Walmart_Forecasting

November 23, 2023

1 Walmart Sales Prediction

1.1 Visualizing and Analysing the data

1.1.1 Importing the libraries

[ ]:
```
import numpy as np
import pandas as pd
import scipy. stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error
from datetime import datetime
import math
```

1.1.2 Reading the dataset

[ ]:
```
train=pd.read_csv('train.csv')
features=pd.read_csv('features.csv')
stores = pd.read_csv('stores.csv')
```

[ ]:
```
train.head()
```

[ ]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False

[ ]:
```
features.head()
```

[ ]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	\
0	1	2010-02-05	42.31	2.572	NaN	NaN	
1	1	2010-02-12	38.51	2.548	NaN	NaN	
2	1	2010-02-19	39.93	2.514	NaN	NaN	
3	1	2010-02-26	46.63	2.561	NaN	NaN	

```
4      1  2010-03-05        46.50       2.625        NaN        NaN

   MarkDown3  MarkDown4  MarkDown5         CPI  Unemployment  IsHoliday
0        NaN        NaN        NaN  211.096358         8.106      False
1        NaN        NaN        NaN  211.242170         8.106       True
2        NaN        NaN        NaN  211.289143         8.106      False
3        NaN        NaN        NaN  211.319643         8.106      False
4        NaN        NaN        NaN  211.350143         8.106      False
```

[ ]:
```
stores.head()
```

[ ]:
```
   Store Type    Size
0      1    A  151315
1      2    A  202307
2      3    B   37392
3      4    A  205863
4      5    B   34875
```

1.1.3 Descriptive Analysis

[ ]:
```
train.describe()
```

[ ]:
```
               Store           Dept   Weekly_Sales
count  421570.000000  421570.000000  421570.000000
mean       22.200546      44.260317   15981.258123
std        12.785297      30.492054   22711.183519
min         1.000000       1.000000   -4988.940000
25%        11.000000      18.000000    2079.650000
50%        22.000000      37.000000    7612.030000
75%        33.000000      74.000000   20205.852500
max        45.000000      99.000000  693099.360000
```

[ ]:
```
features.describe()
```

[ ]:
```
              Store  Temperature   Fuel_Price      MarkDown1      MarkDown2  \
count  8190.000000  8190.000000  8190.000000    4032.000000    2921.000000
mean     23.000000    59.356198     3.405992    7032.371786    3384.176594
std      12.987966    18.678607     0.431337    9262.747448    8793.583016
min       1.000000    -7.290000     2.472000   -2781.450000    -265.760000
25%      12.000000    45.902500     3.041000    1577.532500      68.880000
50%      23.000000    60.710000     3.513000    4743.580000     364.570000
75%      34.000000    73.880000     3.743000    8923.310000    2153.350000
max      45.000000   101.950000     4.468000  103184.980000  104519.540000

            MarkDown3      MarkDown4      MarkDown5          CPI  Unemployment
count    3613.000000    3464.000000    4050.000000  7605.000000   7605.000000
mean     1760.100180    3292.935886    4132.216422   172.460809      7.826821
```

```
std     11276.462208   6792.329861   13086.690278    39.738346      1.877259
min      -179.260000      0.220000    -185.170000   126.064000      3.684000
25%         6.600000    304.687500    1440.827500   132.364839      6.634000
50%        36.260000   1176.425000    2727.135000   182.764003      7.806000
75%       163.150000   3310.007500    4832.555000   213.932412      8.567000
max    149483.310000  67474.850000  771448.100000   228.976456     14.313000
```

[ ]:
```
stores.describe()
```

[ ]:
```
            Store           Size
count   45.000000      45.000000
mean    23.000000  130287.600000
std     13.133926   63825.271991
min      1.000000   34875.000000
25%     12.000000   70713.000000
50%     23.000000  126512.000000
75%     34.000000  202307.000000
max     45.000000  219622.000000
```

