[1]:
```
# Install necessary packages and upgrade them together to manage dependencies better
!pip install pandas-profiling numpy matplotlib seaborn opendatasets scikit-learn jovian --quiet --upgrade
```

[2]:
```
import os
import jovian
import matplotlib
import opendatasets as od
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (10, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

DOWNLOADING DATA:

[5]:
```
od.download('https://www.kaggle.com/c/rossmann-store-sales')
```

```
Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds
Your Kaggle username:
  adityadutta200021
Your Kaggle Key:
  ........
Downloading rossmann-store-sales.zip to .\rossmann-store-sales
100%|██████████| 6.99M/6.99M [00:00<00:00, 30.3MB/s]

Extracting archive .\rossmann-store-sales/rossmann-store-sales.zip to .\rossmann-store-sales
```

[6]:
```
os.listdir('rossmann-store-sales')
```

[6]:
```
['sample_submission.csv', 'store.csv', 'test.csv', 'train.csv']
```

[7]:
```
ross_df = pd.read_csv('./rossmann-store-sales/train.csv', low_memory=False)
```

[8]:
```
ross_df
```

[8]:

	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

[9]:
```
store_df = pd.read_csv('./rossmann-store-sales/store.csv')
```

[10]:
```
store_df
```

[10]:

	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,A
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

1115 rows × 10 columns

[11]:
```
merged_df = ross_df.merge(store_df, how='left', on='Store')
merged_df
```

[11]:

	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

[12]:
```
merged_df.shape
```

[12]:
```
(1017209, 18)
```

[13]:
```
test_df = pd.read_csv('rossmann-store-sales/test.csv')
```

[14]:
```
merged_test_df = test_df.merge(store_df, how='left', on='Store')
```

[15]:
```
merged_test_df
```

[15]:

	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

CLEANING DATA:

[16]:
```
merged_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
```

[17]:
```
round(merged_df.describe().T,2)
```

[17]:

	count	mean	std	min	25%	50%	75%	max
Store	1017209.0	558.43	321.91	1.0	280.0	558.0	838.0	1115.0
DayOfWeek	1017209.0	4.00	2.00	1.0	2.0	4.0	6.0	7.0
Sales	1017209.0	5773.82	3849.93	0.0	3727.0	5744.0	7856.0	41551.0
Customers	1017209.0	633.15	464.41	0.0	405.0	609.0	837.0	7388.0
Open	1017209.0	0.83	0.38	0.0	1.0	1.0	1.0	1.0
Promo	1017209.0	0.38	0.49	0.0	0.0	0.0	1.0	1.0
SchoolHoliday	1017209.0	0.18	0.38	0.0	0.0	0.0	0.0	1.0
CompetitionDistance	1014567.0	5430.09	7715.32	20.0	710.0	2330.0	6890.0	75860.0
CompetitionOpenSinceMonth	693861.0	7.22	3.21	1.0	4.0	8.0	10.0	12.0
CompetitionOpenSinceYear	693861.0	2008.69	5.99	1900.0	2006.0	2010.0	2013.0	2015.0
Promo2	1017209.0	0.50	0.50	0.0	0.0	1.0	1.0	1.0
Promo2SinceWeek	509178.0	23.27	14.10	1.0	13.0	22.0	37.0	50.0
Promo2SinceYear	509178.0	2011.75	1.66	2009.0	2011.0	2012.0	2013.0	2015.0

[18]:
```
merged_df.duplicated().sum()
```

[18]:
```
0
```

[19]:
```
merged_df['Date'] = pd.to_datetime(merged_df.Date)
```

[20]:
```
merged_test_df['Date'] = pd.to_datetime(merged_test_df.Date)
```

[21]:
```
merged_df.Date.min(), merged_df.Date.max()
```

[21]:
```
(Timestamp('2013-01-01 00:00:00'), Timestamp('2015-07-31 00:00:00'))
```

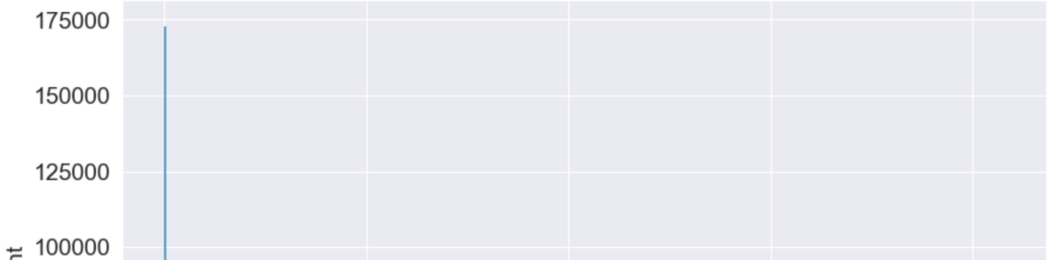
[22]:
```
merged_test_df.Date.min(), merged_test_df.Date.max()
```

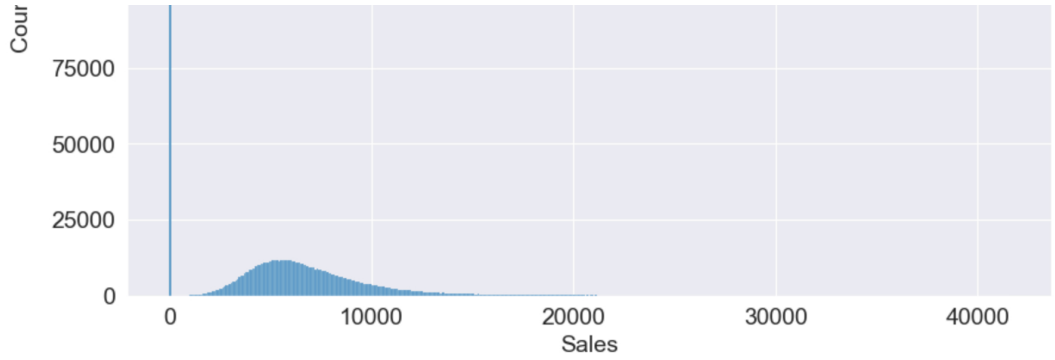
[22]:
```
(Timestamp('2015-08-01 00:00:00'), Timestamp('2015-09-17 00:00:00'))
```

EXPLORATORY DATA ANALYSIS AND VISUALIZATON:

[23]:
```
sns.histplot(data = merged_df, x = 'Sales')
```

[23]:
```
<Axes: xlabel='Sales', ylabel='Count'>
```





[24]:
```
merged_df.Open.value_counts()
```

[24]:
```
Open
1    844392
0    172817
Name: count, dtype: int64
```

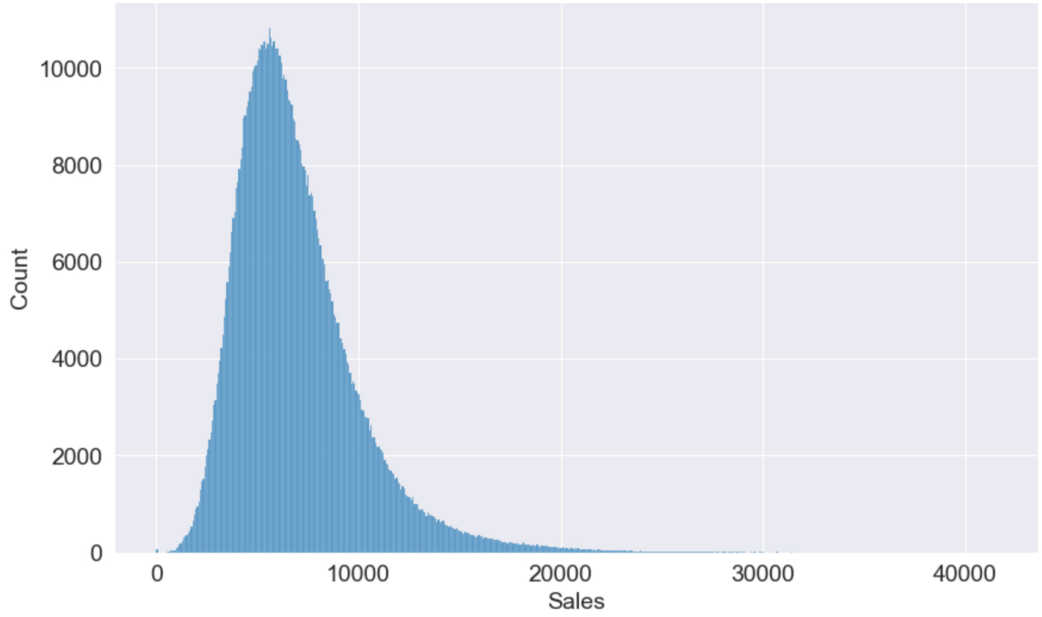
[26]:
```
merged_df.Sales.value_counts()[0]
```

[26]:
```
172871
```

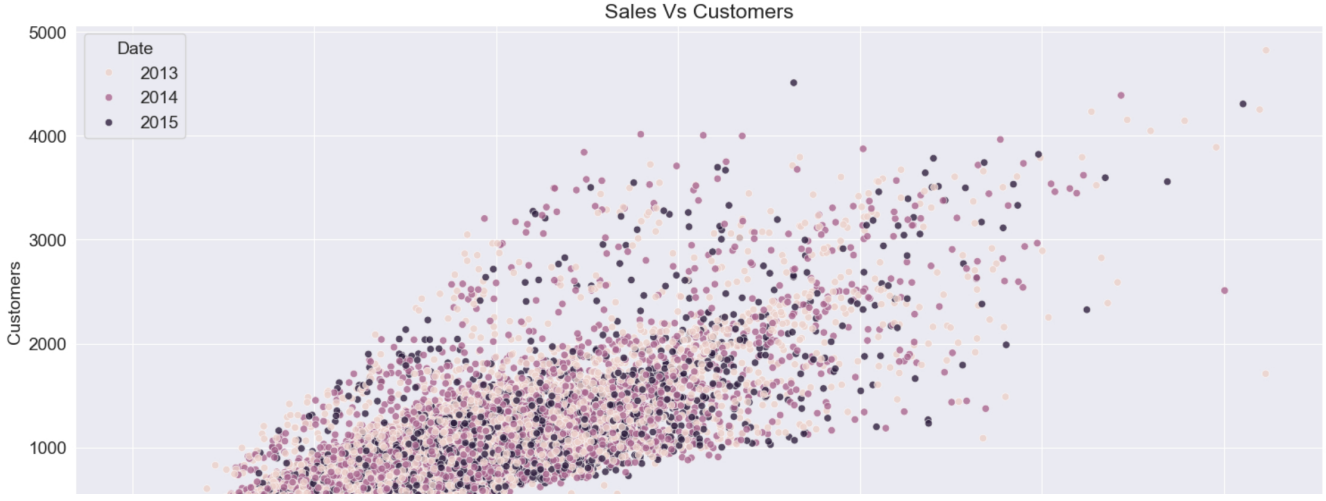
[27]:
```
merged_df = merged_df[merged_df.Open == 1].copy()
```

[28]:
```
sns.histplot(data = merged_df, x = 'Sales')
```

[28]:
```
<Axes: xlabel='Sales', ylabel='Count'>
```



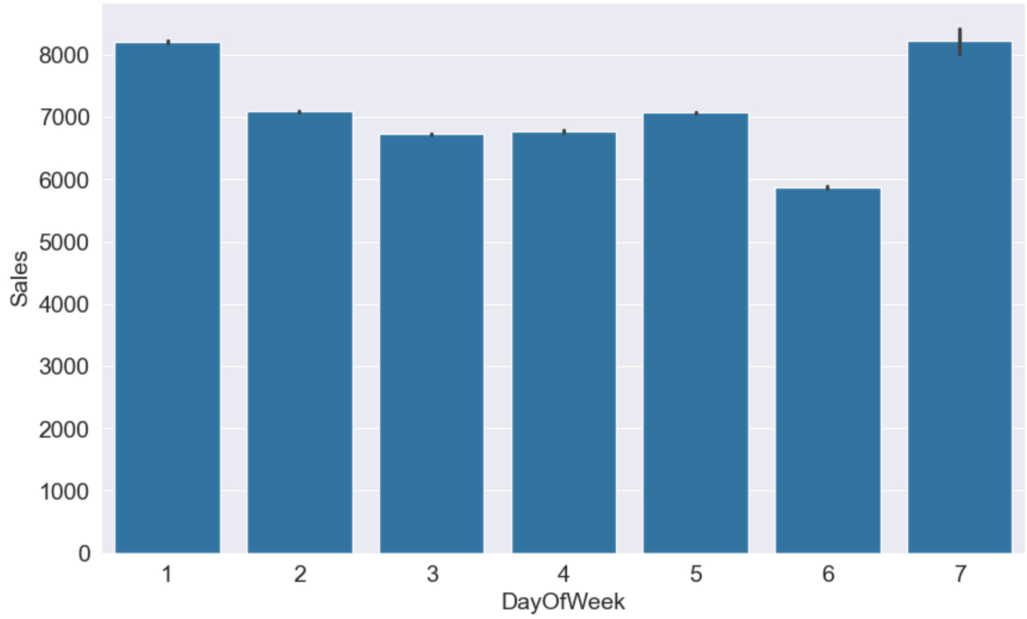
[29]:
```
plt.figure(figsize=(18,8))
temp_df = merged_df.sample(40000)
sns.scatterplot(x=temp_df.Sales, y=temp_df.Customers, hue=temp_df.Date.dt.year, alpha=0.8)
plt.title("Sales Vs Customers")
plt.show()
```



[30]:
```
plt.figure(figsize=(18,8))
temp_df = merged_df.sample(10000)
sns.scatterplot(x=temp_df.Store, y=temp_df.Sales, hue=temp_df.Date.dt.year, alpha=0.8)
plt.title("Stores Vs Sales")
plt.show()
```

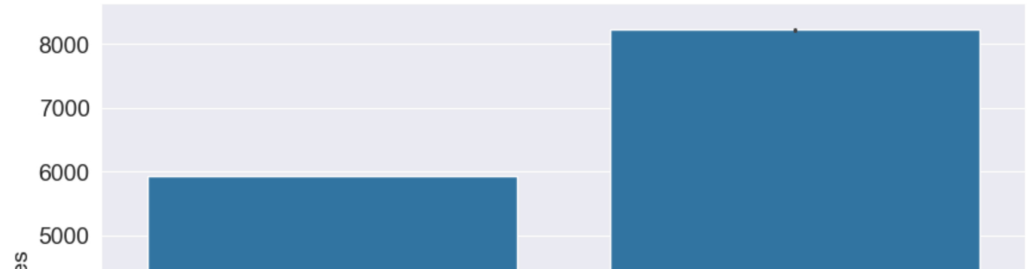
[31]:
```
sns.barplot(data=merged_df, x='DayOfWeek', y='Sales')
```

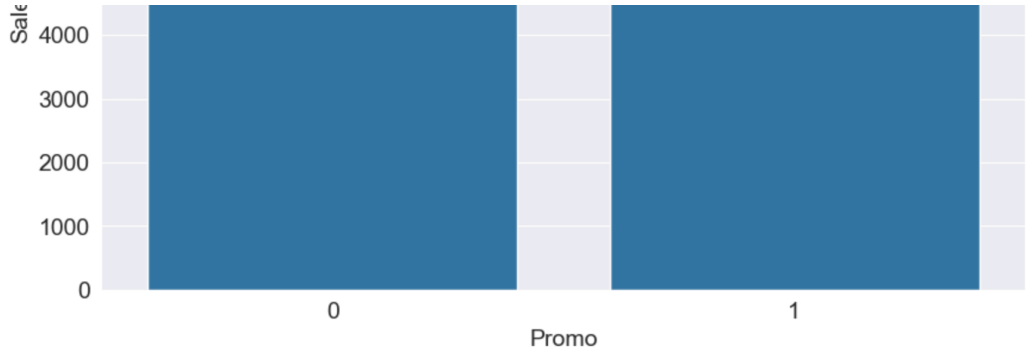
[31]:
```
<Axes: xlabel='DayOfWeek', ylabel='Sales'>
```



[32]:
```
sns.barplot(data=merged_df, x='Promo', y='Sales')
```

[32]:
```
<Axes: xlabel='Promo', ylabel='Sales'>
```





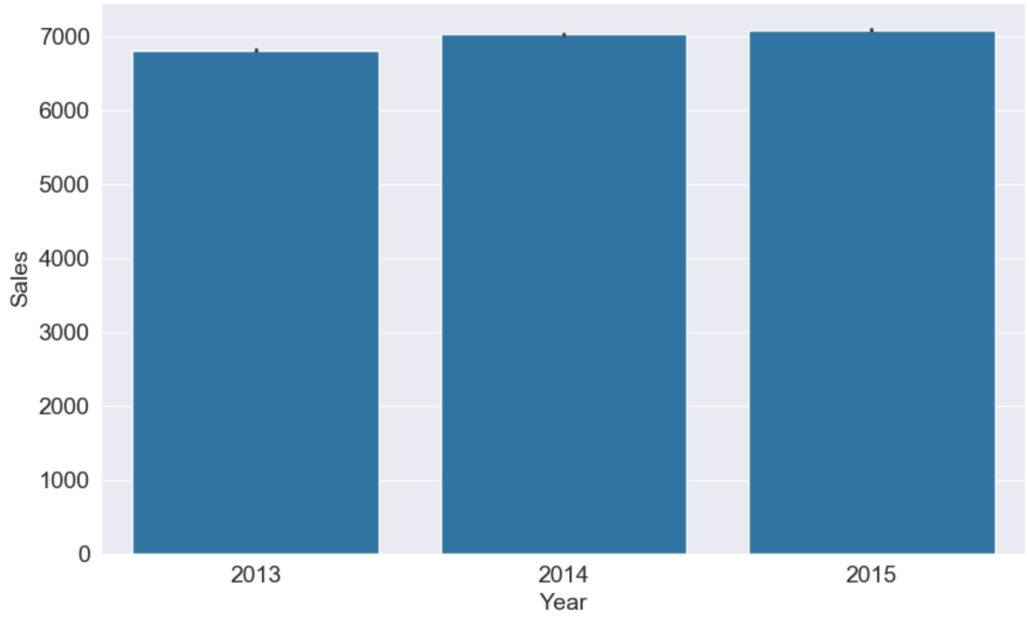
FEATURE ENGINEERING:

[35]:
```
merged_df['Day'] = merged_df.Date.dt.day
merged_df['Month'] = merged_df.Date.dt.month
merged_df['Year'] = merged_df.Date.dt.year
```

[36]:
```
merged_test_df['Day'] = merged_test_df.Date.dt.day
merged_test_df['Month'] = merged_test_df.Date.dt.month
merged_test_df['Year'] = merged_test_df.Date.dt.year
```

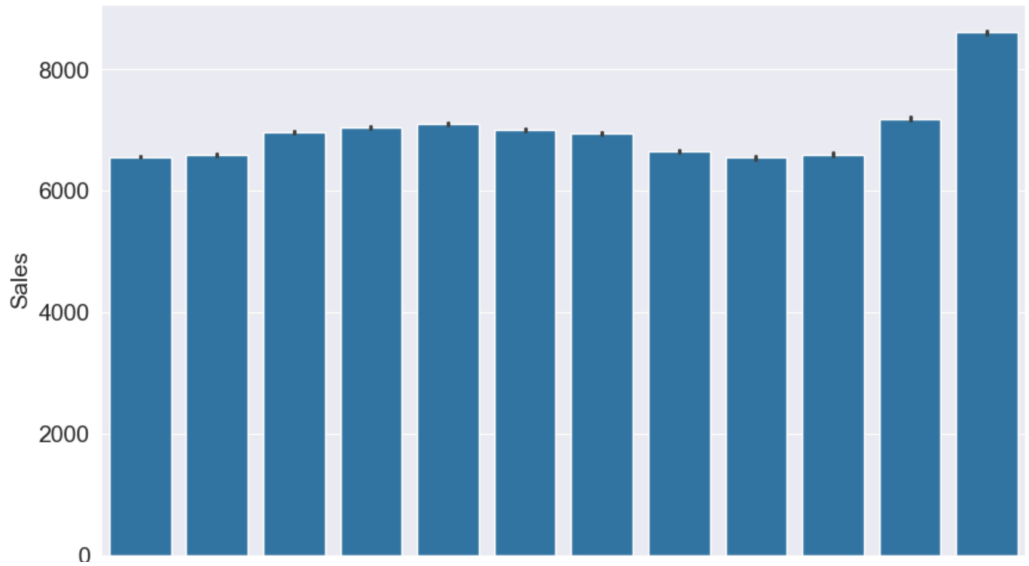
[37]:
```
sns.barplot(data = merged_df, x = 'Year', y = 'Sales')
```

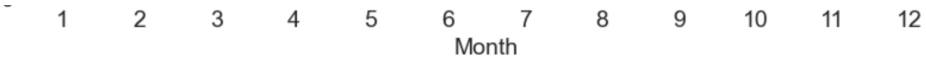
[37]:
```
<Axes: xlabel='Year', ylabel='Sales'>
```



[38]:
```
sns.barplot(data = merged_df, x = 'Month', y = 'Sales')
```

[38]:
```
<Axes: xlabel='Month', ylabel='Sales'>
```





TRAIN / TEST / VALIDATION SPLIT:

[39]:
```
len(merged_df)
```

[39]:
```
844392
```

[40]:
```
train_size = int(.75 * len(merged_df))
train_size
```

[40]:
```
633294
```

[41]:
```
sorted_df = merged_df.sort_values('Date')
train_df, val_df = sorted_df[:train_size], sorted_df[train_size:]
```

[42]:
```
len(train_df), len(val_df)
```

[42]:
```
(633294, 211098)
```

[43]:
```
 train_df.Date.min(), train_df.Date.max()
```

[43]:
```
(Timestamp('2013-01-01 00:00:00'), Timestamp('2014-12-10 00:00:00'))
```

[44]:
```
val_df.Date.min(), val_df.Date.max()
```

[44]:
```
(Timestamp('2014-12-10 00:00:00'), Timestamp('2015-07-31 00:00:00'))
```

[45]:
```
merged_test_df.Date.min(), merged_test_df.Date.max()
```

[45]:
```
(Timestamp('2015-08-01 00:00:00'), Timestamp('2015-09-17 00:00:00'))
```

[47]:
```
train_df.columns
```

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

INPUT AND TARGET COLUMNS:

[48]:
```
input_cols = ['Store', 'DayOfWeek', 'Promo', 'StateHoliday', 'StoreType', 'Assortment', 'Day', 'Month', 'Year']
```

[49]:
```
target_col = 'Sales'
```

[50]:
```
merged_df[input_cols].nunique()
```

[50]:
```
Store          1115
DayOfWeek         7
Promo             2
StateHoliday      4
StoreType         4
Assortment        3
Day              31
Month            12
Year              3
dtype: int64
```

[51]:
```
train_inputs = train_df[input_cols].copy()
train_targets = train_df[target_col].copy()
```

[52]:
```
val_inputs = val_df[input_cols].copy()
val_targets = val_df[target_col].copy()
```

[53]:
```
test_inputs = merged_test_df[input_cols].copy()
# Test data does not have targets
```

[54]:
```
numeric_cols = ['Store', 'Day', 'Month', 'Year']
categorical_cols = ['DayOfWeek', 'Promo', 'StateHoliday', 'StoreType', 'Assortment']
```

IMPUTING, SCALING AND ENCODE:

[55]:
```
from sklearn.impute import SimpleImputer
```

[56]:
```
imputer = SimpleImputer(strategy = 'mean').fit(train_inputs[numeric_cols])
```

[57]:
```
train_inputs[numeric_cols] = imputer.transform(train_inputs[numeric_cols])
val_inputs[numeric_cols] = imputer.transform(val_inputs[numeric_cols])
test_inputs[numeric_cols] = imputer.transform(test_inputs[numeric_cols])
```

[58]:
```
from sklearn.preprocessing import MinMaxScaler
```

[59]:
```
scaler = MinMaxScaler().fit(train_inputs[numeric_cols])
```

[60]:
```
train_inputs[numeric_cols] = scaler.transform(train_inputs[numeric_cols])
val_inputs[numeric_cols] = scaler.transform(val_inputs[numeric_cols])
test_inputs[numeric_cols] = scaler.transform(test_inputs[numeric_cols])
```

[61]:
```
from sklearn.preprocessing import OneHotEncoder
```

[63]:
```
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore').fit(train_inputs[categorical_cols])
encoded_cols = list(encoder.get_feature_names_out(categorical_cols))
```

[64]:
```
encoded_cols
```

[64]:
```
['DayOfWeek_1',
 'DayOfWeek_2',
 'DayOfWeek_3',
 'DayOfWeek_4',
 'DayOfWeek_5',
 'DayOfWeek_6',
 'DayOfWeek_7',
 'Promo_0',
 'Promo_1',
 'StateHoliday_0',
 'StateHoliday_a',
 'StateHoliday_b',
 'StateHoliday_c',
 'StoreType_a',
 'StoreType_b',
 'StoreType_c',
 'StoreType_d',
 'Assortment_a',
 'Assortment_b',
 'Assortment_c']
```

[65]:
```
train_inputs[encoded_cols] = encoder.transform(train_inputs[categorical_cols])
val_inputs[encoded_cols] = encoder.transform(val_inputs[categorical_cols])
test_inputs[encoded_cols] = encoder.transform(test_inputs[categorical_cols])
```

[67]:
```
X_train = train_inputs[numeric_cols + encoded_cols]
X_val = val_inputs[numeric_cols + encoded_cols]
X_test = test_inputs[numeric_cols + encoded_cols]
```

[68]:
```
X_train
```

[68]:

	Store	Day	Month	Year	DayOfWeek_1	DayOfWeek_2	DayOfWeek_3	DayOfWeek_4	DayOfWeek_5	DayOfWeek_6	...	StateHoliday_a	StateHoliday_b	State
1017190	0.983842	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
1016179	0.075404	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
1016353	0.231598	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
1016356	0.234291	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
1016368	0.245063	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
256632	0.667864	0.3	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	
256642	0.677738	0.3	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	
256634	0.669659	0.3	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	
256633	0.668761	0.3	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	
256636	0.671454	0.3	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	

633294 rows × 24 columns

[69]:
```
train_targets.mean()
```

[69]:
```
6873.17964168301
```

FIXED/RANDOM GUESS:

[70]:
```
def return_mean(inputs):
    return np.full(len(inputs), merged_df.Sales.mean())
```

[71]:
```
train_preds = return_mean(X_train)
train_preds
```

[71]:
```
array([6955.51429076, 6955.51429076, 6955.51429076, ..., 6955.51429076,
       6955.51429076, 6955.51429076])
```

[72]:
```
train_targets
```

[72]:
```
1017190     5961
1016179     4220
1016353     6851
1016356    17267
1016368     3102
           ...
256632      6897
```

```
256642     15736
256634      7444
256633      5207
256636      3587
Name: Sales, Length: 633294, dtype: int64
```

[73]:
```
from sklearn.metrics import mean_squared_error
```

[74]:
```
mean_squared_error(train_preds, train_targets, squared = False)
```

[74]:
```
3082.450443277419
```

BASELINE ML MODEL:

[75]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, train_targets)
```

[75]:
```
▾ LinearRegression
LinearRegression()
```

[79]:
```
X_train.columns
```

[79]:
```
Index(['Store', 'Day', 'Month', 'Year', 'DayOfWeek_1', 'DayOfWeek_2',
       'DayOfWeek_3', 'DayOfWeek_4', 'DayOfWeek_5', 'DayOfWeek_6',
       'DayOfWeek_7', 'Promo_0', 'Promo_1', 'StateHoliday_0', 'StateHoliday_a',
       'StateHoliday_b', 'StateHoliday_c', 'StoreType_a', 'StoreType_b',
       'StoreType_c', 'StoreType_d', 'Assortment_a', 'Assortment_b',
       'Assortment_c'],
      dtype='object')
```

[76]:
```
linreg.coef_
```

[76]:
```
array([ 1.06228002e+02, -1.00361780e+02,  6.54013573e+02,  1.05702070e+02,
         1.81511137e+15,  1.81511137e+15,  1.81511137e+15,  1.81511137e+15,
         1.81511137e+15,  1.81511137e+15,  1.81511137e+15, -1.34642904e+13,
        -1.34642904e+13, -2.04959818e+14, -2.04959818e+14, -2.04959818e+14,
        -2.04959818e+14, -6.03680426e+15, -6.03680426e+15, -6.03680426e+15,
        -6.03680426e+15,  8.48537704e+14,  8.48537704e+14,  8.48537704e+14])
```

[80]:
```
train_preds = linreg.predict(X_train)
```

[81]:
```
train_preds
```

[81]:
```
array([ 7110., 10574.,  7030., ...,  6254.,  6009.,  5625.])
```

[82]:
```
mean_squared_error(train_preds, train_targets, squared = False)
```

[82]:
```
2741.590556725826
```

[83]:
```
val_preds = linreg.predict(X_val)
val_preds
```

[83]:
```
array([5625.5, 5625.5, 6009.5, ..., 8650. , 8404.5, 7845. ])
```

[84]:
```
mean_squared_error(val_preds, val_targets, squared = False)
```

[84]:
```
2817.5374275240547
```

EXPLORING MODELLING STRATEGIES:

[85]:
```
def try_model(model):
    # Fit the model
    model.fit(X_train, train_targets)

    # Generate predictions
    train_preds = model.predict(X_train)
    val_preds = model.predict(X_val)

    # Compute RMSE
    train_rmse = mean_squared_error(train_targets, train_preds, squared=False)
    val_rmse = mean_squared_error(val_targets, val_preds, squared=False)
    return train_rmse, val_rmse
```

LINEAR MODELS:

[86]:
```
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet, SGDRegressor
```

[88]:
```
try_model(LinearRegression())
```

[88]:
```
(2741.590556725826, 2817.5374275240547)
```

[89]:
```
try_model(Ridge())
```

[89]:
```
(2741.587195081444, 2817.7784665409426)
```

[90]:
```
try_model(Lasso())
```

[90]:
```
(2741.7143904640566, 2817.9472924045217)
```

[93]:
```
try_model(ElasticNet(alpha = 0.6))
```

[93]:
```
(2843.6585896792467, 2935.8621000493936)
```

[92]:
```
try_model(SGDRegressor())
```

[92]:
```
(2742.031705336479, 2816.9480861084653)
```

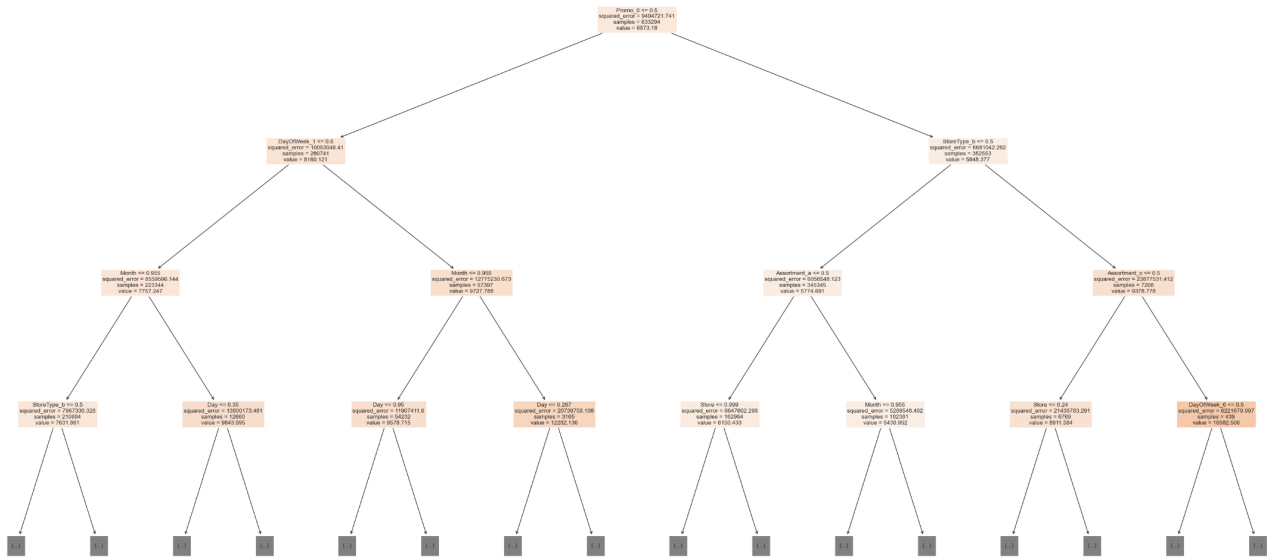
TREE BASED REGRESSORS:

[94]:
```
from sklearn.tree import DecisionTreeRegressor, plot_tree
```

[95]:
```
tree = DecisionTreeRegressor(random_state=42)
try_model(tree)
```

[95]:
```
(0.0, 1559.7378600480247)
```

[96]:
```
plt.figure(figsize=(40, 20))
plot_tree(tree, max_depth=3, filled=True, feature_names=numeric_cols+encoded_cols);
```



[97]:
```
from sklearn.ensemble import RandomForestRegressor
```

[98]:
```
%%time
rf = RandomForestRegressor(random_state=42, n_jobs=-1)
try_model(rf)
```

```
CPU times: total: 9min 37s
Wall time: 44.1 s
```

[98]:
```
(474.9198974666898, 1371.7321923469985)
```

[99]:
```
X_train.columns
```

[99]:
```
Index(['Store', 'Day', 'Month', 'Year', 'DayOfWeek_1', 'DayOfWeek_2',
       'DayOfWeek_3', 'DayOfWeek_4', 'DayOfWeek_5', 'DayOfWeek_6',
       'DayOfWeek_7', 'Promo_0', 'Promo_1', 'StateHoliday_0', 'StateHoliday_a',
       'StateHoliday_b', 'StateHoliday_c', 'StoreType_a', 'StoreType_b',
       'StoreType_c', 'StoreType_d', 'Assortment_a', 'Assortment_b',
       'Assortment_c'],
      dtype='object')
```

FEATURE IMPORTANCE:

[101]:
```
rf.feature_importances_
```

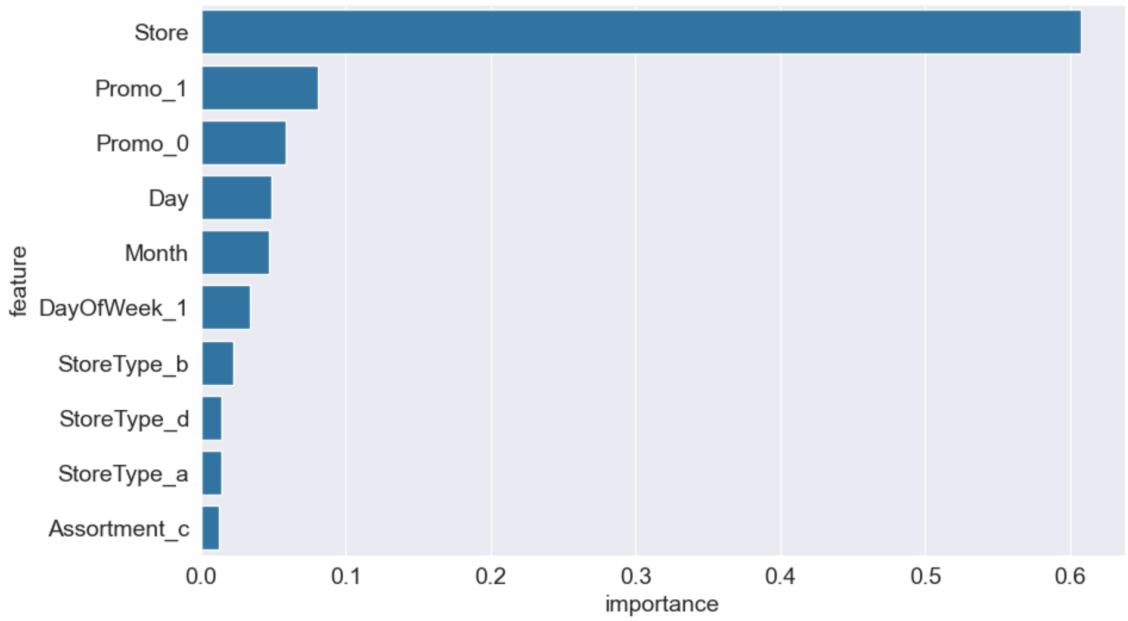
[101]:
```
array([6.07412101e-01, 4.88393196e-02, 4.71254150e-02, 9.30282826e-03,
       3.36941035e-02, 4.40591456e-03, 2.43735761e-03, 2.11467096e-03,
       5.71418981e-03, 6.96554452e-03, 4.47739863e-03, 5.83469269e-02,
       8.05201285e-02, 1.18767484e-03, 2.11472202e-04, 2.45172539e-05,
       1.88388455e-05, 1.38995265e-02, 2.25678813e-02, 1.04169805e-02,
       1.39945667e-02, 8.34303630e-03, 5.50125164e-03, 1.24783549e-02])
```

[105]:
```
importance_df = pd.DataFrame({
    'feature': X_train.columns,
    'importance': rf.feature_importances_
}).sort_values('importance', ascending = False)
importance_df.head(10)
```

[105]:
```
feature  importance
```

	feature	importance
0	Store	0.607412
12	Promo_1	0.080520
11	Promo_0	0.058347
1	Day	0.048839
2	Month	0.047125
4	DayOfWeek_1	0.033694
18	StoreType_b	0.022568
20	StoreType_d	0.013995
17	StoreType_a	0.013900
23	Assortment_c	0.012478

[106]:
```
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```



NEW INPUT PRE-PROCESSING:

[107]:
```
def predict_input(model, single_input):
    if single_input['Open'] == 0:
        return 0.
    input_df = pd.DataFrame([single_input])
    input_df['Date'] = pd.to_datetime(input_df.Date)
    input_df['Day'] = input_df.Date.dt.day
    input_df['Month'] = input_df.Date.dt.month
    input_df['Year'] = input_df.Date.dt.year
    input_df[numeric_cols] = imputer.transform(input_df[numeric_cols])
    input_df[numeric_cols] = scaler.transform(input_df[numeric_cols])
    input_df[encoded_cols] = encoder.transform(input_df[categorical_cols])
    X_input = input_df[numeric_cols + encoded_cols]
    pred = model.predict(X_input)[0]
    return pred
```

[108]:
```
sample_input = {'Id': 1,
 'Store': 1,
 'DayOfWeek': 4,
 'Date': '2015-09-17 00:00:00',
 'Open': 1.0,
 'Promo': 1,
 'StateHoliday': '0',
 'SchoolHoliday': 0,
 'StoreType': 'c',
 'Assortment': 'a',
 'CompetitionDistance': 1270.0,
 'CompetitionOpenSinceMonth': 9.0,
 'CompetitionOpenSinceYear': 2008.0,
 'Promo2': 0,
 'Promo2SinceWeek': np.nan,
 'Promo2SinceYear': np.nan,
 'PromoInterval': np.nan}

sample_input
```

[108]:
```
{'Id': 1,
 'Store': 1,
 'DayOfWeek': 4,
```

```
 'Date': '2015-09-17 00:00:00',
 'Open': 1.0,
 'Promo': 1,
 'StateHoliday': '0',
 'SchoolHoliday': 0,
 'StoreType': 'c',
 'Assortment': 'a',
 'CompetitionDistance': 1270.0,
 'CompetitionOpenSinceMonth': 9.0,
 'CompetitionOpenSinceYear': 2008.0,
 'Promo2': 0,
 'Promo2SinceWeek': nan,
 'Promo2SinceYear': nan,
 'PromoInterval': nan}
```

[109]:
```
predict_input(rf, sample_input)
```

[109]:
```
4258.01
```

[110]:
```
X_test
```

[110]:

	Store	Day	Month	Year	DayOfWeek_1	DayOfWeek_2	DayOfWeek_3	DayOfWeek_4	DayOfWeek_5	DayOfWeek_6	...	StateHoliday_a	StateHoliday_b	S
0	0.000000	0.533333	0.727273	2.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	
1	0.001795	0.533333	0.727273	2.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	
2	0.005386	0.533333	0.727273	2.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	
3	0.006284	0.533333	0.727273	2.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	
4	0.007181	0.533333	0.727273	2.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41083	0.996409	0.000000	0.636364	2.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	
41084	0.997307	0.000000	0.636364	2.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	
41085	0.998205	0.000000	0.636364	2.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	
41086	0.999102	0.000000	0.636364	2.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	
41087	1.000000	0.000000	0.636364	2.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	

41088 rows × 24 columns

[111]:
```
test_preds = rf.predict(X_test)
test_preds
```

[111]:
```
array([ 4258.01,  7713.83,  8775.7 , ...,  5954.68, 20823.07,  6748.48])
```

[112]:
```
submission_df = pd.read_csv('./rossmann-store-sales/sample_submission.csv')
```

[114]:
```
submission_df
```

[114]:

	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

[118]:
```
submission_df['Sales'] = test_preds * test_df['Open'].astype('float')
submission_df.fillna(0, inplace=True)
```

[119]:
```
submission_df
```

[119]:

	Id	Sales
0	1	4258.01
1	2	7713.83
2	3	8775.70
3	4	6775.37
4	5	6492.47

41083	41084	3052.40
41084	41085	9008.27
41085	41086	5954.68
41086	41087	20823.07
41087	41088	6748.48

41088 rows × 2 columns

[120]:
```
submission_df.to_csv('submission.csv', index=None)
```

[122]:
```
print(submission_df.head())
```

```
   Id    Sales
0   1  4258.01
1   2  7713.83
2   3  8775.70
3   4  6775.37
4   5  6492.47
```

[123]:
```
from IPython.display import FileLink
```

[124]:
```
FileLink('submission.csv')
```

[124]: submission.csv

[ ]: