[8]:
```
#restart the kernel after installation
!pip install numpy pandas-profiling matplotlib seaborn --quiet
```

[10]:
```
!pip install jovian opendatasets xgboost graphviz lightgbm scikit-learn xgboost lightgbm --upgrade --quiet
```

DOWNLOADING THE DATA:

[11]:
```
import os
import opendatasets as od
import pandas as pd
pd.set_option("display.max_columns", 120)
pd.set_option("display.max_rows", 120)
```

[12]:
```
od.download('https://www.kaggle.com/c/rossmann-store-sales')
```

```
Skipping, found downloaded files in ".\rossmann-store-sales" (use force=True to force download)
```

[13]:
```
os.listdir('rossmann-store-sales')
```

[13]:
```
['sample_submission.csv', 'store.csv', 'test.csv', 'train.csv']
```

[14]:
```
ross_df = pd.read_csv('./rossmann-store-sales/train.csv', low_memory=False)
store_df = pd.read_csv('./rossmann-store-sales/store.csv')
test_df = pd.read_csv('./rossmann-store-sales/test.csv')
submission_df = pd.read_csv('./rossmann-store-sales/sample_submission.csv')
```

[15]:
```
ross_df
```

[15]:

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

1017209 rows × 9 columns

[16]:
```
test_df
```

[16]:

	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
0	1	1	4	2015-09-17	1.0	1	0	0
1	2	3	4	2015-09-17	1.0	1	0	0
2	3	7	4	2015-09-17	1.0	1	0	0
3	4	8	4	2015-09-17	1.0	1	0	0
4	5	9	4	2015-09-17	1.0	1	0	0
...	...	...	...	...	...	...	...	...
41083	41084	1111	6	2015-08-01	1.0	0	0	0
41084	41085	1112	6	2015-08-01	1.0	0	0	0
41085	41086	1113	6	2015-08-01	1.0	0	0	0
41086	41087	1114	6	2015-08-01	1.0	0	0	0
41087	41088	1115	6	2015-08-01	1.0	0	0	1

41088 rows × 8 columns

[17]:
```
store_df
```

[17]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	Prom
0	1	c	a	1270.0	9.0	2008.0	0	NaN	NaN	
1	2	a	a	570.0	11.0	2007.0	1	13.0	2010.0	Jan,/

2	3	a	a	14130.0	12.0	2006.0	1	14.0	2011.0	Jan,A
3	4	c	c	620.0	9.0	2009.0	0	NaN	NaN	
4	5	a	a	29910.0	4.0	2015.0	0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
1110	1111	a	a	1900.0	6.0	2014.0	1	31.0	2013.0	Jan,A
1111	1112	c	c	1880.0	4.0	2006.0	0	NaN	NaN	
1112	1113	a	c	9260.0	NaN	NaN	0	NaN	NaN	
1113	1114	a	c	870.0	NaN	NaN	0	NaN	NaN	
1114	1115	d	c	5350.0	NaN	NaN	1	22.0	2012.0	Mar,Jul

1115 rows × 10 columns

[18]:
```
submission_df
```

[18]:

	Id	Sales
0	1	0
1	2	0
2	3	0
3	4	0
4	5	0
...	...	...
41083	41084	0
41084	41085	0
41085	41086	0
41086	41087	0
41087	41088	0

41088 rows × 2 columns

[19]:
```
merged_df = ross_df.merge(store_df, how = 'left', on = 'Store' )
merged_test_df = test_df.merge(store_df, how = 'left', on = 'Store')
```

[20]:
```
merged_df
```

[20]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceM
0	1	5	2015-07-31	5263	555	1	1	0	1	c	a	1270.0	
1	2	5	2015-07-31	6064	625	1	1	0	1	a	a	570.0	
2	3	5	2015-07-31	8314	821	1	1	0	1	a	a	14130.0	
3	4	5	2015-07-31	13995	1498	1	1	0	1	c	c	620.0	
4	5	5	2015-07-31	4822	559	1	1	0	1	a	a	29910.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1017204	1111	2	2013-01-01	0	0	0	0	a	1	a	a	1900.0	
1017205	1112	2	2013-01-01	0	0	0	0	a	1	c	c	1880.0	
1017206	1113	2	2013-01-01	0	0	0	0	a	1	a	c	9260.0	
1017207	1114	2	2013-01-01	0	0	0	0	a	1	a	c	870.0	
1017208	1115	2	2013-01-01	0	0	0	0	a	1	d	c	5350.0	

1017209 rows × 18 columns

[21]:
```
merged_test_df
```

[21]:

	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	Compet
0	1	1	4	2015-09-17	1.0	1	0	0	c	a	1270.0	9.0	
1	2	3	4	2015-09-17	1.0	1	0	0	a	a	14130.0	12.0	

2	3	7	4	2015-09-17	1.0	1	0	0	a	c	24000.0	4.0
3	4	8	4	2015-09-17	1.0	1	0	0	a	a	7520.0	10.0
4	5	9	4	2015-09-17	1.0	1	0	0	a	c	2030.0	8.0
...	...	...	...	...	...	...	...	...	...	...	...	...
41083	41084	1111	6	2015-08-01	1.0	0	0	0	a	a	1900.0	6.0
41084	41085	1112	6	2015-08-01	1.0	0	0	0	c	c	1880.0	4.0
41085	41086	1113	6	2015-08-01	1.0	0	0	0	a	c	9260.0	NaN
41086	41087	1114	6	2015-08-01	1.0	0	0	0	a	c	870.0	NaN
41087	41088	1115	6	2015-08-01	1.0	0	0	1	d	c	5350.0	NaN

41088 rows × 17 columns

PREPROCESSING AND FEATURE ENGINEERING:

[22]:
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 147.5+ MB
```

DATE:

[23]:
```
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
```

[25]:
```
split_date(merged_df)
split_date(merged_test_df)
```

[26]:
```
merged_df
```

[26]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceM
0	1	5	2015-07-31	5263	555	1	1	0	1	c	a	1270.0	
1	2	5	2015-07-31	6064	625	1	1	0	1	a	a	570.0	
2	3	5	2015-07-31	8314	821	1	1	0	1	a	a	14130.0	
3	4	5	2015-07-31	13995	1498	1	1	0	1	c	c	620.0	
4	5	5	2015-07-31	4822	559	1	1	0	1	a	a	29910.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1017204	1111	2	2013-01-01	0	0	0	0	a	1	a	a	1900.0	

1017205	1112	2	2013-01-01	0	0	0	0	a	1	c	c	1880.0
1017206	1113	2	2013-01-01	0	0	0	0	a	1	a	c	9260.0
1017207	1114	2	2013-01-01	0	0	0	0	a	1	a	c	870.0
1017208	1115	2	2013-01-01	0	0	0	0	a	1	d	c	5350.0

1017209 rows × 22 columns

STORE OPEN/ CLOSED:

[27]:
```
merged_df[merged_df.Open == 0].Sales.value_counts()
```

[27]:
```
0    172817
Name: Sales, dtype: int64
```

[28]:
```
merged_df = merged_df[merged_df.Open == 1].copy()
```

[29]:
```
merged_df.sample(10)
```

[29]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMc
11825	676	2	2015-07-21	9000	2248	1	0	0	0	b	b	1410.0	
281760	596	4	2014-11-13	4363	625	1	1	0	0	c	a	290.0	
291097	582	1	2014-11-03	10234	1102	1	1	0	0	a	a	120.0	
525029	650	2	2014-03-18	7187	684	1	1	0	0	a	a	1420.0	
227592	133	4	2015-01-08	6029	632	1	1	0	0	a	a	270.0	
248471	1040	5	2014-12-19	12121	1027	1	1	0	0	a	a	4030.0	
34874	310	2	2015-06-30	13045	930	1	1	0	0	a	c	2290.0	
338615	382	6	2014-09-13	5407	533	1	0	0	0	c	c	26130.0	
449844	170	6	2014-05-24	3857	442	1	0	0	0	a	a	1070.0	
607410	521	5	2014-01-03	5116	452	1	0	0	1	d	a	18610.0	

COMPETITION:

[30]:
```
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)
```

[32]:
```
comp_months(merged_df)
comp_months(merged_test_df)
```

[33]:
```
merged_df
```

[33]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceM
0	1	5	2015-07-31	5263	555	1	1	0	1	c	a	1270.0	
1	2	5	2015-07-31	6064	625	1	1	0	1	a	a	570.0	
2	3	5	2015-07-31	8314	821	1	1	0	1	a	a	14130.0	
3	4	5	2015-07-31	13995	1498	1	1	0	1	c	c	620.0	
4	5	5	2015-07-31	4822	559	1	1	0	1	a	a	29910.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1016776	682	2	2013-01-01	3375	566	1	0	a	1	b	a	150.0	
1016827	733	2	2013-01-01	10765	2377	1	0	a	1	b	b	860.0	

1016863	769	2	2013-01-01	5035	1248	1	0	a	1	b	b	840.0
1017042	948	2	2013-01-01	4491	1039	1	0	a	1	b	b	1430.0
1017190	1097	2	2013-01-01	5961	1405	1	0	a	1	b	b	720.0

844392 rows × 23 columns

ADDITIONAL PROMOTION:

[42]:
```
def check_promo_month(row):
    month2str = {1:'Jan', 2:'Feb', 3:'Mar', 4:'Apr', 5:'May', 6:'Jun',
                 7:'Jul', 8:'Aug', 9:'Sept', 10:'Oct', 11:'Nov', 12:'Dec'}
    try:
        months = (row['PromoInterval'] or '').split(',')
        if row['Promo2Open'] and month2str[row['Month']] in months:
            return 1
        else:
            return 0
    except Exception:
        return 0

