

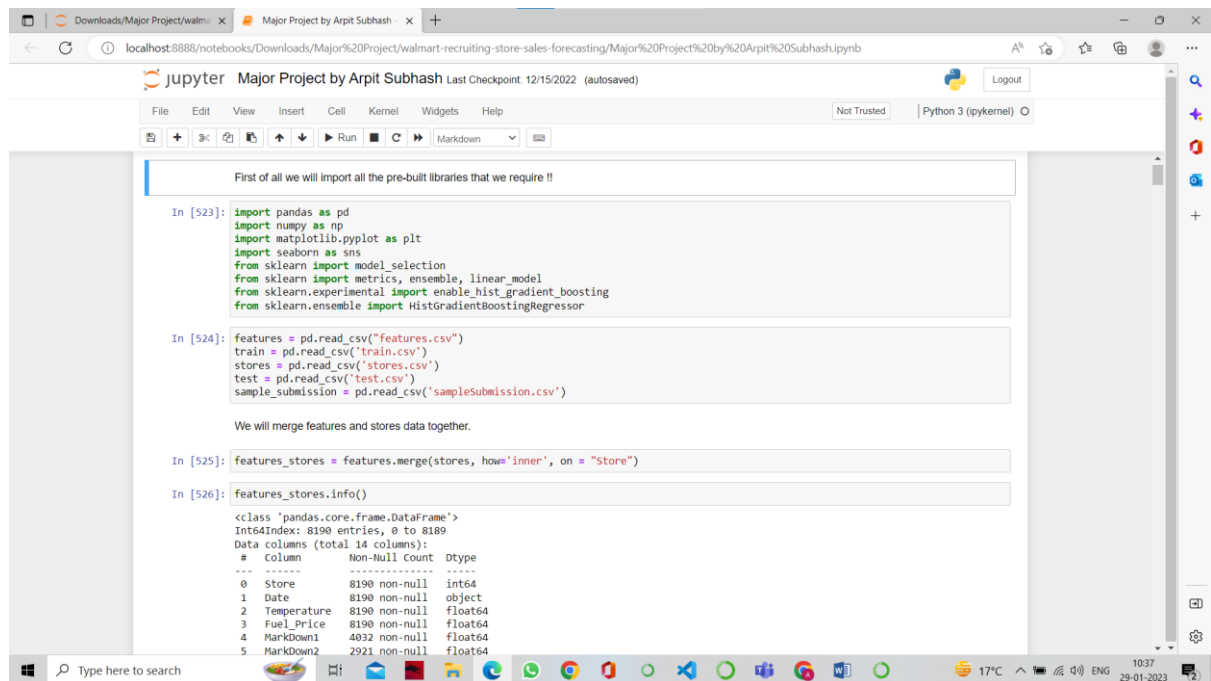
Report on Data Science project

Name: Arpit Subhash

Topic: Prediction on Walmart's departments of different stores through all of the world

Table of contents:

1. Libraries – I have used libraries like pandas, numpy, matplotlib, seaborn, sklearn kit (model_selection, ensemble, linear_model, eli5).
2. Data loading – features, stores, train, test (csv files).



The screenshot shows a Jupyter Notebook titled "Major Project by Arpit Subhash" with the following code cells:

```
First of all we will import all the pre-built libraries that we require !!
```

```
In [523]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import model_selection
from sklearn import metrics, ensemble, linear_model
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingRegressor
```

```
In [524]: features = pd.read_csv('features.csv')
train = pd.read_csv('train.csv')
stores = pd.read_csv('stores.csv')
test = pd.read_csv('test.csv')
sample_submission = pd.read_csv('sampleSubmission.csv')
```

We will merge features and stores data together.

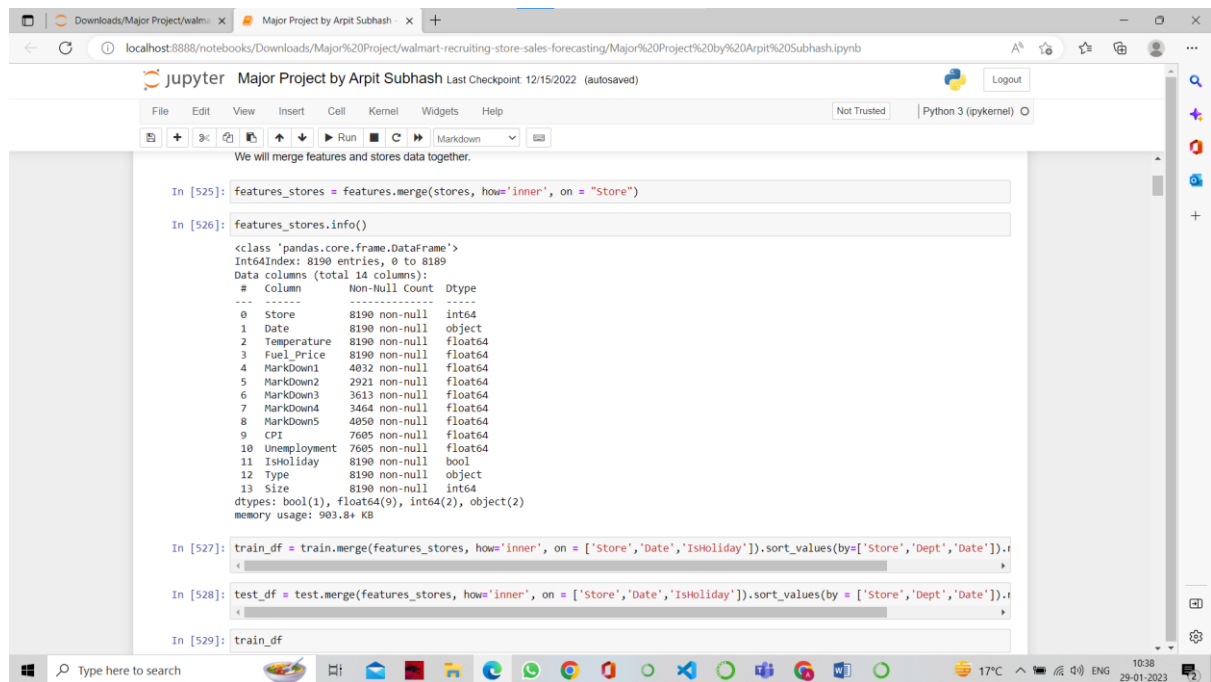
```
In [525]: features_stores = features.merge(stores, how='inner', on = "Store")
```

```
In [526]: features_stores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8190 entries, 0 to 8189
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  --
 0   Store         8190 non-null   int64  
 1   Date          8190 non-null   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
```

3. EDA – First of all I converted the date columns to date time, then I added some basic features related to all types of days that is, data, time, week, month, year and I merged four data sets to form two

new data sets that is, train with features and stores and test with features and stores.

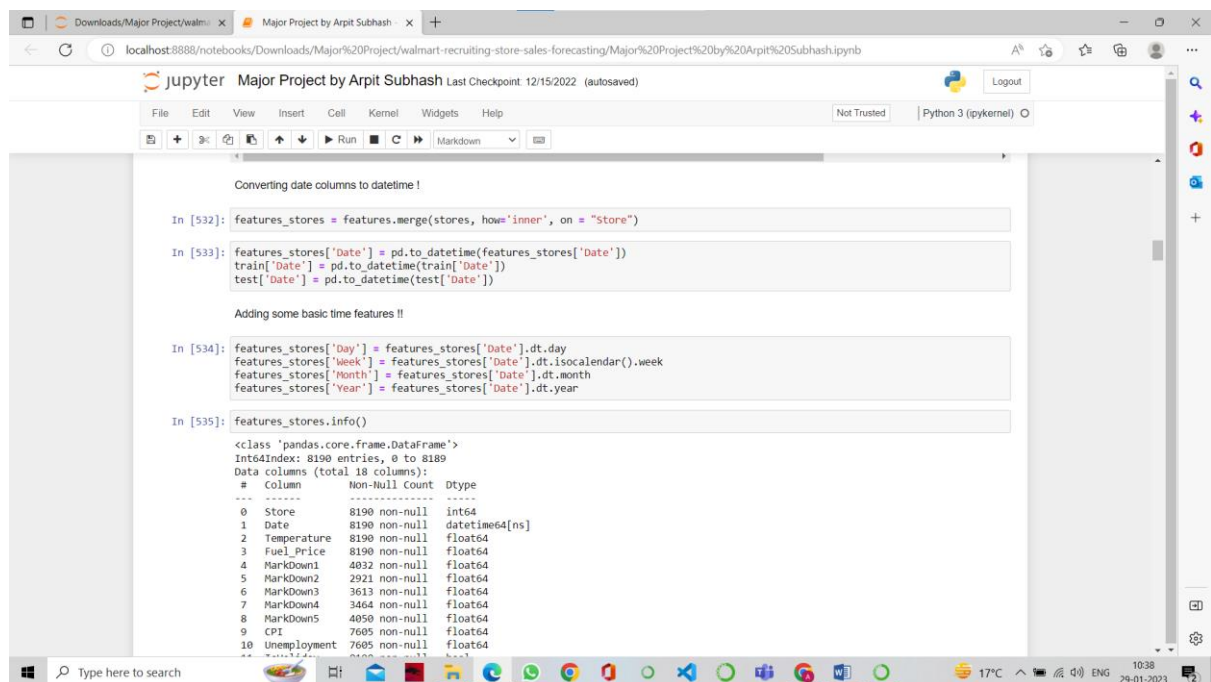


This screenshot shows a Jupyter Notebook interface with the title 'Major Project by Arpit Subhash'. The notebook is running on a local host. The code in the cells performs the following actions:

- Cell 525: `features_stores = features.merge(stores, how='inner', on = "Store")`
- Cell 526: `features_stores.info()`
- Cell 527: `train_df = train.merge(features_stores, how='inner', on = ['Store', 'Date', 'IsHoliday']).sort_values(by= ['Store', 'Dept', 'Date'])`
- Cell 528: `test_df = test.merge(features_stores, how='inner', on = ['Store', 'Date', 'IsHoliday']).sort_values(by= ['Store', 'Dept', 'Date'])`
- Cell 529: `train_df`

The output of cell 526 shows the following information:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8190 entries, 0 to 8189
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0    Store      8190 non-null   int64
1    Date       8190 non-null   object
2    Temperature 8190 non-null   float64
3    Fuel_Price 8190 non-null   float64
4    Markdown1  4832 non-null   float64
5    Markdown2  2921 non-null   float64
6    Markdown3  3613 non-null   float64
7    Markdown4  3464 non-null   float64
8    Markdown5  4050 non-null   float64
9    CPI        7605 non-null   float64
10   Unemployment 7605 non-null   float64
11   IsHoliday  8190 non-null   bool
12   Type       8190 non-null   object
13   Size       8190 non-null   int64
dtypes: bool(1), float64(9), int64(2), object(2)
memory usage: 903.8+ KB
```



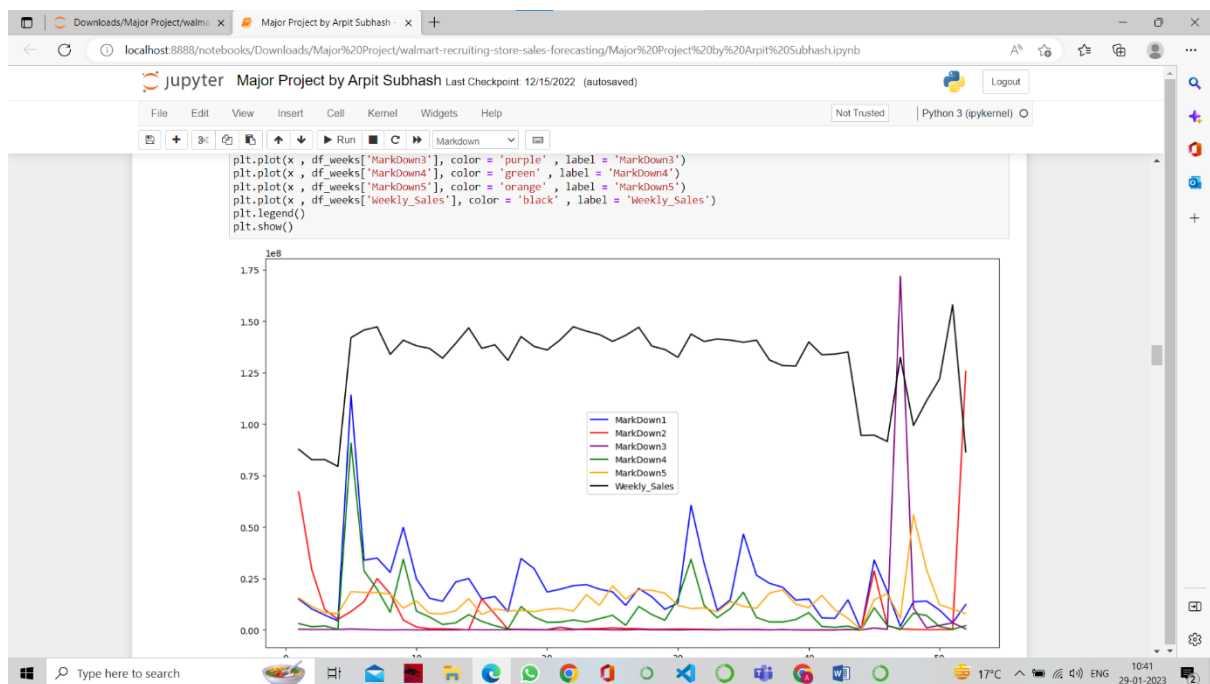
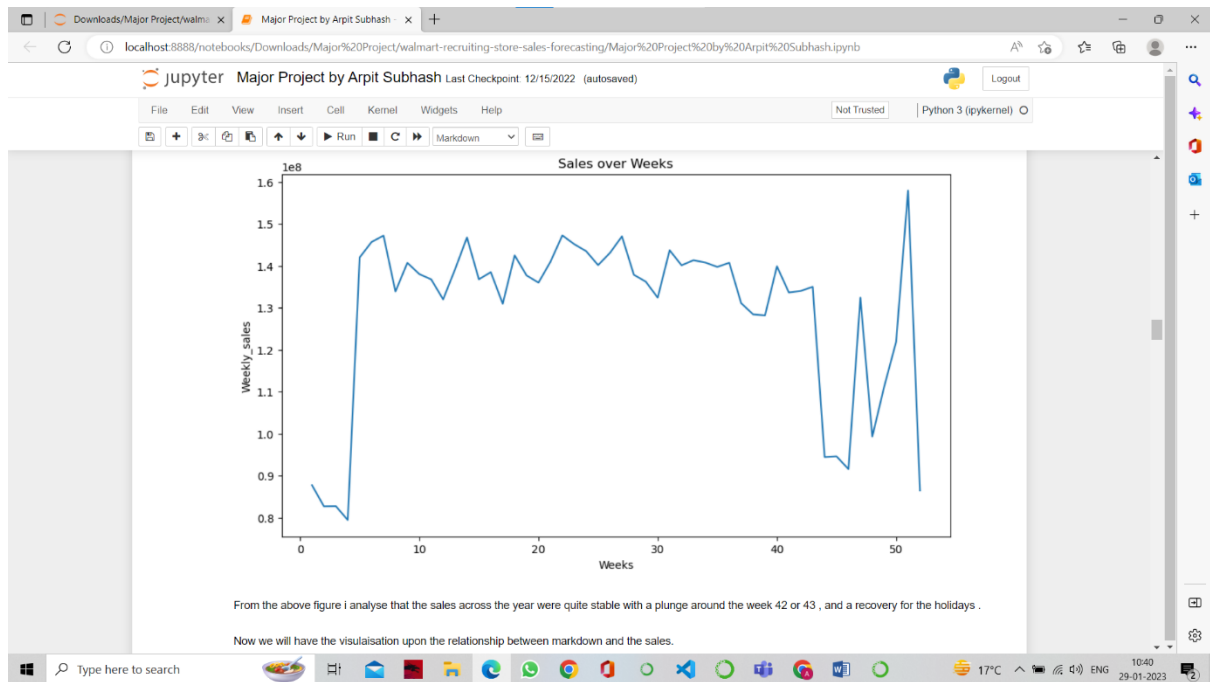
This screenshot shows the continuation of the Jupyter Notebook. The code in the cells performs the following actions:

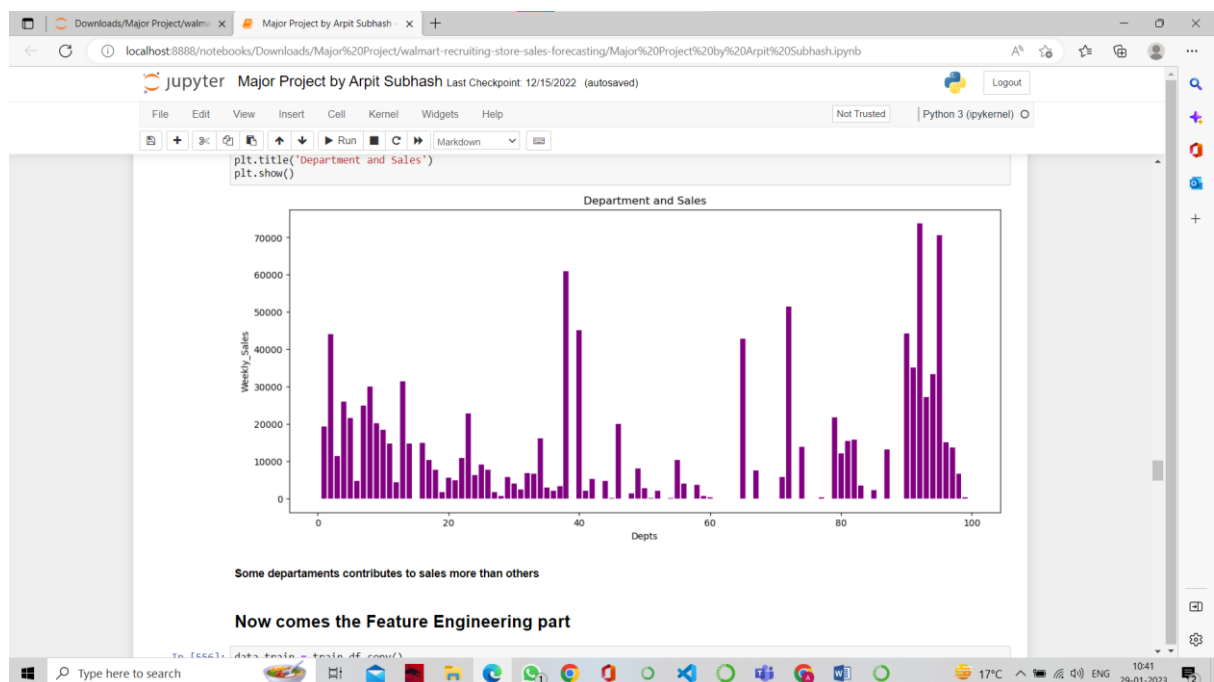
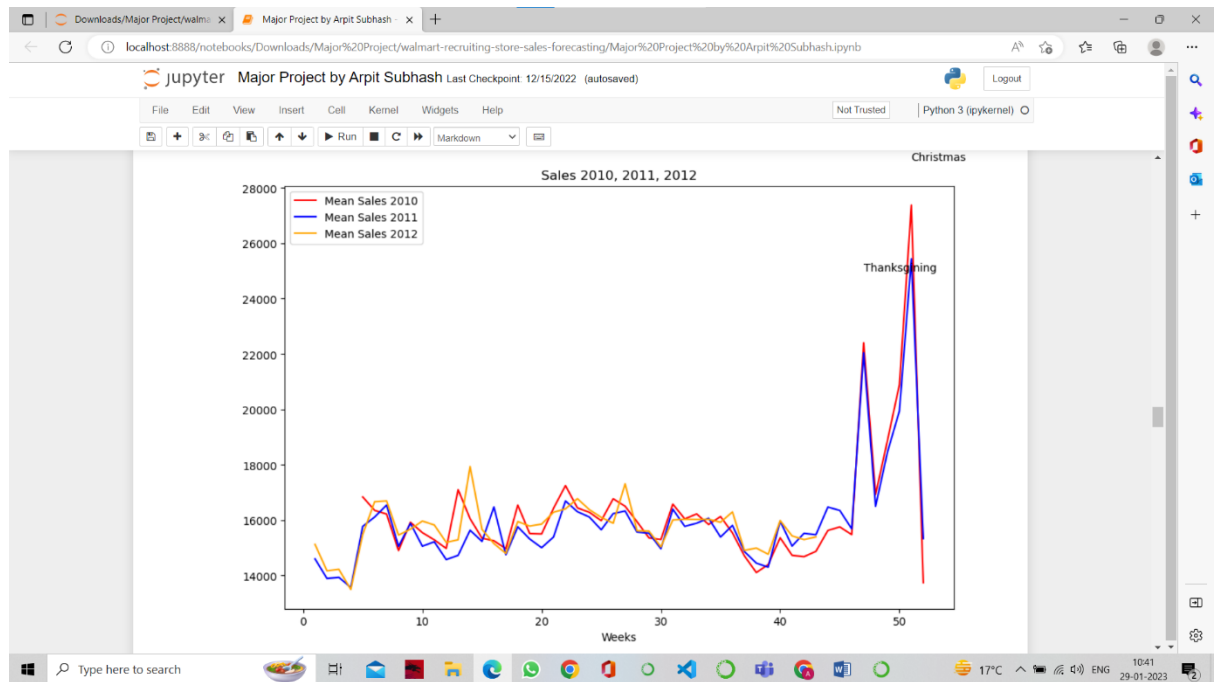
- Cell 532: `features_stores = features.merge(stores, how='inner', on = "Store")`
- Cell 533: `features_stores['Date'] = pd.to_datetime(features_stores['Date'])`
`train['Date'] = pd.to_datetime(train['Date'])`
`test['Date'] = pd.to_datetime(test['Date'])`
- Cell 534: `features_stores['Day'] = features_stores['Date'].dt.day`
`features_stores['Week'] = features_stores['Date'].dt.isocalendar().week`
`features_stores['Month'] = features_stores['Date'].dt.month`
`features_stores['Year'] = features_stores['Date'].dt.year`
- Cell 535: `features_stores.info()`

The output of cell 535 shows the following information:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8190 entries, 0 to 8189
Data columns (total 18 columns):
#   Column      Non-Null Count  Dtype
---  -
0    Store      8190 non-null   int64
1    Date       8190 non-null   datetime64[ns]
2    Temperature 8190 non-null   float64
3    Fuel_Price 8190 non-null   float64
4    Markdown1  4832 non-null   float64
5    Markdown2  2921 non-null   float64
6    Markdown3  3613 non-null   float64
7    Markdown4  3464 non-null   float64
8    Markdown5  4050 non-null   float64
9    CPI        7605 non-null   float64
10   Unemployment 7605 non-null   float64
11   IsHoliday  8190 non-null   bool
12   Type       8190 non-null   object
13   Size       8190 non-null   int64
14   Day        8190 non-null   int64
15   Week       8190 non-null   int64
16   Month      8190 non-null   int64
17   Year       8190 non-null   int64
dtypes: bool(1), datetime64(1), float64(9), int64(7), object(1)
memory usage: 1.1+ MB
```

4. Data Visualisation – sales over weeks, sales vs all the markdowns, sales over weeks of year 2010, 2011, 2012 and relationship between store size and sales and relationship between department and sales.





5. Feature Engineering – Since Thanksgiving and Christmas were the most important holidays, I tried some basic feature engineering on the days like Super Bowl day and Labor day. Also I added all the markdown to form two parts data train and data test.

The screenshot shows a Jupyter Notebook with the following code and text:

Since Thanksgiving and christmas are the most important holidays , so we will try some feature engineerings on these features and also Superbowl day and Labor day.

```
In [557]: data_train['Days_to_Thanksgiving'] = (pd.to_datetime(train_df['Year']).astype(str)+"-11-24", format="%Y-%m-%d") - pd.to_datetime(train_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d") - pd.to_datetime(train_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d")
data_train['Days_to_Christmas'] = (pd.to_datetime(train_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d") - pd.to_datetime(train_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d") - pd.to_datetime(train_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d")

In [558]: data_test['Days_to_Thanksgiving'] = (pd.to_datetime(test_df['Year']).astype(str)+"-11-24", format="%Y-%m-%d") - pd.to_datetime(test_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d") - pd.to_datetime(test_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d")
data_test['Days_to_Christmas'] = (pd.to_datetime(test_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d") - pd.to_datetime(test_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d") - pd.to_datetime(test_df['Year']).astype(str)+"-12-24", format="%Y-%m-%d")

In [559]: data_train['SuperBowlWeek'] = train_df['week'].apply(lambda x: 1 if x == 6 else 0)
data_train['LaborDay'] = train_df['week'].apply(lambda x: 1 if x == 36 else 0)
data_train['Thanksgiving'] = train_df['week'].apply(lambda x: 1 if x == 47 else 0)
data_train['Christmas'] = train_df['week'].apply(lambda x: 1 if x == 52 else 0)

In [560]: data_test['SuperBowlWeek'] = test_df['week'].apply(lambda x: 1 if x == 6 else 0)
data_test['LaborDay'] = test_df['week'].apply(lambda x: 1 if x == 36 else 0)
data_test['Thanksgiving'] = test_df['week'].apply(lambda x: 1 if x == 47 else 0)
data_test['Christmas'] = test_df['week'].apply(lambda x: 1 if x == 52 else 0)
```

Markdowns

```
In [561]: data_train['MarkDownsSum'] = train_df['MarkDown1'] + train_df['MarkDown2'] + train_df['MarkDown3'] + train_df['MarkDown4'] + train_df['MarkDown5']
data_test['MarkDownsSum'] = test_df['MarkDown1'] + test_df['MarkDown2'] + test_df['MarkDown3'] + test_df['MarkDown4'] + test_df['MarkDown5']
```

Filling missing values

```
In [563]: data_train.isna().sum()[data_train.isna().sum() > 0].sort_values(ascending=False)
```

6. Preprocessing – Since all the Markdowns were having null values so I filled all the null values.

The screenshot shows the continuation of the Jupyter Notebook with the following code and output:

Filling missing values

```
In [563]: data_train.isna().sum()[data_train.isna().sum() > 0].sort_values(ascending=False)
```

Out[563]:

MarkDownsSum	324514
MarkDown2	310322
MarkDown4	286603
MarkDown3	284479
MarkDown1	270889
MarkDown5	270138
dtype:	int64

```
In [564]: data_test.isna().sum()[data_test.isna().sum() > 0].sort_values(ascending=False)
```

Out[564]:

CPI	38162
Unemployment	38162
MarkDownsSum	37457
MarkDown2	28627
MarkDown4	12888
MarkDown3	9829
MarkDown1	149
dtype:	int64

```
In [565]: data_train.fillna(0, inplace=True)
In [566]: data_test['CPI'].fillna(data_test['CPI'].mean(),inplace=True)
data_test['Unemployment'].fillna(data_test['Unemployment'].mean(),inplace=True)
In [567]: data_test.fillna(0, inplace=True)
```

Now we will encode the Categorical data

7. Encoding the categorical data – Like if there is holiday then it is 1 otherwise 0 and some more.

The screenshot shows a Jupyter Notebook with the following code and output:

```
Now we will encode the Categorical data
```

```
In [568]: data_train['IsHoliday'] = data_train['IsHoliday'].apply(lambda x: 1 if x == True else 0)
data_test['IsHoliday'] = data_test['IsHoliday'].apply(lambda x: 1 if x == True else 0)

In [569]: data_train['Type'] = data_train['Type'].apply(lambda x: 1 if x == 'A' else (2 if x == 'B' else 3))
data_test['Type'] = data_test['Type'].apply(lambda x: 1 if x == 'A' else (2 if x == 'B' else 3))
```

```
Now comes the features selecting part
```

```
In [570]: features = [feature for feature in data_train.columns if feature not in ('Date', 'Weekly_Sales')]

In [571]: X = data_train[features].copy()
y = data_train.Weekly_Sales.copy()

In [572]: X.shape
Out[572]: (421570, 25)

In [573]: y.shape
Out[573]: (421570,)

In [574]: data_sample = data_train.copy().sample(frac=0.25)
X_sample = data_sample.drop(['Date', 'Weekly_Sales'], axis='columns').copy()
y_sample = data_sample.Weekly_Sales.copy()

In [575]: X_sample.shape
Out[575]: (105392, 25)
```

8. Feature Selection - First I selected the top 5 features that were Dept, Size, Store, CPI, Markdown3 by using one library eli5.

The screenshot shows a Jupyter Notebook with the following code and output:

```
In [581]: features_weights = eli5.show_weights(permutation_test=permutation_test, feature_names=X_test.columns.tolist())
features_weights
```

```
Out[581]:
```

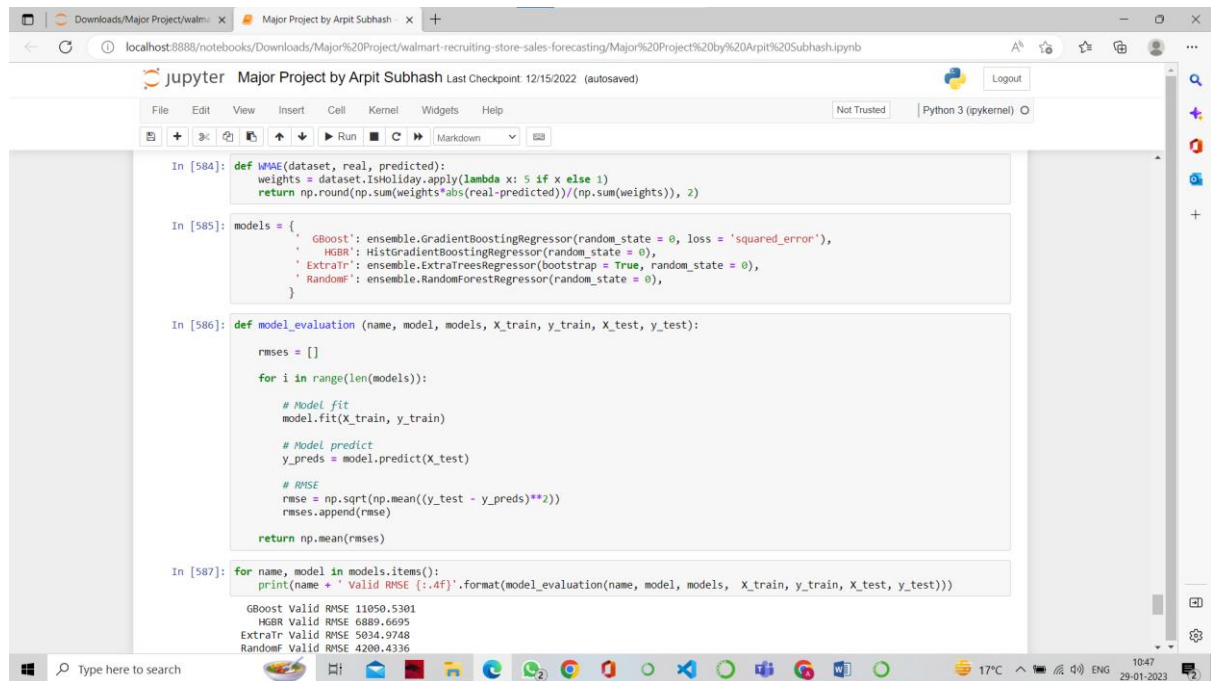
Weight	Feature
1.1769 ± 0.0179	Dept
0.3444 ± 0.0139	Size
0.0443 ± 0.0018	Store
0.0126 ± 0.0007	CPI
0.0097 ± 0.0023	Markdown3
0.0059 ± 0.0013	Thanksgiving
0.0042 ± 0.0003	Type
0.0035 ± 0.0008	Days_to_Thanksgiving
0.0029 ± 0.0006	Week
0.0025 ± 0.0002	Days_to_Christmas
0.0007 ± 0.0001	Unemployment
0.0007 ± 0.0000	Christmas
0.0002 ± 0.0000	Markdown4
0.0001 ± 0.0001	Day
0.0000 ± 0.0000	Month
0.0000 ± 0.0001	Temperature
0.0000 ± 0.0000	Markdown1
0 ± 0.0000	Fuel_Price
0 ± 0.0000	LaborDay
0 ± 0.0000	IsHoliday
0 ± 0.0000	Year
0 ± 0.0000	Markdown2
0 ± 0.0000	SuperBowlWeek
0 ± 0.0000	Markdown5
-0.0000 ± 0.0000	MarkdownsSum

```
In [582]: f_importances = pd.Series(dict(zip(X_test.columns.tolist(), perm.feature_importances_)).sort_values(ascending=False))
f_importances
```

```
Out[582]:
```

Dept	1.176899
Size	0.344450
Store	0.044278
CPI	0.012603
Markdown3	0.009741
Thanksgiving	0.005913
Type	0.004222
Days_to_Thanksgiving	0.003534
Week	0.002900

9. Models – Then I used the models like GBoost, HGBR, ExtraTrees, Random Forest and it comes out that the Random Forest was giving most accurate value or the least error.



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [584]: def WME(dataset, real, predicted):
weights = dataset.IsHoliday.apply(lambda x: 5 if x else 1)
return np.round(np.sum(weights*abs(real-predicted))/(np.sum(weights)), 2)

In [585]: models = {
    'GBoost': ensemble.GradientBoostingRegressor(random_state = 0, loss = 'squared_error'),
    'HGBR': HistGradientBoostingRegressor(random_state = 0),
    'ExtraTr': ensemble.ExtraTreesRegressor(bootstrap = True, random_state = 0),
    'Random': ensemble.RandomForestRegressor(random_state = 0),
}

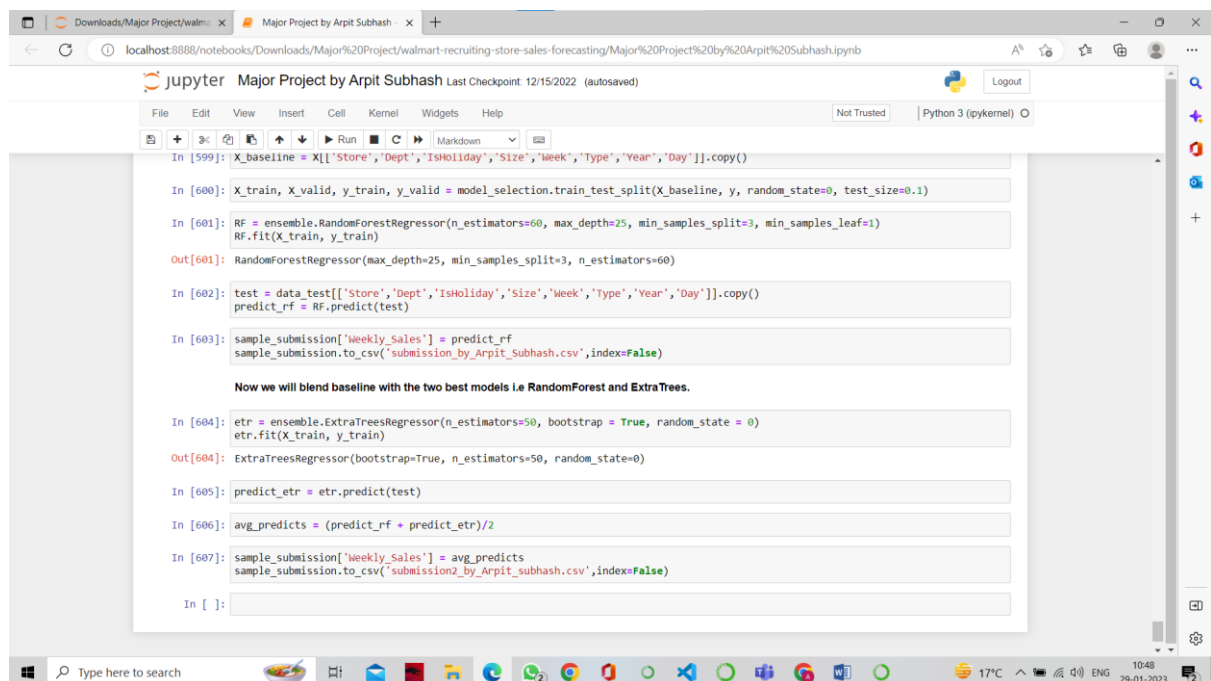
In [586]: def model_evaluation (name, model, models, X_train, y_train, X_test, y_test):
    rmses = []
    for i in range(len(models)):
        # Model fit
        model.fit(X_train, y_train)
        # Model predict
        y_preds = model.predict(X_test)
        # RMSE
        rmse = np.sqrt(np.mean((y_test - y_preds)**2))
        rmses.append(rmse)
    return np.mean(rmses)

In [587]: for name, model in models.items():
    print(name + " Valid RMSE {:.4f}".format(model_evaluation(name, model, models, X_train, y_train, X_test, y_test)))
```

Output:

```
GBoost Valid RMSE 11050.5301
HGBR Valid RMSE 6889.6695
ExtraTr Valid RMSE 5034.9748
Random Valid RMSE 4200.4336
```

10. Baseline model – At last I created a baseline model with Random Forest and Extra Trees Regressor.



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [599]: X_baseline = X[['Store', 'Dept', 'IsHoliday', 'Size', 'Week', 'Type', 'Year', 'Day']].copy()

In [600]: X_train, X_valid, y_train, y_valid = model_selection.train_test_split(X_baseline, y, random_state=0, test_size=0.1)

In [601]: RF = ensemble.RandomForestRegressor(n_estimators=60, max_depth=25, min_samples_split=3, min_samples_leaf=1)
RF.fit(X_train, y_train)

Out[601]: RandomForestRegressor(max_depth=25, min_samples_split=3, n_estimators=60)

In [602]: test = data_test[['Store', 'Dept', 'IsHoliday', 'Size', 'Week', 'Type', 'Year', 'Day']].copy()
predict_rf = RF.predict(test)

In [603]: sample_submission['weekly_sales'] = predict_rf
sample_submission.to_csv('submission_by_Arpit_Subhash.csv', index=False)

Now we will blend baseline with the two best models i.e RandomForest and ExtraTrees.

In [604]: etr = ensemble.ExtraTreesRegressor(n_estimators=50, bootstrap = True, random_state = 0)
etr.fit(X_train, y_train)

Out[604]: ExtraTreesRegressor(bootstrap=True, n_estimators=50, random_state=0)

In [605]: predict_etr = etr.predict(test)

In [606]: avg_predicts = (predict_rf + predict_etr)/2

In [607]: sample_submission['weekly_sales'] = avg_predicts
sample_submission.to_csv('submission2_by_Arpit_subhash.csv', index=False)

In [ ]:
```