Demand Forecasting in Retail Domain

Vellore Institute of Technology, Chennai campus

School of Computer Science and Engineering (SCOPE)

M.Tech CSE with Specialization in Big Data Analytics

ABSTRACT:

Forecasting is a useful technique that can help to understand how historical data influences the future. This is done by looking at past data, defining the patterns, and producing short or long-term predictions. Forecasting is used in multiple areas within retail Such as in Supply chain we use demand forecasting where we have enough inventory available in the store to meet future demand, prevent out of stock and Workforce planning similarly in Finance we have sales forecasting where we use for budget planning and goal setting. Here we use AR model for forecasting.

TIME SERIES ANALYSIS:

Any time series can be broken down into its individual components:

Trend: Increase or decrease in the series of data over longer a period .A newly launched product can be expected to have a positive slope as we can expect sales to keep increasing over time

Seasonality: Fluctuations in the pattern due to seasonal determinants over a period such as a day, week, month, season. For example --Ice creams sell well during summer seasons

Cyclical variations: Occurs when data exhibit rises and falls at irregular intervals. eCommerce example - we can expect low sales after events (big billion day) as a lot of people would have made purchases

Random or irregular variations: Instability due to random factors that do not repeat in the pattern. Competitor increased the price for a couple of weeks because of which our demand got increased

SYSTEM REQUIREMENT:

- Windows 10
- Collab from google for running python

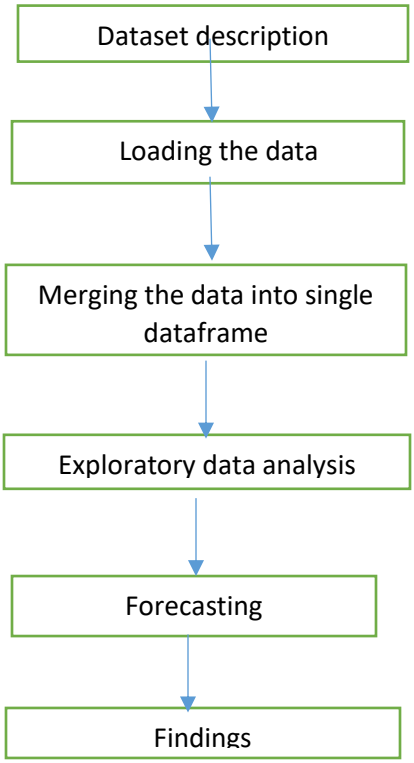
ALGORITHM:

ARMODEL

- Autoregressive models operate under the premise that past values have an effect on current values, which makes the statistical technique popular for analyzing nature, economics, and other processes that vary over time
- We forecast the variable of interest using a linear combination of *past values of the variable*. The term *auto* regression indicates that it is a regression of the variable against itself. Thus, an autoregressive model of order p can be written as

$$Y_t = \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_0 Y_0 + \epsilon_t$$

METHODOLOGY:



DATASET DESCRIPTION:

Sales data-set.csv:
Anonymized information about the 45 stores, indicating the type and size of store.
Stores data-set.csv:
Historical sales data, which covers to 2010-02-05 to 2012-11-01.
Features data-set.csv:
Contains additional data related to the store, department, and regional activity for the given dates.

Column name	Description	Unique value	Missing value
Type	Type – type of the sales	Type- 45	Type- 0
Store	Store- the store number	Store – 45	Store – 0
Size	Size- size of the sales	Size – 45	Size – 0
Store	Store - the store number	Store - 143	Store - 0
Dept	Dept - the department number	Dept - 143	Dept - 0
Date	Date - the week	Date - 143	Date - 0
Weekly Sales	Weekly Sales - sales for the given department in the given store.	Weekly Sales – 143	Weekly Sales- 0

IsHoliday	IsHoliday - whether the week is a special holiday week.	IsHoliday - 143	IsHoliday -0
Store	Store - the store number	Store - 182	Store - 0
Date	Date - the week	Date – 182	Date - 0
Temperature	Temperature - average temperature in the region	Temperature - 182	Temperature - 0
Fuel_Price	Fuel_Price - cost of fuel in the region	Fuel_Price - 182	Fuel_Price - 0
MarkDown1-5	MarkDown1-5 - anonymized data related to promotional markdowns. MarkDown data is only available after Nov 2011, and is not available for all stores all the time. Any missing value is marked with an NA	MarkDown1-5 – 182	MarkDown1 - 51% Markdown 2 -64% Markdown 3- 56% Markdown 4- 58% Markdown 5- 51%
CPI	CPI - the consumer price index	CPI - 182	CPI - 7%
Unemployment	Unemployment - the unemployment rate	Unemployment-182	Unemployment-7%
IsHoliday	IsHoliday - whether the week is a special holiday week.	IsHoliday - 182	IsHoliday - 0

Exploring the data set

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[3] df_stores = pd.read_csv('stores data-set.csv')
    df_features = pd.read_csv('Features data set.csv', parse_dates = ['Date'])
    df_sales = pd.read_csv('sales data-set.csv', parse_dates = ['Date'])
```

Cleaning and preprocessing

```
[4] df_sales.head()
```

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

```
[5] df_sales.Date.value_counts()

2011-12-23    3027
2011-11-25    3021
2011-12-16    3013
2011-09-12    3010
2012-02-17    3007
              ...
2010-09-07    2903
2010-08-20    2901
2010-07-16    2901
2010-08-27    2898
2010-08-13    2896
Name: Date, Length: 143, dtype: int64
```

```
[27] df_all.describe()
```

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
count	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.00
mean	22.200546	60.090059	3.361027	2590.392585	913.428550	468.814001	1086.569677	1662.772385	171.201947	7.960289
std	12.785297	18.447931	0.458515	6052.257540	5086.843826	5528.817592	3894.194587	4207.629321	39.159276	1.863296
min	1.000000	-2.060000	2.472000	0.000000	-265.760000	-29.100000	0.000000	0.000000	126.064000	3.879000
25%	11.000000	46.680000	2.933000	0.000000	0.000000	0.000000	0.000000	0.000000	132.022667	6.891000
50%	22.000000	62.090000	3.452000	0.000000	0.000000	0.000000	0.000000	0.000000	182.318780	7.866000
75%	33.000000	74.280000	3.738000	2809.050000	39.560000	6.500000	436.670000	2168.040000	212.416993	8.572000
max	45.000000	100.140000	4.468000	88646.760000	104519.540000	141630.610000	67474.850000	108519.280000	227.232807	14.313000

```
df_sales.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  datetime64[ns]
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool
dtypes: bool(1), datetime64[ns](1), float64(1), int64(2)
memory usage: 13.3 MB
```

```
[5]  df_sales.Date.value_counts()
```

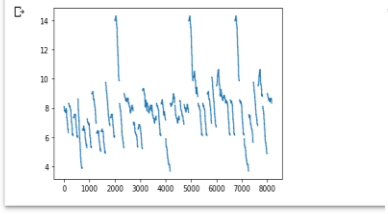
```
2011-12-23    3027
2011-11-25    3021
2011-12-16    3013
2011-09-12    3010
2012-02-17    3007
              ...
2010-09-07    2903
2010-08-20    2901
2010-07-16    2901
2010-08-27    2898
2010-08-13    2896
Name: Date, Length: 143, dtype: int64
```

```
df_sales.info()
```

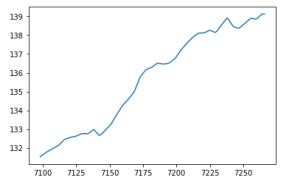
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  datetime64[ns]
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool
dtypes: bool(1), datetime64[ns](1), float64(1), int64(2)
memory usage: 13.3 MB
```

Visual Insights

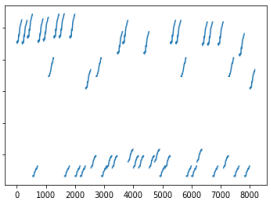
```
df_features.Unemployment.plot();
```



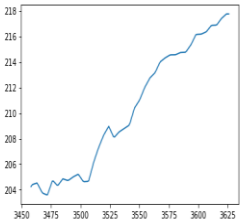
```
[12] df_features[df_features.Store == 40].CPI.plot();
```



```
[10] df_features.CPI.plot();
```



```
df_features[df_features.Store == 20].CPI.plot();
```



```
df_features.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
0	1	2010-05-02	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	False
1	1	2010-12-02	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	True
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	False
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	False
4	1	2010-05-03	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	False

Missing value Imputation

```
[8]  df_features.isna().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
```

```
[6] df_features['MarkDown1'] = df_features['MarkDown1'].fillna(0)
    df_features['MarkDown2'] = df_features['MarkDown2'].fillna(0)
    df_features['MarkDown3'] = df_features['MarkDown3'].fillna(0)
    df_features['MarkDown4'] = df_features['MarkDown4'].fillna(0)
    df_features['MarkDown5'] = df_features['MarkDown5'].fillna(0)
```

```
df_features.head(10)
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
0	1	05/02/2010	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106	False
1	1	12/02/2010	38.51	2.548	0.0	0.0	0.0	0.0	0.0	211.242170	8.106	True
2	1	19/02/2010	39.93	2.514	0.0	0.0	0.0	0.0	0.0	211.289143	8.106	False
3	1	26/02/2010	46.63	2.561	0.0	0.0	0.0	0.0	0.0	211.319643	8.106	False
4	1	05/03/2010	46.50	2.625	0.0	0.0	0.0	0.0	0.0	211.350143	8.106	False
5	1	12/03/2010	57.79	2.667	0.0	0.0	0.0	0.0	0.0	211.380643	8.106	False
6	1	19/03/2010	54.58	2.720	0.0	0.0	0.0	0.0	0.0	211.215635	8.106	False
7	1	26/03/2010	51.45	2.732	0.0	0.0	0.0	0.0	0.0	211.018042	8.106	False
8	1	02/04/2010	62.27	2.719	0.0	0.0	0.0	0.0	0.0	210.820450	7.808	False
9	1	09/04/2010	65.86	2.770	0.0	0.0	0.0	0.0	0.0	210.622857	7.808	False

SAMPLE CODE:

```
[7] # splitting date into 3 columns denoting Year, Month and Day respectively
    df['Year'] = df.Date.apply(lambda x: int(str(x)[:4]))
    df['Month'] = df.Date.apply(lambda x: int(str(x)[5:7]))
    df['Year-Month'] = df.Date.apply(lambda x: str(x)[:7])
    df['Day'] = df.Date.apply(lambda x: int(str(x)[8:10]))
```

```
[8] #df.groupby(['Year','Month']).Fuel_Price.mean()
    plot_no = 1
    _ = plt.subplots(figsize = (20,10))
    _ = plt.xticks(rotation = 60)
    _ = sns.lineplot(data = df, x = 'Year-Month',y = 'Fuel_Price')
    _ = plt.title('LinePlot showing the change in fuel price over the span of 3 years', fontsize=20)
    plt.savefig(str(plot_no)+'_plot.png')
    plot_no +=1
```

```
[9] r = 5 #lets round off the temperature in the range of r
    df['Temperature_r'] = df.sort_values(by=['Temperature']).Temperature.apply(lambda x : x - x %r)
```

```
[10] _ = plt.subplots(figsize = (20,10))
     _ = plt.ylim(0.1,1.45)
     plots = sns.barplot(data = df, x = 'IsHoliday', y = 'Fuel_Price', hue = 'Type')
     _ = plt.title('BarPlot showing the change in fuel price with respect the type of the store with holidays grouped')
     for bar in plots.patches:
       _ = plots.annotate(format(bar.get_height(), '.2f'),
                          (bar.get_x() + bar.get_width() / 2,
                           bar.get_height()) - (bar.get_height()-3.1)/2), ha='center', va='center',
                          size=15, xytext=(0, 0),bbox=dict(boxstyle="round4,pad=0.6", fc="w", ec="black", lw=2),
                          textcoords='offset points')
     plt.savefig(str(plot_no)+'_plot.png')
     plot_no +=1
```

```
[11] _ = plt.subplots(figsize = (20,10))
     _ = sns.lineplot(data = df, x = 'Type', y = 'Fuel_Price', hue = 'IsHoliday',style = 'IsHoliday', markers = True, ci = 68)
     _ = plt.title('LinePlot showing the change in fuel price with respect the type of the store')
     plt.savefig(str(plot_no)+'_plot.png')
     plot_no +=1
```

```
[13] _ = plt.subplots(figsize = (20,10))
     _ = sns.lineplot(data = df, x = 'Temperature_r', y = 'CPI', hue = 'Type',style = 'Type', markers = True, ci = 68)
     _ = plt.title('Lineplot showing the change in CPI with respect to the change in temperature')
     plt.savefig(str(plot_no)+'_plot.png')
     plot_no +=1
```

```
     _ = plt.subplots(figsize = (20,10))
     _ = sns.lineplot(data = df, x = 'Date', y = 'Fuel_Price')
     _ = plt.title('Lineplot showing the change in fuel price in each month over the span of 3 years')
     plt.savefig(str(plot_no)+'_plot.png')
     plot_no +=1
```

```
[15] _ = plt.subplots(figsize = (20,10))
     _ = sns.countplot(data = df,x='Year',hue='Month')
     _ = plt.title('Barplot showing the observation counts for each recorded month')
     plt.savefig(str(plot_no)+'_plot.png')
     plot_no +=1
```

```
12] _ = plt.subplots(figsize = (20,10))
    _ = sns.lineplot(data = df, x = 'Temperature_r', y = 'Fuel_Price', hue = 'IsHoliday',style = 'IsHoliday', markers = True, ci = 68)
    _ = plt.xlabel('Temperature range')
    _ = plt.title('Lineplot showing the change in fuel price with respect to the change in temperature')
    plt.savefig(str(plot_no)+'_plot.png')
    plot_no +=1
```

```
[16] _ = plt.subplots(figsize = (20,10))
     plots = sns.barplot(data = df, x = 'Type', y = 'Fuel_Price')
     for bar in plots.patches:
         _ = plots.annotate(format(bar.get_height(), '.2f'),
                            (bar.get_x() + bar.get_width() / 2,
                             bar.get_height()), ha='center', va='center',
                            size=15, xytext=(0, 23),
                            textcoords='offset points');
     _ = plt.ylim(2.5,3.5)
     _ = plt.title('Barplot showing the change in Fuel price with respect to the type of the store')
     plt.savefig(str(plot_no)+'_plot.png')
     plot_no +=1
```

```
[19] _ = df[['Date', 'Temperature', 'Fuel_Price', 'CPI', 'Unemployment', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']].plot(x = 'Date', subplots
     plt.savefig(str(plot_no)+'_plot.png')
     plot_no +=1
```

```
[17] df_rolled_mean = df.set_index('Date').rolling(window = 2948).mean().reset_index()
     df_rolled_std = df.set_index('Date').rolling(window = 2948).std().reset_index()
```

```
     fig,ax = plt.subplots(figsize = (20,10))
     _ = sns.lineplot(data = df, x = 'Year-Month', y = 'Weekly_Sales', ax = ax, ci = 1)
     _ = plt.xticks(rotation = 60)
     _ = plt.title('Lineplot showing the change in Weekly_Sales in each month over the span of 3 years')
     plt.savefig(str(plot_no)+'_plot.png')
     plot_no +=1
```

```
df_average_sales_week = df.groupby(by=['Date'], as_index=False)['Weekly_Sales'].sum()
df_average_sales = df_average_sales_week.sort_values('Weekly_Sales', ascending=False)

_ = plt.figure(figsize=(20,8))
_ = plt.plot(df_average_sales_week.Date, df_average_sales_week.Weekly_Sales)
_ = plt.title('Data spread of total weekly sales volume of the retail chain')
_ = plt.xlabel('Date')
_ = plt.ylabel('Weekly Sales')
plt.savefig(str(plot_no)+'_plot.png')
plot_no +=1
```

Forecasting:

```
from statsmodels.graphics.tsaplots import acf, pacf, plot_acf, plot_pacf

fig, axes = plt.subplots(1,2, figsize=(20,5))
_ = plot_acf(ts, lags=64, ax=axes[0])
_ = plot_pacf(ts, lags=64, ax=axes[1])
plt.savefig(str(plot_no)+'_plot.png')
plot_no +=1
```

```
from sklearn.linear_model import LinearRegression

def fit_ar_model(ts, orders):
    X=np.array([ ts.values[(i-orders)].squeeze() if i >= np.max(orders) else np.array(len(orders) * [np.nan]) for i in range(len(ts))])
    mask = ~np.isnan(X[:,:1]).squeeze()
    Y= ts.values
    lin_reg=LinearRegression()
    lin_reg.fit(X[mask],Y[mask])
    print(lin_reg.coef_, lin_reg.intercept_)
    print('Score factor: %.2f' % lin_reg.score(X[mask],Y[mask]))
    return lin_reg.coef_, lin_reg.intercept_

def predict_ar_model(ts, orders, coef, intercept):
    return np.array([np.sum(np.dot(coef, ts.values[(i-orders)].squeeze())) + intercept  if i >= np.max(orders) else np.nan for i in range(len(ts))])
```

```
orders=np.array([1,6,52])
coef, intercept = fit_ar_model(ts,orders)
pred=pd.DataFrame(index=ts.index, data=predict_ar_model(ts, orders, coef, intercept))
_ = plt.figure(figsize=(20,5))
_ = plt.plot(ts, 'o')
_ = plt.plot(pred)
plt.savefig(str(plot_no)+'_plot.png')
plot_no +=1
```

```
diff=(ts['Weekly_Sales']-pred[0])/ts['Weekly_Sales']

print('AR Residuals: avg %.2f, std %.2f' % (diff.mean(), diff.std()))

_ = plt.figure(figsize=(20,5))
_ = plt.plot(diff, c='orange')
_ = plt.grid()
plt.savefig(str(plot_no)+'_plot.png')
plot_no +=1
```

```
df20=df.where( df['Store'] == 20)
df20=df20.dropna()
df20=df20.groupby(by=['Date'], as_index=False)['Weekly_Sales'].sum()
df20 = df20.set_index('Date')
df20.head()
```

```
_ = plt.figure(figsize=(20,5))
_ = plt.plot(df20.index, df20.values)
plt.savefig(str(plot_no)+'_plot.png')
plot_no +=1
```

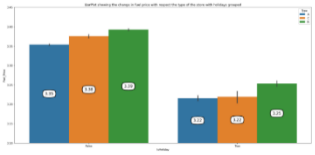
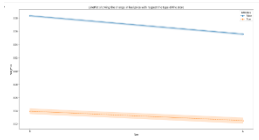
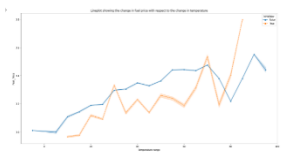
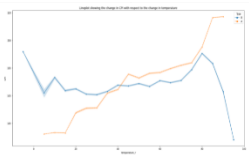
GETTING THE TEMPERATURE FUEL PRICE AND CPI FOR THE STORE 20

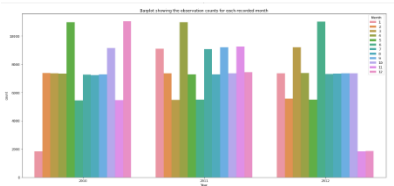
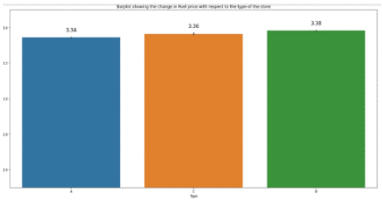
```
dfext=df.where( df['Store'] == 20)
dfext=dfext.dropna()
dfext=dfext.groupby(by=['Date'], as_index=False)[['Temperature', 'Fuel_Price', 'CPI', 'Unemp
loyment',
                                                  'MarkDown1', 'MarkDown2', 'MarkDown3', 'Ma
rkDown4', 'MarkDown5']].mean()
dfext = dfext.set_index('Date')
dfext.head()
```

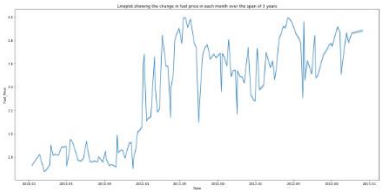
CORRELATION

```
corr = dfext.corr()
_ = plt.figure(figsize=(10,10))
sns.heatmap(corr,
            annot=True, fmt=".3f",
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values);
plt.savefig(str(plot_no)+'_plot.png')
plot_no +=1
```

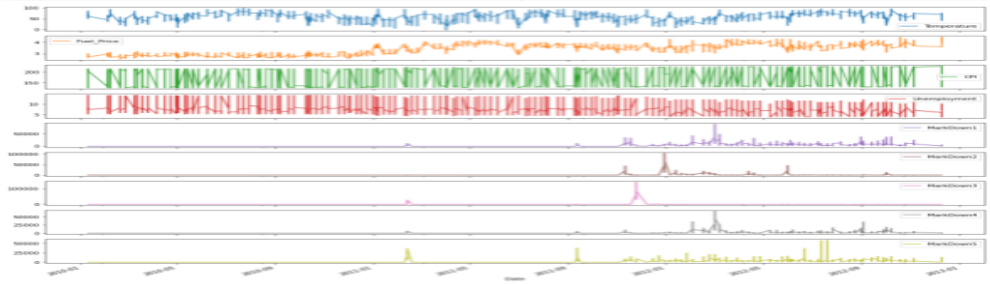
OUTPUT GRAPH:

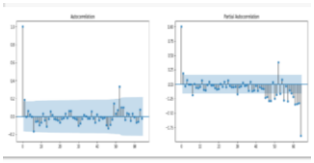



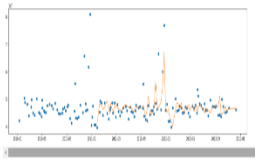


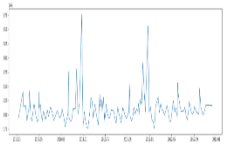
GAINING INSIGHTS OFMARKDOWN1-5,UNEMPLOYMENT,CPI,FUELPRICE,TEMPERATURE OVER THE PERIOD OF TIME

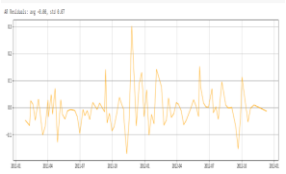


Forecasting- Graphs:









GETTING THE TEMPERATURE FUEL PRICE AND CPI FOR THE STORE 20

Date	Temperature	Fuel_Price	CPI	Unemployment	MarkDown1	MarkDown2	MarkDown3	MarkDown4	M
2010-01-10	61.08	2.707	204.885097	7.484	0.0	0.0	0.0	0.0	0.(
2010-02-04	51.00	2.850	204.025284	7.856	0.0	0.0	0.0	0.0	0.(
2010-02-07	70.10	2.815	204.485058	7.527	0.0	0.0	0.0	0.0	0.(
2010-02-19	25.43	2.745	204.432100	8.187	0.0	0.0	0.0	0.0	0.(
2010-02-26	32.32	2.754	204.463087	8.187	0.0	0.0	0.0	0.0	0.(

CORRELATION

TOP PERFORMING STORE IN TERMS OF SALES

	Type	Weekly_Sales
0	A	4.331015e+09
1	B	2.000701e+09
2	C	4.055035e+08

	Store	Weekly_Sales
19	20	3.013978e+08
3	4	2.995440e+08
13	14	2.889999e+08

FINDINGS:

- The fuel price increases with increase in temperature steadily during workdays and unevenly during holidays
- There is no significant pattern in the data points spread each month in the dataset. However, one noticeable cue is that no sales data is recorded / happened during the month of September in 2013.
- There was a peak during the end of the years 2010 and 2011 but not during 2012. This might be due to comparatively very less observations during the last 2 months in 2012.

INFERENCE:

From the basic analytics and forecasting we can suggest the shop owner that they can stock up the high selling products of various season in the low temperature period itself as fuel price tends to increase as the temperature increases which in turn will reflect in the stock prices. Similarly Store number 19 has the highest weekly sales from which all the other stores can take up same trend in order to increase the sales.