1.2 Data Preprocessing

1.2.1 Checking for null values

[ ]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  object
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool
dtypes: bool(1), float64(1), int64(2), object(1)
memory usage: 13.3+ MB
```

[ ]:
```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8190 entries, 0 to 8189
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         8190 non-null   int64
 1   Date          8190 non-null   object
 2   Temperature   8190 non-null   float64
 3   Fuel_Price    8190 non-null   float64
```

```
 4   MarkDown1     4032 non-null   float64
 5   MarkDown2     2921 non-null   float64
 6   MarkDown3     3613 non-null   float64
 7   MarkDown4     3464 non-null   float64
 8   MarkDown5     4050 non-null   float64
 9   CPI           7605 non-null   float64
 10  Unemployment  7605 non-null   float64
 11  IsHoliday     8190 non-null   bool
dtypes: bool(1), float64(9), int64(1), object(1)
memory usage: 712.0+ KB
```

[ ]:
```
stores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Store   45 non-null     int64
 1   Type    45 non-null     object
 2   Size    45 non-null     int64
dtypes: int64(2), object(1)
memory usage: 1.2+ KB
```

[ ]:
```
train.isnull().sum()
```

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

[ ]:
```
features.isnull().sum()
```

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

[ ]:
```
stores.isnull().sum()
```

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

[ ]:
```
data = train.merge(features, on=['Store', 'Date'], how='inner').merge(stores,␣
↪on=['Store'], how='inner')
print(data.shape)
```

```
(421570, 17)
```

[ ]:
```
data['MarkDown1'] = data['MarkDown1'].replace(np.nan, 0)
data['MarkDown2'] = data['MarkDown2'].replace(np.nan, 0)
data['MarkDown3'] = data['MarkDown3'].replace(np.nan, 0)
data['MarkDown4'] = data['MarkDown4'].replace(np.nan, 0)
data['MarkDown5'] = data['MarkDown5'].replace(np.nan, 0)
```

1.2.2 Handling Negative Values

[ ]:
```
data.describe()
```

[ ]:

	Store	Dept	Weekly_Sales	Temperature	\
count	421570.000000	421570.000000	421570.000000	421570.000000	
mean	22.200546	44.260317	15981.258123	60.090059	
std	12.785297	30.492054	22711.183519	18.447931	
min	1.000000	1.000000	-4988.940000	-2.060000	
25%	11.000000	18.000000	2079.650000	46.680000	
50%	22.000000	37.000000	7612.030000	62.090000	
75%	33.000000	74.000000	20205.852500	74.280000	
max	45.000000	99.000000	693099.360000	100.140000	

	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	\
count	421570.000000	421570.000000	421570.000000	421570.000000	
mean	3.361027	2590.074819	879.974298	468.087665	
std	0.458515	6052.385934	5084.538801	5528.873453	
min	2.472000	0.000000	-265.760000	-29.100000	
25%	2.933000	0.000000	0.000000	0.000000	
50%	3.452000	0.000000	0.000000	0.000000	
75%	3.738000	2809.050000	2.200000	4.540000	
max	4.468000	88646.760000	104519.540000	141630.610000	

	MarkDown4	MarkDown5	CPI	Unemployment	\
count	421570.000000	421570.000000	421570.000000	421570.000000	
mean	1083.132268	1662.772385	171.201947	7.960289	
std	3894.529945	4207.629321	39.159276	1.863296	
min	0.000000	0.000000	126.064000	3.879000	

```
25%         0.000000       0.000000     132.022667       6.891000
50%         0.000000       0.000000     182.318780       7.866000
75%       425.290000    2168.040000     212.416993       8.572000
max     67474.850000  108519.280000     227.232807      14.313000

                Size
count   421570.000000
mean    136727.915739
std      60980.583328
min      34875.000000
25%      93638.000000
50%     140167.000000
75%     202505.000000
max     219622.000000
```

[ ]:
```
data = data[data['Weekly_Sales'] >= 0]
```

[ ]:
```
data.describe()
```

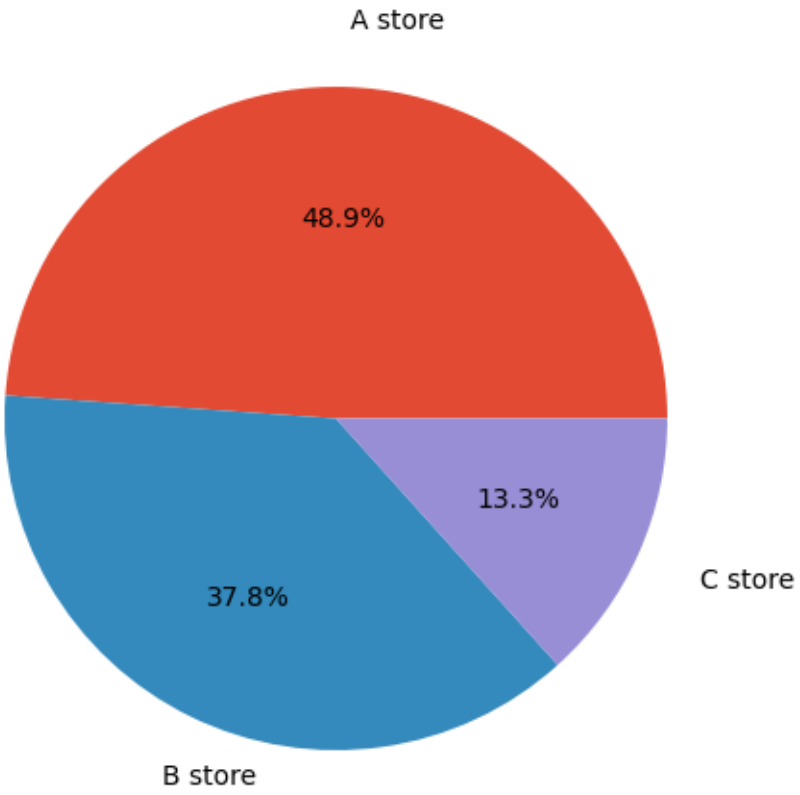
[ ]:
```
                Store           Dept   Weekly_Sales    Temperature  \
count  420285.000000  420285.000000  420285.000000  420285.000000
mean       22.195477      44.242771   16030.329773      60.090474
std        12.787213      30.507197   22728.500149      18.448260
min         1.000000       1.000000       0.000000      -2.060000
25%        11.000000      18.000000    2117.560000      46.680000
50%        22.000000      37.000000    7659.090000      62.090000
75%        33.000000      74.000000   20268.380000      74.280000
max        45.000000      99.000000  693099.360000     100.140000

          Fuel_Price      MarkDown1      MarkDown2      MarkDown3  \
count  420285.000000  420285.000000  420285.000000  420285.000000
mean        3.360888    2590.187246     878.803239     468.771234
std         0.458523    6053.225499    5076.525234    5533.593113
min         2.472000       0.000000    -265.760000     -29.100000
25%         2.933000       0.000000       0.000000       0.000000
50%         3.452000       0.000000       0.000000       0.000000
75%         3.738000    2801.500000       2.400000       4.540000
max         4.468000   88646.760000  104519.540000  141630.610000

           MarkDown4      MarkDown5            CPI   Unemployment  \
count  420285.000000  420285.000000  420285.000000  420285.000000
mean     1083.462694    1662.706138     171.212152       7.960077
std      3895.801513    4205.946641      39.162280       1.863873
min         0.000000       0.000000     126.064000       3.879000
25%         0.000000       0.000000     132.022667       6.891000
50%         0.000000       0.000000     182.350989       7.866000
75%       425.290000    2168.040000     212.445487       8.567000
```

```
max     67474.850000  108519.280000     227.232807      14.313000
```

```
                Size
count   420285.000000
mean    136749.569176
std      60992.688568
min      34875.000000
25%      93638.000000
50%     140167.000000
75%     202505.000000
max     219622.000000
```

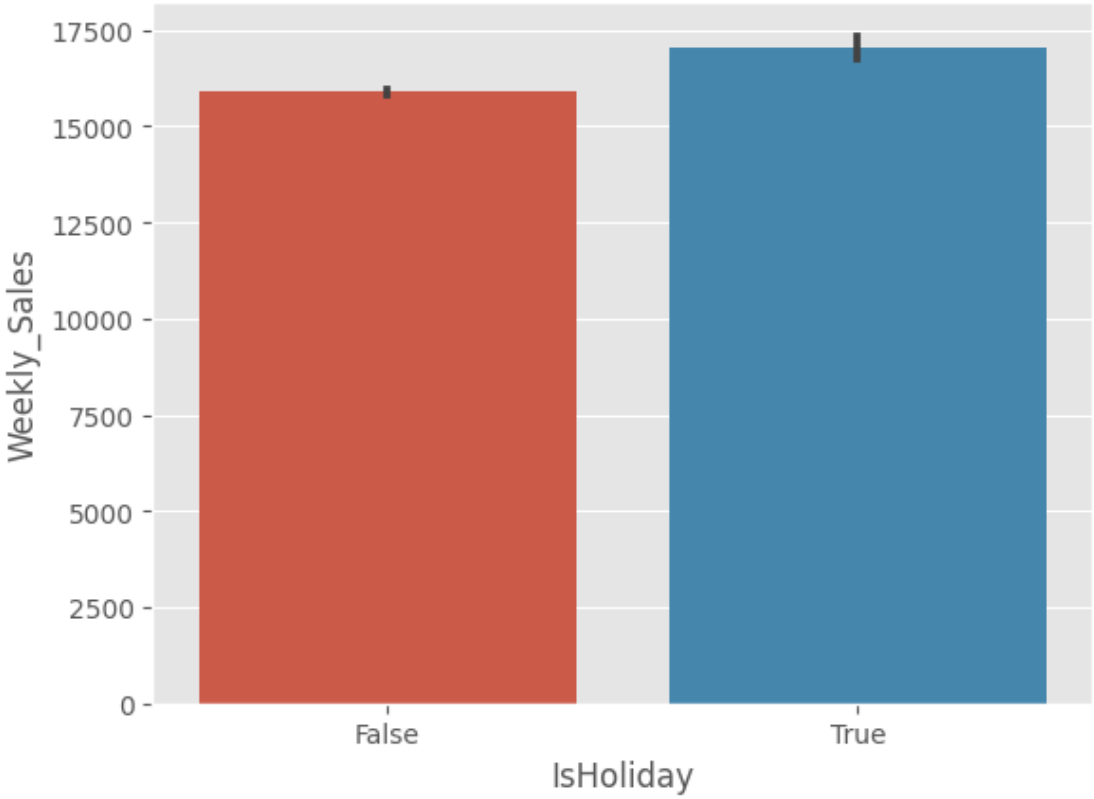
1.2.3 Exploratory Data Analysis

[ ]:
```
sorted_type = stores.groupby( 'Type')
plt.style.use('ggplot')
labels=['A store','B store','C store']
sizes=sorted_type.describe()['Size'].round(1)
sizes=[(22/(17+6+22))*100, (17/(17+6+22))*100, (6/(17+6+22))*100]
fig, axes = plt.subplots(1,1, figsize=(7,7))
axes.pie(sizes,
        labels=labels,
        explode=(0.0,0,0),
        autopct='%1.1f%%',
        pctdistance=0.6,
        labeldistance=1.2,
        radius=0.8,
        center=(0.5,0.5))
plt. show()
```



[ ]:
```
# Weekly Sales on Holidays
holiday = train['Weekly_Sales'].loc[train['IsHoliday']== True]
#Weekly Sales on Non-holidays.
non_holiday = train['Weekly_Sales'].loc[train['IsHoliday']== False]
sns.barplot (x='IsHoliday', y='Weekly_Sales', data=train)
```

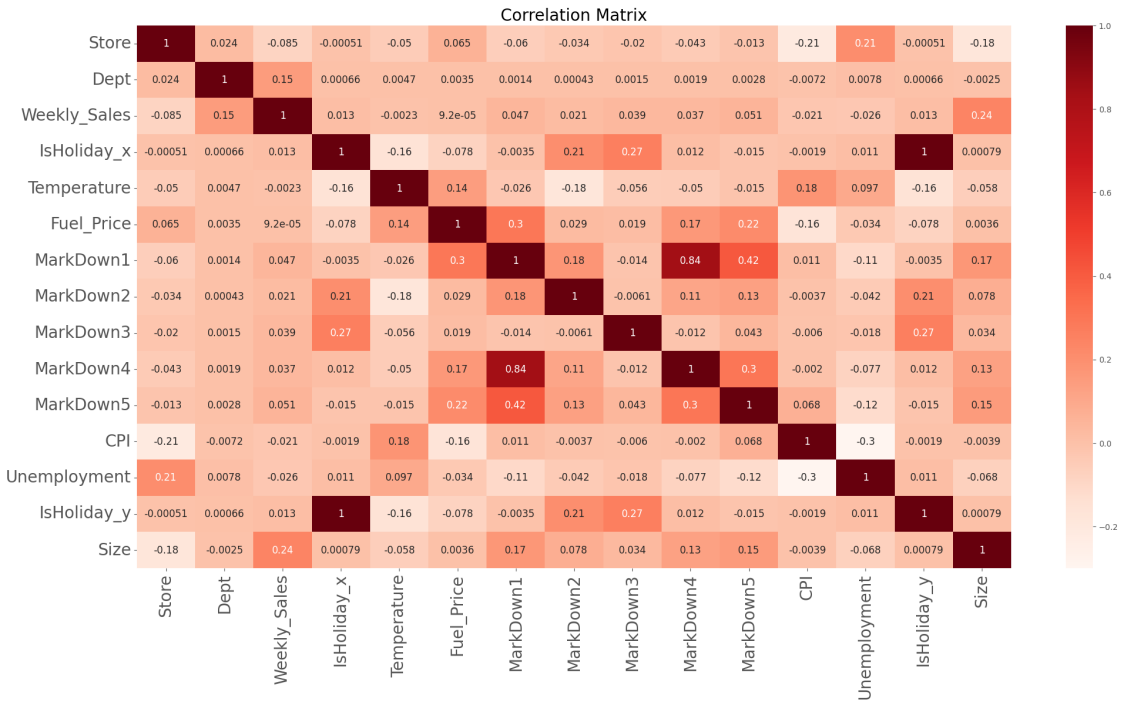
[ ]:
```
<Axes: xlabel='IsHoliday', ylabel='Weekly_Sales'>
```



[ ]:
```
plt.figure(figsize=(24,12))
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
sns.heatmap(data.corr(), cmap='Reds', annot=True, annot_kws={'size':12})
plt.title('Correlation Matrix', fontsize=20)
```

```
<ipython-input-27-bed6b132cd93>:4: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(data.corr(), cmap='Reds', annot=True, annot_kws={'size':12})
```

[ ]:
```
Text(0.5, 1.0, 'Correlation Matrix')
```



1.2.4 Handling Categorical Values

[ ]:
```
data=pd.get_dummies (data,columns=['Type'])
```

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

[ ]:
```
data['month'] = data['Date'].dt.month
data['Year'] = data[ 'Date'].dt.year
```

[ ]:
```
data[['Date', 'month', 'Year']].head()
```

[ ]:
```
        Date  month  Year
0 2010-02-05      2  2010
1 2010-02-05      2  2010
2 2010-02-05      2  2010
3 2010-02-05      2  2010
4 2010-02-05      2  2010
```

[ ]:
```
data['dayofweek_name'] = data['Date'].dt.day_name()
data[['Date', 'dayofweek_name']].head()
```

[ ]:
```
        Date dayofweek_name
0 2010-02-05         Friday
1 2010-02-05         Friday
2 2010-02-05         Friday
```

```
3 2010-02-05          Friday
4 2010-02-05          Friday
```

[ ]:
```
data['is_weekend'] = data['dayofweek_name'].apply(lambda x: 1 if x in␣
 ↪['Sunday', 'Saturday'] else 0)
data[['Date','is_weekend']].head()
```

[ ]:
```
        Date  is_weekend
0 2010-02-05           0
1 2010-02-05           0
2 2010-02-05           0
3 2010-02-05           0
4 2010-02-05           0
```

[ ]:
```
data["IsHoliday_x"] = data["IsHoliday_x"].astype(int)
del data['dayofweek_name']
```

[ ]:
```
print(data.head())
```

```
   Store  Dept       Date  Weekly_Sales  IsHoliday_x  Temperature  Fuel_Price  \
0      1     1 2010-02-05      24924.50            0        42.31       2.572
1      1     2 2010-02-05      50605.27            0        42.31       2.572
2      1     3 2010-02-05      13740.12            0        42.31       2.572
3      1     4 2010-02-05      39954.04            0        42.31       2.572
4      1     5 2010-02-05      32229.38            0        42.31       2.572

   MarkDown1  MarkDown2  MarkDown3  …         CPI  Unemployment  \
0        0.0        0.0        0.0  …  211.096358         8.106
1        0.0        0.0        0.0  …  211.096358         8.106
2        0.0        0.0        0.0  …  211.096358         8.106
3        0.0        0.0        0.0  …  211.096358         8.106
4        0.0        0.0        0.0  …  211.096358         8.106

   IsHoliday_y    Size  Type_A  Type_B  Type_C  month  Year  is_weekend
0        False  151315       1       0       0      2  2010           0
1        False  151315       1       0       0      2  2010           0
2        False  151315       1       0       0      2  2010           0
3        False  151315       1       0       0      2  2010           0
4        False  151315       1       0       0      2  2010           0

[5 rows x 22 columns]
```

[ ]:
```
data.to_csv('merged_data.csv', index=False)
```

1.2.5 Splitting data into train and test

[ ]:
```
df = pd.read_csv("merged_data.csv", keep_default_na=False, na_values=[""])
print(df.columns)
```

```
Index(['Store', 'Dept', 'Date', 'Weekly_Sales', 'IsHoliday_x', 'Temperature',
       'Fuel_Price', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4',
       'MarkDown5', 'CPI', 'Unemployment', 'IsHoliday_y', 'Size', 'Type_A',
       'Type_B', 'Type_C', 'month', 'Year', 'is_weekend'],
      dtype='object')
```

[ ]:
```
X = df.loc[:, df.columns != 'Weekly_Sales']
y = df.loc[:, df.columns == 'Weekly_Sales']
X = X[["Store", "Dept", "Size", "IsHoliday_x", "CPI", "Temperature", "Type_B",␣
 ↪"Type_C", "MarkDown4", "month", "Year"]]
 = y.values.reshape(-1, 1)
print (X.head())
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```
   Store  Dept    Size  IsHoliday_x         CPI  Temperature  Type_B  Type_C  \
0      1     1  151315            0  211.096358        42.31       0       0
1      1     2  151315            0  211.096358        42.31       0       0
2      1     3  151315            0  211.096358        42.31       0       0
3      1     4  151315            0  211.096358        42.31       0       0
4      1     5  151315            0  211.096358        42.31       0       0

   MarkDown4  month  Year
0        0.0      2  2010
1        0.0      2  2010
2        0.0      2  2010
3        0.0      2  2010
4        0.0      2  2010
```

1.3 Model Building

1.3.1 Random Forest

[ ]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor (n_estimators=50, max_depth=20, min_samples_split=3,␣
 ↪min_samples_leaf=1)
rf.fit(X_train, y_train)
print ('Accuracy:',rf.score(X_test, y_test)*100, '%')
y_pred = rf.predict(X_test)
```

```
<ipython-input-74-ca2511d819d8>:3: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  rf.fit(X_train, y_train)
```

```
Accuracy: 96.8495179504525 %
```

[ ]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import explained_variance_score
print('MSE: ', mean_squared_error(y_test, y_pred, squared=True))
print('RMSE: ', mean_squared_error(y_test, y_pred, squared=False))
print('MAE: ', mean_absolute_error (y_test, y_pred))
print('R2: ', explained_variance_score (y_test, y_pred))
```

```
MSE:  16414124.70616664
RMSE:  4051.434894721454
MAE:  1644.4428512985985
R2:  0.9684952121274976
```

[ ]:
```
print ('Training Accuracy:',rf.score(X_train, y_train)*100, '%')
```

```
Training Accuracy: 99.11550833618617 %
```

1.3.2 XgBoost

[ ]:
```
import xgboost as xgb
import warnings
xg_reg = xgb.XGBRegressor(objective='reg:squarederror', nthread= 4,␣
 ↪n_estimators= 500, max_depth= 4, learning_rate= 0.5)
xg_reg.fit(X_train, y_train)
```

[ ]:
```
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=0.5, 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=500, n_jobs=None, nthread=4,
             num_parallel_tree=None, …)
```

[ ]:
```
pred = xg_reg.predict(X_train)
y_pred = xg_reg.predict(X_test)
print ('Accuracy: ',xg_reg.score(X_test, y_test)*100, '%')
```

```
Accuracy:  94.08906350198728 %
```

[ ]:
```
print('MSE: ', mean_squared_error(y_test, y_pred, squared=True))
print('RMSE: ', mean_squared_error(y_test, y_pred, squared=False))
print('MAE: ', mean_absolute_error (y_test, y_pred))
print('R2: ', explained_variance_score (y_test, y_pred))
```

```
MSE:  30796191.593139805
RMSE:  5549.43164595617
MAE:  3068.662971047117
R2:  0.9408906894277229
```

[ ]:
```
print('Training Accuracy: ', xg_reg.score(X_train, y_train)*100, '%')
```

```
Training Accuracy:  94.08686109190809 %
```

1.3.3 Comparing the models

[ ]:
```
from prettytable import PrettyTable
tb = PrettyTable()
tb.field_names = ["Model", "Training Accuracy", "Testing Accuracy", "RMSE",␣
 ↪"MAE" ]
tb.add_row([ "Random Forest", 99.11, 96.84, 4056.09, 1647.11])
tb.add_row(["XgBoost", 94.08, 94.08, 5549.43, 3068.66])
print (tb)
```

```
+---------------+-------------------+------------------+---------+---------+
|     Model     | Training Accuracy | Testing Accuracy |   RMSE  |   MAE   |
+---------------+-------------------+------------------+---------+---------+
| Random Forest |       99.11       |      96.84       | 4056.09 | 1647.11 |
|    XgBoost    |       94.08       |      94.08       | 5549.43 | 3068.66 |
+---------------+-------------------+------------------+---------+---------+
```

[ ]:
```
from sklearn.model_selection import cross_val_score
rf = RandomForestRegressor(n_estimators=58, max_depth=27, min_samples_split=3,␣
 ↪min_samples_leaf=1)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

[ ]:
```
cv = cross_val_score(rf,X,y,cv=6)
np.mean(cv)
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
```

```
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
/usr/local/lib/python3.10/dist-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

[ ]: 0.7280028435919697

[ ]:
```
from sklearn.model_selection import cross_val_score
xg_reg = xgb.XGBRegressor(objective='reg:squarederror', nthread= 4,␣
 ↪n_estimators= 500, max_depth= 4, learning_rate= 0.5)
xg_reg.fit(X_train, y_train)
pred=xg_reg.predict(X_train)
y_pred = xg_reg.predict(X_test)
```

[ ]:
```
cv = cross_val_score(xg_reg,X,y,cv=6)
np.mean(cv)
```

[ ]: 0.7482499257941506

1.4 Saving the model

[ ]:
```
import pickle
pickle.dump(rf, open( 'final_model.pkl', 'wb'))
```

[ ]: