Case - Vanderbilt University Elective Surgery Schedule

Overview

- The given dataset represents a 48-week surgery schedule
- Includes number of surgical cases scheduled from one to 28 days before surgery

Problem Statement

- There was a major problem of variation in the number of surgeries performed daily.
- Knowing the expected number of surgeries would allow the Operating Room managers to schedule the surgeries properly with the correct number of staff which helps to match the demand.

Potential causes

- Surgeries are generally scheduled way earlier.
- Sometimes, no surgeries are scheduled in a week
- 6% add-on case - 6% of total cases were unscheduled (Urgent cases that must be accommodated)

Objective:

- We are using the elective surgery schedule from a few weeks prior to the actual surgery date predict the final number of surgeries to be performed using Linear Regression model.
  - Resolves the issue of variation in daily surgical case volume
  - Improves the problems of surgical staff scheduling.

Exploratory Data Analysis

- We have checked the presence of null values and there are no null values in our data
- We used describe function to know more statistical metrics about our data
- The average number of surgery cases increases as we move closer to the actual date
- Average number of actual surgeries is 116 with a standard deviation of 17.63

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

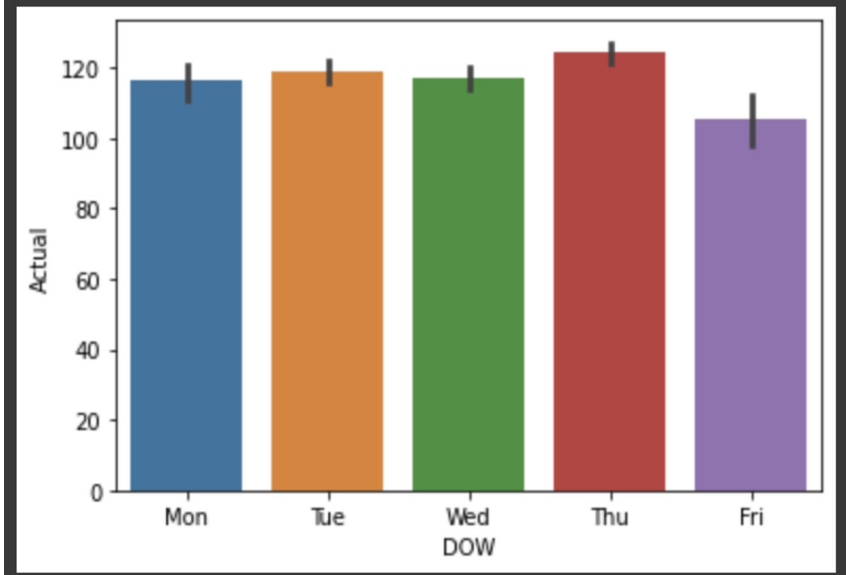
```
SurgDate     0
DOW          0
T - 28       0
T - 21       0
T - 14       0
T - 13       0
T - 12       0
T - 11       0
T - 10       0
T - 9        0
T - 8        0
T - 7        0
T - 6        0
T - 5        0
T - 4        0
T - 3        0
T - 2        0
T - 1        0
Actual       0
dtype: int64
```

```
df.describe()
```

	T - 28	T - 21	T - 14	T - 13	T - 12	T - 11	T - 10	T - 9	T - 8	T - 7	T - 6	T - 5	T - 4	T - 3	T - 2	T - 1	Actual
count	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000
mean	34.261411	47.240664	64.439834	67.817427	70.502075	72.365145	74.946058	78.041494	82.336100	86.000000	89.269710	92.091286	94.688797	97.373444	101.165975	110.008299	116.381743
std	9.387610	11.321079	13.495891	14.200934	14.873109	14.970786	15.096674	15.054842	15.594841	16.089075	16.911962	17.370472	17.468264	17.589816	17.547004	17.785057	17.629388
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.000000	11.000000
25%	28.000000	41.000000	58.000000	62.000000	64.000000	66.000000	68.000000	71.000000	76.000000	80.000000	82.000000	84.000000	88.000000	91.000000	95.000000	102.000000	110.000000
50%	35.000000	48.000000	65.000000	69.000000	72.000000	74.000000	77.000000	79.000000	85.000000	87.000000	91.000000	94.000000	97.000000	100.000000	103.000000	112.000000	117.000000
75%	40.000000	55.000000	73.000000	76.000000	80.000000	82.000000	84.000000	86.000000	92.000000	95.000000	99.000000	103.000000	106.000000	107.000000	111.000000	119.000000	126.000000
max	57.000000	73.000000	93.000000	99.000000	102.000000	106.000000	106.000000	112.000000	113.000000	118.000000	121.000000	121.000000	124.000000	127.000000	131.000000	139.000000	145.000000

Exploratory Data Analysis

```
df2 = df[['DOW', 'Actual']]
ax = sns.barplot(x='DOW',y='Actual', data=df)
```



Total Surgical Volume Based on the day of week

- ❏ We can say that highest surgical Volume can be seen on Thursdays.

  Possibly due to add-on cases

- ❏ However the difference is less when compared to other days of the week

Anova

- The p value obtained from ANOVA analysis is significant ($p < 0.05$).
- Therefore, we can conclude that there are significant differences among treatments
- We reject the null hypothesis and accept alternative hypothesis that total surgical case volume differ by day of week (DOW)

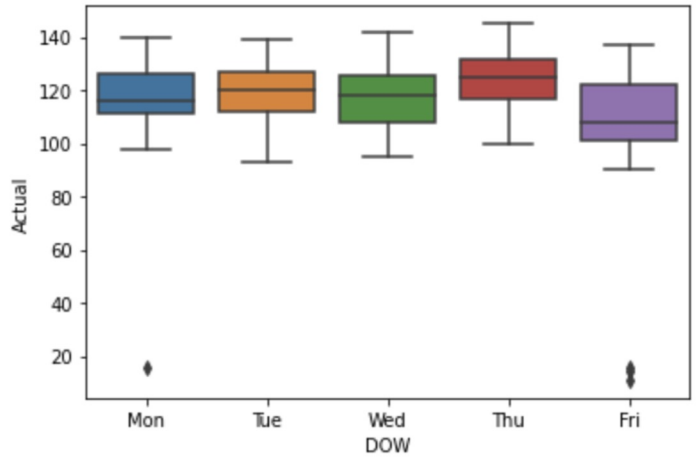
```
import statsmodels.api as sm
from statsmodels.formula.api import ols

model = ols('Actual ~ DOW', data=df).fit()
aov_table = sm.stats.anova_lm(model, typ=2)
aov_table
```

	sum_sq	df	F	PR(>F)
DOW	8909.054042	4.0	8.002734	0.000005
Residual	65681.825626	236.0	NaN	NaN

Exploratory Data Analysis-Box Plot

```
order=["Mon","Tue","Wed","Thu","Fri"]
sns.boxplot(x=df.DOW,y=df.Actual,order=order)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff4d8055ac0>
```



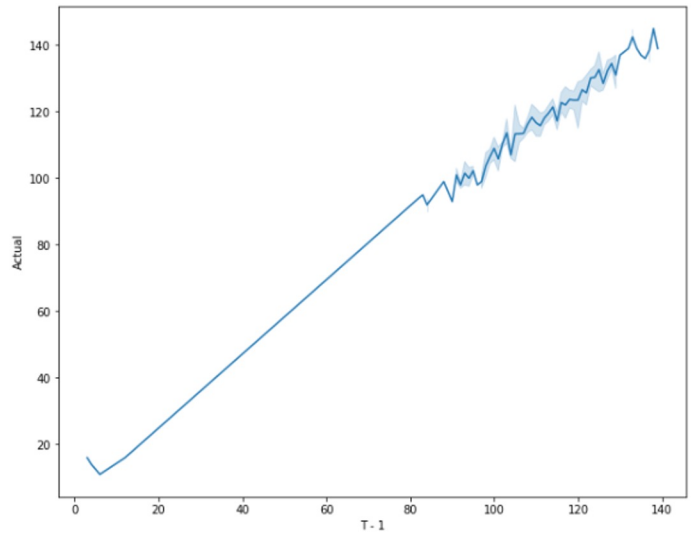
The boxplot shows the day effects on actual volume

- ❏ Highest number of surgeries on Thursdays.
- ❏ Lowest number of surgeries on Fridays.

Exploratory Data Analysis

```
plt.figure(figsize=(10,8))
sns.lineplot(x="T - 1", y="Actual", data = df)
```

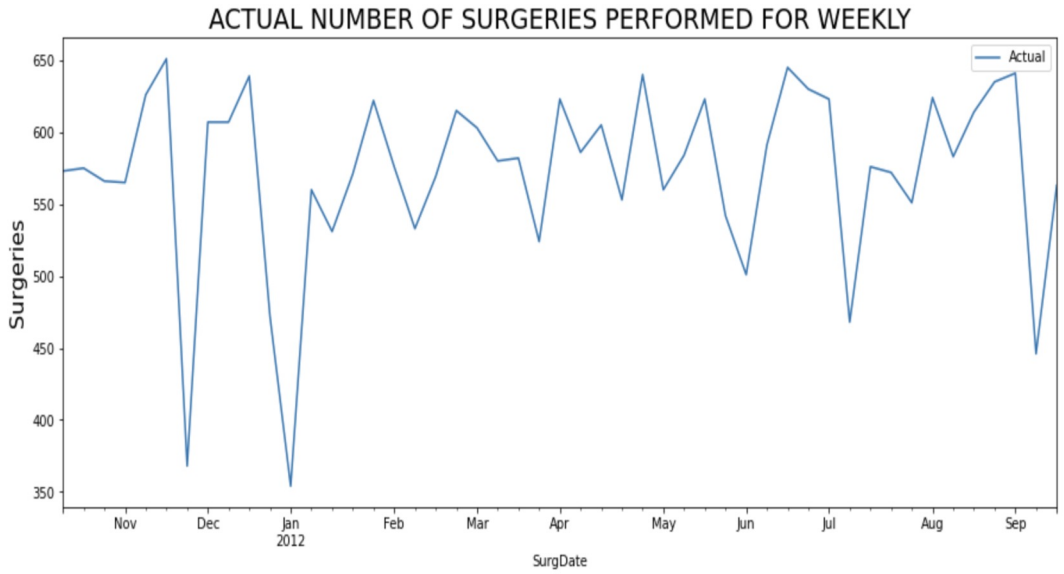
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c37efd580>
```



- Here we can see there is a linear relationship between T-1 and Actual variable.
- As the value of T-1 increases, Actual value also increases.

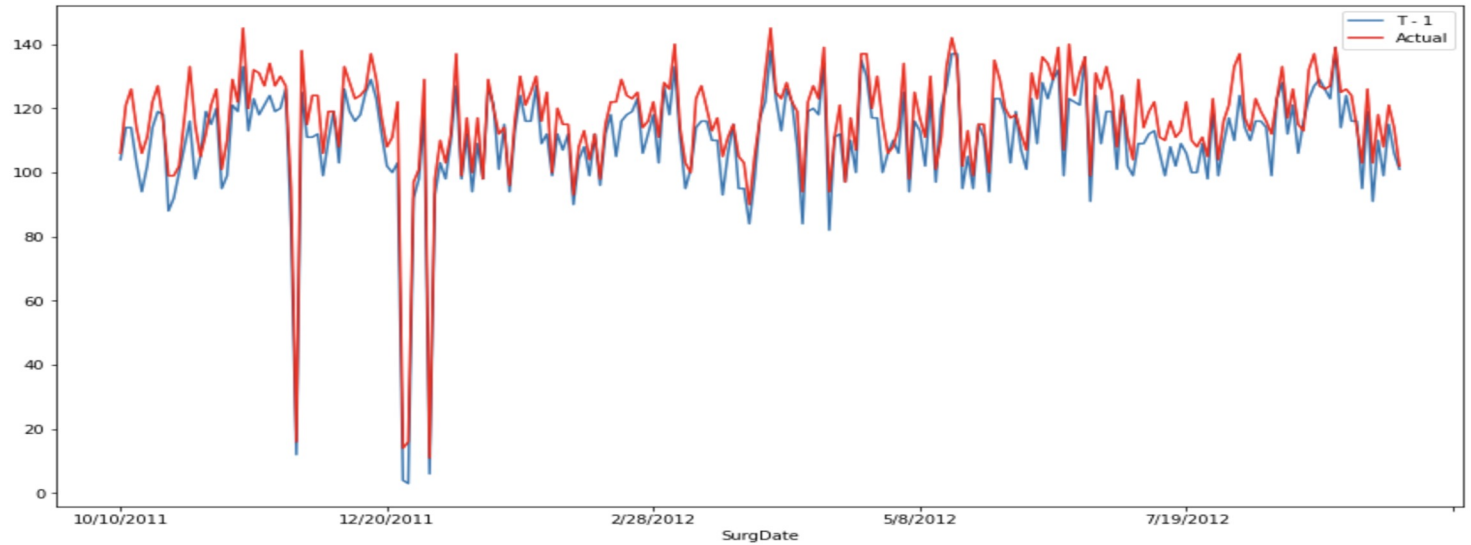
This line plot shows the actual no of surgeries performed per week

```
weekly=Vanderbilt.resample('W', on ='SurgDate').sum()
weekly.plot(figsize=(15,6))
plt.title('ACTUAL NUMBER OF SURGERIES PERFORMED FOR WEEKLY', fontsize=20)
plt.ylabel('Surgeries', fontsize=16)
plt.show()
```

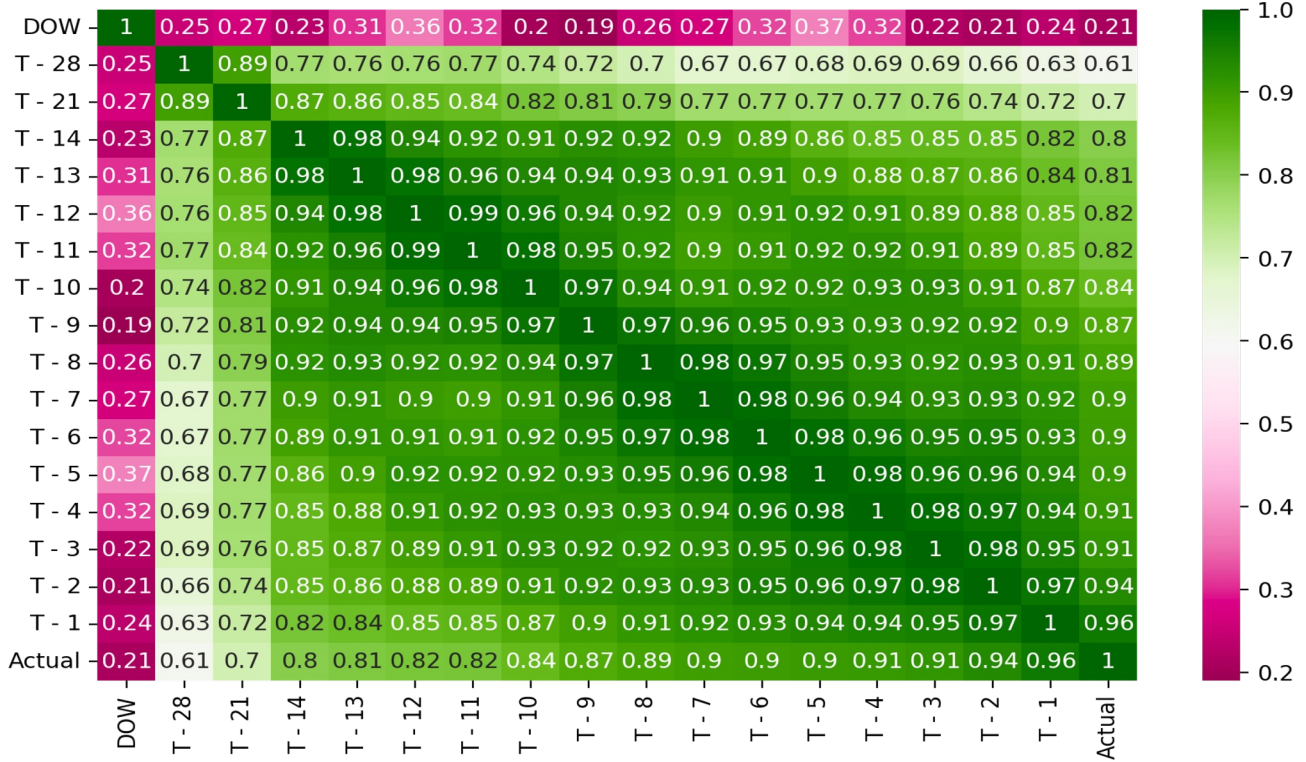


Line plot showing the difference between Scheduled surgeries and Actual

```
plt.figure(figsize=(15,8))
ax = plt.gca()
df.plot(kind='line',x='SurgDate',y='T - 1',ax=ax)
df.plot(kind='line',x='SurgDate',y='Actual', color='red', ax=ax)
plt.show()
```



HEATMAP



Significant Correlations:

Corr(T-1,Actual)=0.96

Corr(T-3,Actual)=0.91

Corr(T-7,Actual)=0.89

Corr(T-14,Actual)=0.80

Corr(T-21,Actual)=0.70

Corr(T-28,Actual)=0.61

Insights:

- The average number of surgery cases increases as we approach the actual day
- As the gap from the actual surgery day increase, the correlations have a tendency to decrease.
- Almost all the correlations are significantly high. However, there are strong correlations between the days nearer to the actual date and the actual date of surgery.

Model - Linear Regression

```
[ ] #X_train1, X_test1, y_train1, y_test1 = train_test_split(x, y, test_size = 0.2,random_state=42)
```

```
[ ] sc = SimpleImputer()
    X_train_standard = sc.fit_transform(X_train)
    X_test_standard = sc.transform(X_test)
```

```
[ ] model = LinearRegression().fit(X_train_standard,y_train)
```

```
print(model)
```

```
LinearRegression()
```

```
[ ] print('Score: ', model.score(X_train_standard,y_train))
```

```
Score:  0.9368866310298756
```

```
[ ] y_pred=model.predict(X_test_standard)
```

```
[ ] print('Score: ', model.score(X_test_standard,y_test))
```

```
Score:  0.8309649894242269
```

Model Evaluation

```
[45] pred_train = model.predict(X_train_standard)
```

```
[46] pred_test = model.predict(X_test_standard)
```

```
[47] print('MAE:', metrics.mean_absolute_error(y_train, pred_train))
     print('MSE:', metrics.mean_squared_error(y_train, pred_train))
     print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, pred_train)))
```

```
MAE: 3.446847951353822
MSE: 20.353258303699363
RMSE: 4.511458556132303
```

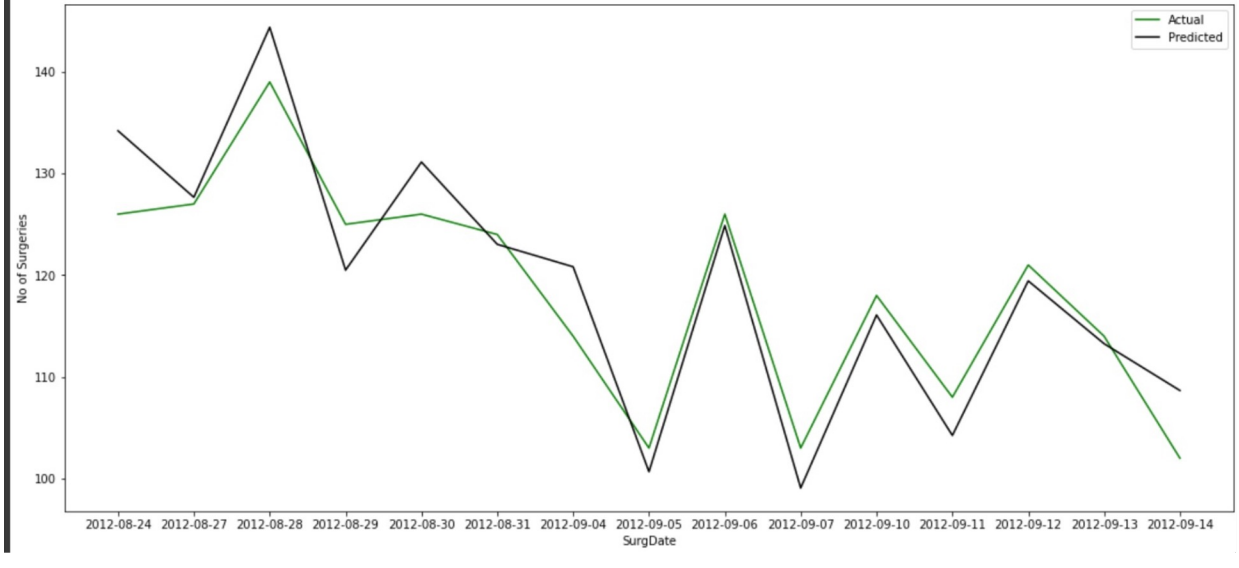
```
print('MAE:', metrics.mean_absolute_error(y_test, pred_test))
print('MSE:', metrics.mean_squared_error(y_test, pred_test))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_test)))
```

```
MAE: 3.5781002028067634
MSE: 18.521729558822713
RMSE: 4.303687902116359
```

Actual Vs Predicted

```
plt.figure(figsize=(18,8))
plt.plot(v_data['SurgDate'],y_test,color='g',label='Actual')
plt.xlabel('SurgDate')
plt.ylabel('No of Surgeries')
plt.plot(v_data['SurgDate'], y_pred,color='k',label='Predicted')
plt.xticks(v_data['SurgDate'],updated)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f9c323f54f0>
```



Predicted cases

```
result = pd.concat([df, Predicted], ignore_index=True, axis=1)
result.columns = ['SurgDate', 'DOW', 'T - 28', 'T - 21', 'T - 14', 'T - 13', 'T - 12', 'T - 11',' T - 10', 'T - 9',
                  'T - 8', 'T - 7', 'T - 6', 'T - 5', 'T - 4', 'T - 3', 'T - 2', 'T - 1', 'Actual', 'Predicted']
result['Predicted'] = round(result['Predicted'])
```

```
final = result.dropna()
final.tail(10)
```

	SurgDate	DOW	T - 28	T - 21	T - 14	T - 13	T - 12	T - 11	T - 10	T - 9	T - 8	T - 7	T - 6	T - 5	T - 4	T - 3	T - 2	T - 1	Actual	Predicted
5	20111017	1	41	56	65	69	72	73	77	78	78	80	86	85	86	92	96	102	111	123.0
6	20111018	3	44	55	69	74	79	83	83	83	93	92	96	103	105	105	107	114	122	121.0
7	20111019	4	32	40	62	66	71	73	73	84	86	87	89	96	96	96	102	119	127	101.0
8	20111020	2	33	44	62	66	67	67	79	77	88	90	98	98	98	105	111	118	116	125.0
9	20111021	0	20	32	48	48	48	47	52	55	59	61	61	61	69	72	70	88	99	99.0
10	20111024	1	32	44	55	59	61	66	69	69	69	69	74	77	83	85	86	92	99	116.0
11	20111025	3	40	51	61	65	70	70	70	70	75	80	83	89	94	96	96	100	102	104.0
12	20111026	4	26	38	54	63	65	65	65	68	74	81	89	94	94	94	103	108	117	119.0
13	20111027	2	34	54	74	75	75	75	78	80	85	91	91	94	94	96	104	116	133	113.0
14	20111028	0	32	39	61	61	61	61	64	69	72	74	74	74	77	84	83	98	115	109.0

Feature Importance

Based on the feature importance

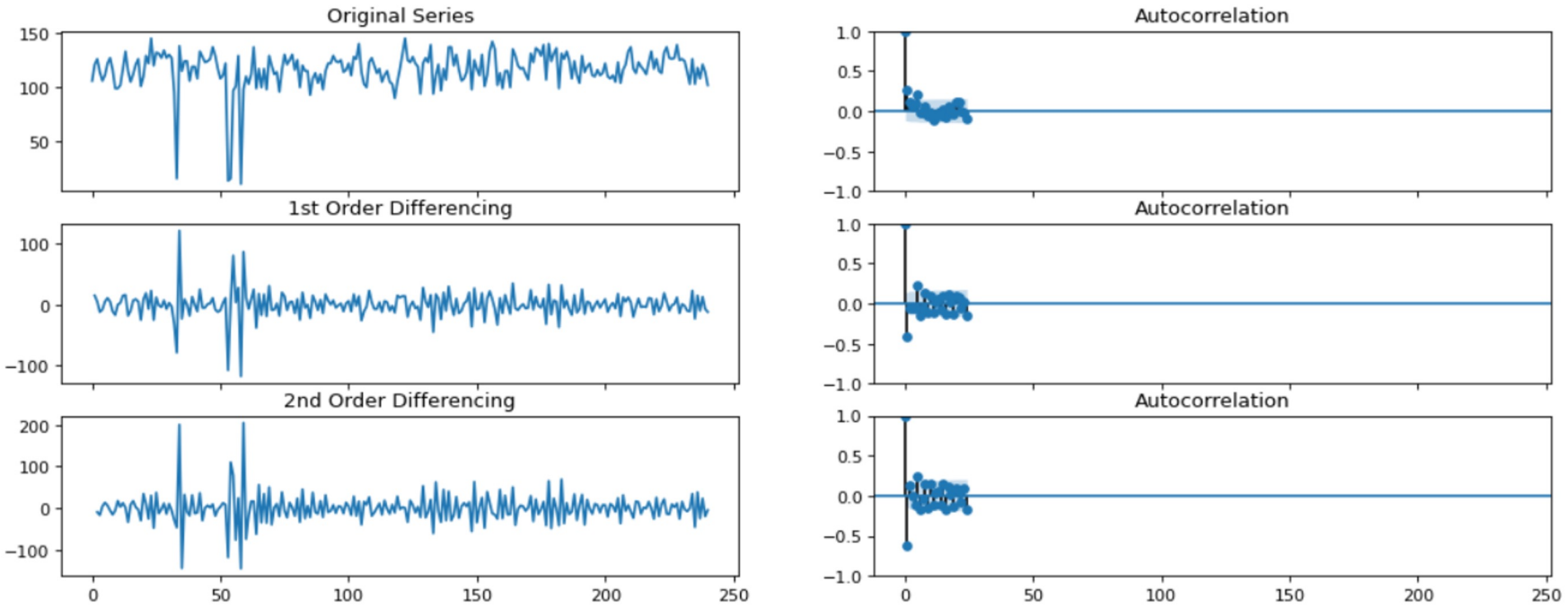
We can use the feature T-6 for predicting as it has better score

```
importance = model.coef_
# summarize feature importance
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()
```

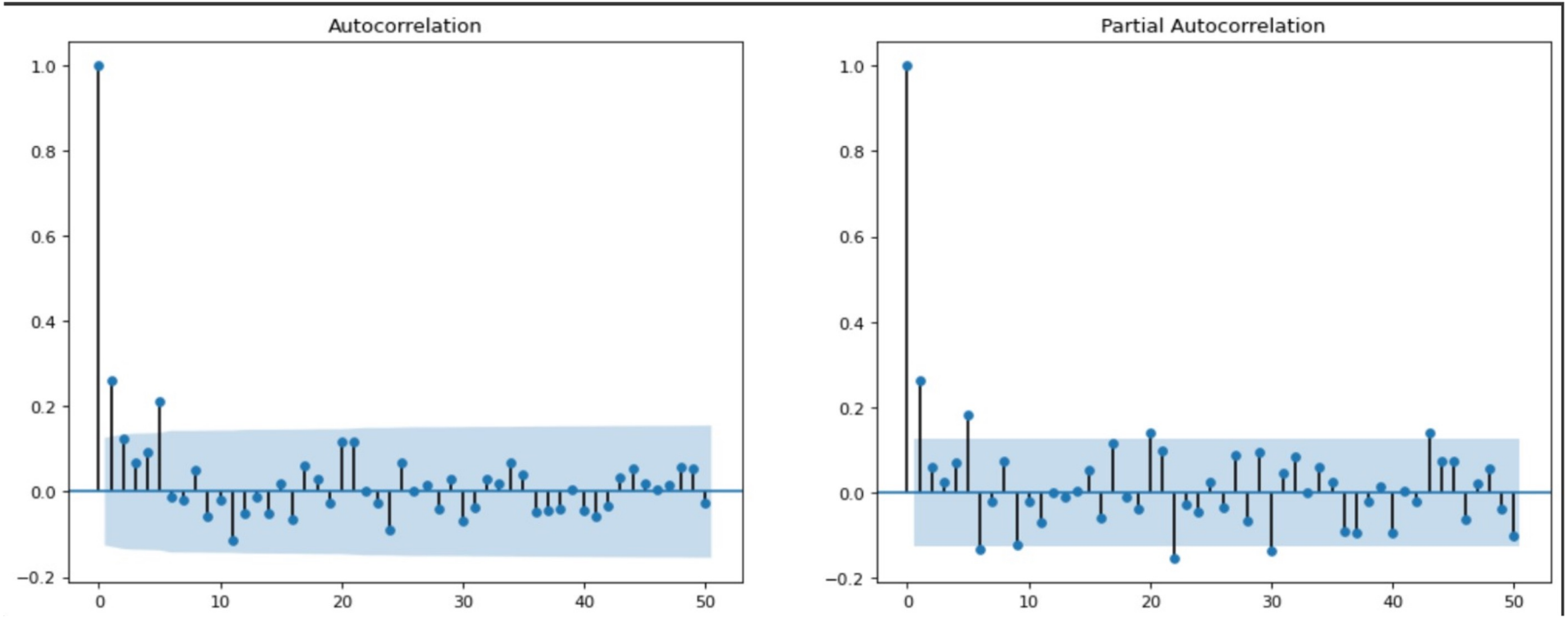
```
Feature: 0, Score: -0.00015
Feature: 1, Score: -0.35011
Feature: 2, Score: -0.03896
Feature: 3, Score: 0.05189
Feature: 4, Score: -0.11509
Feature: 5, Score: 0.14332
Feature: 6, Score: 0.10250
Feature: 7, Score: -0.21973
Feature: 8, Score: -0.08244
Feature: 9, Score: 0.18406
Feature: 10, Score: 0.01031
Feature: 11, Score: 0.11313
Feature: 12, Score: -0.17968
Feature: 13, Score: -0.00251
Feature: 14, Score: 0.14787
Feature: 15, Score: -0.13715
Feature: 16, Score: 0.06076
Feature: 17, Score: 0.93963
```

ARIMA

```
[ ]  fig, axes = plt.subplots(3, 2,figsize=(16,6),dpi=80, sharex=True)
     axes[0, 0].plot(df.Actual); axes[0, 0].set_title('Original Series')
     plot_acf(df.Actual, ax=axes[0, 1])



     # 1st Differencing
     axes[1, 0].plot(df.Actual.diff()); axes[1, 0].set_title('1st Order Differencing')
     plot_acf(df.Actual.diff().dropna(), ax=axes[1, 1])



     # 2nd Differencing
     axes[2, 0].plot(df.Actual.diff().diff()); axes[2, 0].set_title('2nd Order Differencing')
     plot_acf(df.Actual.diff().diff().dropna(), ax=axes[2, 1])


     plt.show()
```


Original Series
1st Order Differencing
2nd Order Differencing
Autocorrelation
Autocorrelation
Autocorrelation

Auto Correlation Vs Partial Correlation

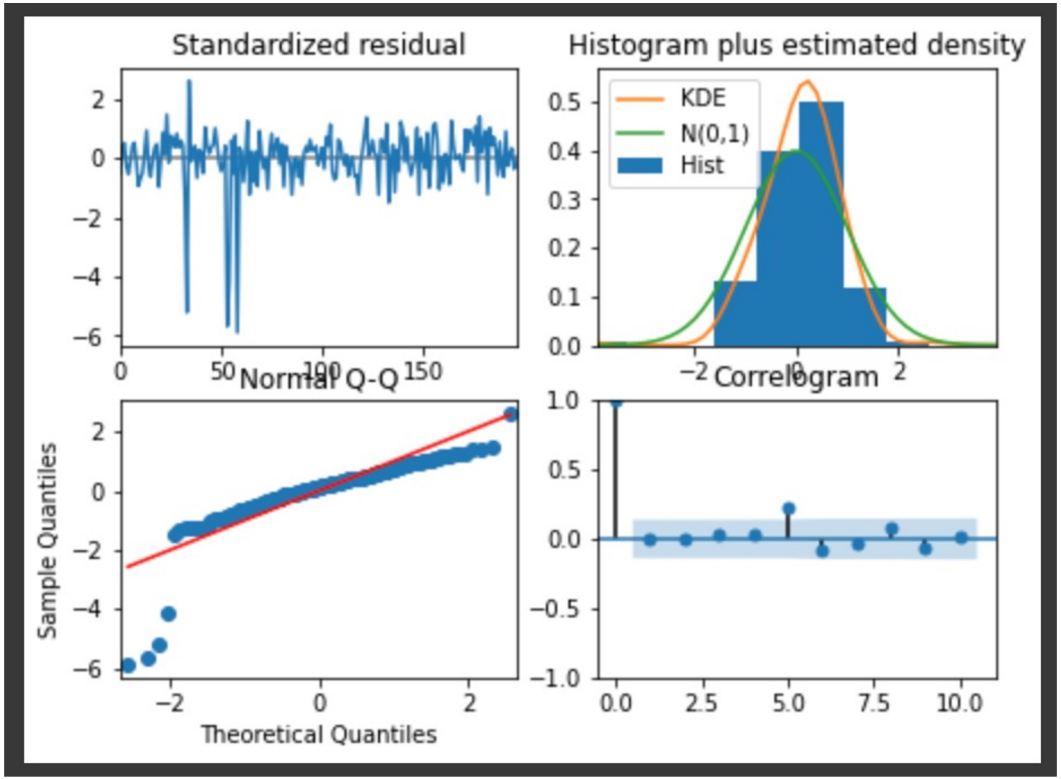


ARIMA MODEL

```
[ ] from statsmodels.tsa.arima.model import ARIMA
    model = ARIMA(df.Actual, order=(1,0,0))
    model_fit = model.fit()
    print(model_fit.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                 Actual   No. Observations:                  241
Model:                 ARIMA(1, 0, 0)   Log Likelihood               -1024.521
Date:                Tue, 21 Feb 2023   AIC                           2055.043
Time:                        05:31:02   BIC                           2065.497
Sample:                             0   HQIC                          2059.254
                                  - 241
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        116.3674      2.130     54.626      0.000     112.192     120.543
ar.L1          0.2612      0.031      8.323      0.000       0.200       0.323
sigma2       288.2230     12.226     23.575      0.000     264.261     312.185
===================================================================================
Ljung-Box (L1) (Q):                   0.05   Jarque-Bera (JB):              2924.98
Prob(Q):                              0.82   Prob(JB):                         0.00
Heteroskedasticity (H):               0.19   Skew:                            -2.96
Prob(H) (two-sided):                  0.00   Kurtosis:                        19.01
===================================================================================
```

Plots


Standardized residual
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Correlogram
Sample Quantiles
Theoretical Quantiles

Thank you!