Imports

In [1]:

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

In [2]:

```
data = pd.read_csv('/content/combine_rating_all_vehicle.csv')
data.head()
```

Out[2]:

	Rating	Model Name	Type
0	1.0	TVS iQube	2-wheeler
1	1.0	TVS iQube	2-wheeler
2	3.0	TVS iQube	2-wheeler
3	1.0	TVS iQube	2-wheeler
4	1.0	TVS iQube	2-wheeler

In [3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1113 entries, 0 to 1112
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Rating      1113 non-null   float64
 1   Model Name  1113 non-null   object 
 2   Type        1113 non-null   object 
dtypes: float64(1), object(2)
memory usage: 26.2+ KB
```

Pre Processing

In [4]:

```
col = 'Model Name'
modelCounts = pd.DataFrame(data[col].value_counts())
modelCounts.reset_index(inplace=True)
modelCounts.columns = ['Model Name', 'Count']
modelCounts.head()
```

Out[4]:

	Model Name	Count
0	Hero Electric Flash	102
1	Okinawa Praise	95
2	Hero Electric Optima	82
3	tata nexon ev	75
4	Tata Nexon EV	74

In [5]:

```
temp = modelCounts.sort_values(by=['Model Name']).reset_index()
temp = temp[list(temp.columns[1:])]
```

```
temp.head()
```

Out[5]:

	Model Name	Count
0	Ampere Magnus EX	28
1	Ampere Magnus Pro	22
2	Ampere REO	24
3	Ampere Zeal	13
4	Ather 450X	30

In [6]:

```
modelRating = pd.DataFrame(data.groupby(['Model Name', 'Type']).mean()).reset_inde
modelRating.head()
```

Out[6]:

	Model Name	Type	Rating
0	Ampere Magnus EX	2-wheeler	3.964286
1	Ampere Magnus Pro	2-wheeler	3.090909
2	Ampere REO	2-wheeler	2.583333
3	Ampere Zeal	2-wheeler	2.846154
4	Ather 450X	2-wheeler	3.666667

In [7]:

```
df = pd.concat([modelRating, temp], axis=1)
df = df.T.drop_duplicates().T

df['NewType'] = df['Type'].apply(lambda x: x.split('-')[0])
df.drop_duplicates(keep='first', inplace=True)
df.head()
```

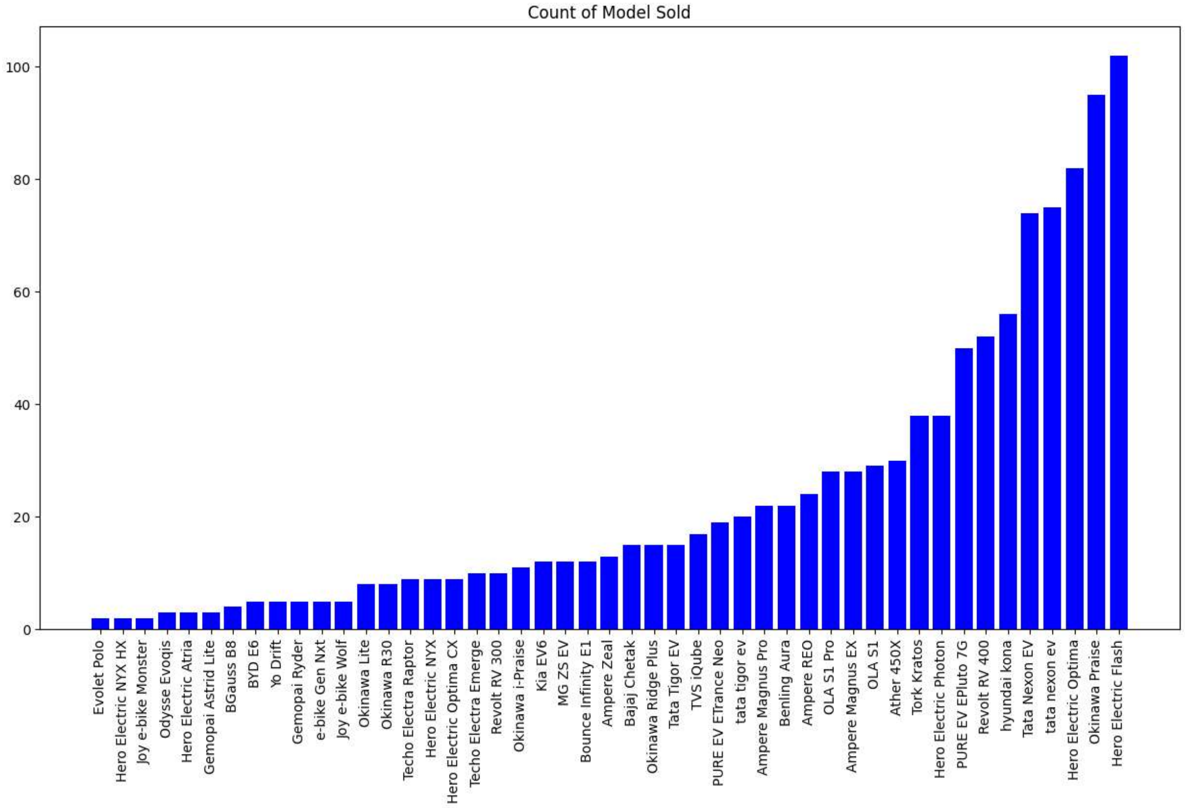
Out[7]:

	Model Name	Type	Rating	Count	NewType
0	Ampere Magnus EX	2-wheeler	3.964286	28	2
1	Ampere Magnus Pro	2-wheeler	3.090909	22	2
2	Ampere REO	2-wheeler	2.583333	24	2
3	Ampere Zeal	2-wheeler	2.846154	13	2
4	Ather 450X	2-wheeler	3.666667	30	2

EDA

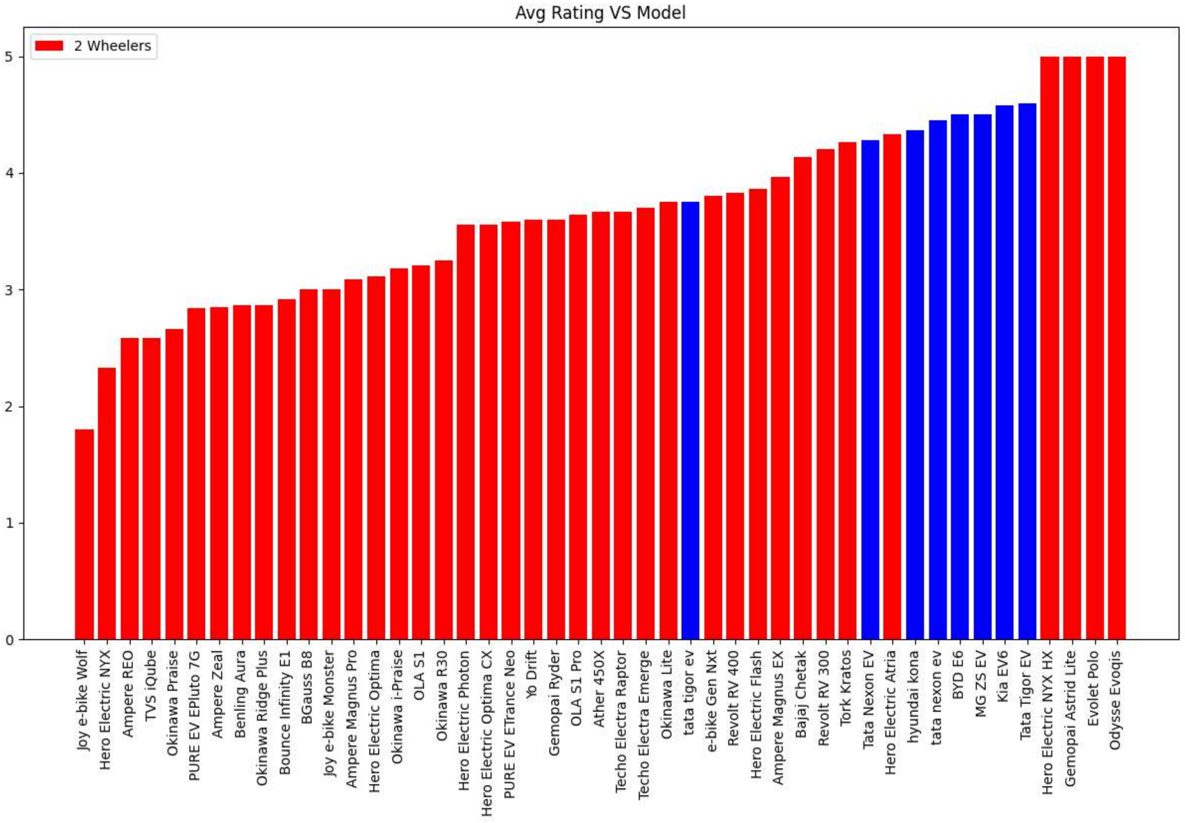
In [8]:

```
d = df.sort_values(by=['Count'])

plt.figure(figsize=(15, 8))
plt.title('Count of Model Sold')
plt.bar(d['Model Name'], d['Count'], color = 'blue')
plt.xticks(rotation=90)
plt.show()
```



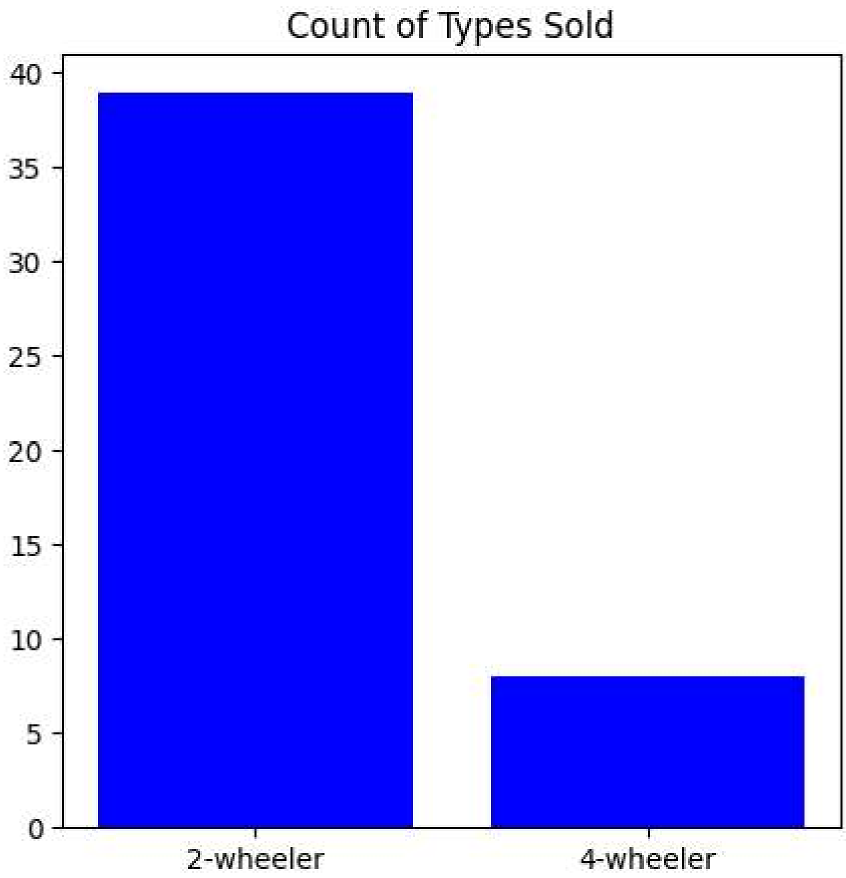
In [9]:

```
d = df.sort_values(by=['Rating'])
d['color'] = df['NewType'].apply(lambda x: 'red' if x == '2' else 'blue')

plt.figure(figsize=(15, 8))
plt.title('Avg Rating VS Model')
plt.bar(d['Model Name'], d['Rating'], color = d['color'])
plt.legend(['2 Wheelers', '4 Wheelers'], loc='best')
plt.xticks(rotation=90)
plt.show()
```



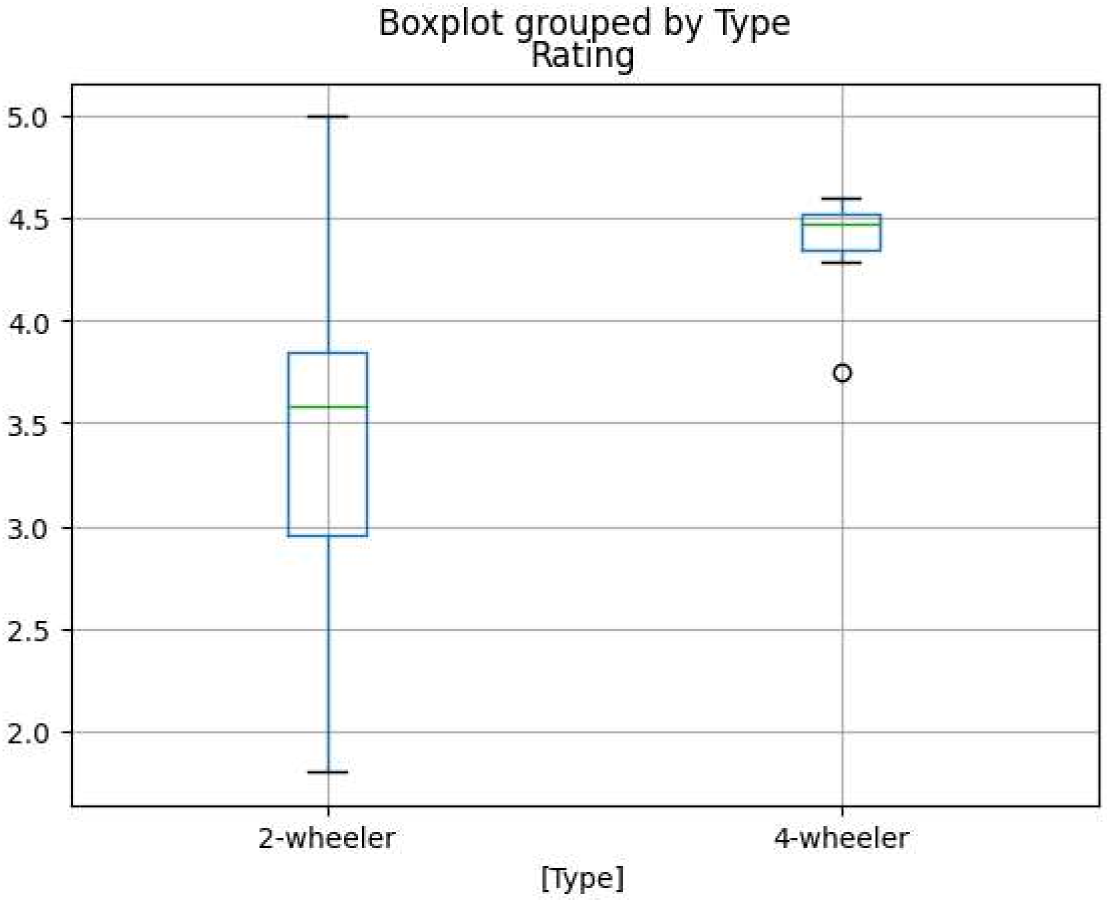
In [10]:

```
d = pd.DataFrame(df['Type'].value_counts()).reset_index()

plt.figure(figsize=(5, 5))
plt.title('Count of Types Sold')
plt.bar(d['index'], d['Type'], color = 'blue')
plt.show()
```

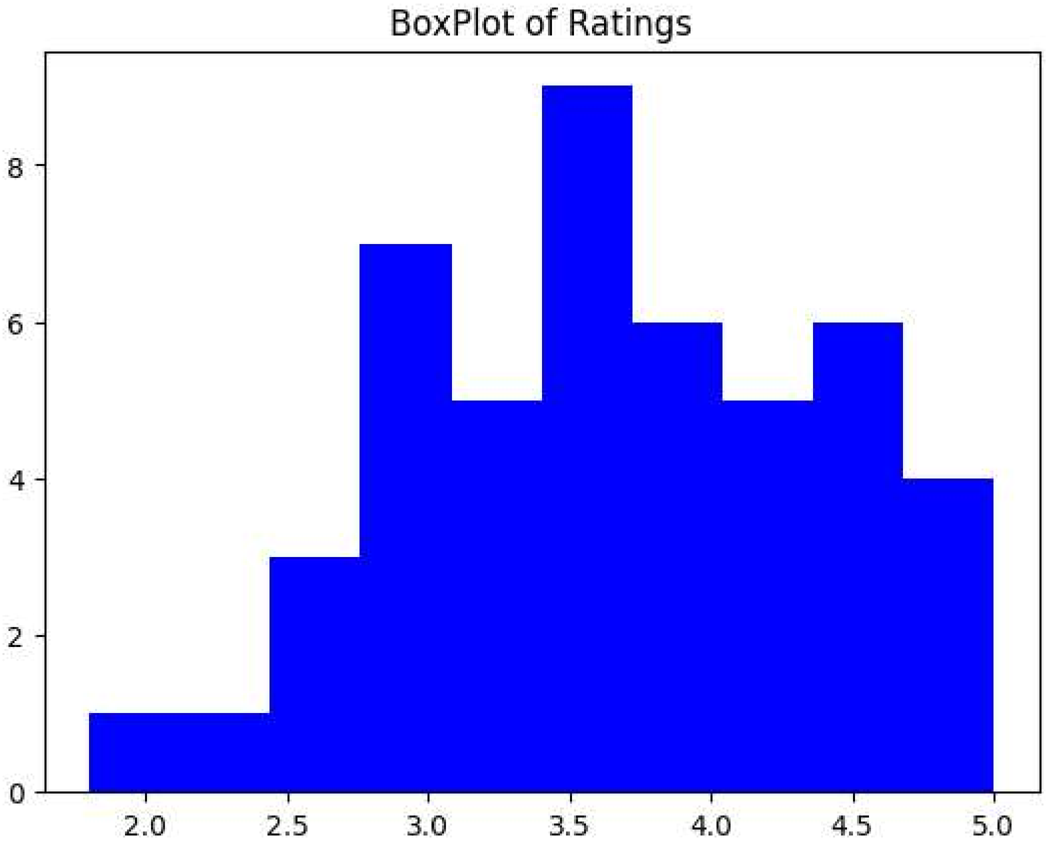


In [11]:

```
d = df.copy()
d.boxplot(column=['Rating'], by=['Type'])
plt.show()
```



In [12]:
```
plt.title('BoxPlot of Ratings')
plt.hist(x = df['Rating'], bins = 10, color = 'blue')
plt.show()
```



Resampling for Clustering

In [13]:

```
from sklearn.utils import resample

wheels_2 = df[df['NewType'] == '2']
wheels_4 = df[df['NewType'] == '4']

wheels_2_downsample = resample(wheels_2,
             replace=True,
             n_samples=len(wheels_4),
             random_state=np.random.randint(1, 101))

print(wheels_2_downsample.shape)
print(wheels_4.shape)
```

```
(8, 5)
(8, 5)
```

In [14]:

```
D = pd.concat([wheels_2_downsample, wheels_4], axis=0).reset_index()
D = D[list(D)[1:]]
# D.drop_duplicates(keep='first', inplace=True)
D
```

Out[14]:

	Model Name	Type	Rating	Count	NewType
0	Ampere Magnus EX	2-wheeler	3.964286	28	2
1	Hero Electric Optima CX	2-wheeler	3.555556	9	2
2	Ather 450X	2-wheeler	3.666667	30	2
3	Okinawa Ridge Plus	2-wheeler	2.866667	15	2
4	Ampere Magnus EX	2-wheeler	3.964286	28	2
5	Bajaj Chetak	2-wheeler	4.133333	15	2
6	Bajaj Chetak	2-wheeler	4.133333	15	2
7	Techo Electra Emerge	2-wheeler	3.7	10	2
8	BYD E6	4-wheeler	4.5	5	4
9	Kia EV6	4-wheeler	4.583333	12	4
10	MG ZS EV	4-wheeler	4.5	12	4
11	Tata Nexon EV	4-wheeler	4.283784	74	4
12	Tata Tigor EV	4-wheeler	4.6	15	4
13	hyundai kona	4-wheeler	4.366071	56	4
14	tata nexon ev	4-wheeler	4.453333	75	4
15	tata tigor ev	4-wheeler	3.75	20	4

Clustering

In [15]:

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

In [16]:

```
X = D[['NewType', 'Rating', 'Count']]
X.head()
```

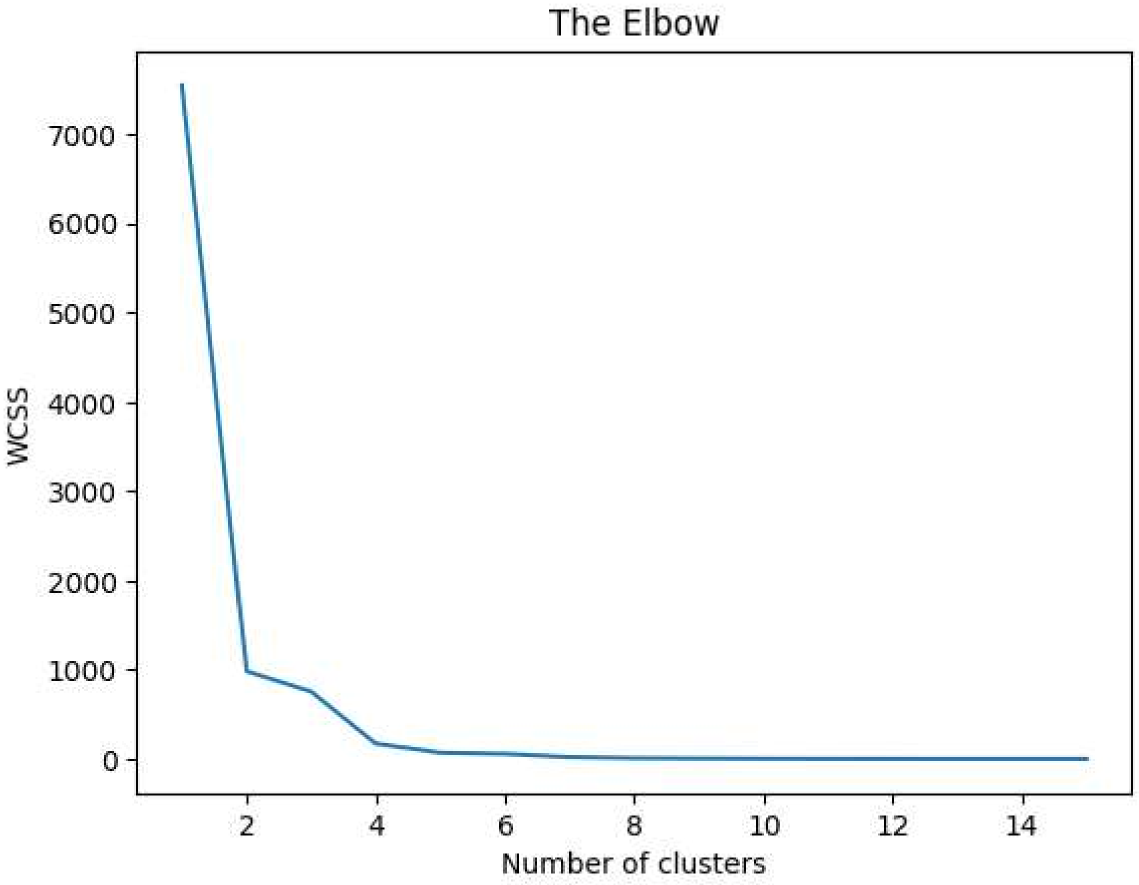
Out[16]:

	NewType	Rating	Count
0	2	3.964286	28
1	2	3.555556	9
2	2	3.666667	30
3	2	2.866667	15
4	2	3.964286	28

In [17]:

```
wcss=[]
r = range(1, 16)
for i in r:
  kmeans = KMeans(i, n_init=1)
  kmeans.fit(X)
  wcss_iter = kmeans.inertia_
  wcss.append(wcss_iter)

number_clusters = r
plt.plot(number_clusters,wcss)
plt.title('The Elbow')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
<ipython-input-17-ca10b049500c>:5: ConvergenceWarning: Number of distinct clusters
(14) found smaller than n_clusters (15). Possibly due to duplicate points in X.
  kmeans.fit(X)
```



In [18]:

```
colors = ['red', 'green', 'blue']
```

Made 2 Clusters

In [19]:

```
kmeans = KMeans(n_clusters=2, random_state=np.random.randint(1, 11), n_init="auto"
```

In [20]:

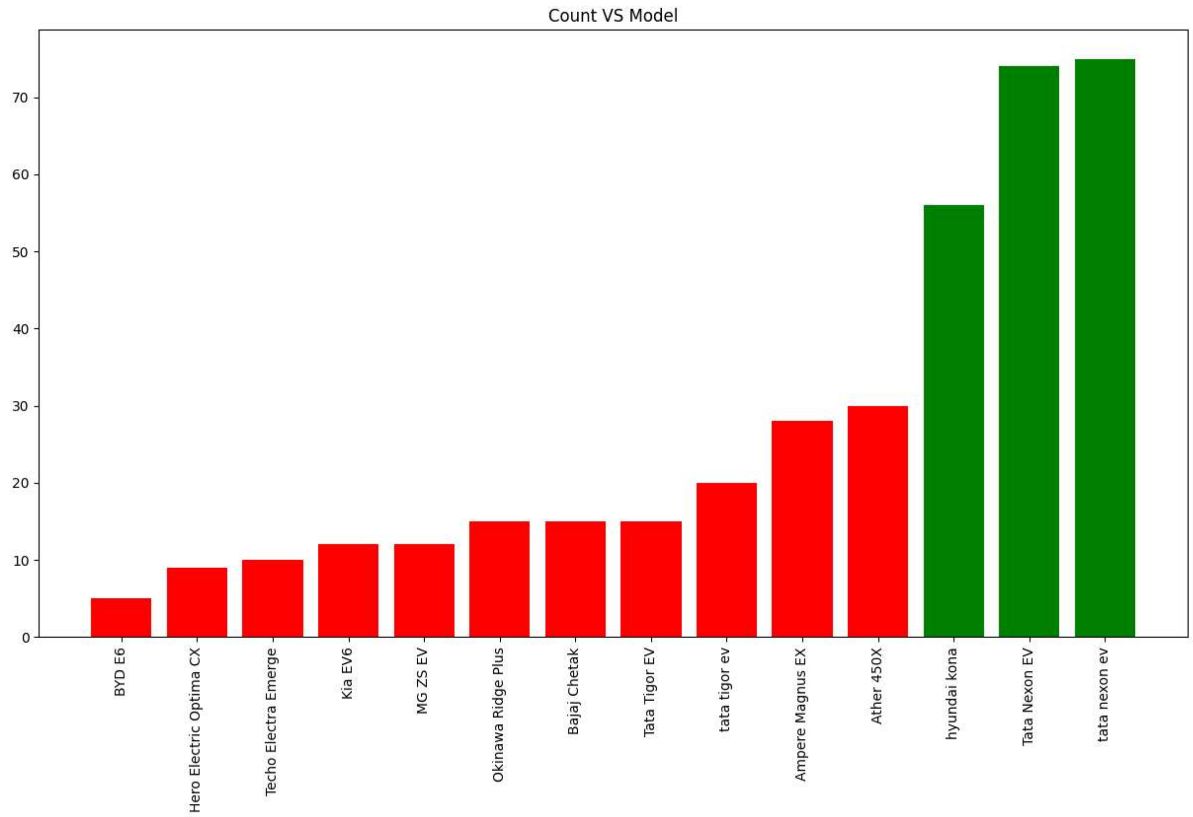
```
data_with_clusters = D.copy()
identified_clusters = kmeans.fit_predict(X)
data_with_clusters['Clusters'] = identified_clusters
```

In [21]:

```
data_with_clusters['color'] = data_with_clusters['Clusters'].apply(lambda x: color
```

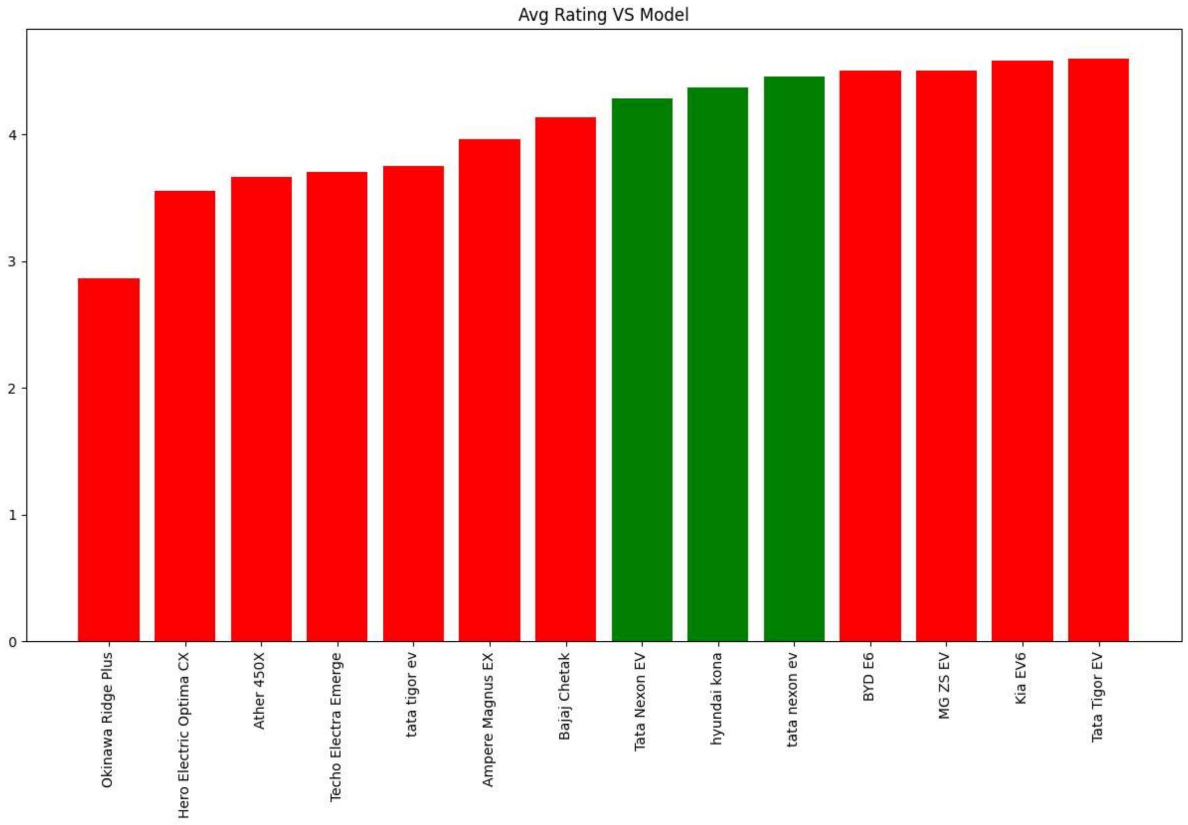
In [22]:

```
y = 'Count'

d = data_with_clusters.sort_values(by=[y])

plt.figure(figsize=(15, 8))
plt.title('Count VS Model')
plt.bar(d['Model Name'], d[y], color = d['color'])
plt.xticks(rotation=90)
plt.show()
```



In [23]:

```
y = 'Rating'

d = data_with_clusters.sort_values(by=[y])

plt.figure(figsize=(15, 8))
plt.title('Avg Rating VS Model')
plt.bar(d['Model Name'], d[y], color = d['color'])
plt.xticks(rotation=90)
plt.show()
```



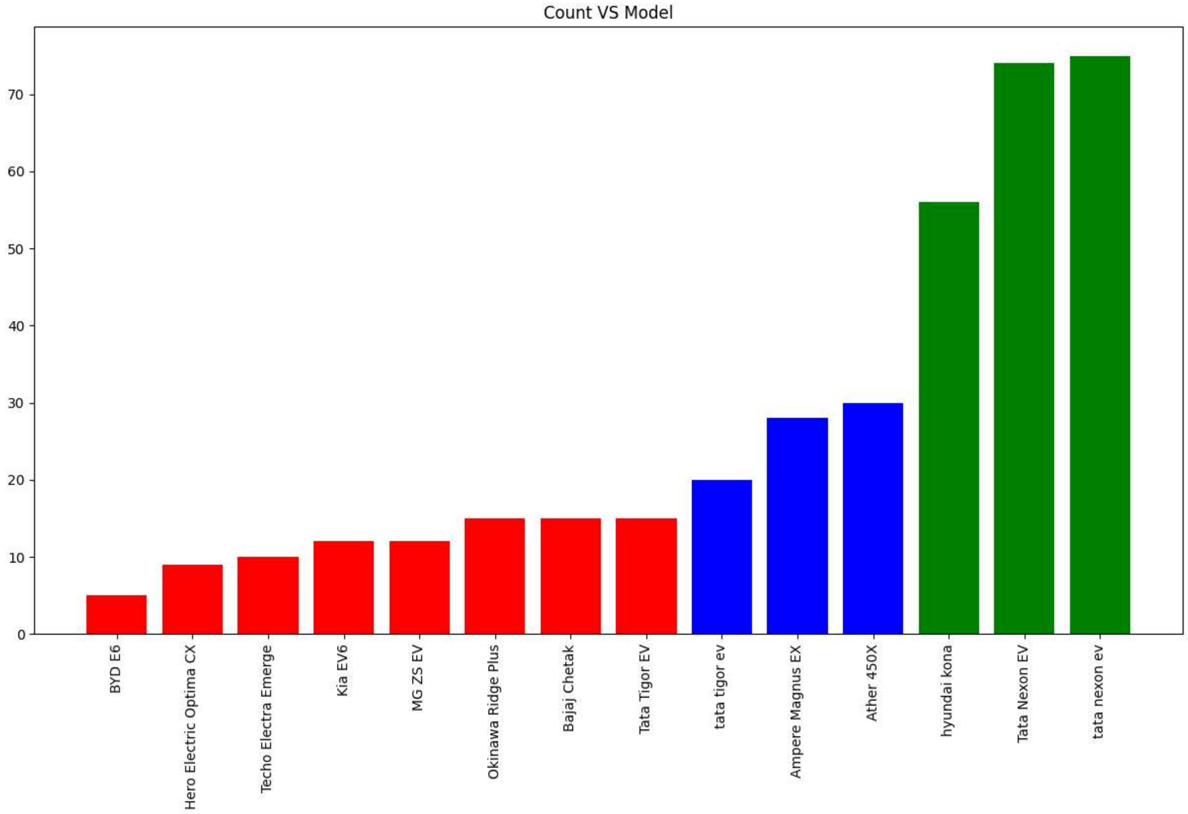
Made 3 Clusters

In [24]:
```
kmeans = KMeans(n_clusters=3, random_state=np.random.randint(1, 11), n_init="auto"
```

In [25]:
```
data_with_clusters = D.copy()
identified_clusters = kmeans.fit_predict(X)
data_with_clusters['Clusters'] = identified_clusters
```

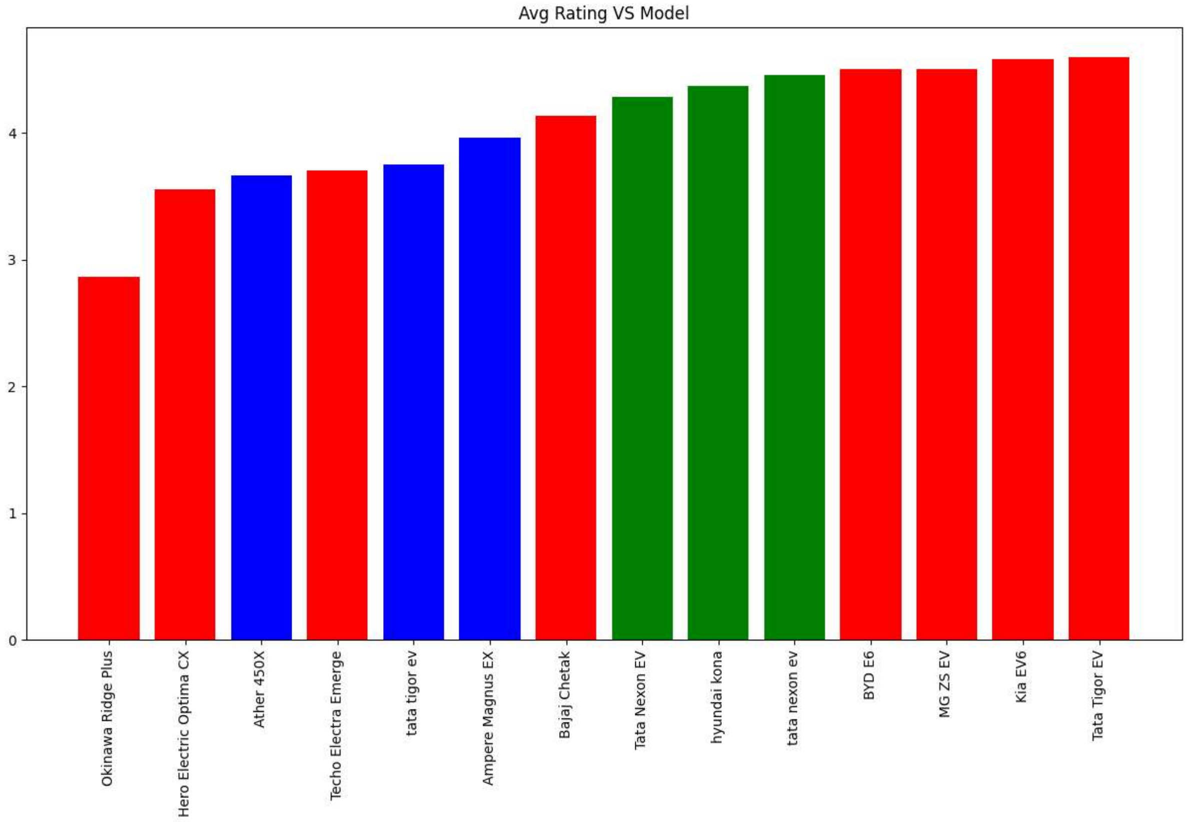
In [26]:
```
data_with_clusters['color'] = data_with_clusters['Clusters'].apply(lambda x: colors
```

In [27]:
```
y = 'Count'

d = data_with_clusters.sort_values(by=[y])

plt.figure(figsize=(15, 8))
plt.title('Count VS Model')
plt.bar(d['Model Name'], d[y], color = d['color'])
plt.xticks(rotation=90)
plt.show()
```



In [28]:

```
y = 'Rating'

d = data_with_clusters.sort_values(by=[y])

plt.figure(figsize=(15, 8))
plt.title('Avg Rating VS Model')
plt.bar(d['Model Name'], d[y], color = d['color'])
plt.xticks(rotation=90)
plt.show()
```



Conclusion: It works well with Count of Vehicle sold, not Rating