In [1]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
```

In [4]:

```python
pokemon=pd.read_excel("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\Excel file\\pokemon.xlsx")
pokemon.head()
```

Out[4]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

In [5]:

```python
#shape of dataset
pokemon.shape
```

Out[5]:

```
(800, 13)
```

In [6]:

```python
# information about dataset
pokemon.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   #           800 non-null    int64 
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      414 non-null    object
 4   Total       800 non-null    int64 
 5   HP          800 non-null    int64 
 6   Attack      800 non-null    int64 
 7   Defense     800 non-null    int64 
 8   Sp. Atk     800 non-null    int64 
 9   Sp. Def     800 non-null    int64 
 10  Speed       800 non-null    int64 
 11  Generation  800 non-null    int64 
 12  Legendary   800 non-null    bool  
dtypes: bool(1), int64(9), object(3)
memory usage: 75.9+ KB
```

In [7]:

```python
#Describe of dataset
pokemon.describe()
```

Out[7]:

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	800.000000	800.00000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.00000
mean	362.813750	435.10250	69.258750	79.001250	73.842500	72.820000	71.902500	68.277500	3.32375
std	208.343798	119.96304	25.534669	32.457366	31.183501	32.722294	27.828916	29.060474	1.66129
min	1.000000	180.00000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000	1.00000
25%	184.750000	330.00000	50.000000	55.000000	50.000000	49.750000	50.000000	45.000000	2.00000
50%	364.500000	450.00000	65.000000	75.000000	70.000000	65.000000	70.000000	65.000000	3.00000
75%	539.250000	515.00000	80.000000	100.000000	90.000000	95.000000	90.000000	90.000000	5.00000
max	721.000000	780.00000	255.000000	190.000000	230.000000	194.000000	230.000000	180.000000	6.00000

In [10]:

```python
# find ot null value in dataset
pokemon.isnull().sum()
```

Out[10]:

```
#               0
Name            0
Type 1          0
Type 2        386
Total           0
HP              0
Attack          0
Defense         0
Sp. Atk         0
Sp. Def         0
Speed           0
Generation      0
Legendary       0
dtype: int64
```

In [12]:

```python
# Changing the column name
pokemon.rename(columns={'Type 1':'primary_type','Type 2':'secondary_type'},inplace=True)
```

In [13]:

```python
pokemon.head()
```

Out[13]:

	#	Name	primary_type	secondary_type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

In [18]:

```python
# Classify the grass pokemon
grass_pokemon=pokemon[pokemon['primary_type']=='Grass']
```

In [19]:

```python
grass_pokemon.head()
```

Out[19]:

	#	Name	primary_type	secondary_type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
48	43	Oddish	Grass	Poison	320	45	50	55	75	65	30	1	False

In [22]:

```python
# Classify the fire pokemon
fire_pokemon=pokemon[pokemon['primary_type']=='Fire']
```

In [23]:

```python
fire_pokemon.head()
```

Out[23]:

	#	Name	primary_type	secondary_type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
5	5	Charmeleon	Fire	NaN	405	58	64	58	80	65	80	1	False
6	6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
7	6	CharizardMega Charizard X	Fire	Dragon	634	78	130	111	130	85	100	1	False
8	6	CharizardMega Charizard Y	Fire	Flying	634	78	104	78	159	115	100	1	False

In [25]:

```python
# Classify the Water pokemon
water_pokemon=pokemon[pokemon['primary_type']=='Water']
```

In [26]:

