Walmart Weekly Sales Prediction on holidays

Team members: Olaoluwa Dara, Adedeji Yusuff, Yousef, Abdoulla Mohamed, David Jose Cortes.

OBJECTIVE: We are working on Walmart data to show the relationship between sales on holidays and other days.

It is expected that sales during holidays would be more than sales during non-holidays.

In [275…

```
#Libraries Importation
import pandas as pd
import numpy as np
from numpy import *
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#Handy for debugging
import gc
import time
import warnings
import os

#Date stuff
from datetime import datetime
from datetime import timedelta

#Preprocessing splitting algorithms
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split


#Model importation
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

#Nice graphing tools
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.offline as py
import plotly.tools as tls
import plotly.graph_objs as go
import plotly.tools as tls

#Machine learning tools
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.utils.validation import check_X_y, check_is_fitted
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import log_loss
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
from scipy import sparse
```

In [276…

```
#Importing Data
df = pd.read_csv(r'C:\MSc\Pattern Recognition\Walmart_data.csv')
```

CHECKING DATA 1. Showing data for first 5 entries

2. Showing data for last 5 entries

In [277…

```
#View the data
df
```

Out[277…

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Mark
0	26	92	8/26/2011	87235.57	False	61.10	3.796	NaN	
1	34	22	3/25/2011	5945.97	False	53.11	3.480	NaN	
2	21	28	12/3/2010	1219.89	False	50.43	2.708	NaN	
3	8	9	9/17/2010	11972.71	False	75.32	2.582	NaN	
4	19	55	5/18/2012	8271.82	False	58.81	4.029	12613.98	
…	…	…	…	…	…	…	…	…	
282446	27	18	10/19/2012	20775.91	False	56.53	4.153	2639.32	
282447	39	36	5/21/2010	5350.00	False	76.67	2.826	NaN	
282448	14	29	4/30/2010	10939.87	False	53.15	2.921	NaN	
282449	15	90	7/1/2011	5013.89	False	67.43	3.916	NaN	
282450	13	32	10/14/2011	8103.12	False	51.74	3.567	NaN	

282451 rows × 16 columns

In [278…

```
#Display the Top 5 rows and headers
df.head()
```

Out[278…

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2
0	26	92	8/26/2011	87235.57	False	61.10	3.796	NaN	NaN

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2
1	34	22	3/25/2011	5945.97	False	53.11	3.480	NaN	NaN
2	21	28	12/3/2010	1219.89	False	50.43	2.708	NaN	NaN
3	8	9	9/17/2010	11972.71	False	75.32	2.582	NaN	NaN
4	19	55	5/18/2012	8271.82	False	58.81	4.029	12613.98	NaN

In [279…

```
df.tail()
```

Out[279…

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	Mark
282446	27	18	10/19/2012	20775.91	False	56.53	4.153	2639.32	
282447	39	36	5/21/2010	5350.00	False	76.67	2.826	NaN	
282448	14	29	4/30/2010	10939.87	False	53.15	2.921	NaN	
282449	15	90	7/1/2011	5013.89	False	67.43	3.916	NaN	
282450	13	32	10/14/2011	8103.12	False	51.74	3.567	NaN	

Showing Statistical information about dataset

In [280…

```
df.describe()
```

Out[280…

	Store	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	Ma
count	282451.000000	282451.000000	282451.000000	282451.000000	282451.000000	100520.000000	742
mean	22.193166	44.286138	15983.429692	60.113640	3.360300	7246.077559	33
std	12.782138	30.503641	22661.092494	18.446485	0.458602	8254.606267	94
min	1.000000	1.000000	-4988.940000	-2.060000	2.472000	0.270000	-2
25%	11.000000	18.000000	2079.330000	46.780000	2.932000	2241.190000	
50%	22.000000	38.000000	7616.550000	62.150000	3.452000	5363.520000	1
75%	33.000000	74.000000	20245.745000	74.290000	3.737000	9235.590000	19
max	45.000000	99.000000	693099.360000	100.140000	4.468000	88646.760000	1045

Feature Engineering: Year & Month

Seperating years from Month and days in order to work with particular days that is either holidays and non-holidays

In [281…
```
#Breaking the dates into Months and Years
df['Year'] = df['Date'].apply(lambda x: x[-4:])
df['Month'] = df['Date'].apply(lambda x: x[3:6])

# Drop the original date column
df = df.drop('Date',  axis=1)
```

In [282…
```
#View the data
df
```

Out[282…

	Store	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	Mar
0	26	92	87235.57	False	61.10	3.796	NaN	NaN	
1	34	22	5945.97	False	53.11	3.480	NaN	NaN	
2	21	28	1219.89	False	50.43	2.708	NaN	NaN	
3	8	9	11972.71	False	75.32	2.582	NaN	NaN	
4	19	55	8271.82	False	58.81	4.029	12613.98	NaN	
…	…	…	…	…	…	…	…	…	
282446	27	18	20775.91	False	56.53	4.153	2639.32	NaN	
282447	39	36	5350.00	False	76.67	2.826	NaN	NaN	
282448	14	29	10939.87	False	53.15	2.921	NaN	NaN	
282449	15	90	5013.89	False	67.43	3.916	NaN	NaN	
282450	13	32	8103.12	False	51.74	3.567	NaN	NaN	

282451 rows × 17 columns

Visualization: Correlations between each columns

This is to show the relationship between all the features and sales, doing this will keep us informed about strong relation between holidays and sales.

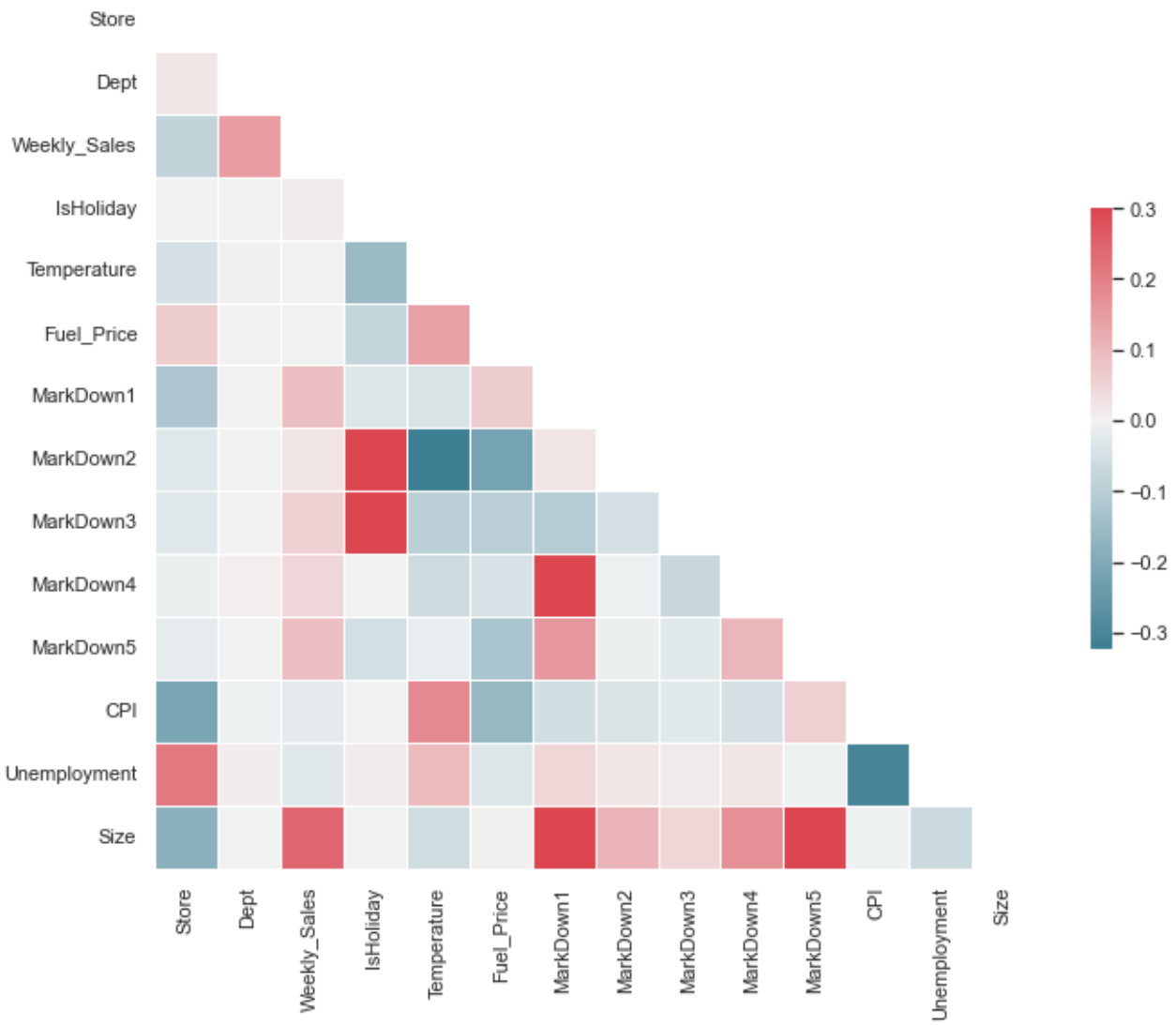
In [283…
```
# corr = df.corr()

# plt.figure(figsize = (12, 10))
# sns.heatmap(corr, annot = True, vmin = -1.0, cmap = 'shrink')
# plt.show()

sns.set(style="white")

# Compute the correlation matrix
corr = df.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
```

```
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
C:\Users\adede\AppData\Local\Temp/ipykernel_43400/3631951013.py:13: DeprecationWarning:

`np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bo
ol` by itself. Doing this will not modify any behavior and is safe. If you specifically
wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
se/1.20.0-notes.html#deprecations
```

Out[283…

```
<AxesSubplot:>
```



In [284…

```
#Dropping useless columns to have a better view of the correlation between each feature
```

```
df = df.drop(['MarkDown1','MarkDown2','MarkDown3','MarkDown4','MarkDown5'],  axis=1)
```

In [285…

```
df
```

Out[285…

	Store	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemployment	Ty
0	26	92	87235.57	False	61.10	3.796	136.213613	7.767	
1	34	22	5945.97	False	53.11	3.480	128.616064	10.398	
2	21	28	1219.89	False	50.43	2.708	211.265543	8.163	
3	8	9	11972.71	False	75.32	2.582	214.878556	6.315	
4	19	55	8271.82	False	58.81	4.029	138.106581	8.150	
…	…	…	…	…	…	…	…	…	
282446	27	18	20775.91	False	56.53	4.153	142.863363	8.000	
282447	39	36	5350.00	False	76.67	2.826	209.392294	8.464	
282448	14	29	10939.87	False	53.15	2.921	181.662036	8.899	
282449	15	90	5013.89	False	67.43	3.916	135.446800	7.806	
282450	13	32	8103.12	False	51.74	3.567	129.770645	6.392	

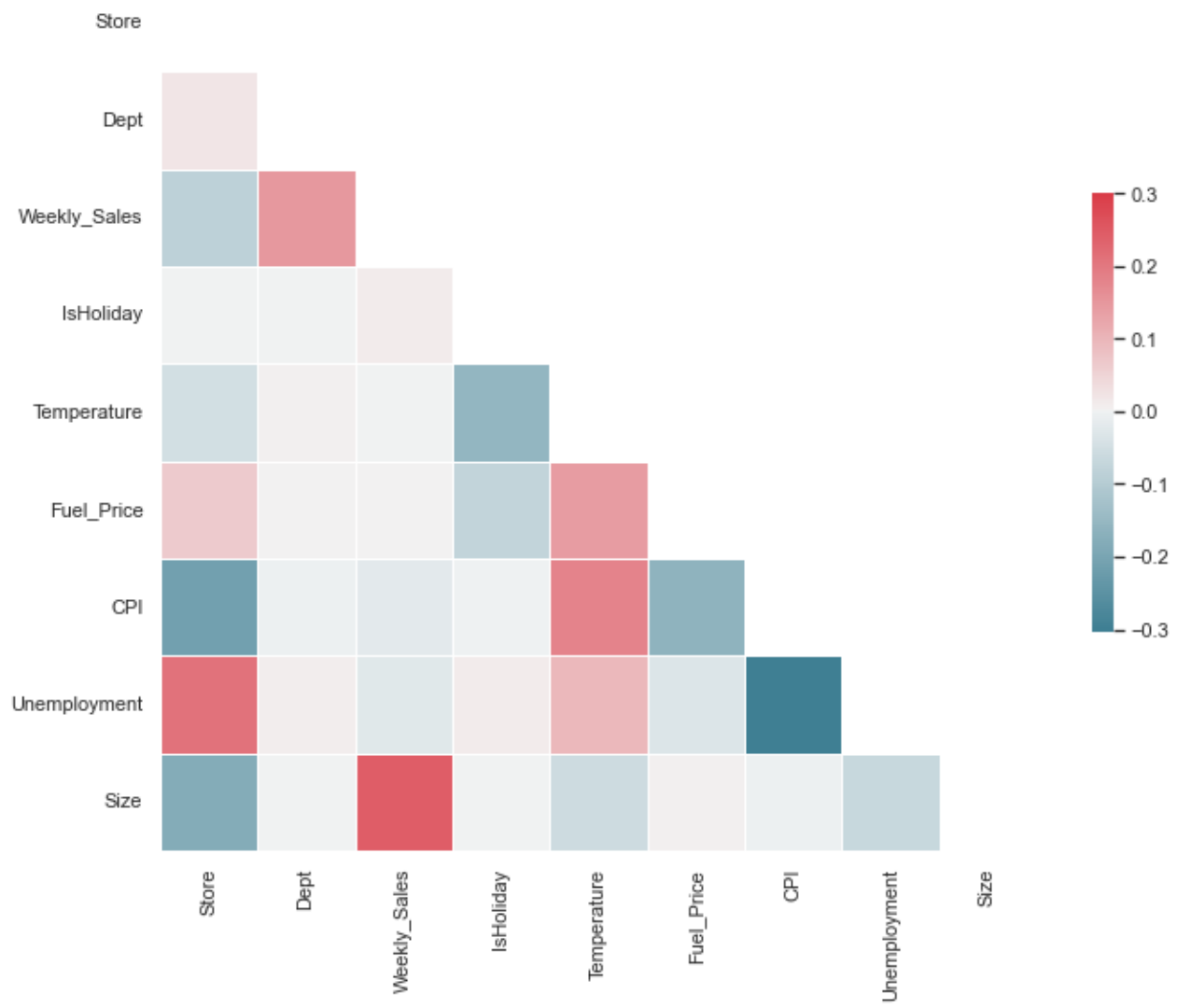
282451 rows × 12 columns

In [286…

```
sns.set(style="white")

# Compute the correlation matrix
corr = df.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
C:\Users\adede\AppData\Local\Temp/ipykernel_43400/3768685617.py:7: DeprecationWarning:

`np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bo
ol` by itself. Doing this will not modify any behavior and is safe. If you specifically
wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
se/1.20.0-notes.html#deprecations
```

Out[286… <AxesSubplot:>



There is a strong positive correlation between Weekly sales and size while there is a strong negative correlation between unemployment and CPI which is of no surprise.

Encoding Store Column

Reducing data to weekly sales that has more than 300,000

In [287…

```
df.loc[df['Weekly_Sales'] >300000]
```

Out[287…

	Store	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemployment	Ty
29352	23	72	369830.98	True	34.95	3.070	132.836933	5.287	
41292	39	72	339700.62	True	67.75	2.735	210.515276	8.476	
66187	18	72	353008.64	True	40.81	3.070	132.836933	9.331	
83588	10	72	630999.19	True	60.68	3.760	129.836400	7.874	

	Store	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemployment	Ty
93294	4	72	381072.11	True	48.08	2.752	126.669267	7.127	
97641	10	72	404245.03	False	57.06	3.236	126.983581	9.003	
116197	27	72	368484.19	True	47.88	3.689	140.421786	7.906	
121958	12	72	359995.60	True	47.66	3.162	126.669267	14.313	
140076	22	72	345532.23	True	46.28	3.536	140.421786	7.706	
150574	18	72	305161.38	True	41.97	3.536	136.478800	8.471	
161541	14	72	375948.31	True	48.71	3.492	188.350400	8.523	
171371	4	7	318422.01	False	35.92	3.103	129.984548	5.143	
193697	35	72	649770.18	True	47.88	3.492	140.421786	8.745	
206418	6	72	342578.65	True	65.79	2.735	213.267296	7.007	
210430	6	72	326866.60	True	62.78	3.236	220.041741	6.551	
221706	12	72	360140.66	True	53.25	3.622	129.836400	12.890	
236552	28	72	355356.39	True	47.66	3.162	126.669267	14.313	
258653	10	72	693099.36	True	55.33	3.162	126.669267	9.003	
267714	14	72	313933.22	False	30.59	3.141	182.544590	8.724	
268457	39	72	351553.98	True	66.36	3.236	217.181253	7.716	
280060	22	72	393705.20	True	44.61	3.070	136.689571	8.572	

CLEANING DATA

In [288…

```
df.isnull().sum()
```

Out[288…

```
Store           0
Dept            0
Weekly_Sales    0
IsHoliday       0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
Type            0
Size            0
Year            0
Month           0
dtype: int64
```

In [289…

```
df.fillna(0, inplace=True)
```

In [290…

```
def onehot_encode(df, column, prefix):
    df = df.copy()
    dummies = pd.get_dummies(df[column], prefix=prefix)
```

```
    df = pd.concat([df, dummies], axis = 1)
    df = df.drop(column, axis = 1)
    return df
```

In [291…
```
df = onehot_encode(df, column = 'Store', prefix = 'store')
```

In [292…
```
df["IsHoliday"] = df["IsHoliday"].astype(int)
```

In [293…
```
df
```

Out[293…

	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemployment	Type	
0	92	87235.57	0	61.10	3.796	136.213613	7.767	A	152
1	22	5945.97	0	53.11	3.480	128.616064	10.398	A	158
2	28	1219.89	0	50.43	2.708	211.265543	8.163	B	140
3	9	11972.71	0	75.32	2.582	214.878556	6.315	A	155
4	55	8271.82	0	58.81	4.029	138.106581	8.150	A	203
…	…	…	…	…	…	…	…	…	
282446	18	20775.91	0	56.53	4.153	142.863363	8.000	A	204
282447	36	5350.00	0	76.67	2.826	209.392294	8.464	A	184
282448	29	10939.87	0	53.15	2.921	181.662036	8.899	A	200
282449	90	5013.89	0	67.43	3.916	135.446800	7.806	B	123
282450	32	8103.12	0	51.74	3.567	129.770645	6.392	A	219

282451 rows × 56 columns

In [294…
```
df = df.drop('Type',  axis=1)
```

In [295…
```
#df = df.drop('Year',  axis=1)
```

In [296…
```
df = df.drop('Month',  axis=1)
```

In [297…
```
df
```

Out[297…

	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemployment	Size	Y
0	92	87235.57	0	61.10	3.796	136.213613	7.767	152513	2
1	22	5945.97	0	53.11	3.480	128.616064	10.398	158114	2
2	28	1219.89	0	50.43	2.708	211.265543	8.163	140167	2

	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	CPI	Unemployment	Size	Y
3	9	11972.71	0	75.32	2.582	214.878556	6.315	155078	2
4	55	8271.82	0	58.81	4.029	138.106581	8.150	203819	2
...	...	...	...	...	...	...	...	...	
282446	18	20775.91	0	56.53	4.153	142.863363	8.000	204184	2
282447	36	5350.00	0	76.67	2.826	209.392294	8.464	184109	2
282448	29	10939.87	0	53.15	2.921	181.662036	8.899	200898	2
282449	90	5013.89	0	67.43	3.916	135.446800	7.806	123737	2
282450	32	8103.12	0	51.74	3.567	129.770645	6.392	219622	2

282451 rows × 54 columns

In [298…

```
# print(df.dtypes)
```

Split & Scale Data

In [299…

```
#Assigning our data into variable X & y in a matrix form
y = df['IsHoliday'].copy()
X = df.drop('IsHoliday', axis = 1).copy()
```

In [300…

```
print(X)
```

```
        Dept  Weekly_Sales  Temperature  Fuel_Price         CPI  Unemployment  \
0         92      87235.57        61.10       3.796  136.213613         7.767
1         22       5945.97        53.11       3.480  128.616064        10.398
2         28       1219.89        50.43       2.708  211.265543         8.163
3          9      11972.71        75.32       2.582  214.878556         6.315
4         55       8271.82        58.81       4.029  138.106581         8.150
...      ...           ...          ...         ...         ...           ...
282446    18      20775.91        56.53       4.153  142.863363         8.000
282447    36       5350.00        76.67       2.826  209.392294         8.464
282448    29      10939.87        53.15       2.921  181.662036         8.899
282449    90       5013.89        67.43       3.916  135.446800         7.806
282450    32       8103.12        51.74       3.567  129.770645         6.392

          Size  Year  store_1  store_2  ...  store_36  store_37  store_38  \
0       152513  2011        0        0  ...         0         0         0
1       158114  2011        0        0  ...         0         0         0
2       140167  2010        0        0  ...         0         0         0
3       155078  2010        0        0  ...         0         0         0
4       203819  2012        0        0  ...         0         0         0
...        ...   ...      ...      ...  ...       ...       ...       ...
282446  204184  2012        0        0  ...         0         0         0
282447  184109  2010        0        0  ...         0         0         0
282448  200898  2010        0        0  ...         0         0         0
282449  123737  2011        0        0  ...         0         0         0
```

```
282450  219622  2011        0        0  ...         0         0         0

        store_39  store_40  store_41  store_42  store_43  store_44  store_45
0              0         0         0         0         0         0         0
1              0         0         0         0         0         0         0
2              0         0         0         0         0         0         0
3              0         0         0         0         0         0         0
4              0         0         0         0         0         0         0
...          ...       ...       ...       ...       ...       ...       ...
282446         0         0         0         0         0         0         0
282447         1         0         0         0         0         0         0
282448         0         0         0         0         0         0         0
282449         0         0         0         0         0         0         0
282450         0         0         0         0         0         0         0

[282451 rows x 53 columns]
```

Training data

The dataframe is divided into X_test, X_training and y_test and Y_train and 20% of dataset is used for test and 80% is used for training.

In [301…

```
#Train and Test split
#X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8)
test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
```

In [241…

```
## Feature Scaling i.e transform X so we have mean 0 at each column
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [242…

```
#Train and Test split
#X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8)
test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
```

Modeling and Training Section

In [243…

```
#Defining the models for prediction
log_model = LogisticRegression()
```

```
dec_model = DecisionTreeClassifier()
```

In [244…
```
#Importing LogisticReg classifier
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Out[244…
```
LogisticRegression(random_state=0)
```

In [254…
```
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

Out[254…
```
[[51753   713]
 [ 3917   108]]
0.9180400417765662
```

In [255…
```
#Importing naive_baye classifier
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[255…
```
GaussianNB()
```

In [256…
```
gaussian = GaussianNB() #initialization of the model
```

In [257…
```
gaussian.fit(X_train, y_train) #training the model
```

Out[257…
```
GaussianNB()
```

In [258…
```
y_pred = classifier.predict(X_test)  #training the model
y_pred #testing
```

Out[258…
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [259…
```
y_pred = gaussian.predict(X_test) #training the model
print(y_pred)#testing
```

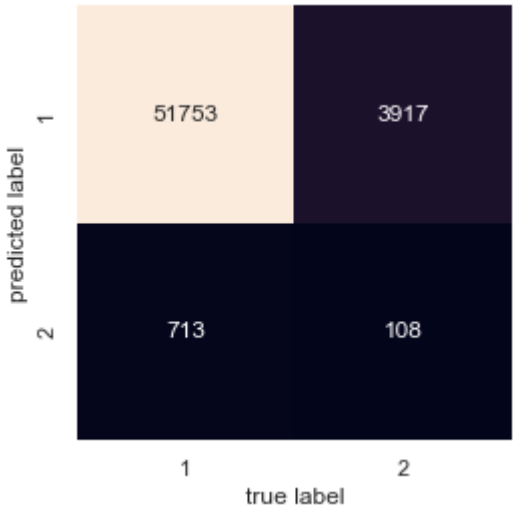
```
[0 0 0 ... 0 0 0]
```

In [260…
```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

accuracy1=accuracy_score(y_test, y_pred)
print(f"Naive Bayes' accuracy: {accuracy_score(y_test, y_pred)}")
print(classification_report(y_test, y_pred))
```

```
Naive Bayes' accuracy: 0.9180400417765662
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     52466
           1       0.13      0.03      0.04      4025

    accuracy                           0.92     56491
   macro avg       0.53      0.51      0.50     56491
weighted avg       0.87      0.92      0.89     56491
```

In [261…

```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(y_test, y_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=[1,2], yticklabels=[1,2])
plt.xlabel('true label')
plt.ylabel('predicted label');
```



In [262…

```
model=DecisionTreeClassifier(criterion= "entropy",splitter='best',max_depth=None ) #spl
model.fit(X_train,y_train)
```

Out[262…

```
DecisionTreeClassifier(criterion='entropy')
```

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

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

In [263…

```
y_pred=model.predict(X_test)
y_pred
```

Out[263…

```
array([0, 0, 0, ..., 0, 0, 0])
```

In [264…

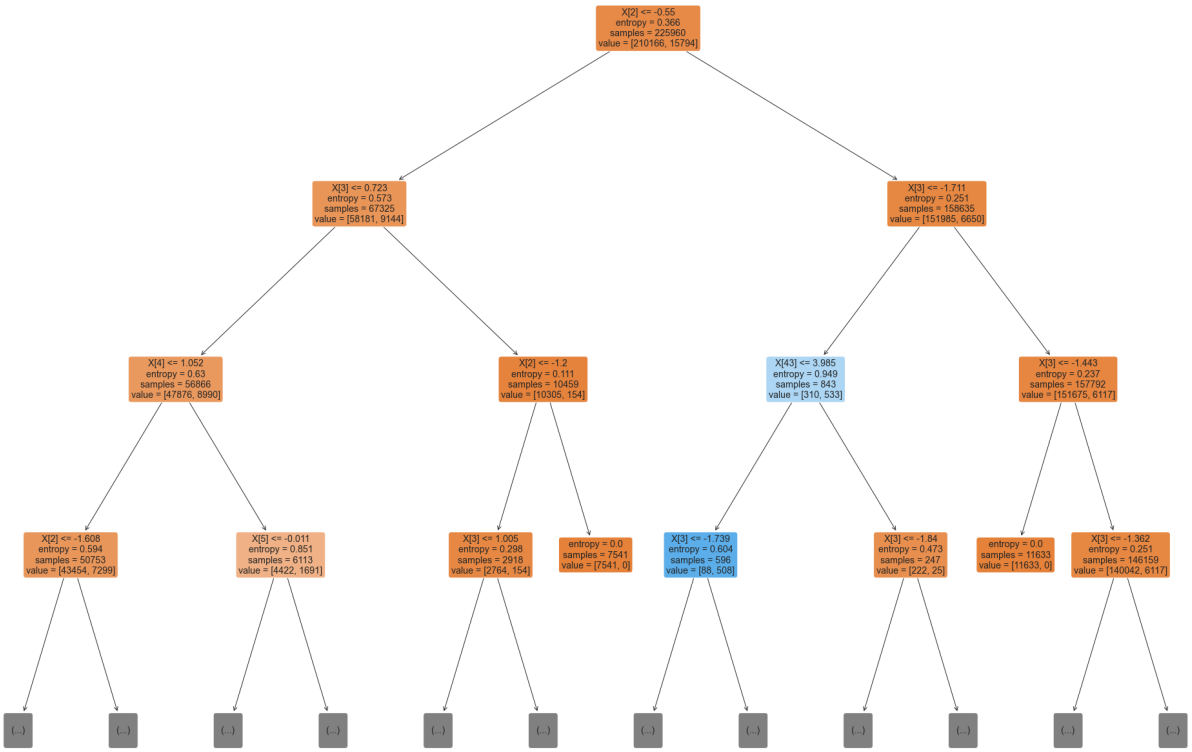
```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

```
accuracy2=accuracy_score(y_test, y_pred)
print(f"Decision Tree's accuracy: {accuracy_score(y_test, y_pred)}")
print(classification_report(y_test, y_pred))
```

```
Decision Tree's accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     52466
           1       1.00      1.00      1.00      4025

    accuracy                           1.00     56491
   macro avg       1.00      1.00      1.00     56491
weighted avg       1.00      1.00      1.00     56491
```

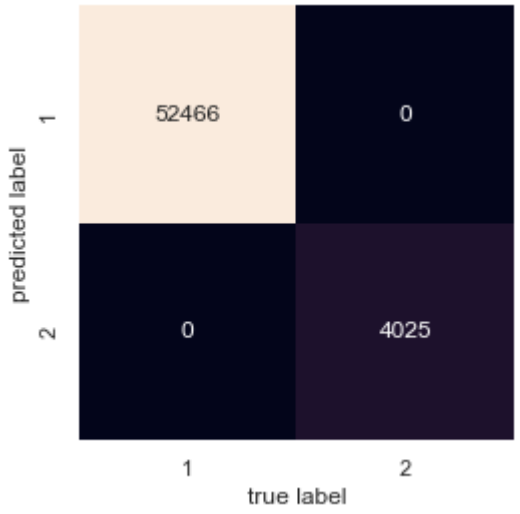
In [265…

```
from sklearn import tree
fig = plt.figure(figsize=(35,25))
tree.plot_tree(model, filled = True, rounded = True, max_depth =3)
plt.show()
```



In [266…

```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(y_test, y_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=[1,2], yticklabels=[1,2])
plt.xlabel('true label')
plt.ylabel('predicted label');
```



In [267…

```
log_model.fit(X_train, y_train)
dec_model.fit(X_train, y_train)

print("Models trained.")
```

```
Models trained.
```

Result

In [268…

```
print("   Logistic Regression Accuracy:", log_model.score(X_test, y_test))
print("         Decision Tree Accuracy:", dec_model.score(X_test, y_test))
```

```
   Logistic Regression Accuracy: 0.9287497123435592
         Decision Tree Accuracy: 1.0
```

Model Evaluation

In [269…

```
print("Accuracy of the Bayes model is {:.2f}%".format(accuracy1*100))
print("Logistic Regression Accuracy is {:.2f}%".format(accuracy1*100))
print("Accuracy of the Decision Tree model {:.2f}%".format(accuracy2*100))
```

```
Accuracy of the Bayes model is 91.80%
Logistic Regression Accuracy is 91.80%
Accuracy of the Decision Tree model 100.00%
```

In [272…

```
#Random forest model specification
regr = RandomForestRegressor(n_estimators=20, criterion='mse', max_depth=None,
                      min_samples_split=2, min_samples_leaf=1,
                      min_weight_fraction_leaf=0.0, max_features='auto',
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, bootstrap=True,
                      oob_score=False, n_jobs=1, random_state=None,
                      verbose=2, warm_start=False)

#Train on data
regr.fit(X_train, y_train.ravel())
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
building tree 1 of 20
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.7s remaining:    0.0s
building tree 2 of 20
building tree 3 of 20
building tree 4 of 20
building tree 5 of 20
building tree 6 of 20
building tree 7 of 20
building tree 8 of 20
building tree 9 of 20
building tree 10 of 20
building tree 11 of 20
building tree 12 of 20
building tree 13 of 20
building tree 14 of 20
building tree 15 of 20
building tree 16 of 20
building tree 17 of 20
building tree 18 of 20
building tree 19 of 20
building tree 20 of 20
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:  1.4min finished
```

Out[272…

```
RandomForestRegressor(n_estimators=20, n_jobs=1, verbose=2)
```

In [273…

```
selector = [
    #'Month',
    'CPI',
    'Fuel_Price',
    'Size',
    'Temperature',
    'Unemployment',
    'IsHoliday',
    'Year']
display(df[selector].describe())
display(df[selector].head())
```

	CPI	Fuel_Price	Size	Temperature	Unemployment	IsHoliday
count	282451.000000	282451.000000	282451.000000	282451.000000	282451.000000	282451.000000
mean	171.207802	3.360300	136730.073220	60.113640	7.968098	0.070168
std	39.160808	0.458602	61002.319363	18.446485	1.868070	0.255430
min	126.064000	2.472000	34875.000000	-2.060000	3.879000	0.000000
25%	132.022667	2.932000	93638.000000	46.780000	6.891000	0.000000
50%	182.350989	3.452000	140167.000000	62.150000	7.866000	0.000000
75%	212.464799	3.737000	202505.000000	74.290000	8.572000	0.000000
max	227.232807	4.468000	219622.000000	100.140000	14.313000	1.000000

	CPI	Fuel_Price	Size	Temperature	Unemployment	IsHoliday	Year
0	136.213613	3.796	152513	61.10	7.767	0	2011

	CPI	Fuel_Price	Size	Temperature	Unemployment	IsHoliday	Year
1	128.616064	3.480	158114	53.11	10.398	0	2011
2	211.265543	2.708	140167	50.43	8.163	0	2010
3	214.878556	2.582	155078	75.32	6.315	0	2010
4	138.106581	4.029	203819	58.81	8.150	0	2012

In [ ]: