Machine Learning
Superstore Sales Prediction

Chole
Ching

TABLE OF CONTENTS

01
Business Question

Business Objectives

Prediction Sales of Store

Based on the historical data of the 45 Walmart stores, and understand the overall growth

Seasonal Sales Pattern

Examine the seasonal pattern for better implementing marketing campaigns

Implication of macro-economic indictors

Significance of the macro-indicators like CPI, Unemployment rate, Fuel price etc. on the sales figure

02
Data Collection

Data Collection

kaggle

M YASSER H · UPDATED 5 MONTHS AGO

▲ 62 | New Notebook | Download (125 kB)

Walmart Dataset

Walmart Store Sales Prediction - Regression Problem

https://www.kaggle.com/datasets/yasserh/walmart-dataset

03 Preprocessing

EDA

Original Dataset:

```
df.head()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

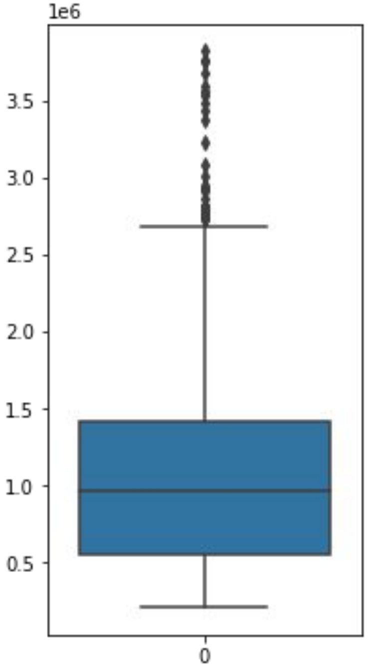
Weekly sales data of :

- 45 Walmart stores in the US
- 143 weeks from 2010-02-05 to 2012-10-26.

EDA

Univariate Analysis

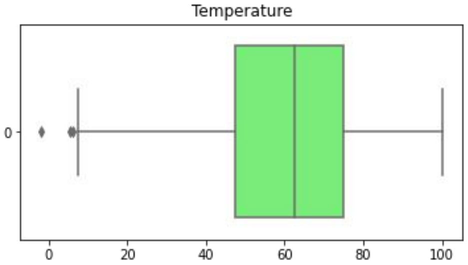
Weekly Sales

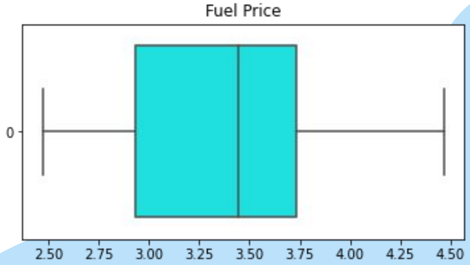


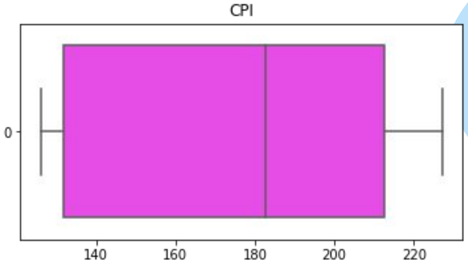
UQ: 1,420,158

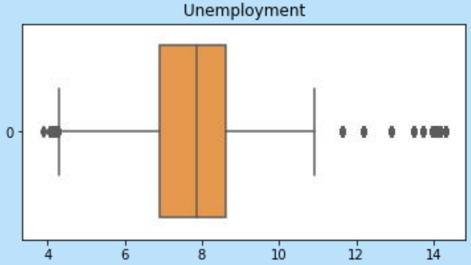
Mean: 1,046,964

LQ: 553,350





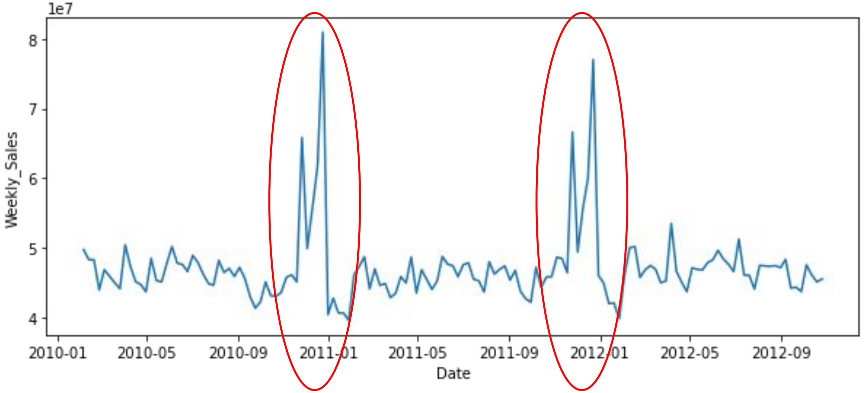


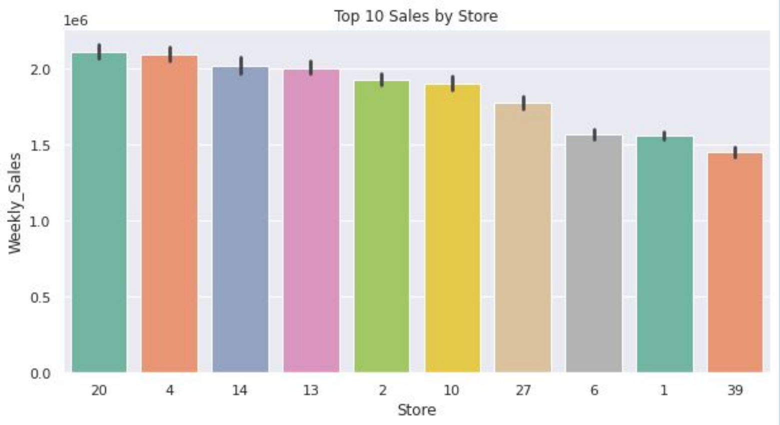


EDA

Bivariate Analysis

Weekly Sales & Date



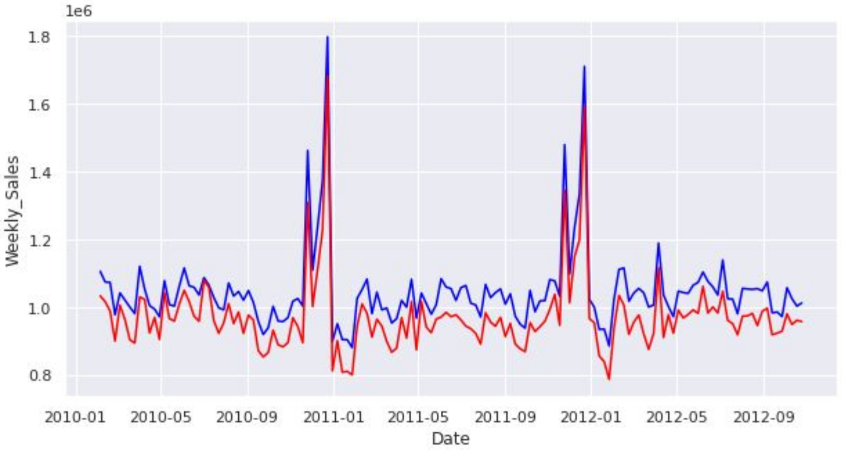


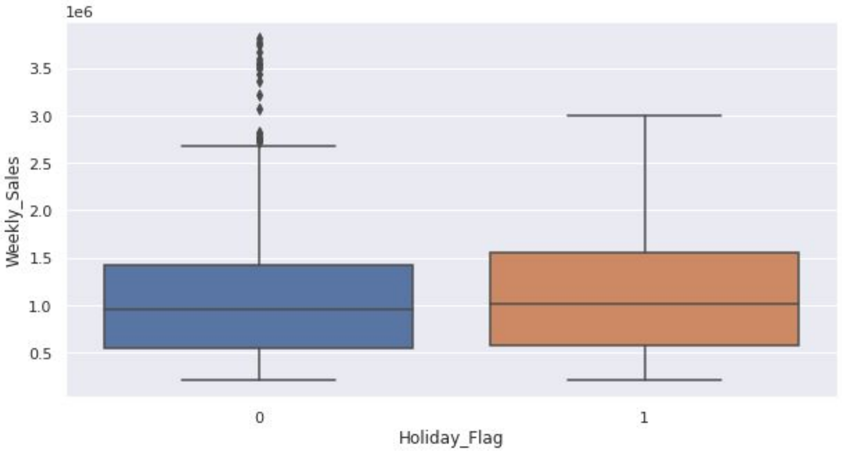
- 2 peaks at the end of every year
- Thanksgiving and Christmas

- Store no. 20 has the highest total sales during 2010-02-05 to 2012-10-26.

EDA

Bivariate Analysis



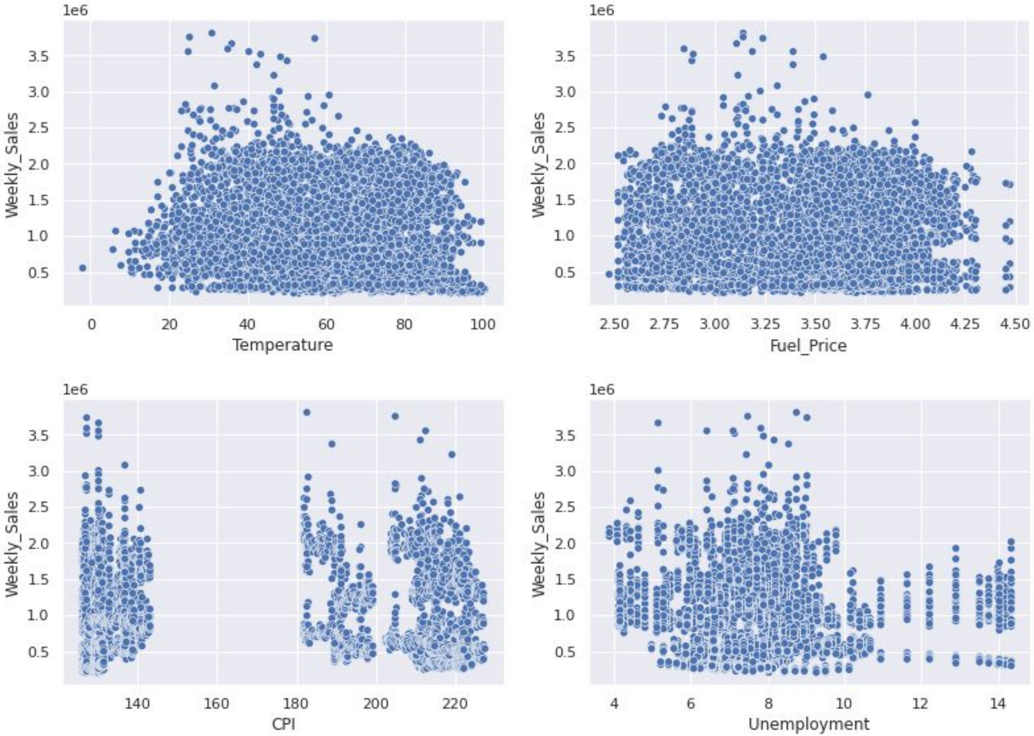


- Mean (blue line) is higher than median (red line)
- indicates that the sales among stores may vary a lot

- Holiday weeks have a higher mean but not significant.
- Non-holiday weeks have a number of outliers with much higher sales.

EDA

Bivariate Analysis



All these features show **no obvious positive/negative relationship** with Weekly Sales

Preprocessing

Time Series Models

Converting 'Date' to datetime object

```
df['Date'] = pd.to_datetime(df['Date'], format="%d-%m-%Y") #Convert the type of Date to datetime
```

Converting into time series

```
dateparse = lambda dates: pd.datetime.strptime(dates, '%Y-%m-%d')
data = pd.read_csv('/content/gdrive/MyDrive/Walmart Sales Project/data/ts_ARIMA_Walmart.csv', parse_dates=['Date'],
                   index_col='Date',date_parser=dateparse)
print ('\n Parsed Data:')
print (data.head())
```

```
 Parsed Data:
            Weekly_Sales
Date
2010-02-05   49750740.50
2010-02-12   48336677.63
2010-02-19   48276993.78
2010-02-26   43968571.13
2010-03-05   46871470.30
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: The pandas.datetime class is deprecated and will be rem
  """Entry point for launching an IPython kernel.
```

```
#Convert to timeseries
ts = data['Weekly_Sales']
ts.head(5)
```

```
Date
2010-02-05    49750740.50
2010-02-12    48336677.63
2010-02-19    48276993.78
2010-02-26    43968571.13
2010-03-05    46871470.30
Name: Weekly_Sales, dtype: float64
```

Regression Models

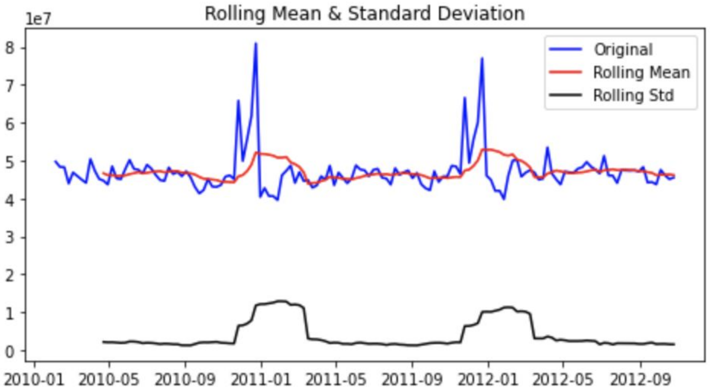
Converting date to numerical features

```
df['weekofyear'] = df['Date'].dt.weekofyear
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month
```

04
Model(s) Creation

Time Series Algorithms

ARIMA




```
Results of Dickey-Fuller Test:
Test Statistic                 -5.908298e+00
p-value                         2.675979e-07
#Lags Used                      4.000000e+00
Number of Observations Used     1.380000e+02
Critical Value (1%)            -3.478648e+00
Critical Value (5%)            -2.882722e+00
Critical Value (10%)           -2.578065e+00
dtype: float64
```

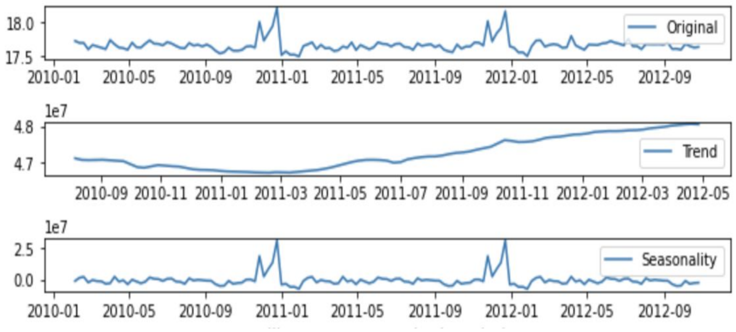
DF Test for stationarity

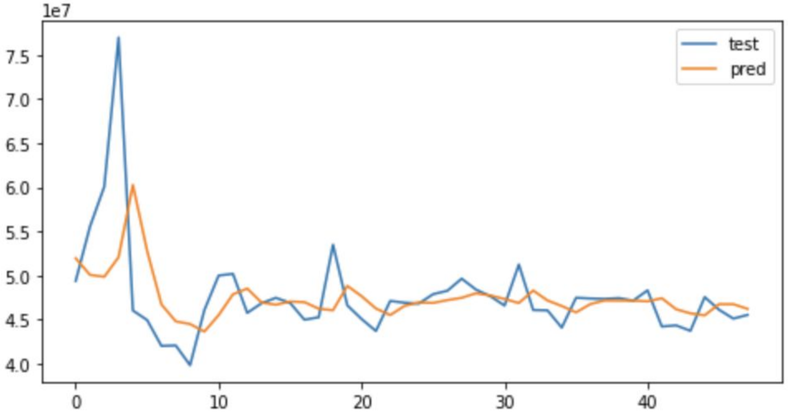
p-value = 0.0267e-05

(<0.05, = **stationary**)

Time Series Algorithms

ARIMA



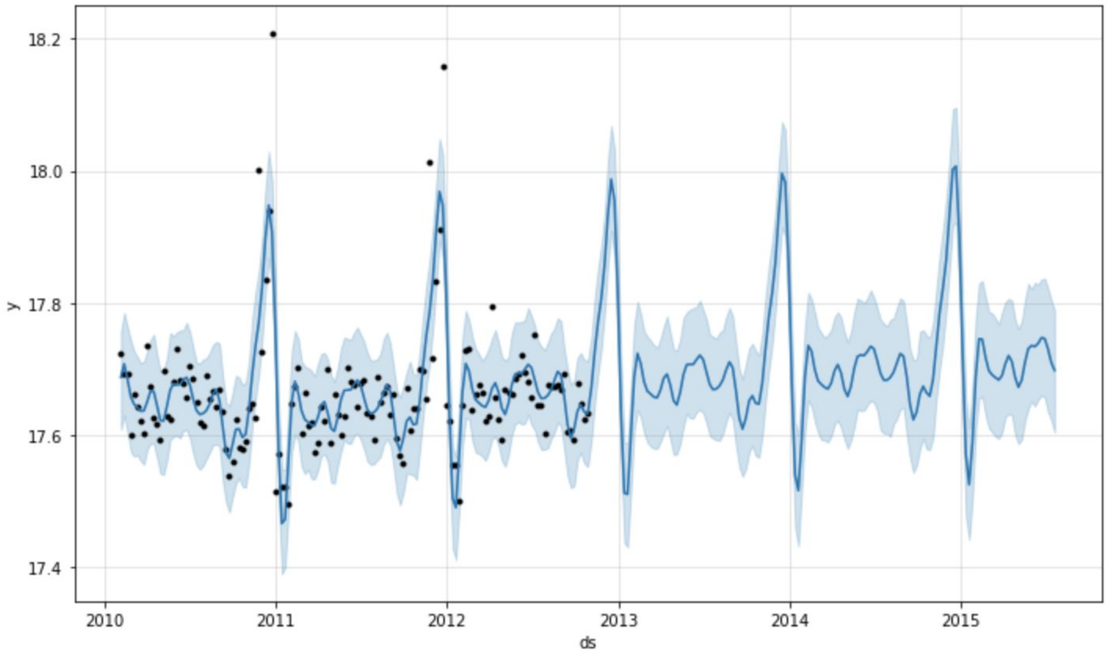



```
rms = sqrt(mean_squared_error(np.exp(test), np.exp(predictions)))
print('Root Mean Squarred Error: %.2f'% rms)
```

```
Root Mean Squarred Error: 5136942.16
```

Time Series Models

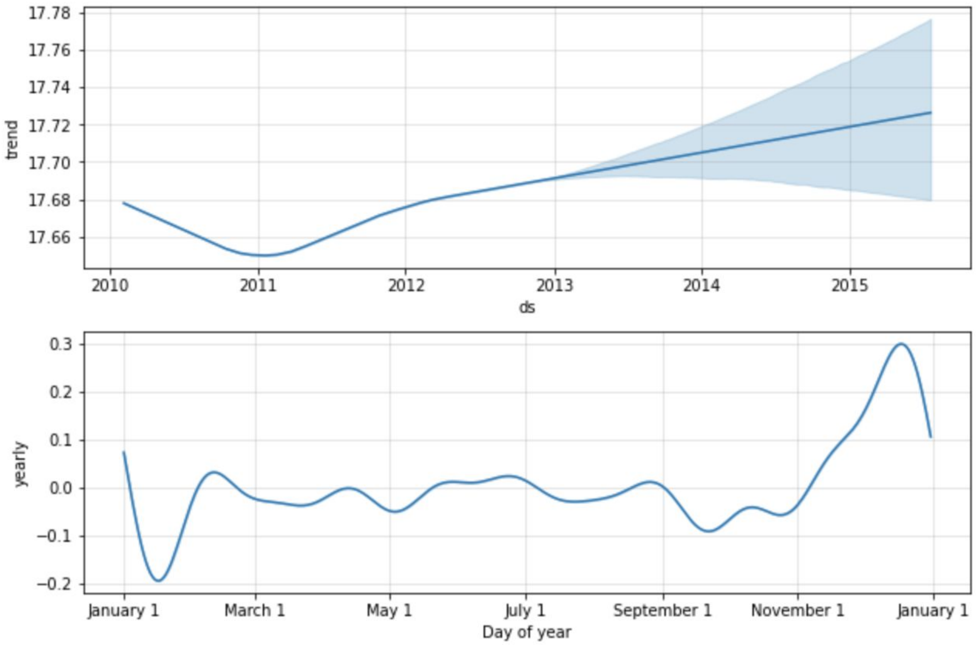
Facebook Prophet



Univariate Forecasting

Time Series Models

Facebook Prophet

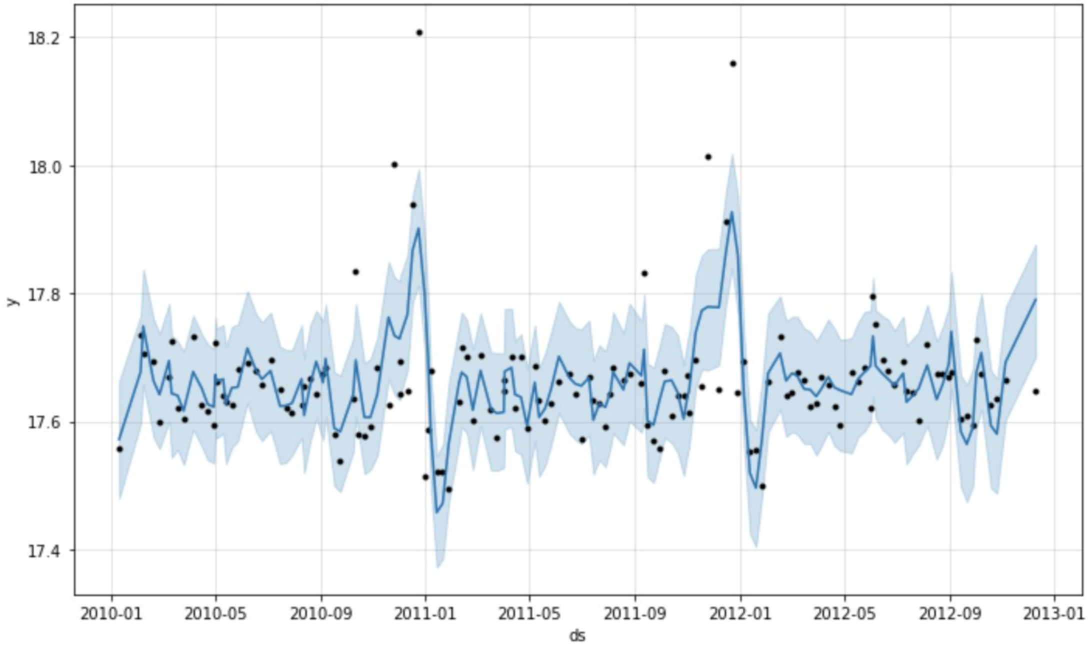


Univariate Forecasting

RMSE : **3451591.61**

Time Series Models

Facebook Prophet



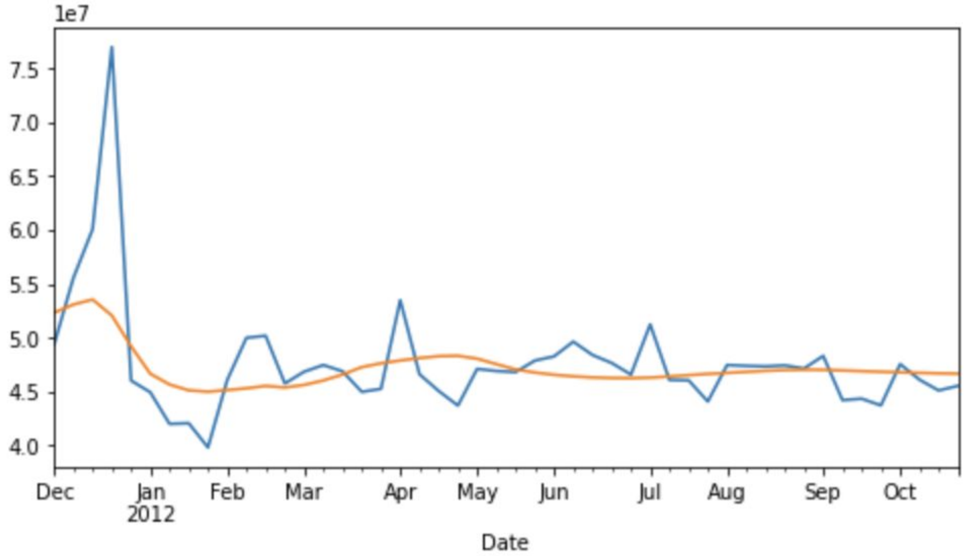
Multivariate Forecasting

```
model_new = Prophet()
model_new.add_regressor('Fuel_Price')
model_new.add_regressor('Unemployment')
```

RMSE : **3886002.06**

Time Series Algorithms

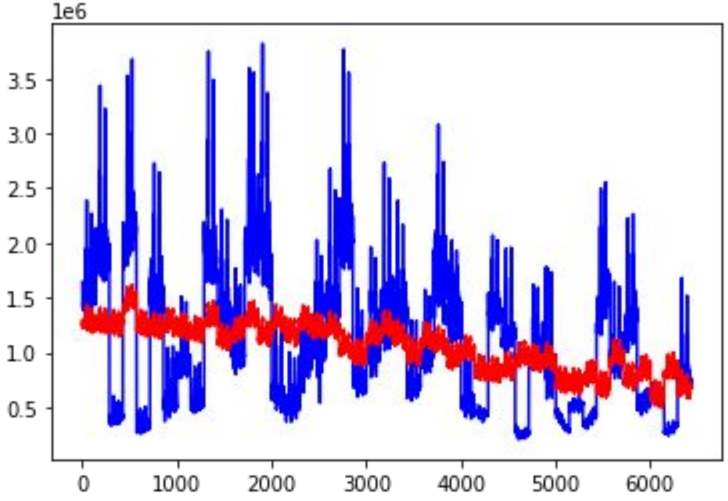
LSTM



RMSE : **5097790.39**

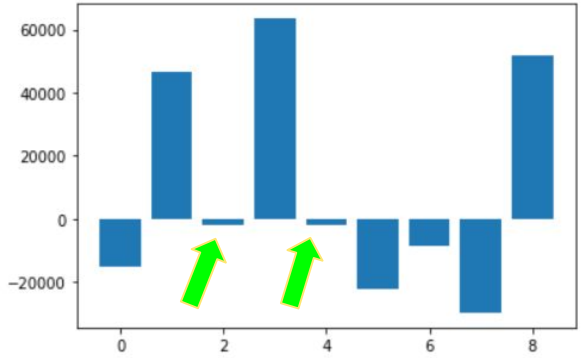
Regression Models

Multiple Linear Regression



Baseline Model:
RMSE : **520420.226516**

```
Feature "Store", Score: -15383.97544
Feature "Holiday_Flag", Score: 46616.62910
Feature "Temperature", Score: -1798.65900
Feature "Fuel_Price", Score: 63491.75705
Feature "CPI", Score: -2105.53047
Feature "Unemployment", Score: -22181.88143
Feature "weekofyear", Score: -8660.31910
Feature "year", Score: -29855.52133
Feature "month", Score: 51783.48883
```

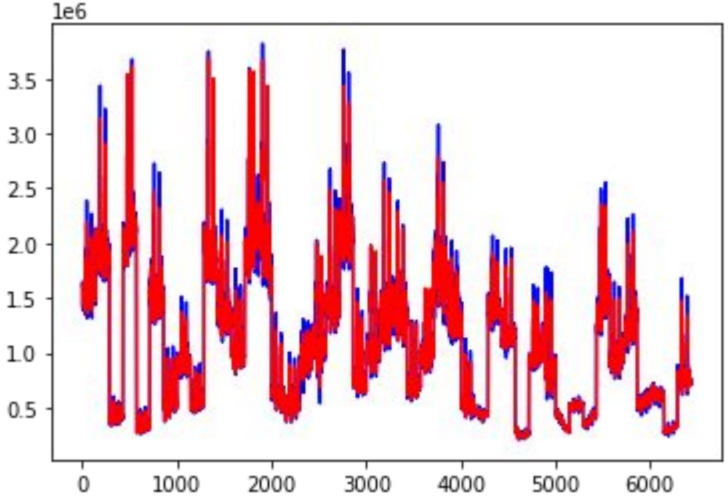


Feature "Temperature' and 'CPI' excluded:

RMSE :
527688.219704

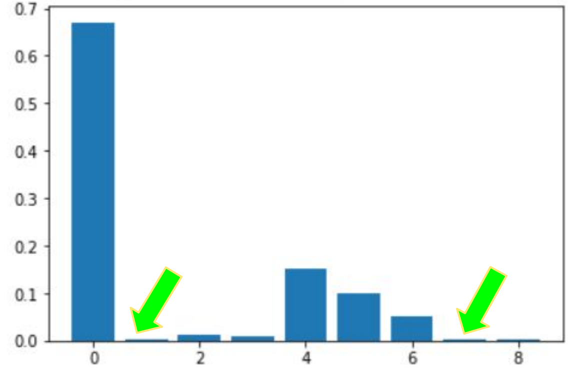
Regression Models

Random Forest Regression



Baseline Model:
RMSE : **37682.047669**

```
Feature "Store", Score: 0.67041
Feature "Holiday_Flag", Score: 0.00155
Feature "Temperature", Score: 0.01251
Feature "Fuel_Price", Score: 0.00890
Feature "CPI", Score: 0.15246
Feature "Unemployment", Score: 0.09956
Feature "weekofyear", Score: 0.05036
Feature "year", Score: 0.00154
Feature "month", Score: 0.00272
```



Feature 'Holiday_Flag', 'year' and month excluded:

RMSE :
37682.047669

Dimentionarity Reduction

PCA
RMSE : 68731.563667

KPCA
RMSE : 68731.563667

```
#Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 6)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

```
[33] explained_variance = pca.explained_variance_ratio_
     explained_variance
```

```
array([6.94759666e-01, 1.55410532e-01, 7.75322368e-02, 7.08812135e-02,
       1.33115187e-03, 8.51998953e-05])
```

```
[34] rf_pca = RandomForestRegressor(n_estimators=500, random_state=1)
     rf_pca.fit(X_train_pca, y_train)
     y_pred_rf_pca = rf_pca.predict(X_test_pca)
```

```
[35] rmse = np.sqrt(MSE(y_test, y_pred_rf_pca))
     print("RMSE : % f" %(rmse))
```

```
RMSE :  68731.563667
```

```
[36] #Applying KPCA
     from sklearn.decomposition import KernelPCA
     kpca = KernelPCA(n_components = 6)
     X_train_kpca = kpca.fit_transform(X_train)
     X_test_kpca = kpca.transform(X_test)
```

```
[37] explained_variance = pca.explained_variance_ratio_
     explained_variance
```

```
array([6.94759666e-01, 1.55410532e-01, 7.75322368e-02, 7.08812135e-02,
       1.33115187e-03, 8.51998953e-05])
```

```
[40] rf_kpca = RandomForestRegressor(n_estimators=500, random_state=1)
     rf_kpca.fit(X_train_kpca, y_train)
     y_pred_rf_kpca = rf_kpca.predict(X_test_kpca)
```

```
[41] rmse = np.sqrt(MSE(y_test, y_pred_rf_kpca))
     print("RMSE : % f" %(rmse))
```

```
RMSE :  68731.563667
```

Hyperparameter Tuning

```
# Fit the grid search to the data
grid_search.fit(X_train, y_train)
grid_search.best_params_
```

```
Fitting 2 folds for each of 1296 candidates, totalling 2592 fits
{'bootstrap': True,
 'max_depth': 80,
 'max_features': 3,
 'min_samples_leaf': 4,
 'min_samples_split': 8,
 'n_estimators': 100,
 'random_state': 4}
```

RMSE : 90392.538069

```
[49] # Fit the grid search to the data
grid_search2.fit(X_train, y_train)
grid_search2.best_params_
```

```
Fitting 2 folds for each of 432 candidates, totalling 864 fits
{'bootstrap': True,
 'max_depth': 70,
 'max_features': 5,
 'min_samples_leaf': 1,
 'min_samples_split': 7
 'n_estimators': 50,
 'random_state': 5}
```

RMSE : 61737.736260

```
# Fit the grid search to the data
grid_search3.fit(X_train, y_train)
grid_search3.best_params_
```

```
{'bootstrap': True,
 'max_depth': 60,
 'max_features': 5,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 40,
 'random_state': 4}
```

RMSE : 40296.371264

```
# Fit the grid search to the data
grid_search4.fit(X_train, y_train)
grid_search4.best_params_
```

```
Fitting 2 folds for each of 36 candidates, totalling 72 fits
{'bootstrap': True,
 'max_depth': 50,
 'max_features': 5,
 'min_samples_leaf': 1,
 'min_samples_split': 3,
 'n_estimators': 35,
 'random_state': 4}
```

RMSE : 44146.195977

```
[70] # Fit the grid search to the data
grid_search5.fit(X_train, y_train)
grid_search5.best_params_
```

```
Fitting 2 folds for each of 48 candidates, totalling 96 fits
{'bootstrap': True,
 'max_depth': 50,
 'max_features': 5,
 'min_samples_leaf': 1,
 'min_samples_split': 3,
 'n_estimators': 30,
 'random_state': 4}
```

RMSE : 44040.803179

The baseline model performs the best among all these models.

05 Model Evaluation and Comparison

Model Evaluation and Comparison

Algorithm	Metrics
Linear Regression	RMSE : 520420.226516
Random Forest Regression	RMSE : 37682.047669
ARIMA	RMSE : 5136942.16
Facebook Prophet	RMSE : 3451591.61
LSTM	RMSE : 5097790.39

06 Conclusion and Future Improvements

Conclusion and Future Improvements

1. Larger dataset for more accurate training

2. Compare with other algorithms such as Deep Learning algorithms

3. Apply the models to product-wise sales forecasting

Thank you!