def promo_cols(df):
    # Months since Promo2 was open
    df['Promo2Open'] = 12 * (df.Year - df.Promo2SinceYear) +  (df.WeekofYear - df.Promo2SinceWeek)*7/30.5
    df['Promo2Open'] = df['Promo2Open'].map(lambda x: 0 if x < 0 else x).fillna(0) * df['Promo2']
    # Whether a new round of promotions was started in the current month
    df['IsPromo2Month'] = df.apply(check_promo_month, axis=1) * df['Promo2']
```

[44]:
```
promo_cols(merged_df)
promo_cols(merged_test_df)
```

[45]:
```
merged_df
```

[45]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceM
0	1	5	2015-07-31	5263	555	1	1	0	1	c	a	1270.0	
1	2	5	2015-07-31	6064	625	1	1	0	1	a	a	570.0	
2	3	5	2015-07-31	8314	821	1	1	0	1	a	a	14130.0	
3	4	5	2015-07-31	13995	1498	1	1	0	1	c	c	620.0	
4	5	5	2015-07-31	4822	559	1	1	0	1	a	a	29910.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1016776	682	2	2013-01-01	3375	566	1	0	a	1	b	a	150.0	
1016827	733	2	2013-01-01	10765	2377	1	0	a	1	b	b	860.0	
1016863	769	2	2013-01-01	5035	1248	1	0	a	1	b	b	840.0	
1017042	948	2	2013-01-01	4491	1039	1	0	a	1	b	b	1430.0	
1017190	1097	2	2013-01-01	5961	1405	1	0	a	1	b	b	720.0	

844392 rows × 25 columns

INPUT AND TARGET COLUMNS:

[47]:
```
merged_df.columns
```

[47]:
```
Index(['Store', 'DayOfWeek', 'Date', 'Sales', 'Customers', 'Open', 'Promo',
       'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment',
       'CompetitionDistance', 'CompetitionOpenSinceMonth',
       'CompetitionOpenSinceYear', 'Promo2', 'Promo2SinceWeek',
       'Promo2SinceYear', 'PromoInterval', 'Year', 'Month', 'Day',
       'WeekofYear', 'CompetitionOpen', 'Promo2Open', 'IsPromo2Month'],
      dtype='object')
```

[53]:
```
input_cols = ['Store', 'DayOfWeek', 'Promo', 'StateHoliday', 'SchoolHoliday',
              'StoreType', 'Assortment', 'CompetitionDistance', 'CompetitionOpen',
              'Day', 'Month', 'Year', 'WeekofYear',  'Promo2',
              'Promo2Open', 'IsPromo2Month']
target_cols = ['Sales']
```

[54]:
```
inputs = merged_df[input_cols].copy()
targets = merged_df[target_cols].copy()
```

[56]:
```
inputs
```

[56]:

	Store	DayOfWeek	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOpen	Day	Month	Year	WeekofYear	Prom
0	1	5	1	0	1	c	a	1270.0	82.0	31	7	2015	31	
1	2	5	1	0	1	a	a	570.0	92.0	31	7	2015	31	
2	3	5	1	0	1	a	a	14130.0	103.0	31	7	2015	31	
3	4	5	1	0	1	c	c	620.0	70.0	31	7	2015	31	
4	5	5	1	0	1	a	a	29910.0	3.0	31	7	2015	31	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1016776	682	2	0	a	1	b	a	150.0	76.0	1	1	2013	1	
1016827	733	2	0	a	1	b	b	860.0	159.0	1	1	2013	1	
1016863	769	2	0	a	1	b	b	840.0	0.0	1	1	2013	1	
1017042	948	2	0	a	1	b	b	1430.0	0.0	1	1	2013	1	
1017190	1097	2	0	a	1	b	b	720.0	130.0	1	1	2013	1	

844392 rows × 16 columns

[57]:
```
targets
```

[57]:

	Sales
0	5263
1	6064
2	8314
3	13995
4	4822
...	...
1016776	3375
1016827	10765
1016863	5035
1017042	4491
1017190	5961

844392 rows × 1 columns

[55]:
```
test_inputs = merged_test_df[input_cols].copy()
```

[60]:
```
numeric_cols = ['Store', 'Promo', 'SchoolHoliday',
             'CompetitionDistance', 'CompetitionOpen', 'Promo2', 'Promo2Open', 'IsPromo2Month',
             'Day', 'Month', 'Year', 'WeekofYear',  ]
categorical_cols = ['DayOfWeek', 'StateHoliday', 'StoreType', 'Assortment']
```

IMPUTE MISSING NUMERICAL DATA:

[61]:
```
inputs[numeric_cols].isna().sum()
```

[61]:
```
Store                   0
Promo                   0
SchoolHoliday           0
CompetitionDistance    2186
CompetitionOpen         0
Promo2                  0
Promo2Open              0
IsPromo2Month           0
Day                     0
Month                   0
Year                    0
WeekofYear              0
dtype: int64
```

[62]:
```
test_inputs[numeric_cols].isna().sum()
```

[62]:
```
Store                  0
Promo                  0
SchoolHoliday          0
CompetitionDistance    96
CompetitionOpen        0
Promo2                 0
Promo2Open             0
IsPromo2Month          0
Day                    0
Month                  0
Year                   0
```

```
Year                    0
WeekofYear              0
dtype: int64
```

[63]:
```
max_distance = inputs.CompetitionDistance.max()
max_distance
```

[63]:
```
75860.0
```

[64]:
```
inputs['CompetitionDistance'].fillna(max_distance, inplace = True)
test_inputs['CompetitionDistance'].fillna(max_distance, inplace = True)
```

SCALE NUMERIC VALUES:

[65]:
```
from sklearn.preprocessing import MinMaxScaler
```

[66]:
```
scaler = MinMaxScaler().fit(inputs[numeric_cols])
```

[67]:
```
inputs[numeric_cols] = scaler.transform(inputs[numeric_cols])
test_inputs[numeric_cols] = scaler.transform(test_inputs[numeric_cols])
```

ENCODE CATEGORICAL COLUMNS:

[68]:
```
from sklearn.preprocessing import OneHotEncoder
```

[72]:
```
encoder = OneHotEncoder(sparse_output = False, handle_unknown = 'ignore').fit(inputs[categorical_cols])
encoded_cols = list(encoder.get_feature_names_out(categorical_cols))
```

[74]:
```
inputs[encoded_cols] = encoder.transform(inputs[categorical_cols])
test_inputs[encoded_cols] = encoder.transform(test_inputs[categorical_cols])
```

[76]:
```
X = inputs[numeric_cols + encoded_cols]
X_test = test_inputs[numeric_cols + encoded_cols]
```

GRADIENT BOOSTING:

[77]:
```
from xgboost import XGBRegressor
```

[78]:
```
model = XGBRegressor(n_jobs = -1, random_state = 42,n_estimators = 20, max_depth = 4 )
```

[79]:
```
model.fit(X, targets)
```

[79]:
```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, 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=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=4, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=20, n_jobs=-1,
             num_parallel_tree=None, random_state=42, ...)
```

PREDICTION:

[81]:
```
preds = model.predict(X)
preds
```

[81]:
```
array([ 7960.1133,  7581.974 ,  7993.401 , ...,  7101.1714,  8794.572 ,
       10820.442 ], dtype=float32)
```

EVALUATION:

[122]:
```
from sklearn.metrics import mean_squared_error, root_mean_squared_error

def rmse(a, b):
    return root_mean_squared_error(a, b)
```

[123]:
```
rmse(preds, targets)
```

[123]:
```
2397.1614431476673
```

[124]:
```
merged_df.Sales.min(), merged_df.Sales.max()
```

[124]:
```
(0, 41551)
```

VISUALIZATION:

[125]:
```
import matplotlib.pyplot as plt
from xgboost import plot_tree
from matplotlib.pylab import rcParams
%matplotlib inline

rcParams['figure.figsize'] = 30,30
```

[126]:
```
trees = model.get_booster().get_dump()
```

[127]:
```
len(trees)
```

[127]:
```
20
```

[128]:
```
print(trees[0])
```

```
0:[Promo<1] yes=1,no=2,missing=2
        1:[StoreType_b<1] yes=3,no=4,missing=4
                3:[Assortment_a<1] yes=7,no=8,missing=8
                        7:[CompetitionDistance<0.00448312238] yes=15,no=16,missing=16
                                15:leaf=221.39238
                                16:leaf=-258.765686
                        8:[WeekofYear<0.921568632] yes=17,no=18,missing=18
                                17:leaf=-487.187103
                                18:leaf=-103.261314
                4:[Assortment_c<1] yes=9,no=10,missing=10
                        9:[Store<0.236086175] yes=19,no=20,missing=20
                                19:leaf=1829.72522
                                20:leaf=385.337738
                        10:[DayOfWeek_6<1] yes=21,no=22,missing=22
                                21:leaf=3160.67676
                                22:leaf=2034.8446
        2:[DayOfWeek_1<1] yes=5,no=6,missing=6
                5:[Month<1] yes=11,no=12,missing=12
                        11:[StoreType_b<1] yes=23,no=24,missing=24
                                23:leaf=209.018616
                                24:leaf=1218.62549
                        12:[Day<0.200000003] yes=25,no=26,missing=26
                                25:leaf=649.005493
                                26:leaf=1140.81042
                6:[Month<1] yes=13,no=14,missing=14
                        13:[CompetitionDistance<0.00342827011] yes=27,no=28,missing=28
                                27:leaf=1190.61548
                                28:leaf=770.427429
                        14:[Day<0.0666666701] yes=29,no=30,missing=30
                                29:leaf=1295.73438
                                30:leaf=1955.04956
```

FEATURE IMPORTANCE:

[129]:
```
importance_df = pd.DataFrame({
    'feature': X.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending = False)
```

[130]:
```
importance_df
```

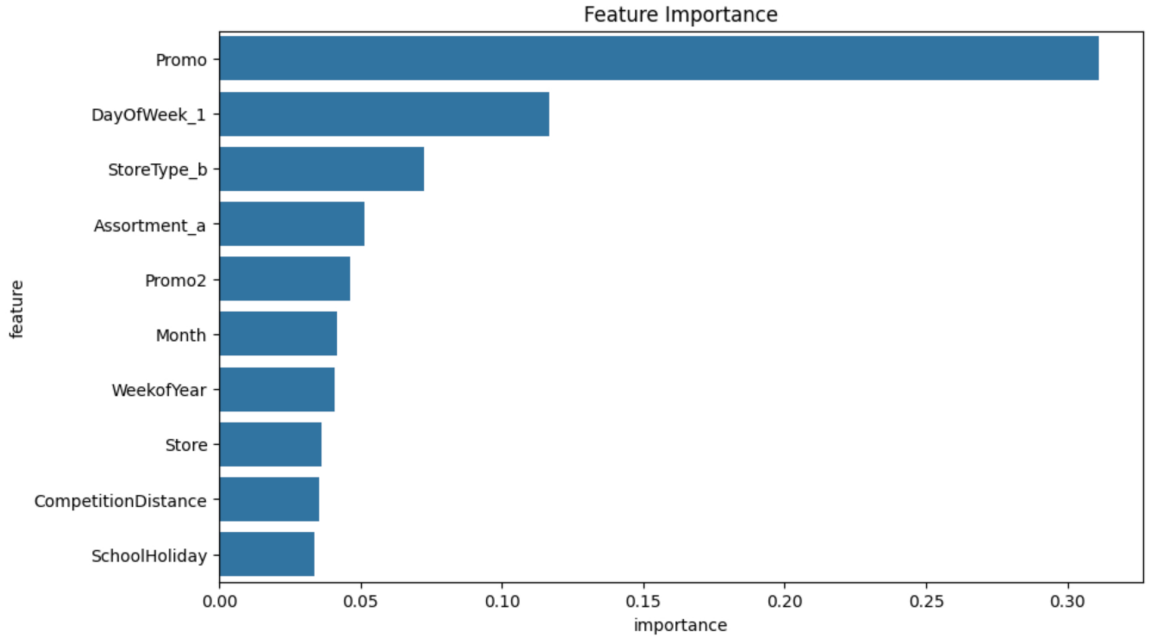
[130]:

	feature	importance
1	Promo	0.311225
12	DayOfWeek_1	0.116910
24	StoreType_b	0.072281
27	Assortment_a	0.051223
5	Promo2	0.046284
9	Month	0.041754
11	WeekofYear	0.040932
0	Store	0.036404
3	CompetitionDistance	0.035390
2	SchoolHoliday	0.033745
6	Promo2Open	0.031847
8	Day	0.027168
4	CompetitionOpen	0.025312
29	Assortment_c	0.018465
18	DayOfWeek_7	0.016332
26	StoreType_d	0.013670
16	DayOfWeek_5	0.013392
13	DayOfWeek_2	0.010915
23	StoreType_a	0.010237
17	DayOfWeek_6	0.009317
10	Year	0.009243

25	StoreType_c	0.008993
14	DayOfWeek_3	0.008059
15	DayOfWeek_4	0.006771
7	IsPromo2Month	0.004134
19	StateHoliday_0	0.000000
20	StateHoliday_a	0.000000
21	StateHoliday_b	0.000000
22	StateHoliday_c	0.000000
28	Assortment_b	0.000000

[131]:
```
import seaborn as sns
plt.figure(figsize = (10,6))
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x = 'importance', y = 'feature')
```

[131]: <Axes: title={'center': 'Feature Importance'}, xlabel='importance', ylabel='feature'>



KFOLD CROSS VALIDATION:

[132]:
```
from sklearn.model_selection import KFold
```

[133]:
```
kfold = KFold(n_splits = 5)
```

[134]:
```
def train_and_evaluate(X_train, train_targets, X_val,val_targets,**params):
    model = XGBRegressor(n_jobs = -1,random_state = 42, **params)
    model.fit(X_train, train_targets)
    train_rmse = rmse(model.predict(X_train),train_targets)
    val_rmse  = rmse(model.predict(X_val),val_targets)
    return model, train_rmse, val_rmse
```

[135]:
```
models = []

for train_idxs, val_idxs in kfold.split(X):
    X_train, train_targets = X.iloc[train_idxs], targets.iloc[train_idxs]
    X_val, val_targets = X.iloc[val_idxs], targets.iloc[val_idxs]
    model, train_rmse, val_rmse = train_and_evaluate(X_train,
                                                     train_targets,
                                                     X_val,
                                                     val_targets,
                                                     max_depth=4,
                                                     n_estimators=20)
    models.append(model)
    print('Train RMSE: {}, Validation RMSE: {}'.format(train_rmse, val_rmse))
```

```
Train RMSE: 2394.7876402581237, Validation RMSE: 2462.850314220323
Train RMSE: 2401.1582743064164, Validation RMSE: 2449.841681576792
Train RMSE: 2411.8926798093644, Validation RMSE: 2391.5910868526794
Train RMSE: 2344.831637340392, Validation RMSE: 2440.0443213469284
Train RMSE: 2391.1546155860246, Validation RMSE: 2460.8741743196815
```

[136]:
```
import numpy as np

def predict_avg(models, inputs):
```

```
    return np.mean([model.predict(inputs) for model in models], axis=0)
```

[138]:
```
preds = predict_avg(models, X)
preds
```

[138]:
```
array([8031.8696, 7536.164 , 8671.131 , ..., 7181.968 , 7955.146 ,
       9629.75  ], dtype=float32)
```

[139]:
```
from sklearn.model_selection import train_test_split
```

[140]:
```
X_train, X_val, train_targets, val_targets = train_test_split(X, targets, test_size=0.1)
```

HYPERPARAMETER TUNING AND REGULARIZATION:

[141]:
```
def test_params(**params):
    model = XGBRegressor(n_jobs=-1, random_state=42, **params)
    model.fit(X_train, train_targets)
    train_rmse = rmse(model.predict(X_train), train_targets)
    val_rmse = rmse(model.predict(X_val), val_targets)
    print('Train RMSE: {}, Validation RMSE: {}'.format(train_rmse, val_rmse))
```

N_ESTIMATORS:

[142]:
```
test_params(n_estimators = 10)
```

```
Train RMSE: 2341.0906769337294, Validation RMSE: 2350.130188809246
```

[144]:
```
test_params(n_estimators = 100)
```

```
Train RMSE: 1171.4514676420645, Validation RMSE: 1182.5898313921737
```

MAX_DEPTH

[146]:
```
test_params(max_depth = 5, n_estimators = 10)
```

```
Train RMSE: 2443.6957248026724, Validation RMSE: 2450.290247176367
```

LEARNING RATE:

[148]:
```
test_params(learning_rate = 0.1, n_estimators = 50)
```

```
Train RMSE: 2197.9126412377545, Validation RMSE: 2207.8230398249584
```

[149]:
```
test_params(learning_rate = 0.99, n_estimators = 50)
```

```
Train RMSE: 1136.2368874010656, Validation RMSE: 1149.767337808948
```

BOOSTER:

[150]:
```
test_params(booster = 'gblinear')
```

```
Train RMSE: 2725.167895333599, Validation RMSE: 2735.2993258194574
```

PUTTING TOGETHER AND MAKING PREDICTIONS:

[151]:
```
model = XGBRegressor(n_jobs=-1, random_state=42, n_estimators=1000,
                     learning_rate=0.2, max_depth=10, subsample=0.9,
                     colsample_bytree=0.7)
```

[152]:
```
model.fit(X,targets)
```

[152]:
```
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=0.2, 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, ...)
```

[155]:
```
X_test
```

[155]:

	Store	Promo	SchoolHoliday	CompetitionDistance	CompetitionOpen	Promo2	Promo2Open	IsPromo2Month	Day	Month	Year	WeekofYear	DayOfWe
0	0.000000	1.0	0.0	0.016482	0.060606	0.0	0.000000	0.0	0.533333	0.727273	1.0	0.725490	
1	0.001795	1.0	0.0	0.186050	0.075758	1.0	0.743169	0.0	0.533333	0.727273	1.0	0.725490	
2	0.005386	1.0	0.0	0.316192	0.020924	0.0	0.000000	0.0	0.533333	0.727273	1.0	0.725490	

3	0.006284	1.0	0.0	0.098892	0.007937	0.0	0.000000	0.0	0.533333	0.727273	1.0	0.725490
4	0.007181	1.0	0.0	0.026503	0.130592	0.0	0.000000	0.0	0.533333	0.727273	1.0	0.725490
...	...	...	...	...	...	...	...	...	...	...	...	...
41083	0.996409	0.0	0.0	0.024789	0.010101	1.0	0.333333	0.0	0.000000	0.636364	1.0	0.588235
41084	0.997307	0.0	0.0	0.024525	0.080808	0.0	0.000000	0.0	0.000000	0.636364	1.0	0.588235
41085	0.998205	0.0	0.0	0.121835	0.000000	0.0	0.000000	0.0	0.000000	0.636364	1.0	0.588235
41086	0.999102	0.0	0.0	0.011208	0.000000	0.0	0.000000	0.0	0.000000	0.636364	1.0	0.588235
41087	1.000000	0.0	1.0	0.070280	0.000000	1.0	0.528689	0.0	0.000000	0.636364	1.0	0.588235

41088 rows × 30 columns

[154]:
```
test_preds = model.predict(X_test)
test_preds
```

[154]:
```
array([ 3976.9543,  8010.5522,  8700.922 , ...,  5796.3257, 21716.41  ,
        7364.0054], dtype=float32)
```

[156]:
```
test_df
```

[156]:

	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
0	1	1	4	2015-09-17	1.0	1	0	0
1	2	3	4	2015-09-17	1.0	1	0	0
2	3	7	4	2015-09-17	1.0	1	0	0
3	4	8	4	2015-09-17	1.0	1	0	0
4	5	9	4	2015-09-17	1.0	1	0	0
...	...	...	...	...	...	...	...	...
41083	41084	1111	6	2015-08-01	1.0	0	0	0
41084	41085	1112	6	2015-08-01	1.0	0	0	0
41085	41086	1113	6	2015-08-01	1.0	0	0	0
41086	41087	1114	6	2015-08-01	1.0	0	0	0
41087	41088	1115	6	2015-08-01	1.0	0	0	1

41088 rows × 8 columns

[157]:
```
submission_df
```

[157]:

	Id	Sales
0	1	0
1	2	0
2	3	0
3	4	0
4	5	0
...	...	...
41083	41084	0
41084	41085	0
41085	41086	0
41086	41087	0
41087	41088	0

41088 rows × 2 columns

[160]:
```
submission_df['Sales'] = test_preds
submission_df
```

[160]:

	Id	Sales
0	1	3976.954346
1	2	8010.552246
2	3	8700.921875
3	4	7396.159668
4	5	7195.354492
...	...	...
41083	41084	2432.904541
41084	41085	6463.385254
41085	41086	5796.325684

41086	41087	21716.410156
41087	41088	7364.005371

41088 rows × 2 columns

[171]:
```
submission_df['Sales'] = submission_df['Sales'] * test_df.Open.fillna(1.)
submission_df
```

[171]:

	Id	Sales
0	1	3976.954346
1	2	8010.552246
2	3	8700.921875
3	4	7396.159668
4	5	7195.354492
...	...	...
41083	41084	2432.904541
41084	41085	6463.385254
41085	41086	5796.325684
41086	41087	21716.410156
41087	41088	7364.005371

41088 rows × 2 columns

[178]:
```
submission_df['Sales'].fillna(1, inplace=True)
```

[179]:
```
submission_df['Sales'].isna().sum()
```

[179]:
```
0
```

[180]:
```
submission_df.to_csv('submission.csv', index=None)
```

[181]:
```
from IPython.display import FileLink
```

[182]:
```
FileLink('submission.csv')
```

[182]: submission.csv

[ ]: