# EV MARKET ANALYSIS-Copy1

July 4, 2022

[1]:

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

## 0.1 Import the dataset

[2]:

```
#data = pd.read_csv('C:/Users/DELL/Desktop/Feynn Lab/EV details/archive/
 ↪EV_CARS_INDIA.csv')
df = pd.read_csv('C:\\Users\\DELL\\Desktop\\python project\\ev_cars.csv')
```

[3]:

```
df
```

[3]:

```
             Brand Name  Battery Capacity(kWh)   Acceleration(sec)  \
0    Audi RS e-tron GT                    93.4                 3.3
1       Audi e-tron GT                    93.4                 4.1
2          Audi e-tron                    95.0                 5.7
3         Tata Nexon EV                   30.2                 9.9
4         Tata Tigor EV                   26.0                 5.7
5  Hyudai Kona Electric                   39.2                 9.7
6         Jaguar I-Pace                   90.0                 4.8
7      Mahindra eVerito                   21.2                11.2
8              MG ZS EV                   44.5                 8.5

   TopSpeed(km/h)  Range(km)  Max Power(kW)   Max Torque(Nm) Transmission  \
0             250        480            500              830    Automatic
1             245        500            523              630    Automatic
2             200        484            300              664    Automatic
3             180        312             96              245    Automatic
4             120        306             55              170    Automatic
5             155        452            103              395    Automatic
6             200        470            294              696    Automatic
7              86        140             33               91    Automatic
8             120        340            107              353    Automatic

   No. of Seats  Charging T(h) No. of Airbags Drive Type   Price(Lakhs)
```

```
0             5              9            Yes        AWD            204
1             5              9            Yes        AWD            179
2             5              9            Yes        AWD            123
3             5              9            Yes        FWD             17
4             5              9            Yes        FWD             14
5             5              7            Yes        FWD             24
6             5             13            Yes        AWD            112
7             5             12            Yes        FWD             10
8             5              8            Yes        FWD             25
```

## 0.2 Exploratory data analysis

```
[4]: df.isna().sum()
```

```
[4]: Brand Name                0
     Battery Capacity(kWh)     0
     Acceleration(sec)         0
     TopSpeed(km/h)            0
     Range(km)                 0
     Max Power(kW)             0
     Max Torque(Nm)            0
     Transmission              0
     No. of Seats              0
     Charging T(h)             0
     No. of Airbags            0
     Drive Type                0
     Price(Lakhs)              0
     dtype: int64
```

```
[5]: df.columns
```

```
[5]: Index(['Brand Name', 'Battery Capacity(kWh)', 'Acceleration(sec)',
            'TopSpeed(km/h)', 'Range(km)', 'Max Power(kW)', 'Max Torque(Nm)',
            'Transmission', 'No. of Seats', 'Charging T(h)', 'No. of Airbags',
            'Drive Type', 'Price(Lakhs)'],
           dtype='object')
```

```
[6]: df.shape
```

```
[6]: (9, 13)
```

```
[7]: df.describe()
```

```
[7]:        Battery Capacity(kWh)  Acceleration(sec)  TopSpeed(km/h)  Range(km)  \
     count               9.000000           9.000000        9.000000   9.000000
     mean               59.211111           6.988889      172.888889 387.111111
     std                32.735472           2.870298       57.152088 121.111978
```

```
min                  21.200000            3.300000      86.000000  140.000000
25%                  30.200000            4.800000     120.000000  312.000000
50%                  44.500000            5.700000     180.000000  452.000000
75%                  93.400000            9.700000     200.000000  480.000000
max                  95.000000           11.200000     250.000000  500.000000

       Max Power(kW)  Max Torque(Nm)  No. of Seats  Charging T(h)  \
count       9.000000        9.000000           9.0       9.000000
mean      223.444444      452.666667           5.0       9.444444
std       189.078761      260.956893           0.0       1.878238
min        33.000000       91.000000           5.0       7.000000
25%        96.000000      245.000000           5.0       9.000000
50%       107.000000      395.000000           5.0       9.000000
75%       300.000000      664.000000           5.0       9.000000
max       523.000000      830.000000           5.0      13.000000

       Price(Lakhs)
count      9.000000
mean      78.666667
std       76.990259
min       10.000000
25%       17.000000
50%       25.000000
75%      123.000000
max      204.000000
```

[8]:

```python
df.groupby('Drive Type')['Brand Name'].agg('count').reset_index()
```

[8]:

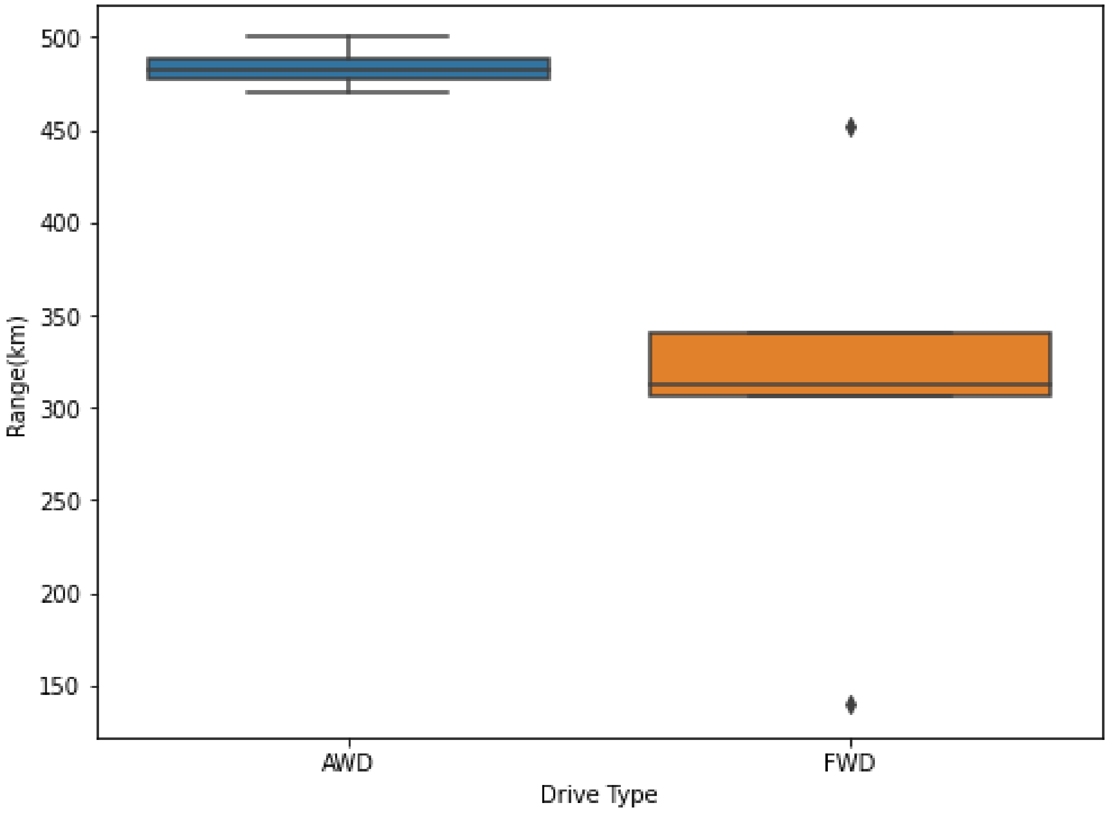
```
  Drive Type  Brand Name
0        AWD           4
1        FWD           5
```

[9]:

```python
fig=plt.figure(figsize=(8,6))
sns.boxplot(data=df,x='Drive Type',y="Range(km)")
```

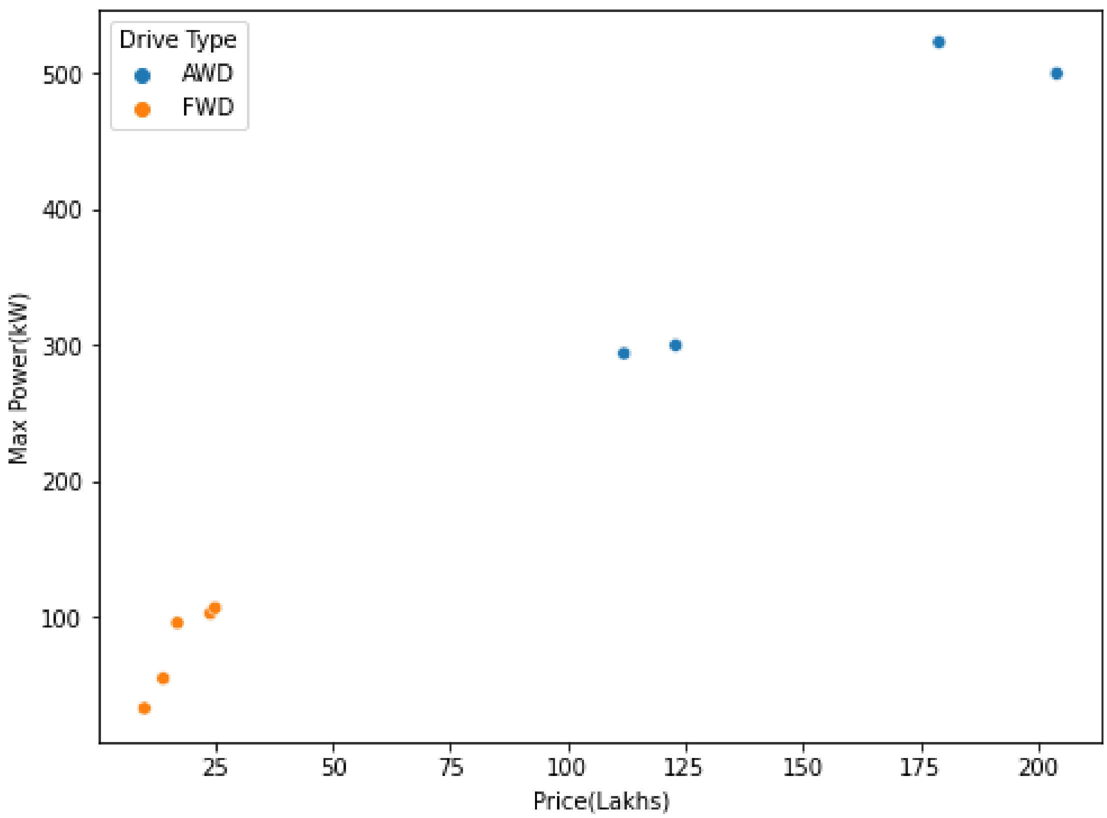
[9]:

```
<AxesSubplot:xlabel='Drive Type', ylabel='Range(km)'>
```



```
[10]: fig=plt.figure(figsize=(8,6))
      sns.scatterplot(data=df,hue='Drive Type',y='Max Power(kW)',x='Price(Lakhs)')
```

```
[10]: <AxesSubplot:xlabel='Price(Lakhs)', ylabel='Max Power(kW)'>
```



### 0.3 Converting all catagorical value's into integer

```
[11]: df['Transmission'] = pd.get_dummies(df['Transmission'])
      df['No. of Airbags'] = pd.get_dummies(df['No. of Airbags'])
```

```
[12]: df['Drive Type'] = df['Drive Type'].map({'AWD':0, 'FWD': 1})
```

```
[13]: df = df.drop(['Brand Name'], axis=1)
```

```
[14]: df.head()
```

```
[14]:    Battery Capacity(kWh)  Acceleration(sec)  TopSpeed(km/h)  Range(km)  \
      0                   93.4                3.3             250        480
      1                   93.4                4.1             245        500
      2                   95.0                5.7             200        484
      3                   30.2                9.9             180        312
      4                   26.0                5.7             120        306

         Max Power(kW)  Max Torque(Nm)  Transmission  No. of Seats  Charging T(h)  \
      0            500             830             1             5              9
```

```
1            523             630             1             5              9
2            300             664             1             5              9
3             96             245             1             5              9
4             55             170             1             5              9

   No. of Airbags  Drive Type  Price(Lakhs)
0               1           0           204
1               1           0           179
2               1           0           123
3               1           1            17
4               1           1            14
```

[15]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Battery Capacity(kWh)  9 non-null      float64
 1   Acceleration(sec)      9 non-null      float64
 2   TopSpeed(km/h)         9 non-null      int64
 3   Range(km)              9 non-null      int64
 4   Max Power(kW)          9 non-null      int64
 5   Max Torque(Nm)         9 non-null      int64
 6   Transmission           9 non-null      uint8
 7   No. of Seats           9 non-null      int64
 8   Charging T(h)          9 non-null      int64
 9   No. of Airbags         9 non-null      uint8
 10  Drive Type             9 non-null      int64
 11  Price(Lakhs)           9 non-null      int64
dtypes: float64(2), int64(8), uint8(2)
memory usage: 866.0 bytes
```

[16]:

```
df['Battery Capacity(kWh)'] = df['Battery Capacity(kWh)'].astype(int)
df['Acceleration(sec)'] = df['Acceleration(sec)'].astype(int)
```

[17]:

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

[18]:

```
X
```

[18]:

```
   Battery Capacity(kWh)  Acceleration(sec)  TopSpeed(km/h)  Range(km)  \
0                     93                  3             250        480
1                     93                  4             245        500
2                     95                  5             200        484
3                     30                  9             180        312
```

```
4                      26                   5             120         306
5                      39                   9             155         452
6                      90                   4             200         470
7                      21                  11              86         140
8                      44                   8             120         340

   Max Power(kW)  Max Torque(Nm)  Transmission  No. of Seats  Charging T(h)  \
0            500             830             1             5              9
1            523             630             1             5              9
2            300             664             1             5              9
3             96             245             1             5              9
4             55             170             1             5              9
5            103             395             1             5              7
6            294             696             1             5             13
7             33              91             1             5             12
8            107             353             1             5              8

   No. of Airbags  Drive Type
0               1           0
1               1           0
2               1           0
3               1           1
4               1           1
5               1           1
6               1           0
7               1           1
8               1           1
```

```
[19]: y
```

```
[19]: 0    204
      1    179
      2    123
      3     17
      4     14
      5     24
      6    112
      7     10
      8     25
      Name: Price(Lakhs), dtype: int64
```

Splitting the data set

```
[20]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2,␣
        ↪random_state=1)
```

##Feature Scaling

```
[21]: from sklearn.preprocessing import StandardScaler
      sc = StandardScaler()
      X_train = sc.fit_transform(X_train)
      X_test = sc.transform(X_test)
```

```
[22]: X_train, X_test
```

```
[22]: (array([[ 1.07640529, -0.83149718,  0.41320705,  0.73350688,  0.33248592,
                0.98160034,  0.        ,  0.        ,  1.72392288,  0.        ,
               -1.15470054],
              [-1.10806426,  1.56517116, -1.597398  , -1.95601834, -1.00551359,
               -1.30880045,  0.        ,  0.        ,  1.1992507 ,  0.        ,
                0.8660254 ],
              [ 1.17138222, -0.83149718,  1.20686694,  0.97800917,  1.50643952,
                0.73173844,  0.        ,  0.        , -0.37476584,  0.        ,
               -1.15470054],
              [ 1.17138222, -1.17387837,  1.29505137,  0.81500764,  1.38853152,
                1.48889572,  0.        ,  0.        , -0.37476584,  0.        ,
               -1.15470054],
              [-0.94976937, -0.48911599, -0.99774386, -0.60310565, -0.89273202,
               -1.00972333,  0.        ,  0.        , -0.37476584,  0.        ,
                0.8660254 ],
              [-0.82313345,  0.88040878,  0.06046932, -0.5542052 , -0.68254819,
               -0.72578934,  0.        ,  0.        , -0.37476584,  0.        ,
                0.8660254 ],
              [-0.53820264,  0.88040878, -0.38045284,  0.5868055 , -0.64666315,
               -0.15792138,  0.        ,  0.        , -1.42411021,  0.        ,
                0.8660254 ]]),
       array([[-0.37990775,  0.53802759, -0.99774386, -0.32600306, -0.62615741,
               -0.31692441,  0.        ,  0.        , -0.89943803,  0.        ,
                0.8660254 ],
              [ 1.23470018, -0.48911599,  0.41320705,  0.84760795,  0.36324453,
                0.86045517,  0.        ,  0.        , -0.37476584,  0.        ,
               -1.15470054]]))
```

```
[26]: from sklearn.linear_model import LogisticRegression
      log_classifier = LogisticRegression(random_state = 0)
      log_classifier.fit(X_train, y_train)
```

```
[26]: LogisticRegression(random_state=0)
```

```
[27]: y_pred = log_classifier.predict(X_test)
```

```
[28]: y_pred
```

```
[28]: array([ 24, 179], dtype=int64)
```

[ ]:

```
```