```python
water_pokemon.head()
```

Out[26]:

	#	Name	primary_type	secondary_type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
9	7	Squirtle	Water	NaN	314	44	48	65	50	64	43	1	False
10	8	Wartortle	Water	NaN	405	59	63	80	65	80	58	1	False
11	9	Blastoise	Water	NaN	530	79	83	100	85	105	78	1	False
12	9	BlastoiseMega Blastoise	Water	NaN	630	79	103	120	135	115	78	1	False
59	54	Psyduck	Water	NaN	320	50	52	48	65	50	55	1	False

In [28]:

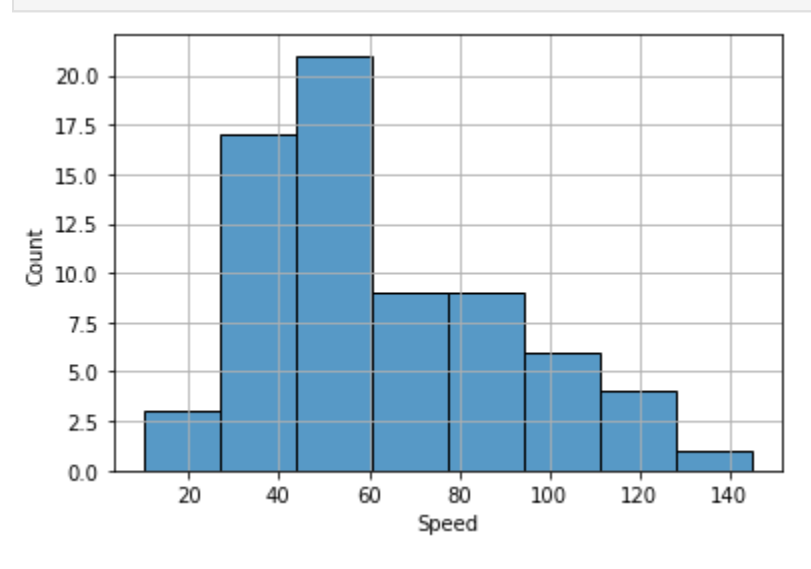
```python
# measuring the types of pokemon
pokemon['primary_type'].value_counts()
```

Out[28]:

```
Water       112
Normal       98
Grass        70
Bug          69
Psychic      57
Fire         52
Electric     44
Rock         44
Dragon       32
Ground       32
Ghost        32
Dark         31
Poison       28
Steel        27
Fighting     27
Ice          24
Fairy        17
Flying        4
Name: primary_type, dtype: int64
```

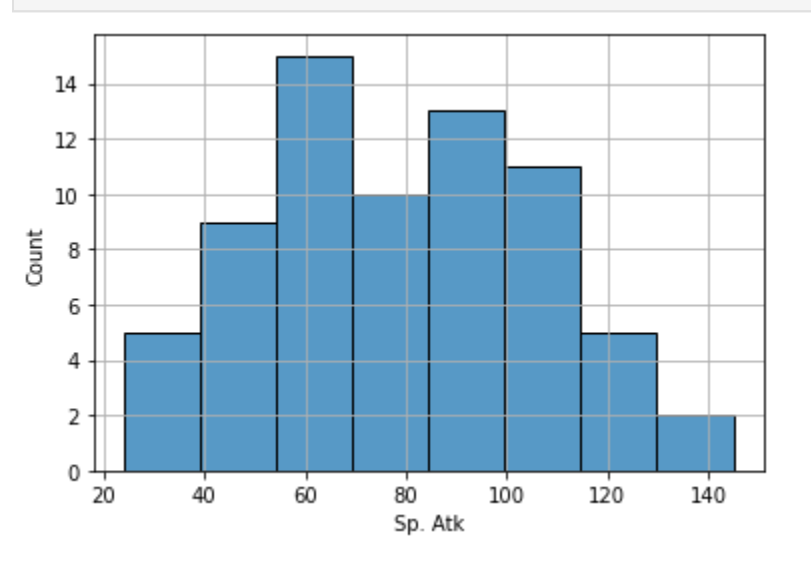
In [31]:

```python
# Histogram
sns.histplot(grass_pokemon['Speed'])
plt.grid(True)
plt.show()
```



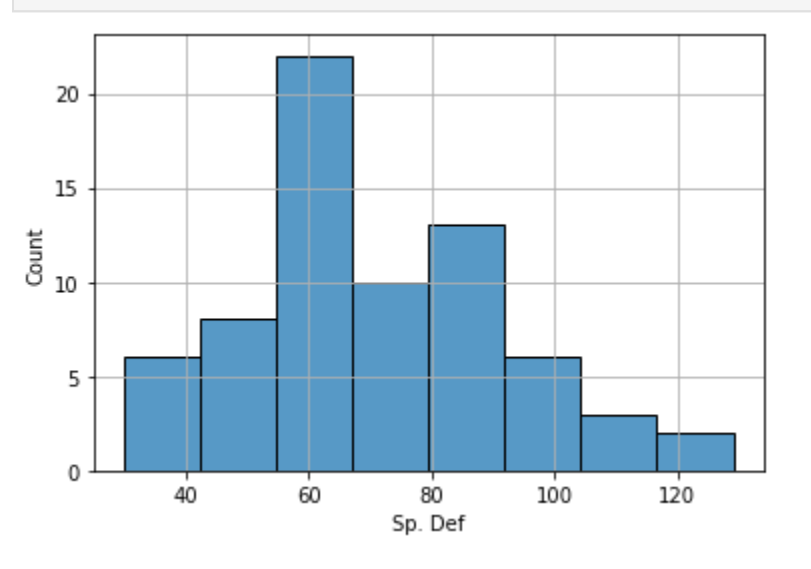
In [32]:

```python
sns.histplot(grass_pokemon['Sp. Atk'])
plt.grid(True)
plt.show()
```



In [33]:

```python
sns.histplot(grass_pokemon['Sp. Def'])
plt.grid(True)
plt.show()
```



In [34]:

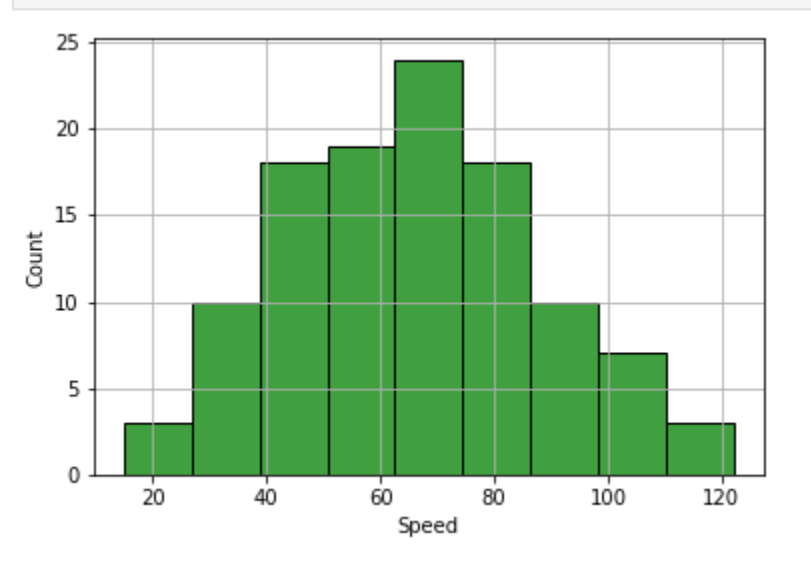
```python
grass_pokemon.describe()
```

Out[34]:

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000
mean	344.871429	421.142857	67.271429	73.214286	70.800000	77.500000	70.428571	61.928571	3.357143
std	200.264385	106.650626	19.516564	25.380520	24.485192	27.244864	21.446645	28.506456	1.579173
min	1.000000	180.000000	30.000000	27.000000	30.000000	24.000000	30.000000	10.000000	1.000000
25%	187.250000	318.500000	51.250000	55.000000	50.000000	57.000000	55.000000	40.000000	2.000000
50%	372.000000	430.000000	65.500000	70.000000	66.000000	75.000000	66.000000	58.500000	3.500000
75%	496.750000	497.000000	75.000000	93.500000	84.500000	99.500000	85.000000	80.000000	5.000000
max	673.000000	630.000000	123.000000	132.000000	131.000000	145.000000	129.000000	145.000000	6.000000

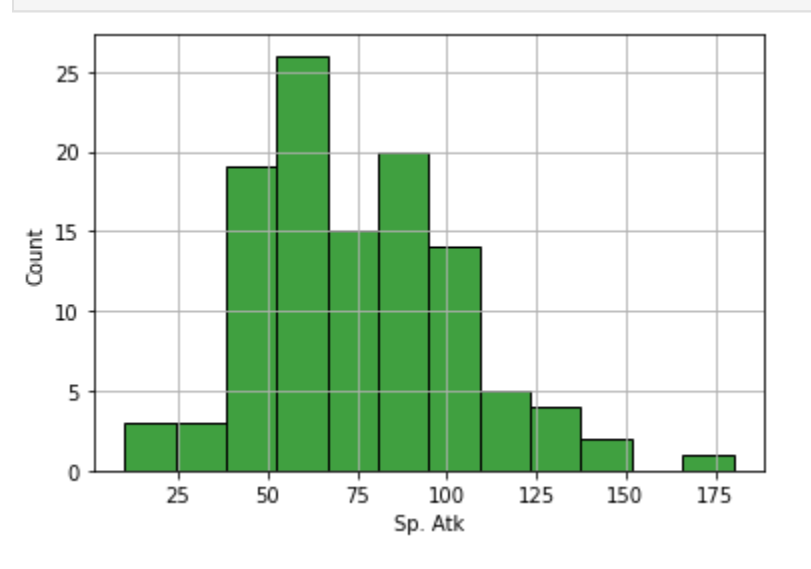
In [36]:

```python
#histogram
sns.histplot(water_pokemon['Speed'],color='green')
plt.grid(True)
plt.show()
```



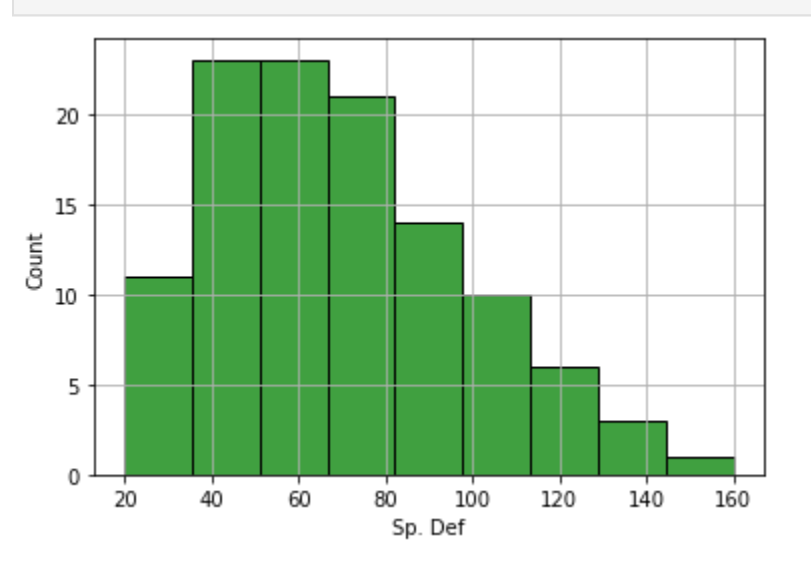
In [37]:

```python
sns.histplot(water_pokemon['Sp. Atk'],color='green')
plt.grid(True)
plt.show()
```



In [38]:

```python
sns.histplot(water_pokemon['Sp. Def'],color='green')
plt.grid(True)
plt.show()
```



In [39]:

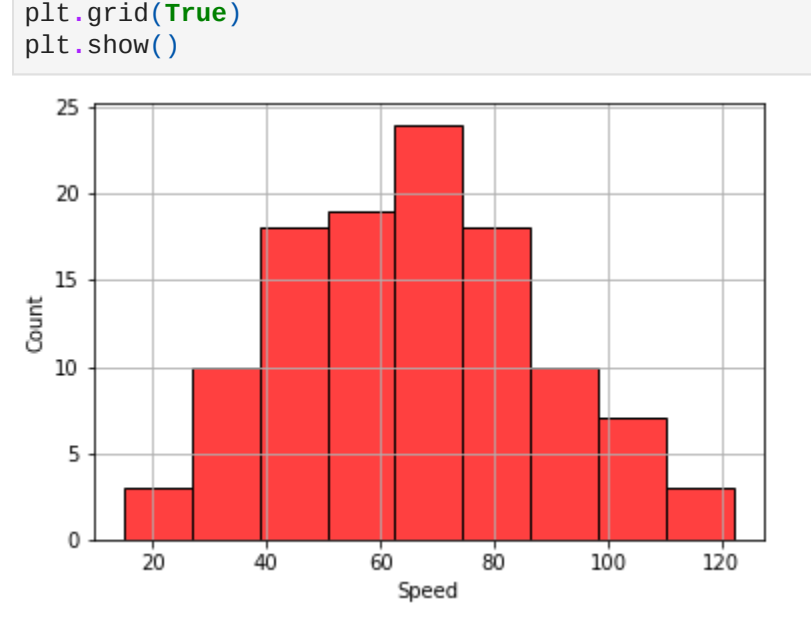
```python
water_pokemon.describe()
```

Out[39]:

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	112.000000	112.000000	112.000000	112.000000	112.000000	112.000000	112.000000	112.000000	112.000000
mean	303.089286	430.455357	72.062500	74.151786	72.946429	74.812500	70.517857	65.964286	2.857143
std	188.440807	113.188266	27.487026	28.377192	27.773809	29.030128	28.460493	23.019353	1.558800
min	7.000000	200.000000	20.000000	10.000000	20.000000	10.000000	20.000000	15.000000	1.000000
25%	130.000000	328.750000	52.250000	53.000000	54.500000	55.000000	50.000000	50.000000	1.000000
50%	275.000000	455.000000	70.000000	72.000000	70.000000	70.000000	65.000000	65.000000	3.000000
75%	456.250000	502.250000	90.250000	92.000000	88.500000	90.500000	89.250000	82.000000	4.000000
max	693.000000	770.000000	170.000000	155.000000	180.000000	180.000000	160.000000	122.000000	6.000000

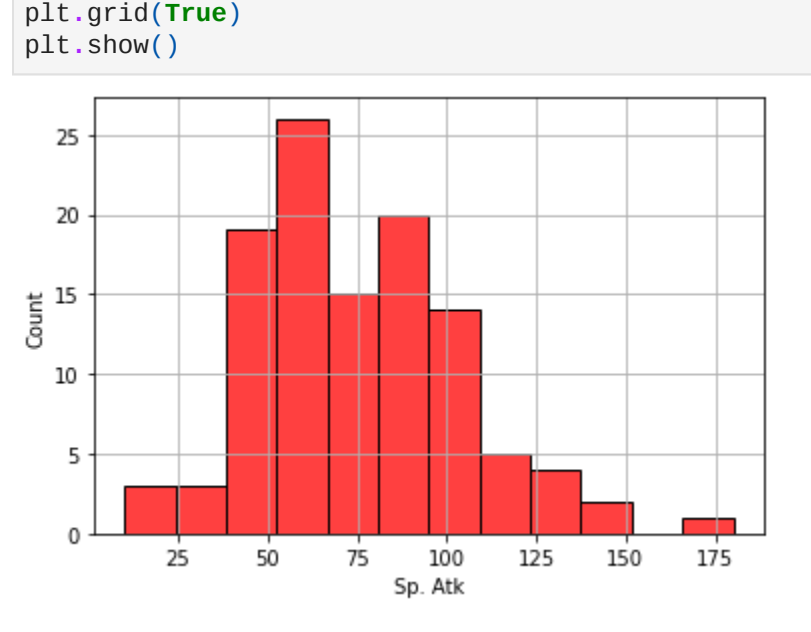
In [42]:

```python
# Histogram
sns.histplot(fire_pokemon['Speed'],color='red')
plt.grid(True)
plt.show()
```



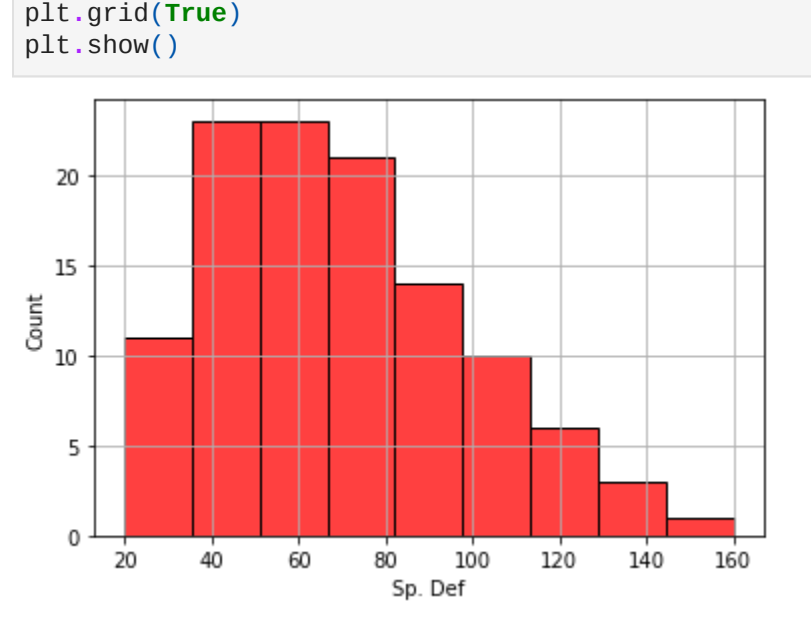
In [43]:

```python
sns.histplot(fire_pokemon['Sp. Atk'],color='red')
plt.grid(True)
plt.show()
```



In [44]:

```python
sns.histplot(fire_pokemon['Sp. Def'],color='red')
plt.grid(True)
plt.show()
```



In [45]:

```python
fire_pokemon.describe()
```

Out[45]:

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	112.000000	112.000000	112.000000	112.000000	112.000000	112.000000	112.000000	112.000000	112.000000
mean	303.089286	430.455357	72.062500	74.151786	72.946429	74.812500	70.517857	65.964286	2.857143
std	188.440807	113.188266	27.487026	28.377192	27.773809	29.030128	28.460493	23.019353	1.558800
min	7.000000	200.000000	20.000000	10.000000	20.000000	10.000000	20.000000	15.000000	1.000000
25%	130.000000	328.750000	52.250000	53.000000	54.500000	55.000000	50.000000	50.000000	1.000000
50%	275.000000	455.000000	70.000000	72.000000	70.000000	70.000000	65.000000	65.000000	3.000000
75%	456.250000	502.250000	90.250000	92.000000	88.500000	90.500000	89.250000	82.000000	4.000000
max	693.000000	770.000000	170.000000	155.000000	180.000000	180.000000	160.000000	122.000000	6.000000

In [48]:

```python
# measuring the legendary vaule in dataset
pokemon['Legendary'].value_counts()
```

Out[48]:

```
False    735
True      65
Name: Legendary, dtype: int64
```

In [49]:

```python
65/800
```

Out[49]:

```
0.08125
```

In [53]:

```python
x=pokemon[['Speed']]
```

In [54]:

```python
y=pokemon[['Legendary']]
```

In [56]:

```python
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3)
print("shape of x_train= ",x_train.shape)
print("shape of x_test= ",x_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of x_train=  (560, 1)
shape of x_test=  (240, 1)
shape of y_train=  (560, 1)
shape of y_test=  (240, 1)
```

In [57]:

```python
from sklearn.tree import DecisionTreeClassifier
```

In [58]:

```python
dtc=DecisionTreeClassifier()
```

In [59]:

```python
dtc.fit(x_train,y_train)
```

Out[59]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [60]:

```python
y_pred=dtc.predict(x_test)
```

In [61]:

```python
from sklearn.metrics import confusion_matrix
```

In [63]:

```python
confusion_matrix(y_test,y_pred)
```

Out[63]:

```
array([[216,   2],
       [ 21,   1]], dtype=int64)
```

In [67]:

```python
# Accuracy of model
[(216+0)/(217+0+22+1)]
```

Out[67]:

```
[0.9]
```

In [68]:

```python
pokemon.head()
```

Out[68]:

	#	Name	primary_type	secondary_type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

In [74]:

```python
x=pokemon[['Defense']]
y=pokemon[['Attack']]
```

In [75]:

```python
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3)
print("shape of x_train= ",x_train.shape)
print("shape of x_test= ",x_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of x_train=  (560, 1)
shape of x_test=  (240, 1)
shape of y_train=  (560, 1)
shape of y_test=  (240, 1)
```

In [76]:

```python
dtc=DecisionTreeClassifier()
```

In [78]:

```python
dtc.fit(x_train,y_train)
```

Out[78]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [79]:

```python
y_pred=dtc.predict(x_test)
```

In [81]:

```python
from sklearn.metrics import mean_squared_error
```

In [82]:

```python
mean_squared_error(y_test,y_pred)
```

Out[82]:

```
1291.575
```

In [ ]: