Classification Problem

Title: To predict value fom the dataset that is categorical in nature.

Multi-class Classification:

Multi-class Classification refers to predicting more than two classes/columns.

To predict Type of Cars from the datset.

Import Libraries

In [1]:
```
from warnings import filterwarnings
filterwarnings("ignore")
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

Collecting Data

Reading in the Cars93.csv file into a pandas dataframe.

In [2]:
```
A = pd.read_csv("C:/Users/aksha/OneDrive/Desktop/Cars93.csv")
```

In [3]:
```
A.head()
```

Out[3]:

	id	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags	...	Passengers	Length	Wheelbase	Width	Turn.circle	Rear.seat.room	Luggage.room	Weight	Origin	Make
0	1	Acura	Integra	Small	12.9	15.9	18.8	25	31	None	...	5	177	102	68	37	26.5	11.0	2705	non-USA	Acura Integra
1	2	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger	...	5	195	115	71	38	30.0	15.0	3560	non-USA	Acura Legend
2	3	Audi	90	Compact	25.9	29.1	32.3	20	26	Driver only	...	5	180	102	67	37	28.0	14.0	3375	non-USA	Audi 90
3	4	Audi	100	Midsize	30.8	37.7	44.6	19	26	NaN	...	6	193	106	70	37	31.0	17.0	3405	non-USA	Audi 100
4	5	BMW	535i	Midsize	23.7	30.0	36.2	22	30	Driver only	...	4	186	109	69	39	27.0	13.0	3640	non-USA	BMW 535i

5 rows × 28 columns

In [4]:
```
Q = []
import re
for i in A.columns:
    Q.append(re.sub("[.]","",i))

A.columns = Q
```

In [5]:
```
Q
```

Out[5]:
```
['id',
 'Manufacturer',
 'Model',
 'Type',
 'MinPrice',
 'Price',
 'MaxPrice',
 'MPGcity',
 'MPGhighway',
 'AirBags',
 'DriveTrain',
 'Cylinders',
 'EngineSize',
 'Horsepower',
 'RPM',
 'Revpermile',
 'Mantransavail',
 'Fueltankcapacity',
 'Passengers',
 'Length',
 'Wheelbase',
 'Width',
 'Turncircle',
 'Rearseatroom',
 'Luggageroom',
 'Weight',
 'Origin',
 'Make']
```

Missing Data Treatment

Cleaning the data by removing NaN values and unnecessary columns from the dataset.

In [6]:
```
A.isnull() #True -- If it is null
           #False -- If it is not null
```

Out[6]:

	id	Manufacturer	Model	Type	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	AirBags	...	Passengers	Length	Wheelbase	Width	Turncircle	Rearseatroom	Luggageroom	Weight	Origin	Make
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	True	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
88	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	True	False	False	False
89	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
90	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
91	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
92	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

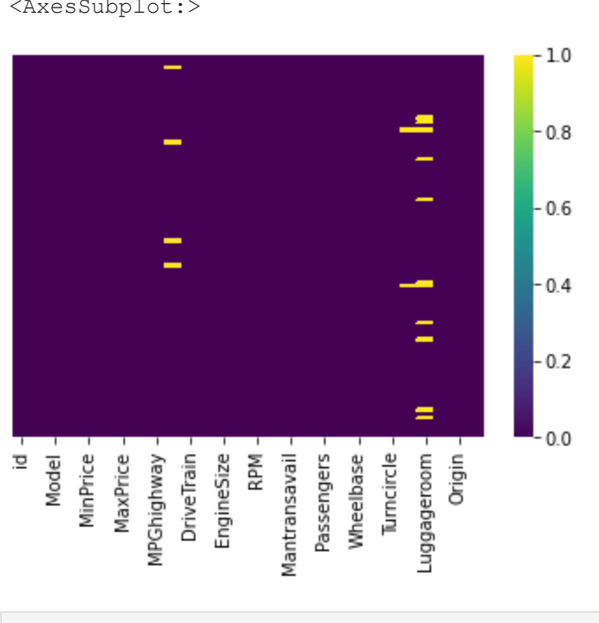
93 rows × 28 columns

In [7]:
```
A.isnull().sum() #airbags,rear.seat.room,luggage.room have missing values in them.
```

Out[7]:
```
id                   0
Manufacturer         0
Model                0
Type                 0
MinPrice             0
Price                0
MaxPrice             0
MPGcity              0
MPGhighway           0
AirBags              4
DriveTrain           0
Cylinders            0
EngineSize           0
Horsepower           0
RPM                  0
Revpermile           0
Mantransavail        0
Fueltankcapacity     0
Passengers           0
Length               0
Wheelbase            0
Width                0
Turncircle           0
Rearseatroom         2
Luggageroom         11
Weight               0
Origin               0
Make                 0
dtype: int64
```

In [8]:
```
#To analyze them visually, use seaborn to create a simple heatmap to see where is the data missing.
sb.heatmap(A.isnull(), yticklabels = False , cmap = "viridis")
```

Out[8]: <AxesSubplot:>



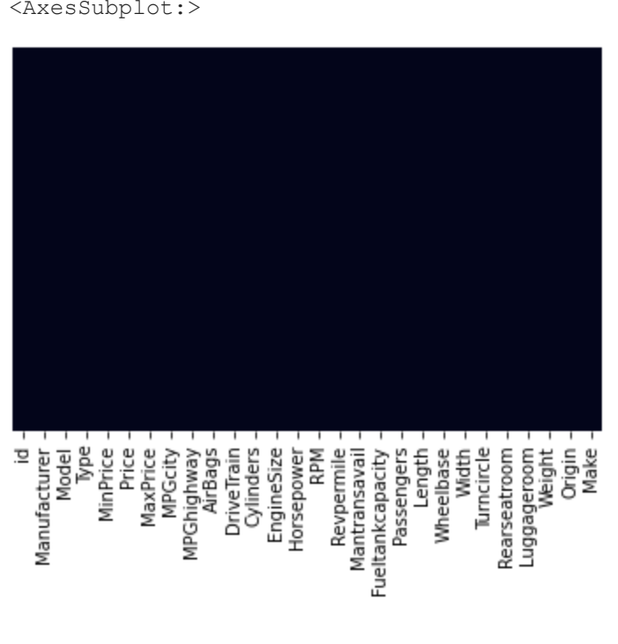
In [9]:
```
#to remove these columns
from PM8 import replacer
replacer(A)
```

In [10]:
```
A.isna().sum()
```

Out[10]:
```
id                  0
Manufacturer        0
Model               0
Type                0
MinPrice            0
Price               0
MaxPrice            0
MPGcity             0
MPGhighway          0
AirBags             0
DriveTrain          0
Cylinders           0
EngineSize          0
Horsepower          0
RPM                 0
Revpermile          0
Mantransavail       0
Fueltankcapacity    0
Passengers          0
Length              0
Wheelbase           0
Width               0
Turncircle          0
Rearseatroom        0
Luggageroom         0
Weight              0
Origin              0
Make                0
dtype: int64
```

In [11]:
```
#to check if the values have been removed or not.
sb.heatmap(A.isnull(), yticklabels=False , cbar = False)
```

Out[11]: <AxesSubplot:>



Dropping Unnecessary Colums

In [12]:
```
Y = A[["Type"]]
X = A.drop(labels=["Type","id","Make","Model","Manufacturer"],axis=1)
```

Categorical & Continuous

In [13]:
```
cat = []
con = []
for i in X.columns:
    if(X[i].dtypes == "object"):
        cat.append(i)
    else:
        con.append(i)
```

In [14]:
```
cat
```

Out[14]:
```
['AirBags', 'DriveTrain', 'Cylinders', 'Mantransavail', 'Origin']
```

In [15]:
```
con
```

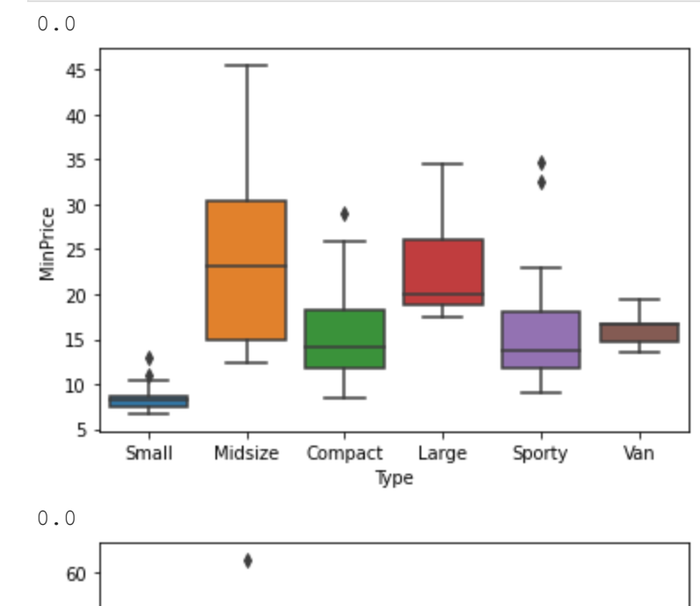
Out[15]:
```
['MinPrice',
 'Price',
 'MaxPrice',
 'MPGcity',
 'MPGhighway',
 'EngineSize',
 'Horsepower',
 'RPM',
 'Revpermile',
 'Fueltankcapacity',
 'Passengers',
 'Length',
 'Wheelbase',
 'Width',
 'Turncircle',
 'Rearseatroom',
 'Luggageroom',
 'Weight']
```

Analyzing data through Boxplot

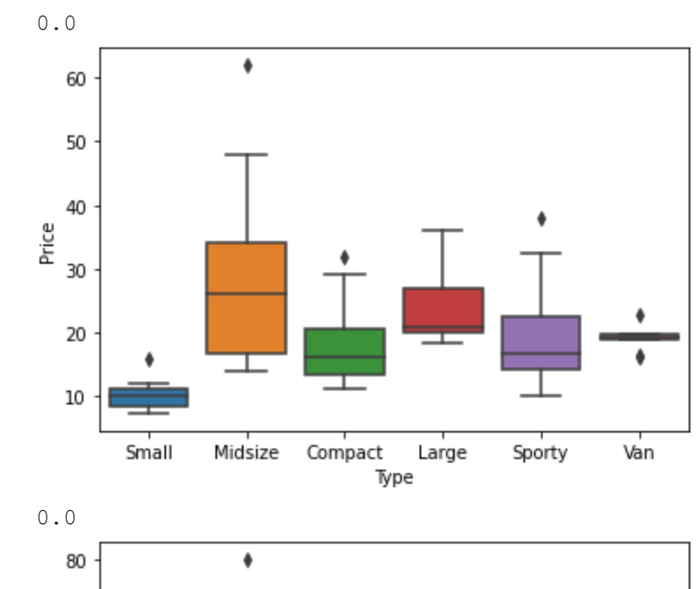
In [16]:
```
from PM8 import ANOVA
from seaborn import boxplot

for i in con:
    print(ANOVA(A,"Type",i))
    boxplot(Y.Type,X[i])
    plt.show()
```

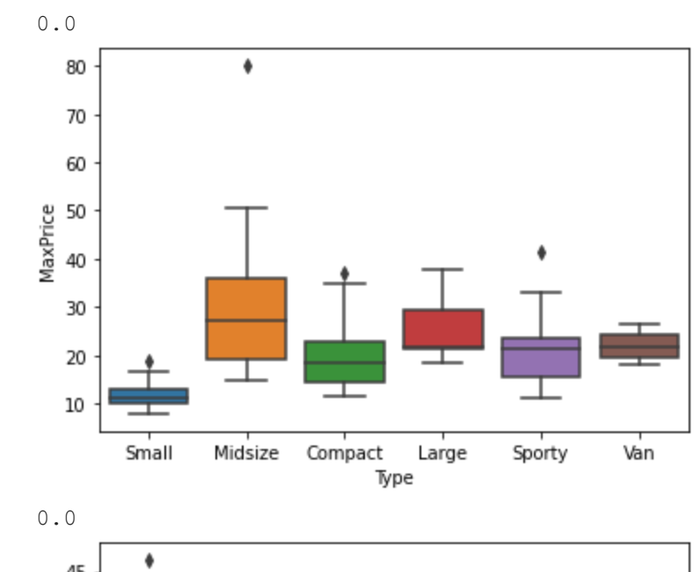
0.0



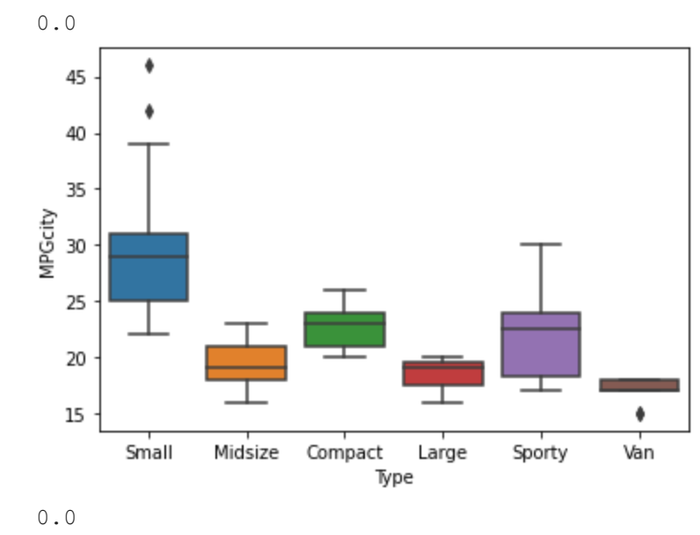
0.0



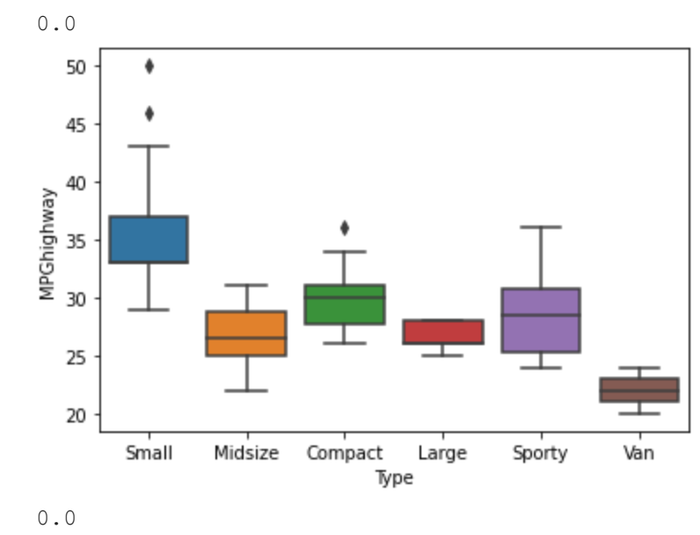
0.0



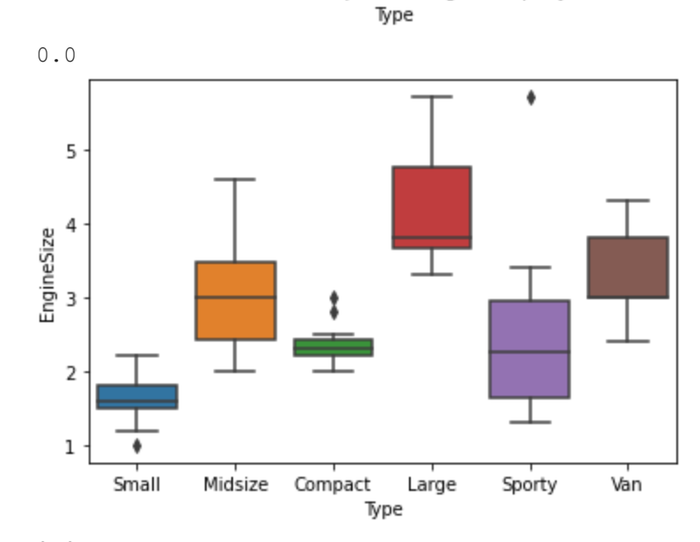
0.0



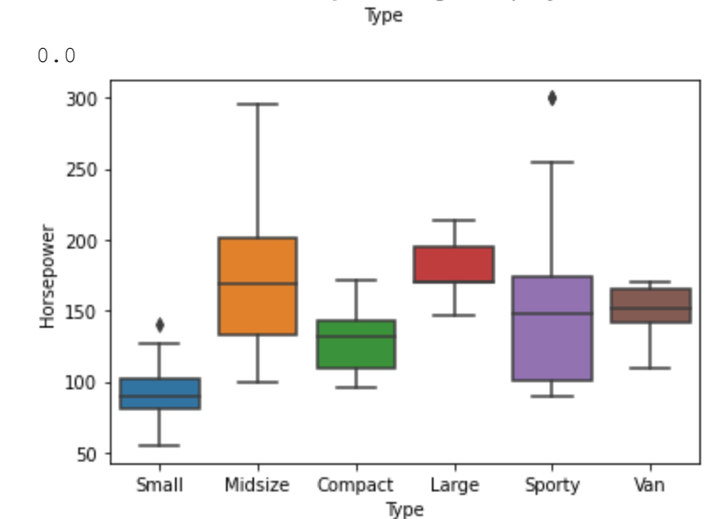
0.0



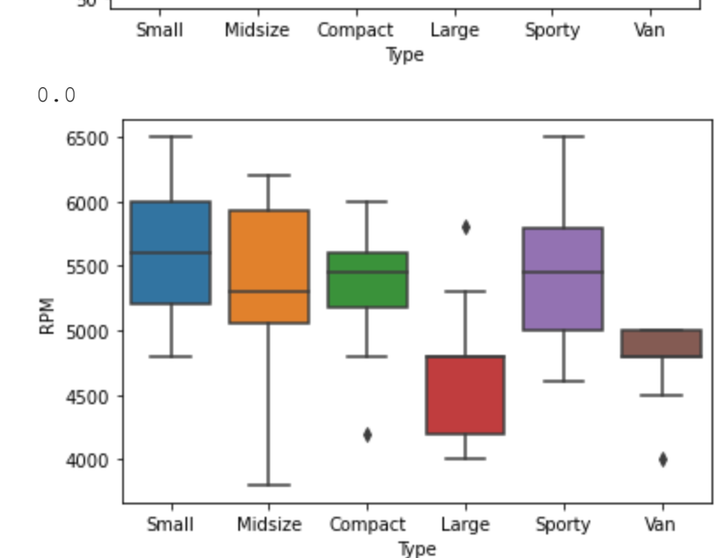
0.0



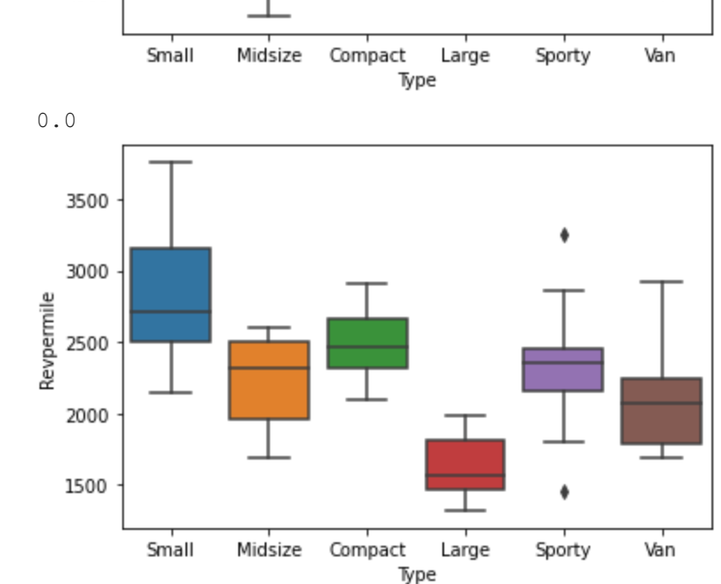
0.0



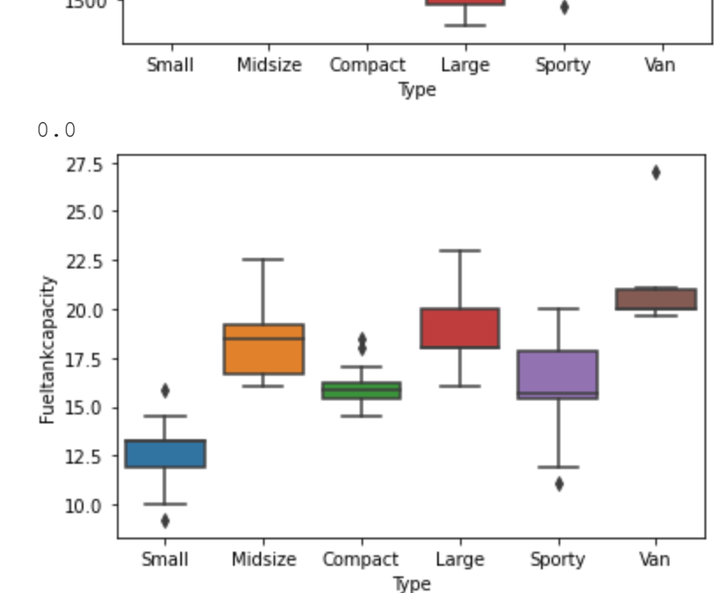
0.0



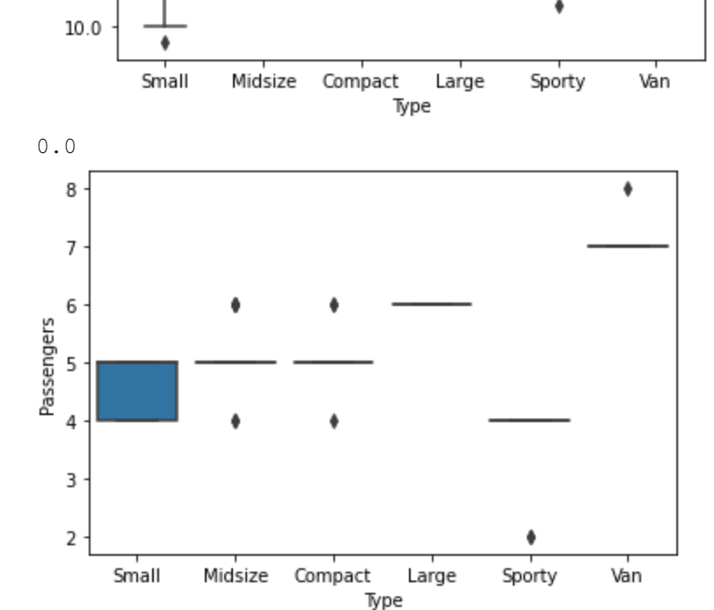
0.0



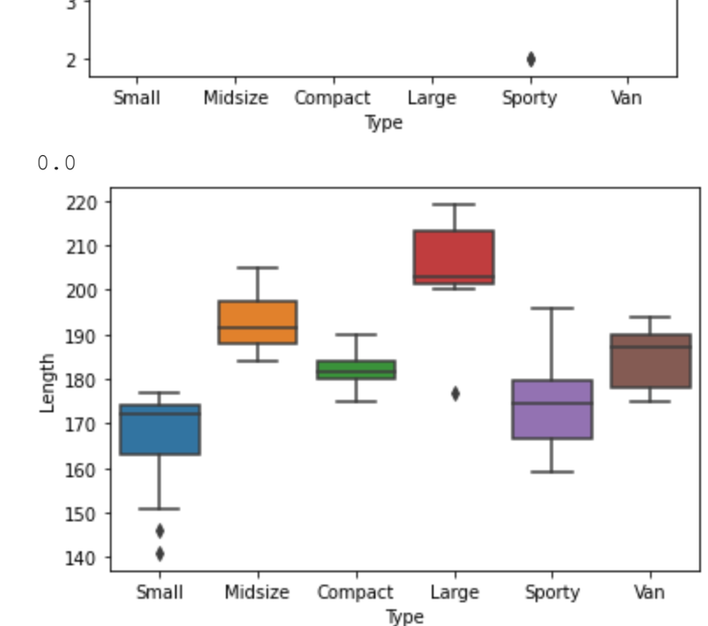
0.0



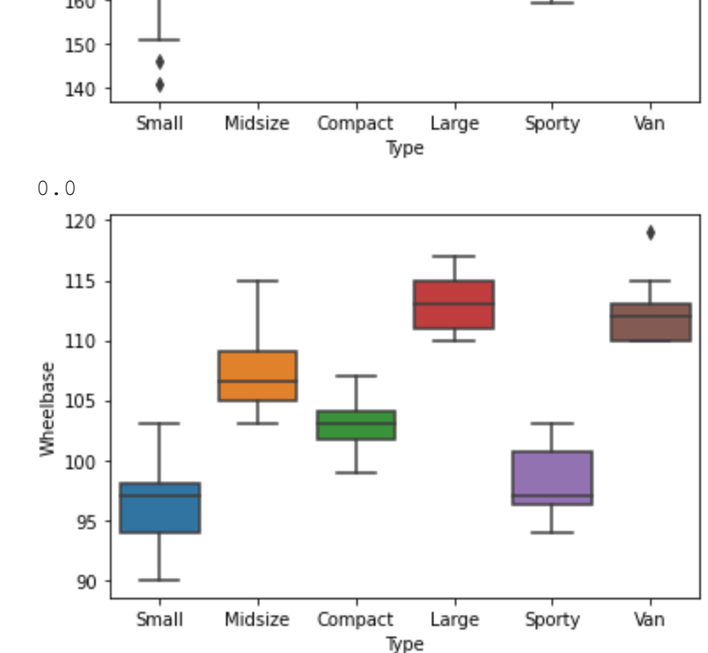
0.0



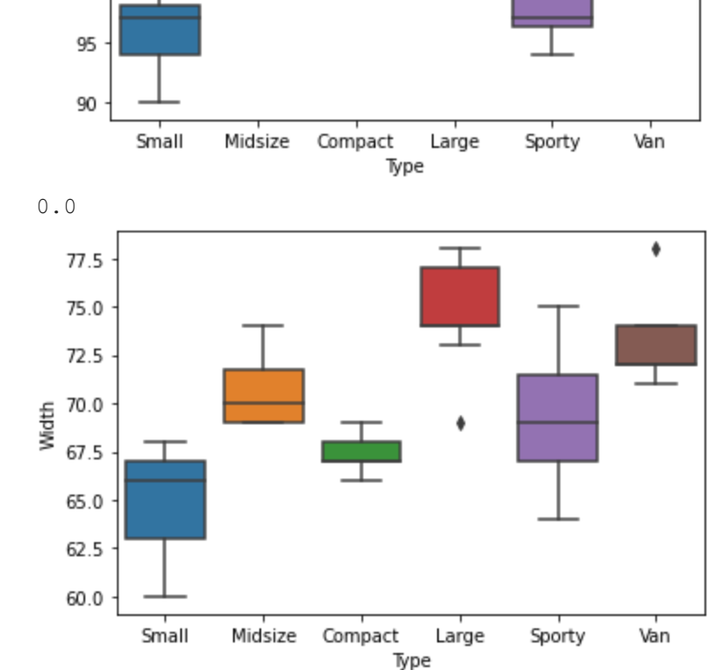
0.0



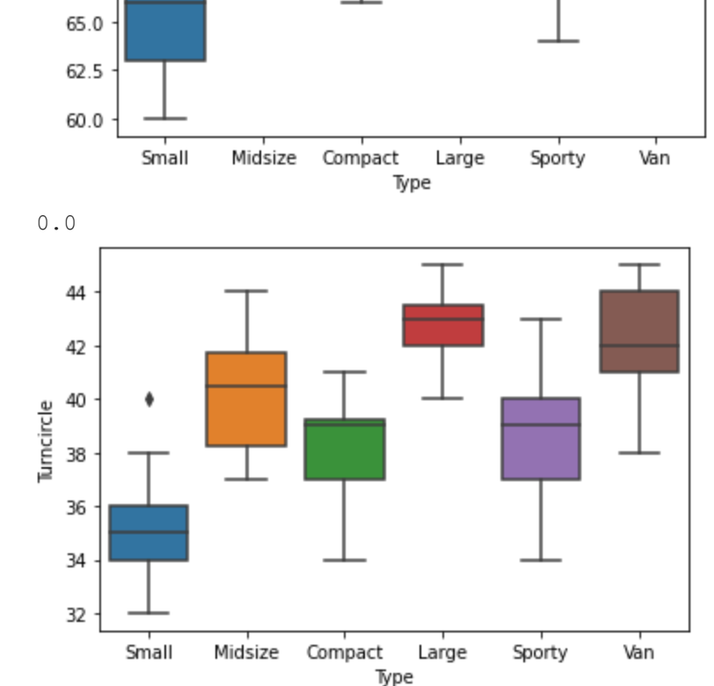
0.0



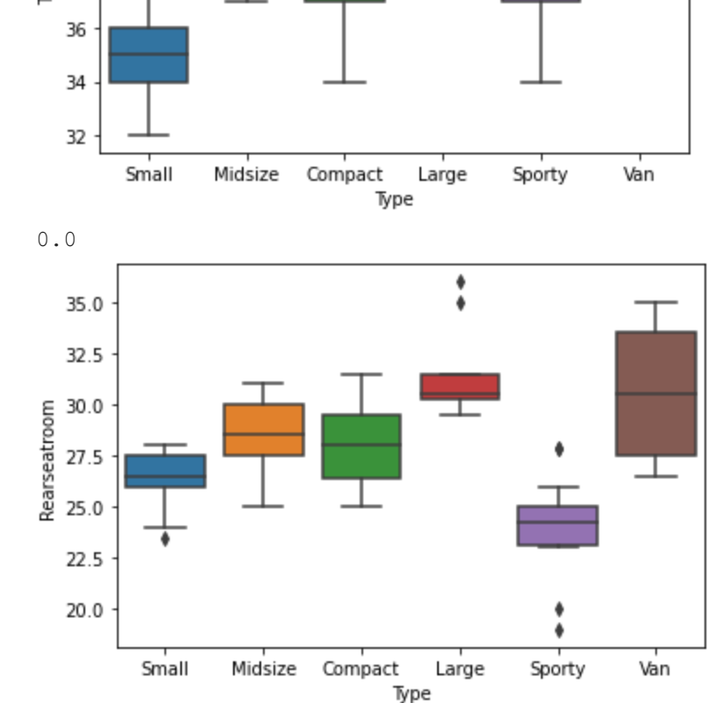
0.0



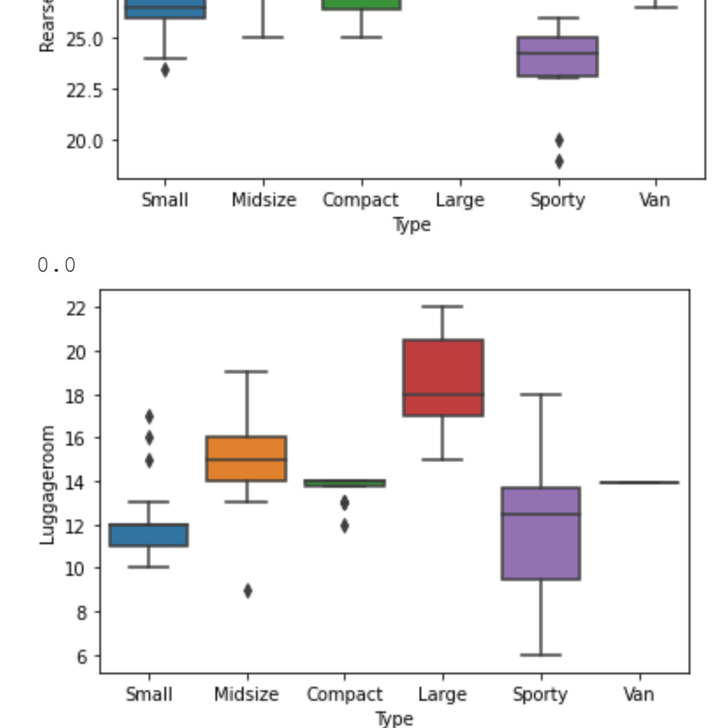
0.0



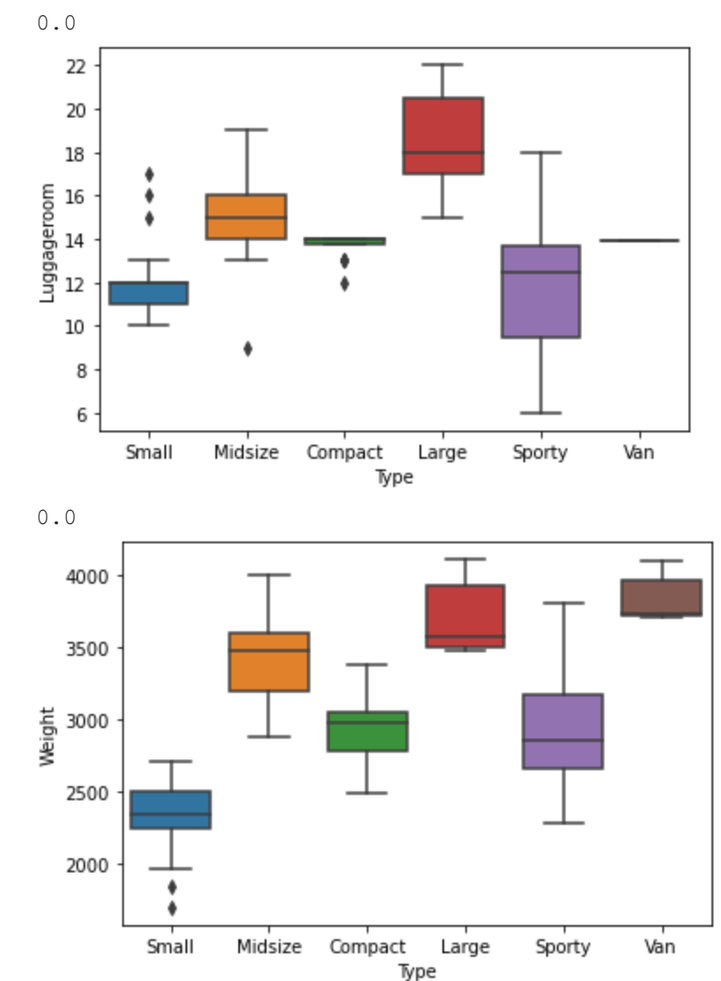
0.0



0.0



0.0



Cross Tabulation

To describe the relationship between two categorical variables, we use a special type of table called cross-tabulation

In [17]:
```
#cat columns
cat
```

Out[17]:
```
['AirBags', 'DriveTrain', 'Cylinders', 'Mantransavail', 'Origin']
```

In [18]:
```
pd.crosstab(Y.Type,A.AirBags)
```

Out[18]:

AirBags	Driver & Passenger	Driver only	None
Type			
Compact	2	9	5
Large	3	8	0
Midsize	6	12	4
Small	0	5	16
Sporty	3	9	2
Van	0	3	6

In [20]:
```
pd.crosstab(Y.Type,A.DriveTrain)
```

Out[20]:

DriveTrain	4WD	Front	Rear
Type			
Compact	1	13	2
Large	0	7	4
Midsize	0	17	5
Small	2	19	0
Sporty	2	7	5
Van	5	4	0

In [21]:
```
pd.crosstab(Y.Type,A.Cylinders)
```

Out[21]:

Cylinders	3	4	5	6	8	rotary
Type						
Compact	0	15	0	1	0	0
Large	0	0	0	7	4	0
Midsize	0	7	1	12	2	0
Small	3	18	0	0	0	0
Sporty	0	8	0	4	1	1
Van	0	1	1	7	0	0

In [22]:
```
pd.crosstab(Y.Type,A.Mantransavail)
```

Out[22]:

Mantransavail	No	Yes
Type		
Compact	2	14
Large	11	0
Midsize	13	9
Small	0	21
Sporty	0	14
Van	6	3

In [23]:
```
pd.crosstab(Y.Type,A.Origin)
```

Out[23]:

Origin	USA	non-USA
Type		
Compact	7	9
Large	11	0
Midsize	10	12
Small	7	14
Sporty	8	6
Van	5	4

In [24]:
```
#convert to dummy values
cat.remove("DriveTrain")
Xcat = pd.get_dummies(A[cat])
```

In [27]:
```
Xcat
```

Out[27]:

	AirBags_Driver & Passenger	AirBags_Driver only	AirBags_None	Cylinders_3	Cylinders_4	Cylinders_5	Cylinders_6	Cylinders_8	Cylinders_rotary	Mantransavail_No	Mantransavail_Yes	Origin_USA	Origin_non-USA
0	0	0	1	0	1	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	1	0	0	0	1	0	1
2	0	1	0	0	0	0	1	0	0	0	1	0	1
3	0	1	0	0	0	0	1	0	0	0	1	0	1
4	0	1	0	0	1	0	0	0	0	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
88	0	0	1	0	0	1	0	0	0	0	1	0	1
89	0	0	1	0	1	0	0	0	0	0	1	0	1
90	0	0	1	0	0	0	1	0	0	0	1	0	1
91	0	1	0	0	1	0	0	0	0	0	1	0	1
92	1	0	0	0	0	1	0	0	0	0	1	0	1

93 rows × 13 columns

Standardization of Continuous Variables

In [28]:
```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
B = pd.DataFrame(ss.fit_transform(A[con]),columns=con)
X = B.join(Xcat)
```

In [29]:
```
X.head()
```

Out[29]:

	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	EngineSize	Horsepower	RPM	Revpermile	Fueltankcapacity	...	Cylinders_3	Cylinders_4	Cylinders_5	Cylinders_6	Cylinders_8	Cylinders_rotary	Mantransavail_No	Mantransavail_Yes	Origin_USA	Origin_non-USA
0	-0.485787	-0.375720	-0.282465	0.471312	0.360925	-0.841022	-0.073484	1.717489	1.129530	-1.062184	...	0	1	0	0	0	0	0	1	0	1
1	1.388017	1.497844	1.531409	-0.781032	-0.770514	0.515869	1.078322	0.369586	0.005661	0.409445	...	0	0	0	1	0	0	0	1	0	1
2	1.008658	0.998227	0.948052	-0.423219	-0.581941	0.128186	0.540813	0.369586	-0.105713	0.072197	...	0	0	0	1	0	0	0	1	0	1
3	1.571949	1.893374	2.069191	-0.602126	-0.581941	0.128186	0.540813	0.369586	0.410659	1.359872	...	0	0	0	1	0	0	0	1	0	1
4	0.755752	1.091905	1.303535	-0.065407	0.172352	0.806631	1.231897	0.706562	0.430909	1.359872	...	0	1	0	0	0	0	0	1	0	1

5 rows × 31 columns

Building a Logistic Regression Model

Train Test Split

Splitting the data into training set and testing set.

In [30]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=0.2,random_state=21)
```

Logistic Regression

Logistic Regression is used as it solves classification problems & is used to predict categorical variable.

In [31]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
model = lr.fit(xtrain,ytrain)
pred = model.predict(xtest)
```

Accuracy Score

In [32]:
```
#confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(ytest,pred)
```

Out[32]:
```
array([[1, 0, 0, 0, 0, 0],
       [0, 3, 0, 0, 0, 0],
       [0, 0, 4, 0, 0, 0],
       [1, 0, 0, 3, 0, 0],
       [0, 0, 0, 0, 5, 0],
       [0, 0, 0, 0, 0, 2]], dtype=int64)
```

In [33]:
```
(1+3+4+3+5+2)/(1+3+4+1+3+5+2)
```

Out[33]:
```
0.9473684210526315
```

In [34]:
```
#accuracy score
from sklearn.metrics import accuracy_score
accuracy_score(ytest,pred)
```

Out[34]:
```
0.9473684210526315
```