This data set includes 898 Pokemon, 1072 including alternate forms, including their number, name, first and second type, the stat total and basic stats: HP, Attack, Defense, Special Attack, Special Defense, and Speed, generation, and legendary status. It has been of great use when teaching statistics to kids.

In [65]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('Pokemon.csv')
df.head()
```

Out[2]:

	#	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 [3]:
```
df.rename(columns={'Type 1':'Primary_Type','Type 2':'Secondary_Type'},inplace=True)
```

In [4]:
```
df.head()
```

Out[4]:

	#	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 [5]:
```
Grass_pokemon=df[df['Primary_Type']=='Grass']
```

In [6]:
```
Grass_pokemon.head()
```

Out[6]:

	#	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 [7]:
```
water_pokemon=df[df['Primary_Type']=='Water']
```

In [8]:
```
water_pokemon.head()
```

Out[8]:

	#	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 [9]:
```
fire_pokemon=df[df['Primary_Type']=='Fire']
```

In [10]:
```
fire_pokemon.head()
```

Out[10]:

	#	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 [11]:
```
df.shape
```

Out[11]: (800, 13)

In [12]:
```
Grass_pokemon.shape
```

Out[12]: (70, 13)

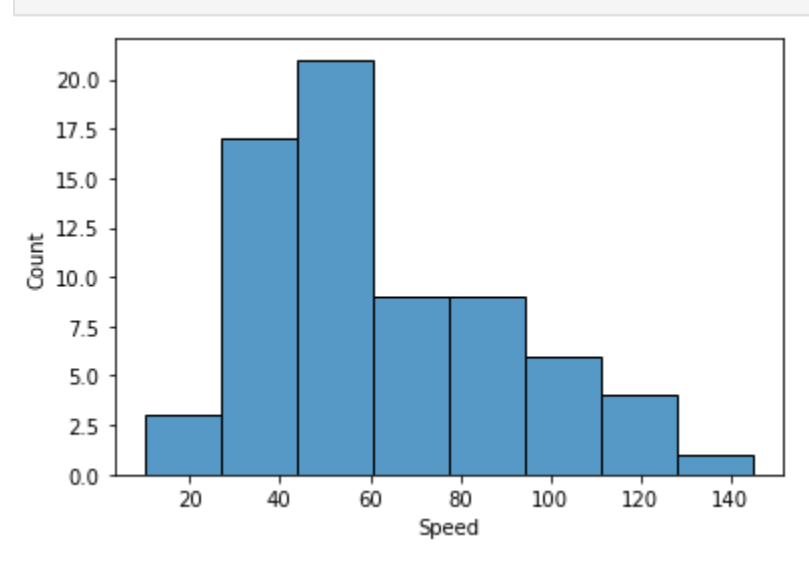
In [13]:
```
water_pokemon.shape
```

Out[13]: (112, 13)

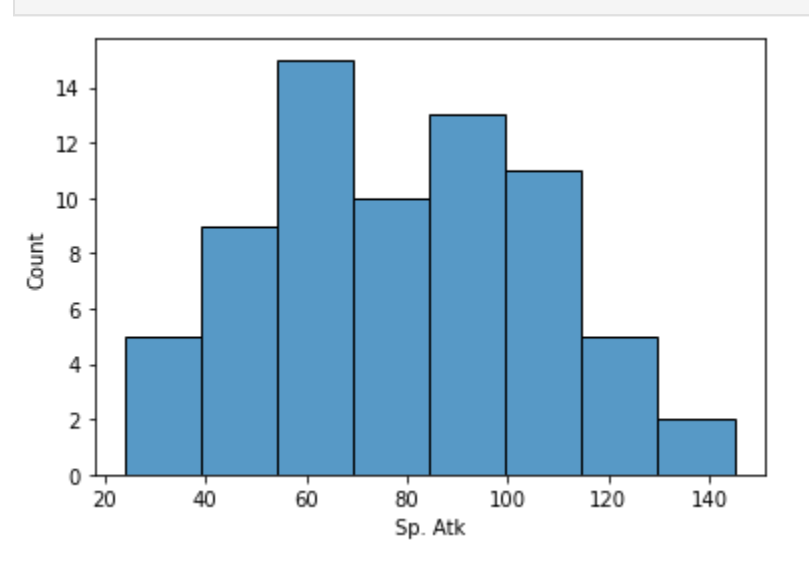
In [14]:
```
fire_pokemon.shape
```

Out[14]: (52, 13)

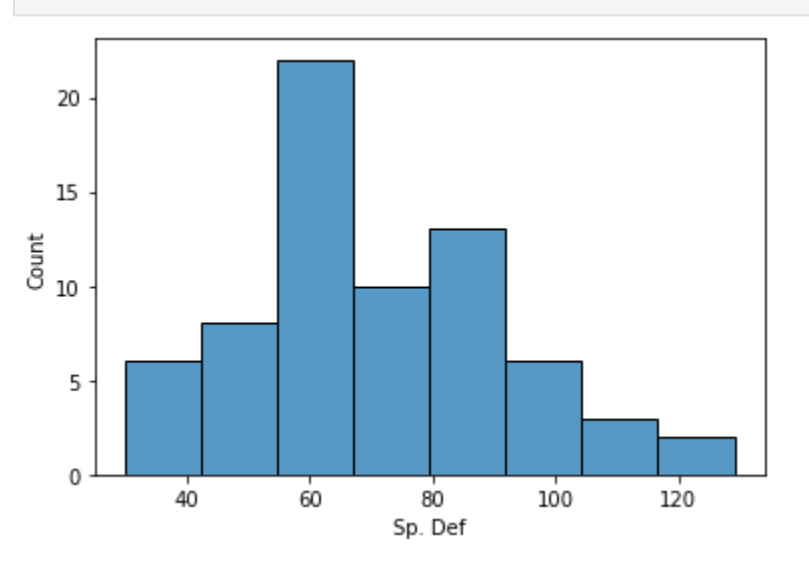
In [15]:
```
sns.histplot(Grass_pokemon['Speed'])
plt.show()
```



In [16]:
```
sns.histplot(Grass_pokemon['Sp. Atk'])
plt.show()
```



In [17]:
```
sns.histplot(Grass_pokemon['Sp. Def'])
plt.show()
```



In [18]:
```
Grass_pokemon.describe()
```

Out[18]:

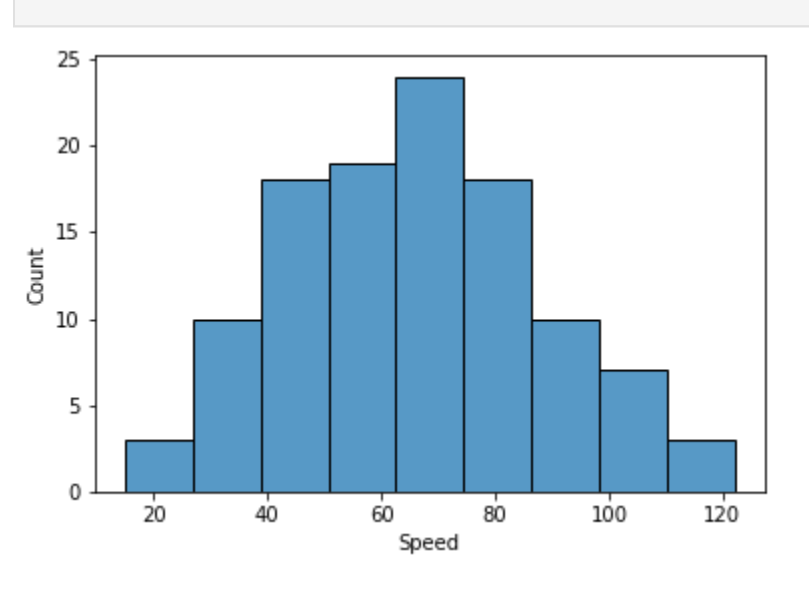
	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000
mean	344.871429	421.142857	67.271429	73.214286	70.800000	77.500000	70.428571	61.928571	3.357143
std	200.264385	106.650626	19.516564	25.380520	24.485192	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 [19]:
```
water_pokemon.head()
```

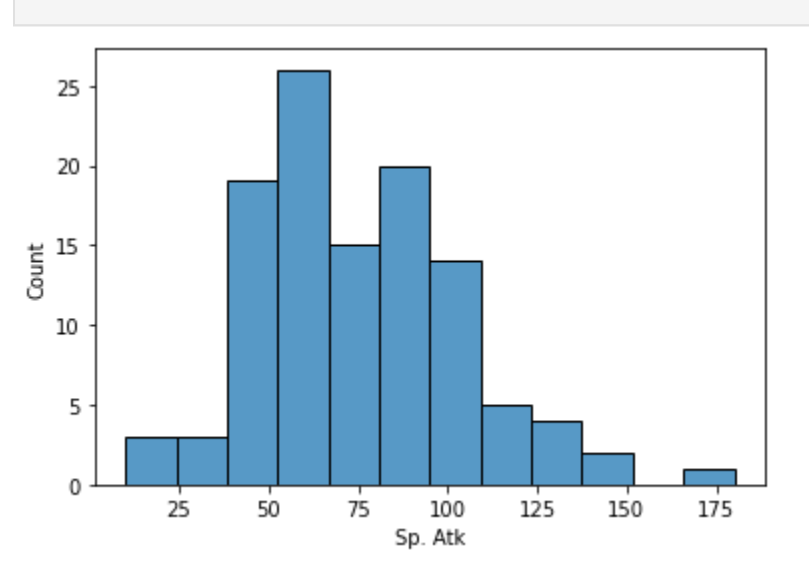
Out[19]:

	#	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

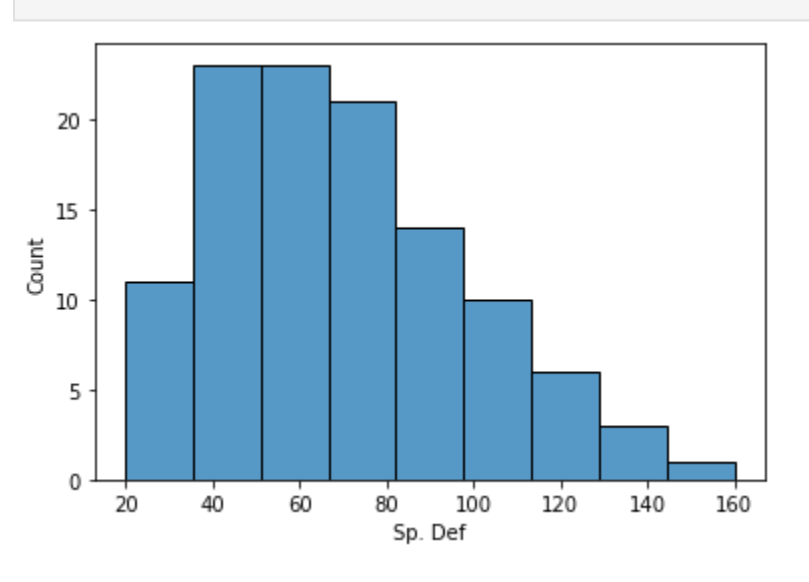
In [20]:
```
sns.histplot(water_pokemon['Speed'])
plt.show()
```



In [21]:
```
sns.histplot(water_pokemon['Sp. Atk'])
plt.show()
```



In [22]:
```
sns.histplot(water_pokemon['Sp. Def'])
plt.show()
```

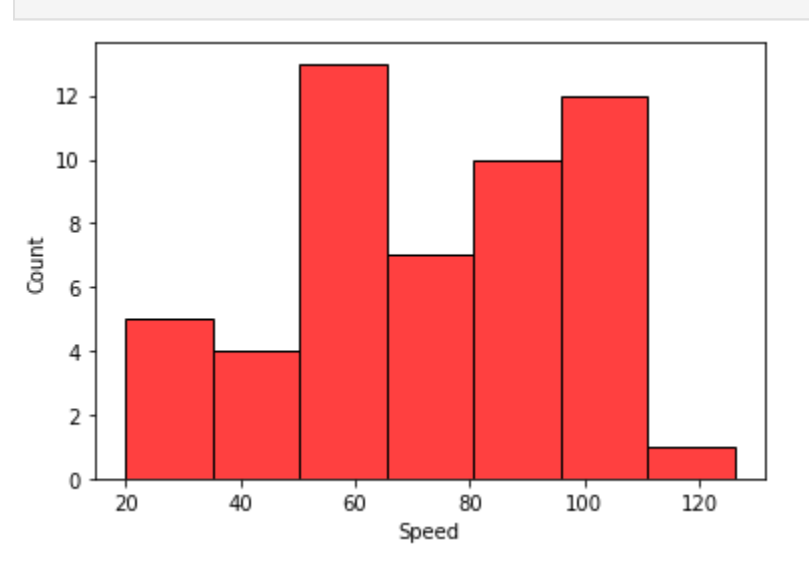


In [23]:
```
water_pokemon.describe()
```

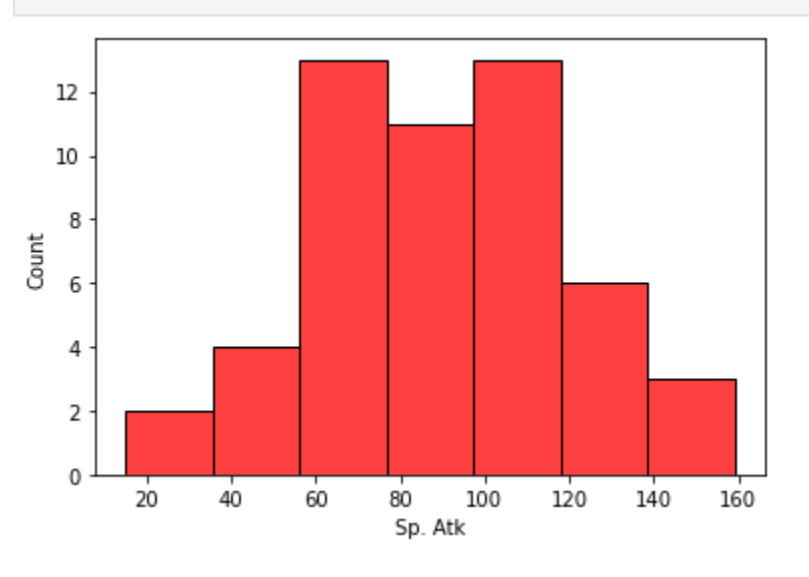
Out[23]:

	#	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

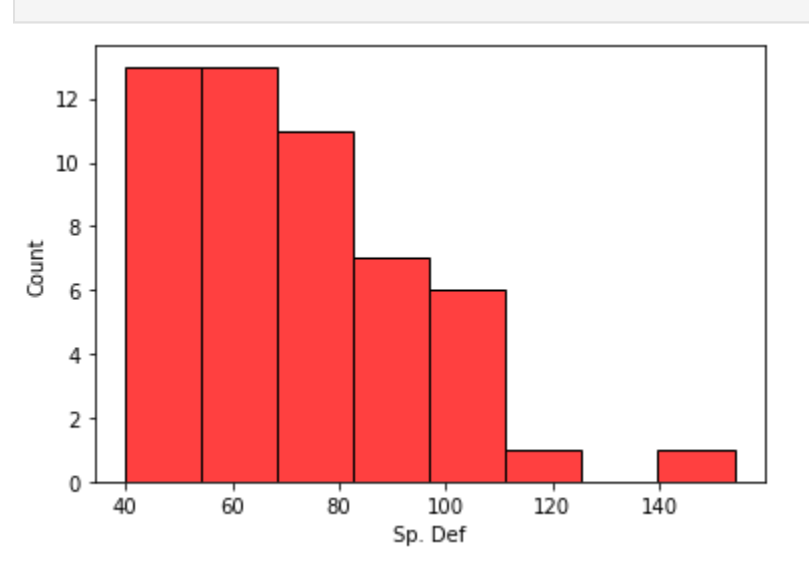
In [24]:
```
sns.histplot(fire_pokemon['Speed'],color='red')
plt.show()
```



In [25]:
```
sns.histplot(fire_pokemon['Sp. Atk'],color='red')
plt.show()
```



In [26]:
```
sns.histplot(fire_pokemon['Sp. Def'],color='red')
plt.show()
```



In [27]:
```
fire_pokemon.describe()
```

Out[27]:

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	52.000000	52.000000	52.000000	52.000000	52.000000	52.000000	52.000000	52.000000	52.000000
mean	327.403846	458.076923	69.903846	84.769231	67.769231	88.980769	72.211538	74.442308	3.211538
std	226.262840	109.760496	19.404123	28.769275	23.658200	30.042121	22.619908	25.245783	1.850665
min	4.000000	250.000000	38.000000	30.000000	37.000000	15.000000	40.000000	20.000000	1.000000
25%	143.500000	368.000000	58.000000	62.250000	51.000000	70.000000	54.750000	60.000000	1.000000
50%	289.500000	482.000000	70.000000	84.500000	64.000000	85.000000	67.500000	78.500000	3.000000
75%	513.250000	534.000000	80.000000	101.000000	78.000000	109.000000	85.000000	96.250000	5.000000
max	721.000000	680.000000	115.000000	160.000000	140.000000	159.000000	154.000000	126.000000	6.000000

In [28]:
```
df.head()
```

Out[28]:

	#	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 [29]:
```
df['Legendary'].value_counts()
```

Out[29]:
```
False    735
True      65
Name: Legendary, dtype: int64
```

In [30]:
```
x = df[['Speed']]
```

In [31]:
```
y = df[['Legendary']]
```

In [32]:
```
from sklearn.model_selection import train_test_split
```

In [33]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [34]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [35]:
```
model = DecisionTreeClassifier()
model
```

Out[35]: DecisionTreeClassifier()

In [36]:
```
model.fit(x_train,y_train)
```

Out[36]: DecisionTreeClassifier()

In [37]:
```
y_pred = model.predict(x_test)
```

In [38]:
```
y_pred
```

Out[38]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False])
```

In [39]:
```
from sklearn.metrics import confusion_matrix
```

In [40]:
```
print(confusion_matrix(y_test,y_pred))
```

```
[[218   2]
 [ 19   1]]
```

In [41]:
```
df.head()
```

Out[41]:

	#	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 [52]:
```
x = df[['Defense']]
```

In [53]:
```
y = df[['Attack']]
```

In [54]:
```
from sklearn.model_selection import train_test_split
```

In [55]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [56]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [57]:
```
model = DecisionTreeRegressor()
model
```

Out[57]: DecisionTreeRegressor()

In [59]:
```
model.fit(x_train,y_train)
```

Out[59]: DecisionTreeRegressor()

In [60]:
```
y_pred = model.predict(x_test)
```

In [61]:
```
y_test.head()
```

Out[61]:

	Attack
226	120
759	105
274	85
764	38
407	95

In [62]:
```
y_pred[0:5]
```

Out[62]:
```
array([ 95.        , 107.77777778,  75.81818182,  45.        ,
       106.12      ])
```

In [63]:
```
from sklearn.metrics import mean_squared_error
```

In [64]:
```
print(mean_squared_error(y_test,y_pred))
```

```
768.1778245082465
```

In [ ]: