In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [2]:
```
data = pd.read_csv(r"C:\DS Assignments\Assignment 4(simple linear regression)\delivery_time.csv")
```

In [3]:
```
data
```

Out[3]:

	Delivery Time	Sorting Time
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
4	29.00	10
5	15.35	6
6	19.00	7
7	9.50	3
8	17.90	10
9	18.75	9
10	19.83	8
11	10.75	4
12	16.68	7
13	11.50	3
14	12.03	3
15	14.88	4
16	13.75	6
17	18.11	7
18	8.00	2
19	17.83	7
20	21.50	5

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Delivery Time  21 non-null     float64
 1   Sorting Time   21 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 464.0 bytes
```

In [5]:
```
data.describe()
```

Out[5]:

	Delivery Time	Sorting Time
count	21.000000	21.000000
mean	16.790952	6.190476
std	5.074901	2.542028
min	8.000000	2.000000
25%	13.500000	4.000000
50%	17.830000	6.000000
75%	19.750000	8.000000
max	29.000000	10.000000

In [6]:
```
data.var()
```

Out[6]:
```
Delivery Time    25.754619
```

```
Sorting Time      6.461905
dtype: float64
```

Correlation

In [7]:
```
data.corr()
```

Out[7]:

	Delivery Time	Sorting Time
Delivery Time	1.000000	0.825997
Sorting Time	0.825997	1.000000

In [8]:
```
import seaborn as sns
sns.distplot(data['Delivery Time'])
```

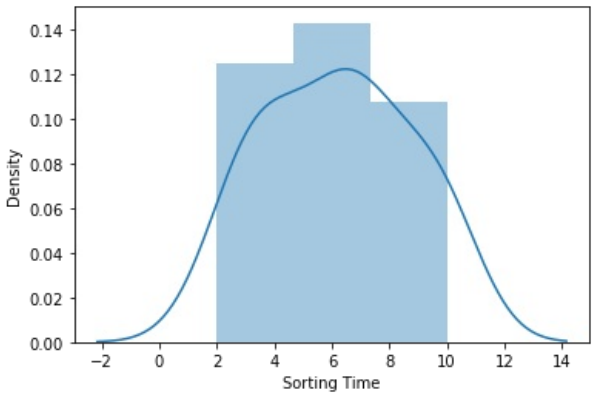
```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:
```
<AxesSubplot:xlabel='Delivery Time', ylabel='Density'>
```

In [9]:
```
import seaborn as sns
sns.distplot(data['Sorting Time'])
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:
```
<AxesSubplot:xlabel='Sorting Time', ylabel='Density'>
```



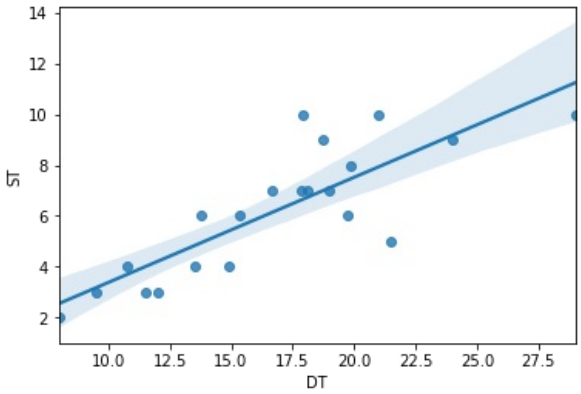
Fitting a Linear Regression Model

In [10]:
```
data=data.rename({'Delivery Time': 'DT'}, axis=1)
data=data.rename({'Sorting Time': 'ST'}, axis=1)
```

In [11]:
```
import statsmodels.formula.api as smf
model = smf.ols("ST~DT",data = data).fit()
```

In [12]:
```
sns.regplot(x="DT", y="ST", data=data)
```

Out[12]:
```
<AxesSubplot:xlabel='DT', ylabel='ST'>
```



In [13]:
```
#Coefficients
model.params
```

Out[13]:
```
Intercept   -0.756673
DT           0.413744
dtype: float64
```

In [14]:
```
#t and p-Values
print(model.tvalues, '\n', model.pvalues)
```

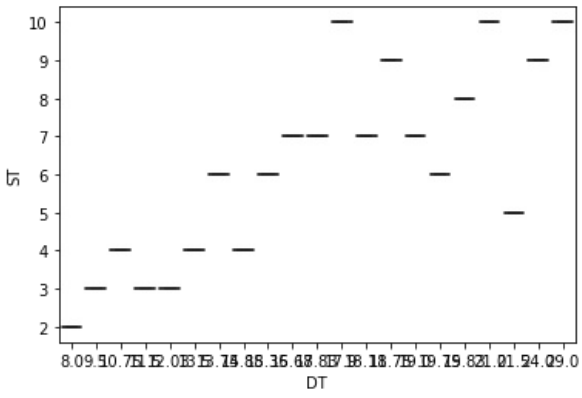
```
Intercept   -0.667290
DT           6.387447
dtype: float64
 Intercept    0.512611
DT           0.000004
dtype: float64
```

In [15]:
```
#R squared values
(model.rsquared,model.rsquared_adj)
```

Out[15]:
```
(0.6822714748417232, 0.6655489208860245)
```

In [53]:
```
import seaborn as sb
sb.boxplot(x ='DT',y ='ST',data = data,color='green')
```

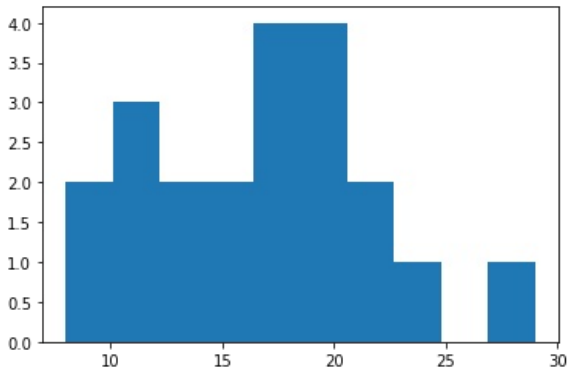
Out[53]:
```
<AxesSubplot:xlabel='DT', ylabel='ST'>
```



In [17]:
```
plt.hist(data.DT)
```

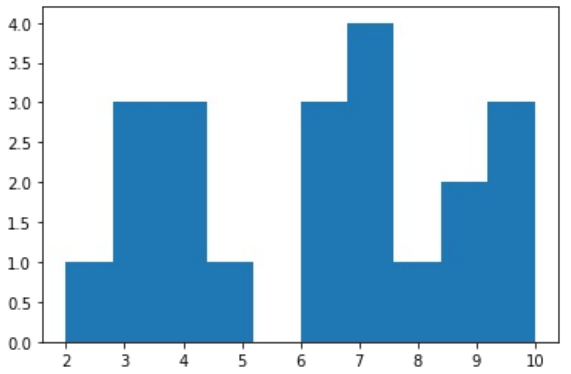
Out[17]:
```
(array([2., 3., 2., 2., 4., 4., 2., 1., 0., 1.]),
 array([ 8. , 10.1, 12.2, 14.3, 16.4, 18.5, 20.6, 22.7, 24.8, 26.9, 29. ]),
```

```
 <BarContainer object of 10 artists>)
```



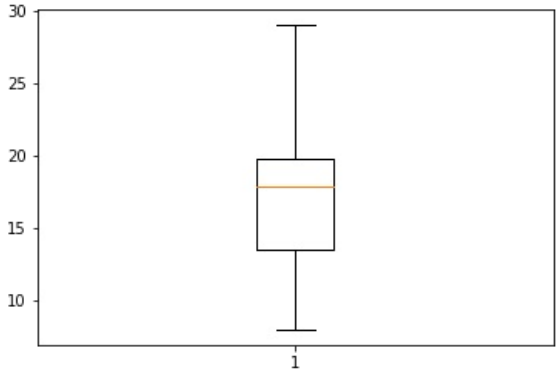
In [18]:
```
plt.hist(data.ST)
```

Out[18]:
```
(array([1., 3., 3., 1., 0., 3., 4., 1., 2., 3.]),
 array([ 2. ,  2.8,  3.6,  4.4,  5.2,  6. ,  6.8,  7.6,  8.4,  9.2, 10. ]),
 <BarContainer object of 10 artists>)
```



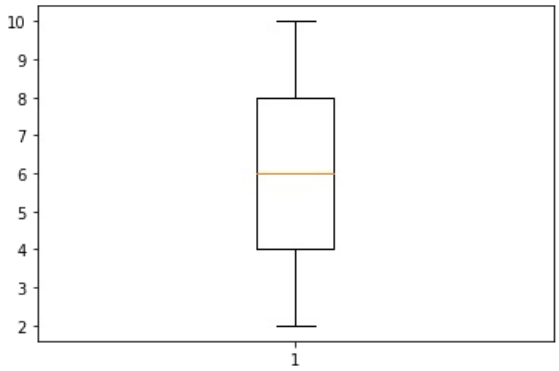
In [19]:
```
plt.boxplot(data.DT)
```

Out[19]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x255c69d5310>,
  <matplotlib.lines.Line2D at 0x255c69d5670>],
 'caps': [<matplotlib.lines.Line2D at 0x255c69d59d0>,
  <matplotlib.lines.Line2D at 0x255c69d5d30>],
 'boxes': [<matplotlib.lines.Line2D at 0x255c69c8f70>],
 'medians': [<matplotlib.lines.Line2D at 0x255c69e00d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x255c69e0430>],
 'means': []}
```



In [20]:
```
plt.boxplot(data.ST)
```

Out[20]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x255c6a2e370>,
  <matplotlib.lines.Line2D at 0x255c6a2e6d0>],
 'caps': [<matplotlib.lines.Line2D at 0x255c6a2ea30>,
  <matplotlib.lines.Line2D at 0x255c6a2ed90>],
 'boxes': [<matplotlib.lines.Line2D at 0x255c6a1ffa0>],
 'medians': [<matplotlib.lines.Line2D at 0x255c6a39130>],
 'fliers': [<matplotlib.lines.Line2D at 0x255c6a39490>],
 'means': []}
```



In [21]:
```
import statsmodels.formula.api as smf
model_train = smf.ols("np.log(DT)~np.log(ST)",data =data).fit()
```

In [22]:
```
model_train.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	np.log(DT)	**R-squared:**	0.772
Model:	OLS	**Adj. R-squared:**	0.760
Method:	Least Squares	**F-statistic:**	64.39
Date:	Wed, 18 Aug 2021	**Prob (F-statistic):**	1.60e-07
Time:	15:56:54	**Log-Likelihood:**	10.291
No. Observations:	21	**AIC:**	-16.58
Df Residuals:	19	**BIC:**	-14.49
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.7420	0.133	13.086	0.000	1.463	2.021
np.log(ST)	0.5975	0.074	8.024	0.000	0.442	0.753

Omnibus:	1.871	**Durbin-Watson:**	1.322
Prob(Omnibus):	0.392	**Jarque-Bera (JB):**	1.170
Skew:	0.577	**Prob(JB):**	0.557
Kurtosis:	2.916	**Cond. No.**	9.08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [26]:
```
data['sortingtime_sq'] = data.ST*data.ST
```

In [27]:
```
data['sortingtime_cb'] = data.ST*data.ST*data.ST
```

In [29]:
```
model_train1 = smf.ols("DT~ST+sortingtime_sq+sortingtime_cb",data = data).fit()
```

In [30]:
```
model_train1.summary()
```

Out[30]:

OLS Regression Results

Dep. Variable:	DT	**R-squared:**	0.703
Model:	OLS	**Adj. R-squared:**	0.651
Method:	Least Squares	**F-statistic:**	13.44
Date:	Wed, 18 Aug 2021	**Prob (F-statistic):**	9.59e-05
Time:	16:03:59	**Log-Likelihood:**	-50.633
No. Observations:	21	**AIC:**	109.3
Df Residuals:	17	**BIC:**	113.4
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.1582	10.987	-0.378	0.710	-27.338	19.021
ST	7.5025	6.370	1.178	0.255	-5.937	20.942
sortingtime_sq	-0.9253	1.106	-0.837	0.414	-3.258	1.407
sortingtime_cb	0.0445	0.059	0.757	0.460	-0.079	0.168

Omnibus:	2.616	**Durbin-Watson:**	1.369
Prob(Omnibus):	0.270	**Jarque-Bera (JB):**	1.428
Skew:	0.630	**Prob(JB):**	0.490
Kurtosis:	3.204	**Cond. No.**	9.50e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.5e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

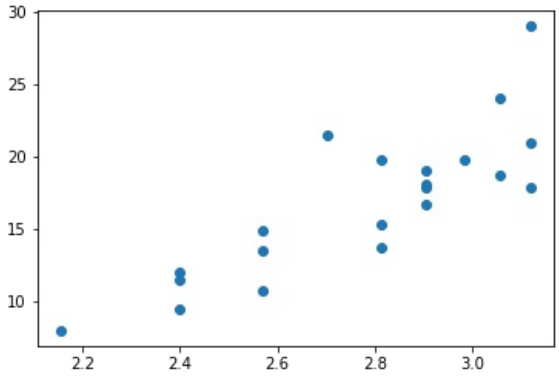
In [31]:
```
model_train.conf_int = 0.05
```

In [34]:
```
pred_train = model_train.predict(data['ST'])
```

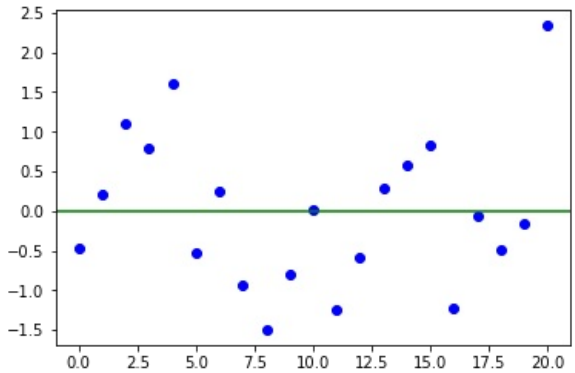
In [36]:
```
resid = pred_train-data.DT
```

In [37]:
```
stand_resid = model_train.resid_pearson
```

In [39]:
```
plt.scatter(pred_train,data.DT)
plt.xlabel = "predicted values"
plt.ylabel = "actual values"
```



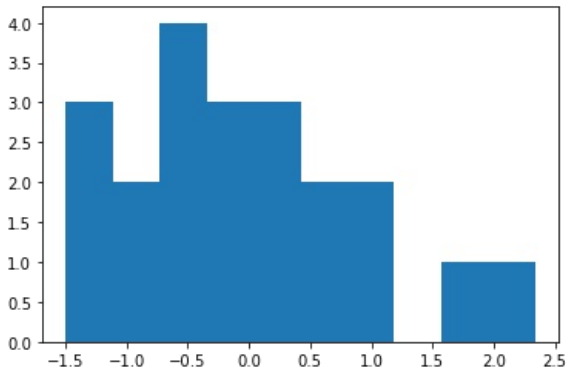
In [40]:
```
plt.plot(model_train.resid_pearson,"bo")
plt.axhline(y=0,color = 'green')
plt.xlabel = "observation number"
plt.ylabel ="standardized value"
```



In [41]:
```
plt.hist(model_train.resid_pearson)
```

Out[41]:
```
(array([3., 2., 4., 3., 3., 2., 2., 0., 1., 1.]),
 array([-1.49533389, -1.11197641, -0.72861893, -0.34526145,  0.03809603,
         0.42145351,  0.80481099,  1.18816846,  1.57152594,  1.95488342,
         2.3382409 ]),
```

<BarContainer object of 10 artists>)



In [42]:
```
np.sqrt(np.mean(resid*resid))    #RSME
```

Out[42]: 14.791938693055753

In [44]:
```
# testing
test_pred = model_train.predict(data)
```

In [45]:
```
test_resid = test_pred-data.DT
```

In [46]:
```
np.sqrt(np.mean(test_resid*test_resid))
```

Out[46]: 14.791938693055753

Predict for new data point

In [54]:
```
#Predict for 9 and 10 daily circulation
newdata=pd.Series([21.00,17.90])
```

In [55]:
```
data_pred=pd.DataFrame(newdata,columns=['DT'])
```

In [56]:
```
model.predict(data_pred)
```

Out[56]:
```
0    7.931943
1    6.649338
dtype: float64
```

In [57]:
```
y= -0.756673+0.413744*(21.00)
y
```

Out[57]: 7.931951000000001

In [58]:
```
y1= -0.756673+0.413744*(17.90)
y1
```

Out[58]: 6.649344599999999