   0.093357    0.0            0.0             0.081355
0.000000
 936213  0.357271    0.0            0.0             0.070411
0.004329
```

```
 535814  0.254937    0.0            0.0             0.031514
0.000000
 386157  0.197487    0.0            1.0             0.178138
0.007215
 ...          ...    ...            ...                  ...
...
 210968  0.209156    0.0            0.0             0.057358
0.000000
 418832  0.338420    0.0            0.0             0.027954
0.015873
 538095  0.300718    1.0            0.0             0.002242
0.000000
 791195  0.296230    0.0            1.0             0.008571
0.000000
 768538  0.976661    1.0            1.0             0.068565
0.036797

         Promo2       Day     Month  Year  WeekOfYear  ...
DayOfWeek_5  \
 461690     1.0  0.433333  0.363636   0.5    0.372549  ...
0.0
 58084      1.0  0.266667  0.454545   1.0    0.450980  ...
0.0
 936213     1.0  0.433333  0.181818   0.0    0.196078  ...
0.0
 535814     0.0  0.233333  0.181818   0.5    0.176471  ...
0.0
 386157     0.0  0.766667  0.545455   0.5    0.568627  ...
0.0
 ...        ...       ...       ...   ...         ...  ...            ..
.
 210968     0.0  0.733333  0.000000   1.0    0.058824  ...
1.0
 418832     0.0  0.666667  0.454545   0.5    0.470588  ...
0.0
 538095     0.0  0.166667  0.181818   0.5    0.176471  ...
0.0
 791195     1.0  0.700000  0.545455   0.0    0.568627  ...
0.0
 768538     0.0  0.366667  0.636364   0.0    0.627451  ...
0.0

         DayOfWeek_6  DayOfWeek_7  StoreType_a  StoreType_b
StoreType_c  \
 461690          0.0          0.0          0.0          0.0
0.0
 58084           0.0          0.0          1.0          0.0
0.0
 936213          0.0          0.0          1.0          0.0
```

```
0.0
 535814          1.0          0.0          1.0          0.0
0.0
 386157          0.0          0.0          0.0          0.0
0.0
 ...             ...          ...          ...          ...          .
..
 210968          0.0          0.0          0.0          0.0
0.0
 418832          1.0          0.0          1.0          0.0
0.0
 538095          0.0          0.0          1.0          0.0
0.0
 791195          0.0          0.0          1.0          0.0
0.0
 768538          0.0          0.0          0.0          0.0
0.0

         StoreType_d  Assortment_a  Assortment_b  Assortment_c
 461690          1.0           1.0           0.0           0.0
 58084           0.0           0.0           0.0           1.0
 936213          0.0           1.0           0.0           0.0
 535814          0.0           1.0           0.0           0.0
 386157          1.0           0.0           0.0           1.0
 ...             ...           ...           ...           ...
 210968          1.0           1.0           0.0           0.0
 418832          0.0           0.0           0.0           1.0
 538095          0.0           1.0           0.0           0.0
 791195          0.0           0.0           0.0           1.0
 768538          1.0           1.0           0.0           0.0

 [759952 rows x 24 columns],
 'X_test':           Store  Promo  SchoolHoliday  CompetitionDistance
CompetitionOpen  \
 0      0.000000    1.0            0.0             0.016482
0.060606
 1      0.001795    1.0            0.0             0.186050
0.075758
 2      0.005386    1.0            0.0             0.316192
0.020924
 3      0.006284    1.0            0.0             0.098892
0.007937
 4      0.007181    1.0            0.0             0.026503
0.130592
 ...         ...    ...            ...                  ...
...
 41083  0.996409    0.0            0.0             0.024789
0.010101
 41084  0.997307    0.0            0.0             0.024525
```

```
0.080808
 41085  0.998205    0.0            0.0             0.121835
0.000000
 41086  0.999102    0.0            0.0             0.011208
0.000000
 41087  1.000000    0.0            1.0             0.070280
0.000000

        Promo2       Day     Month  Year  WeekOfYear  ...  DayOfWeek_5
\
 0         0.0  0.533333  0.727273   1.0    0.725490  ...          0.0

 1         1.0  0.533333  0.727273   1.0    0.725490  ...          0.0

 2         0.0  0.533333  0.727273   1.0    0.725490  ...          0.0

 3         0.0  0.533333  0.727273   1.0    0.725490  ...          0.0

 4         0.0  0.533333  0.727273   1.0    0.725490  ...          0.0

 ...       ...       ...       ...   ...         ...  ...          ...

 41083     1.0  0.000000  0.636364   1.0    0.588235  ...          0.0

 41084     0.0  0.000000  0.636364   1.0    0.588235  ...          0.0

 41085     0.0  0.000000  0.636364   1.0    0.588235  ...          0.0

 41086     0.0  0.000000  0.636364   1.0    0.588235  ...          0.0

 41087     1.0  0.000000  0.636364   1.0    0.588235  ...          0.0


        DayOfWeek_6  DayOfWeek_7  StoreType_a  StoreType_b
StoreType_c  \
 0              0.0          0.0          0.0          0.0
1.0
 1              0.0          0.0          1.0          0.0
0.0
 2              0.0          0.0          1.0          0.0
0.0
 3              0.0          0.0          1.0          0.0
0.0
 4              0.0          0.0          1.0          0.0
0.0
 ...            ...          ...          ...          ...          ..
.
 41083          1.0          0.0          1.0          0.0
0.0
 41084          1.0          0.0          0.0          0.0
```

```
1.0
 41085          1.0          0.0          1.0          0.0
0.0
 41086          1.0          0.0          1.0          0.0
0.0
 41087          1.0          0.0          0.0          0.0
0.0

        StoreType_d  Assortment_a  Assortment_b  Assortment_c
 0              0.0           1.0           0.0           0.0
 1              0.0           1.0           0.0           0.0
 2              0.0           0.0           0.0           1.0
 3              0.0           1.0           0.0           0.0
 4              0.0           0.0           0.0           1.0
 ...            ...           ...           ...           ...
 41083          0.0           1.0           0.0           0.0
 41084          0.0           0.0           0.0           1.0
 41085          0.0           0.0           0.0           1.0
 41086          0.0           0.0           0.0           1.0
 41087          1.0           0.0           0.0           1.0

 [41088 rows x 24 columns],
 'target_col': 'Sales',
 'numeric_cols': ['Store',
  'Promo',
  'SchoolHoliday',
  'CompetitionDistance',
  'CompetitionOpen',
  'Promo2',
  'Day',
  'Month',
  'Year',
  'WeekOfYear'],
 'categorical_cols': ['DayOfWeek', 'StoreType', 'Assortment'],
 'encoded_cols': ['DayOfWeek_1',
  'DayOfWeek_2',
  'DayOfWeek_3',
  'DayOfWeek_4',
  'DayOfWeek_5',
  'DayOfWeek_6',
  'DayOfWeek_7',
  'StoreType_a',
  'StoreType_b',
  'StoreType_c',
  'StoreType_d',
  'Assortment_a',
  'Assortment_b',
  'Assortment_c']}
```

```
rossman_xgboost_model['train_inputs']
```

```
             Store  DayOfWeek  Promo StateHoliday Assortment  \
0        0.000000          5    1.0            0          a
1        0.000898          5    1.0            0          a
2        0.001795          5    1.0            0          a
3        0.002693          5    1.0            0          c
4        0.003591          5    1.0            0          a
...           ...        ...    ...          ...        ...
1016776  0.611311          2    0.0            a          a
1016827  0.657092          2    0.0            a          b
1016863  0.689408          2    0.0            a          b
1017042  0.850090          2    0.0            a          b
1017190  0.983842          2    0.0            a          b

         CompetitionDistance  Day     Month  Year  WeekOfYear  ...  \
0                   0.016482  1.0  0.545455   1.0    0.588235  ...
1                   0.007252  1.0  0.545455   1.0    0.588235  ...
2                   0.186050  1.0  0.545455   1.0    0.588235  ...
3                   0.007911  1.0  0.545455   1.0    0.588235  ...
4                   0.394119  1.0  0.545455   1.0    0.588235  ...
...                      ...  ...       ...   ...         ...  ...
1016776             0.001714  0.0  0.000000   0.0    0.000000  ...
1016827             0.011076  0.0  0.000000   0.0    0.000000  ...
1016863             0.010812  0.0  0.000000   0.0    0.000000  ...
1017042             0.018592  0.0  0.000000   0.0    0.000000  ...
1017190             0.009230  0.0  0.000000   0.0    0.000000  ...

         DayOfWeek_5  DayOfWeek_6  DayOfWeek_7 StoreType_a
StoreType_b  \
0                1.0          0.0          0.0         0.0
0.0
1                1.0          0.0          0.0         1.0
0.0
2                1.0          0.0          0.0         1.0
0.0
3                1.0          0.0          0.0         0.0
0.0
4                1.0          0.0          0.0         1.0
0.0
...              ...          ...          ...         ...          ..
.
1016776          0.0          0.0          0.0         0.0
1.0
1016827          0.0          0.0          0.0         0.0
1.0
1016863          0.0          0.0          0.0         0.0
1.0
1017042          0.0          0.0          0.0         0.0
1.0
1017190          0.0          0.0          0.0         0.0
1.0
```

```
         StoreType_c  StoreType_d  Assortment_a  Assortment_b
Assortment_c
0                1.0          0.0           1.0           0.0
0.0
1                0.0          0.0           1.0           0.0
0.0
2                0.0          0.0           1.0           0.0
0.0
3                1.0          0.0           0.0           0.0
1.0
4                0.0          0.0           1.0           0.0
0.0
...              ...          ...           ...           ...
...
1016776          0.0          0.0           1.0           0.0
0.0
1016827          0.0          0.0           0.0           1.0
0.0
1016863          0.0          0.0           0.0           1.0
0.0
1017042          0.0          0.0           0.0           1.0
0.0
1017190          0.0          0.0           0.0           1.0
0.0

[844392 rows x 28 columns]
```

```
test_preds = rossman_xgboost_model['model'].predict(X_test)

rossman_xgboost_model['model']
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.7, device=None,
early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1,
max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=10, max_leaves=None,
             min_child_weight=None, missing=nan,
monotone_constraints=None,
             multi_strategy=None, n_estimators=1000, n_jobs=-1,
             num_parallel_tree=None, random_state=42, ...)
```

```
test_preds
```

```
array([ 3761.39  ,  7738.725 ,  9194.781 , ...,  7101.3135,
23356.654 ,
        7592.986 ], dtype=float32)
```

```python
submisson_df =pd.read_csv('csv\\sample_submission.csv')

submisson_df['Sales'] = test_preds

submisson_df
```

```
          Id         Sales
0          1   3761.389893
1          2   7738.725098
2          3   9194.781250
3          4   6687.968750
4          5   7371.160645
...      ...           ...
41083  41084   3162.992920
41084  41085   8253.756836
41085  41086   7101.313477
41086  41087  23356.654297
41087  41088   7592.985840

[41088 rows x 2 columns]
```

```python
submisson_df.to_csv('csv\\submisson_df.csv' , index=False)
```