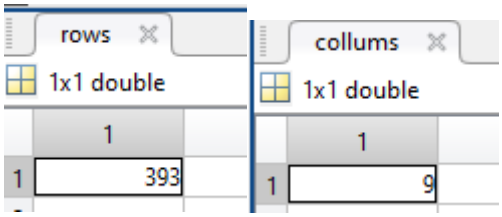
Assessment 1 Big Data

Task 1

```
%opens dataset
fid = fopen('car_data.txt','r');
Data =[];
while 1==1
    %gets line from file
    car = fgetl(fid);
    if car == -1
        break;
    end
    %converts string to array
    a = strsplit(car,'\t');
    Data = [Data;a];
end
i = height(Data);
for j = height(Data):-1:1
    for s = 1:9
        %remove all of the NA values
        Na = strcmp(Data(j,s),'NA');
        if Na == 1
            Data(j,:) = [];
        end
    end
end

%this gets the size of data that has been imported
[rows,collums] = size(Data);
```



The number of columns has stayed the same this is due to me keep the car names inside of the array. I do this as I later will later remover the strings from the array. The number of rows has lowered as I have taken out all data cells that contained a NaN . The code works by first getting the dataset than making an array for the data set. The dataset is than taken line by line and converts the string to array. Next the code will look for a NA value if it =1 than the data is removed. Finally, it outputs the rows and columns.

Task 2

Any row which contained a NA value was removed from the array making it no longer part of the dataset, so it isn't been used. These values wont cause problems as the whole row is removed not just the NA value. I could have converted the NA values to an integer such as 0. However, I believe it

is much easier and faster to do it this way around although some incomplete data is lost in the process.

Task 3

```
%sets which data to be included
q = 1:1:rows;
p = 1:1:8;
%output only double vaules. removes strings
stringA = string(Data);
Dcar = double(stringA(q,p));
Dcar(1,:)= [];
%temporary vaules declared
mpgmeantemporary = 0;
accelerationtemporary = 0;
horsepowertemporary = 0;
weighttemporaryorary = 0;
%individual arrays
mpgarray = Dcar(:,1);
accelerationarray = Dcar(:,6);
horsepowerarray = Dcar(:,4);
weightarray = Dcar(:,5);
%sorted arrays
mpgarray1 = insertionSort(mpgarray);
accelerationarray1 = insertionSort(accelerationarray);
horsearray1 = insertionSort(horsepowerarray);
weightarray1 = insertionSort(weightarray);
%counter
for ctr = 1:392
    mpgmeantemporary = mpgmeantemporary + mpgarray1(ctr);
    accelerationtemporary = accelerationtemporary + accelerationarray1(ctr);
    horsepowertemporary = horsepowertemporary + horsearray1(ctr);
    weighttemporaryorary = weighttemporaryorary + weightarray1(ctr);
end
```

```
%calculate mean
mpgmean = mpgmeantemporary/ctr(1,1);
accelerationmean = accelerationtemporary/ctr(1,1);
horsemean = horsepowertemporary/ctr(1,1);
weightmean = weighttemporaryorary/ctr(1,1);
%median
mpgmedian = median(mpgarray1);
accelmedian = median(accelerationarray1);
horsemedian = median(horsearray1);
weightmedian = uint16(median(weightarray1));
%minimum
mpgmin = mpgarray1(1);
accelmin = accelerationarray1(1);
horsemin = horsearray1(1);
weightmin = weightarray1(1);
%maximum
mpgmax = mpgarray1(392);
accelmax = accelerationarray1(392);
horsemax = horsearray1(392);
weightmax = weightarray1(392);
%standard deviation
mpgstdeviation = stdev(mpgarray1);
acceldeviation = stdev(accelerationarray1);
horsedeviation = stdev(horsearray1);
weightdeviation = stdev(weightarray1);
```

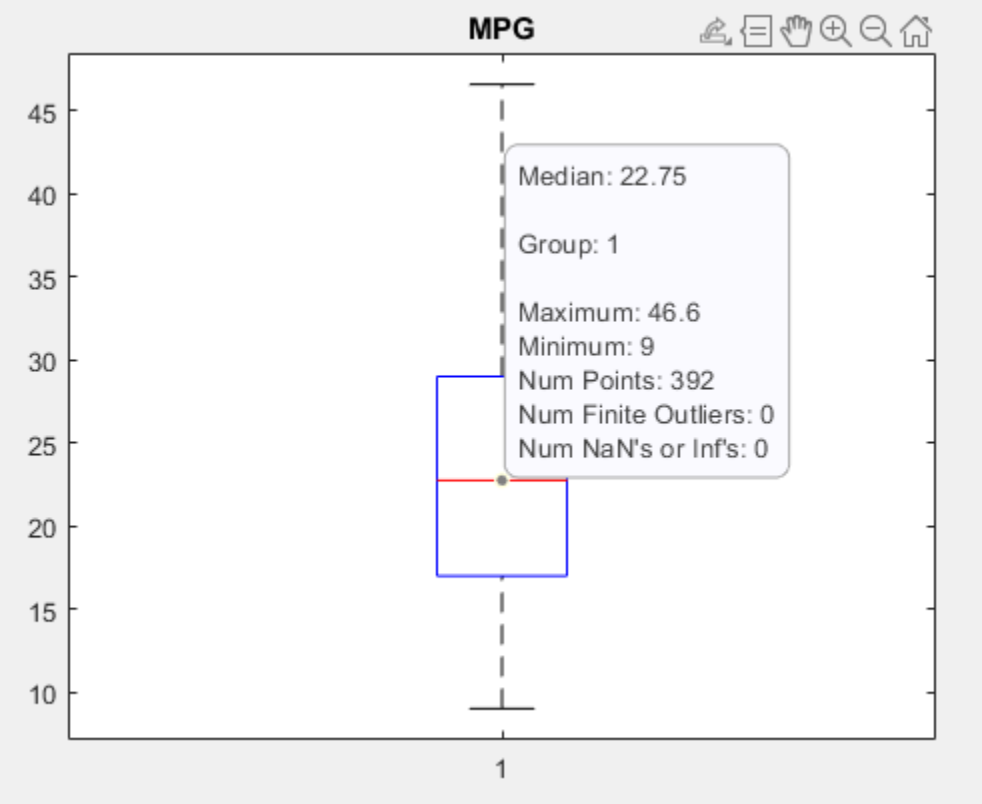
```
%insertionSort algo for sorting
function x = insertionSort(x)
for is=2:length(x)
    b = is;
    while (b > 1 && x(b-1) > x(b))
        [x(b),x(b-1)] = deal(x(b-1),x(b));
        b = b - 1;
    end
end
end
%median function
function med = Median(x)
    n = length(x);
    t = (n+1)/2;
    med = (x(floor(t))+x(ceil(t)))/2;
end
%standard deviation function
function stdeviation = stdev(x)
    AvgX = sumsfunction(x)/length(x);
    stdeviation = sqrt(sumsfunction((x-AvgX).^2)/(length(x)-1));
end
%sum function
function sums = sumsfunction(x)
         sums =0;
         for i = 1 : length(x)
             sums = sums + x(i);
         end
end
```

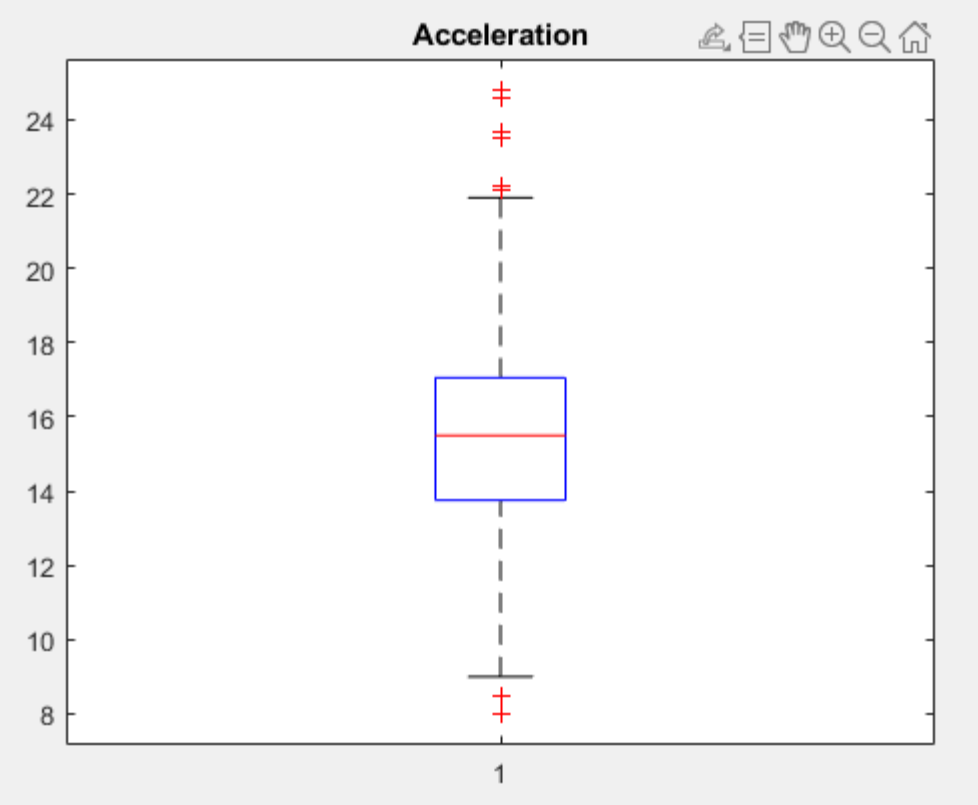
```
%table being made for variables
mean = [mpgmean;accelerationmean;horsemean;weightmean];
median = [mpgmedian;accelmedian;horsemedian;weightmedian];
min = [mpgmin;accelmin;horsemin;weightmin];
max = [mpgmax;accelmax;horsemax;weightmax];
standarddeviation = [mpgstdeviation;acceldeviation;horsedeviation;weightdeviation];
%table variables
variables = ["MPG";"Acceleration";"Horsepower";"Weight"];
%table heading
vars = table(variables,mean,median,min,max,standarddeviation);
disp(vars)
```

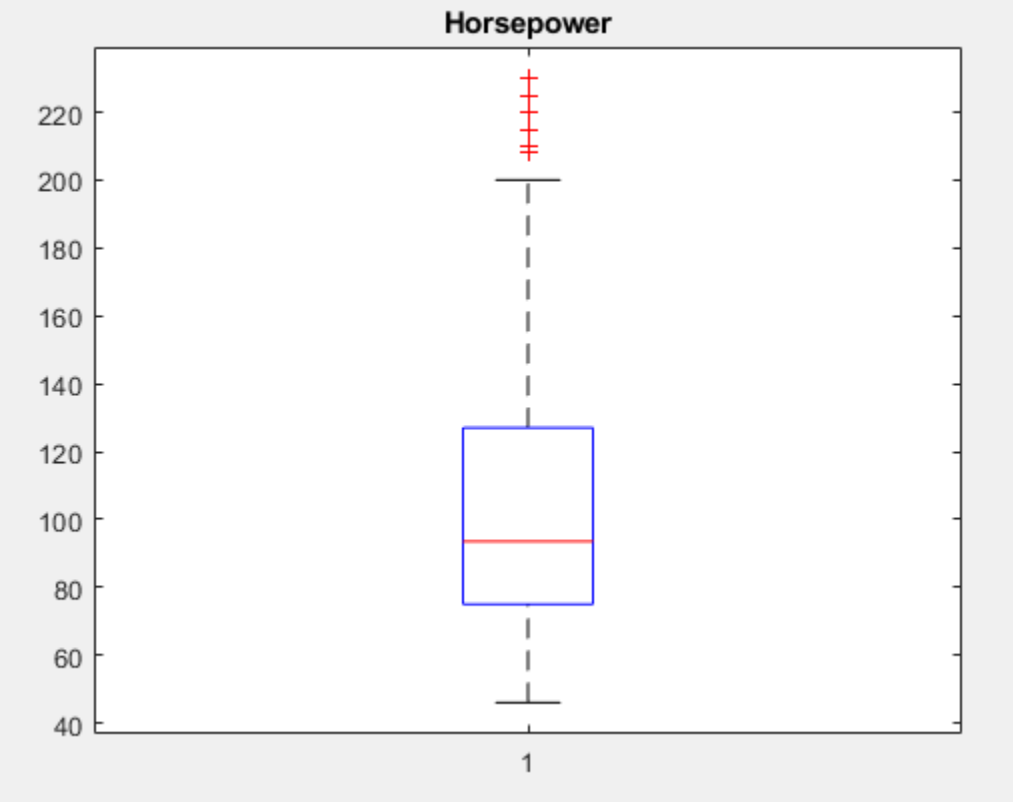
variables	mean	medianval	min	max	standarddeviation
"MPG"	23.446	23	9	46.6	7.805
"Acceleration"	15.541	16	8	24.8	2.7589
"Horsepower"	104.47	94	46	230	38.491
"Weight"	2977.6	2804	1613	5140	849.4

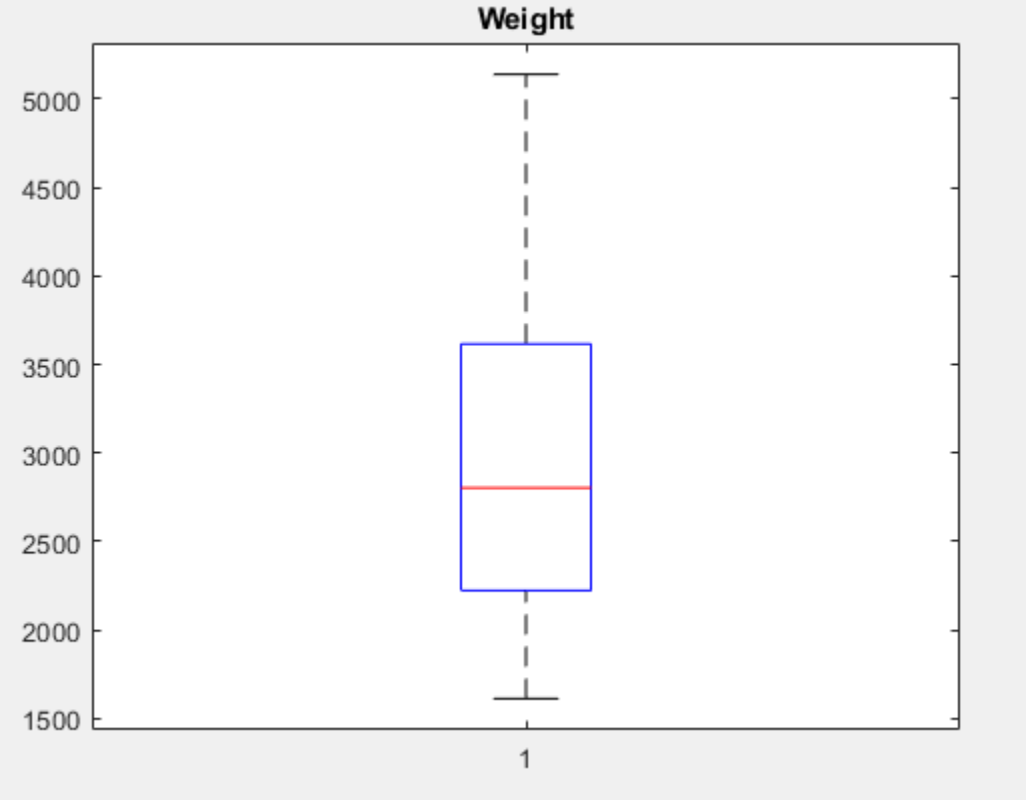
The table above shows the output that my code has when run. It will calculate the following variables using the dataset given mean ,median ,minimum , maximum and standard deviation for MPG , Acceleration ,Horsepower and Weight

Task 4

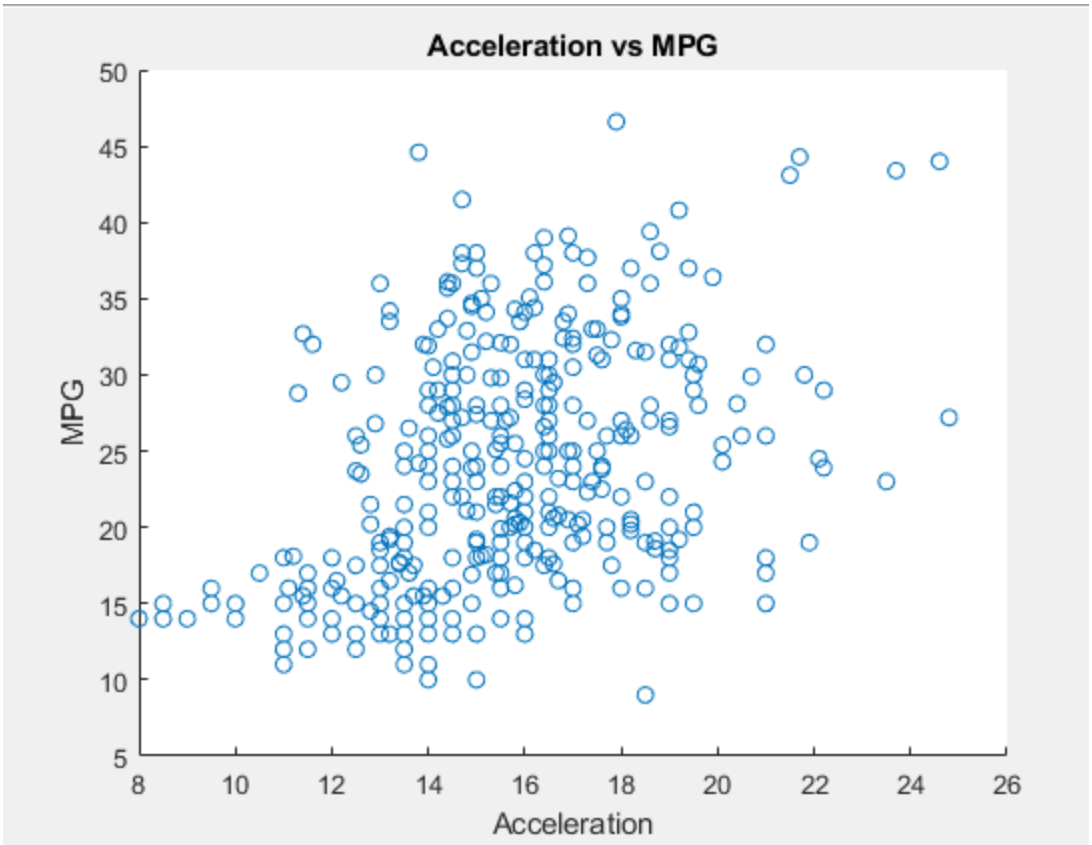

MPG
Median: 22.75
Group: 1
Maximum: 46.6
Minimum: 9
Num Points: 392
Num Finite Outliers: 0
Num NaN's or Inf's: 0
45
40
35
30
25
20
15
10
1


Acceleration
24
22
20
18
16
14
12
10
8
1

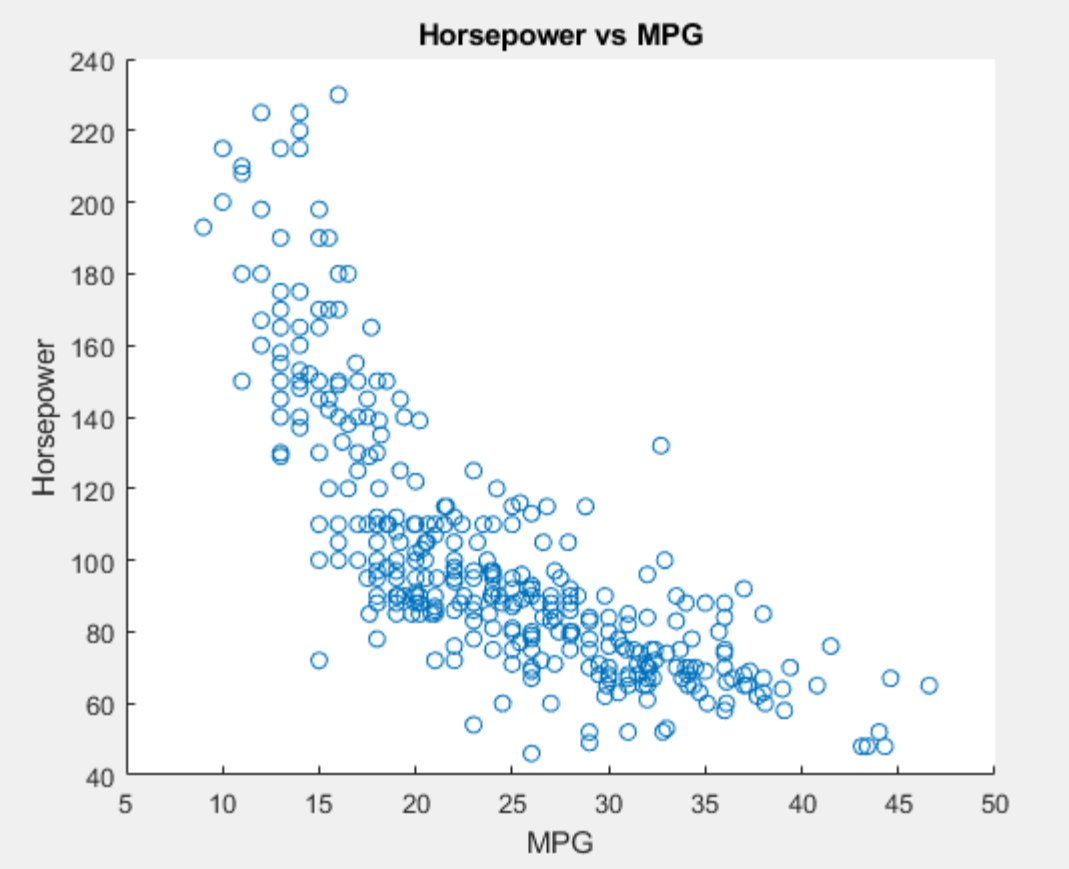




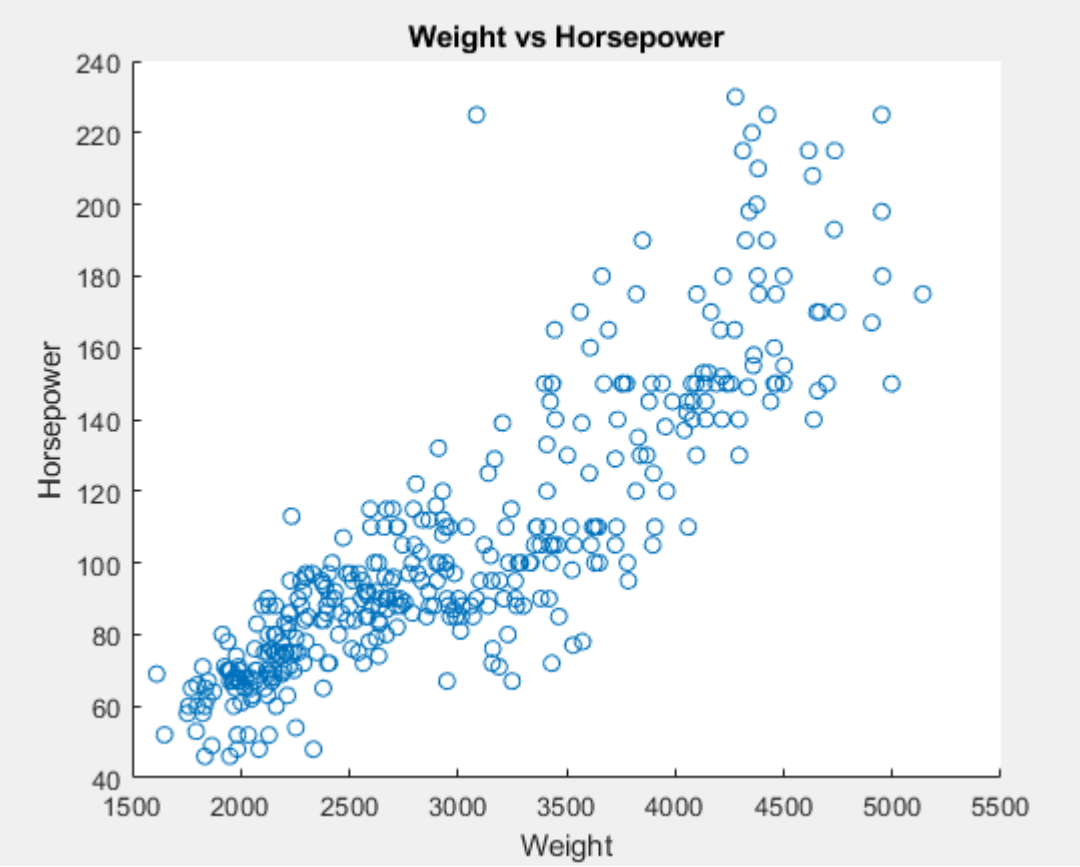
Jamie Godwin 18675053



The scatter plot shown above could be somewhat positively collated or no correlation at all. This is because it has no clear upwards or downwards trend. Generally, as acceleration is increased MPG will also increase however this isn't always the case as some values have higher y's than x's .

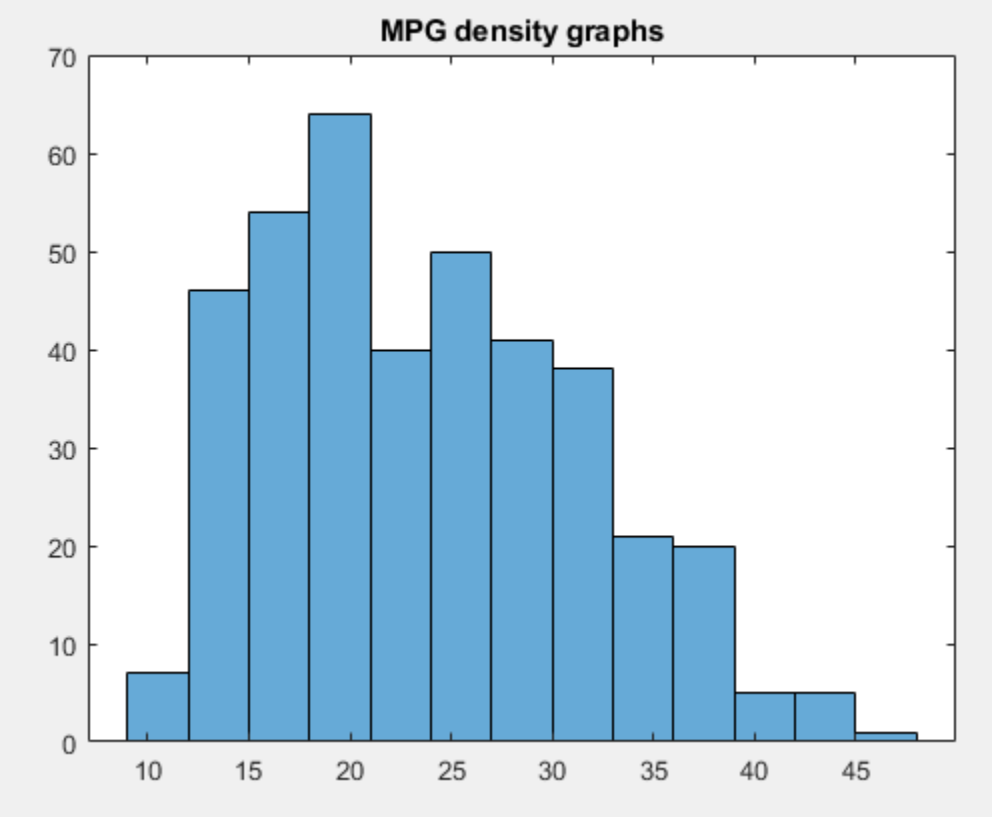


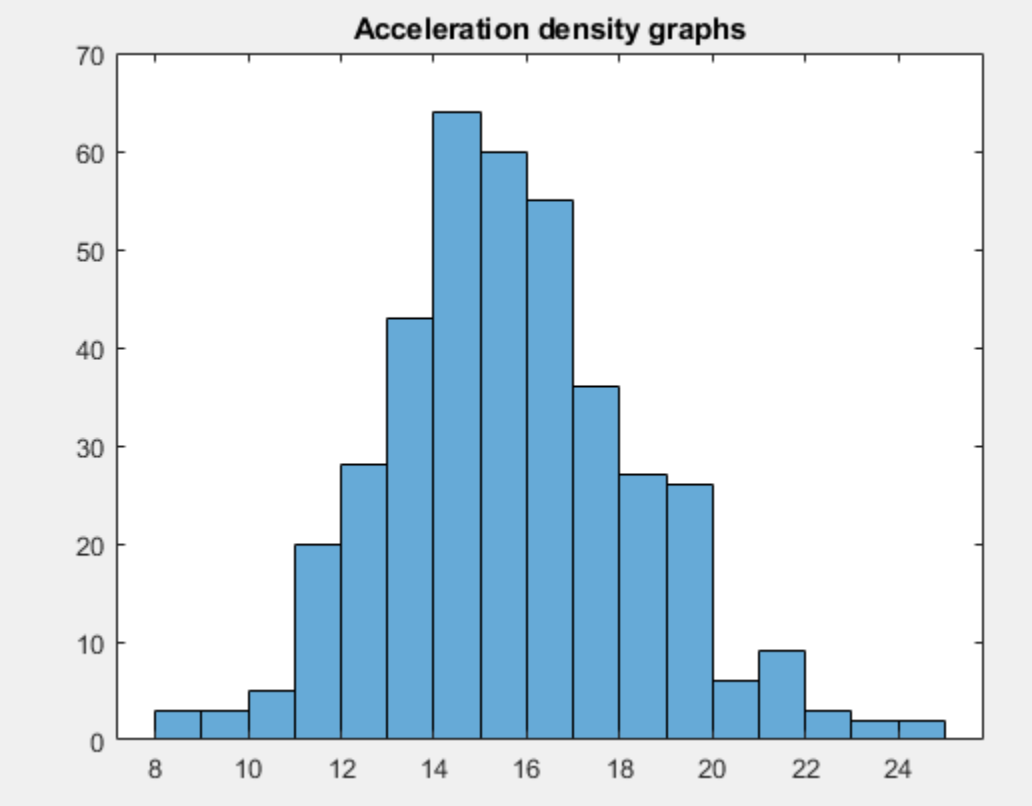
The higher the horsepower the lower MPG this scatter plot has a clear negative correlation between the two variables.



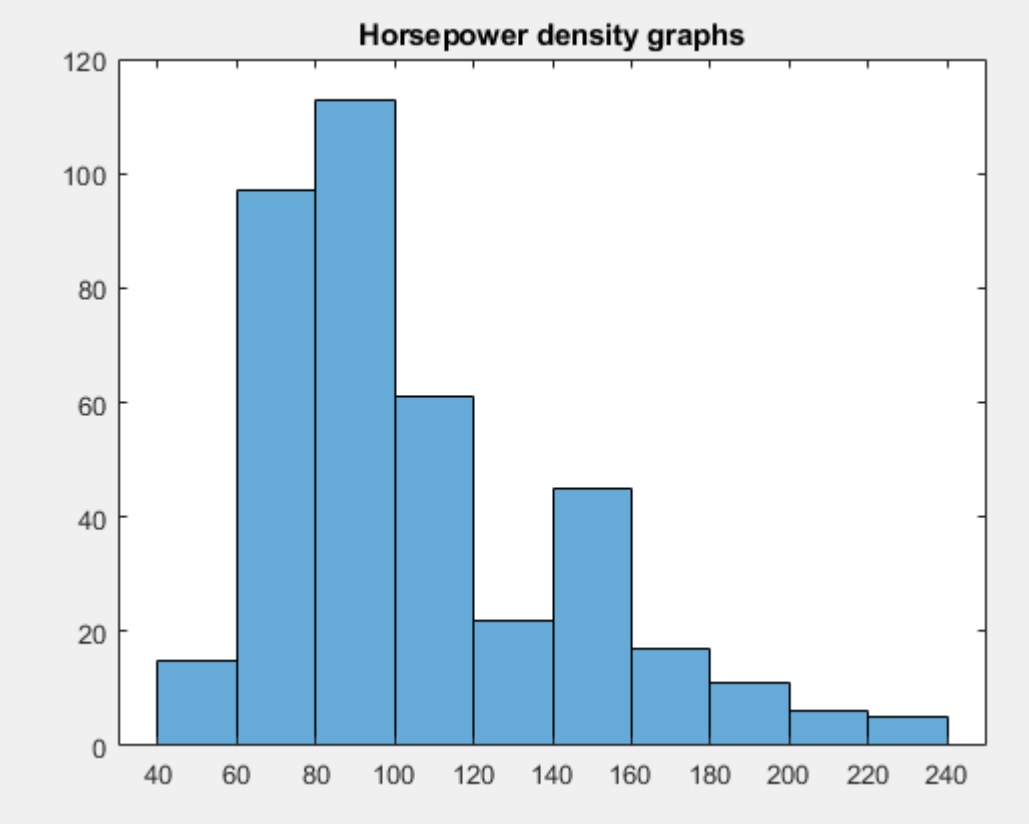
The more weight that the car has the more horsepower this scatter plot shows a clear positive correlation between the two variables. This dose however have a few points that our outliers. Due to the massive difference in weight and horsepower.

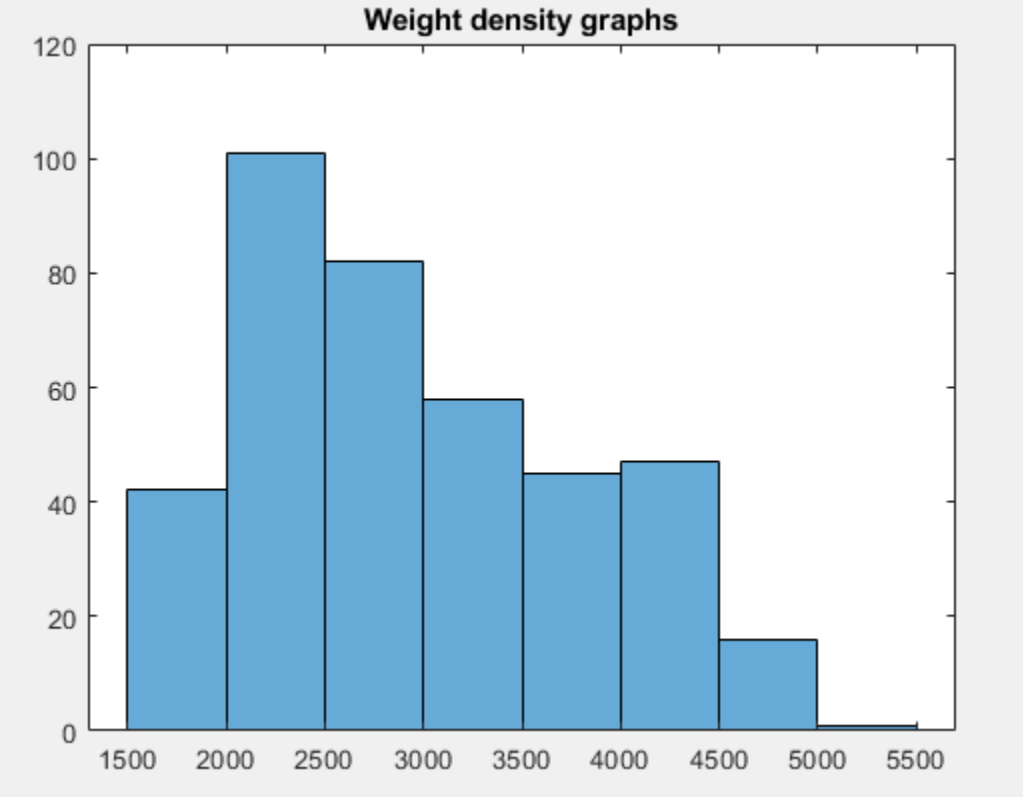
```
%ploting the data using boxplot
figure
boxplot(mpgarray)
title('MPG')
figure
boxplot(accelerationarray)
title('Acceleration')
figure
boxplot(horsearray)
title('Horsepower')
figure
boxplot(weightarray)
title('Weight')
% plotting the scatter graphs
figure
scatter(accelerationarray,mpgarray)
title('Acceleration vs MPG')
xlabel('Acceleration');
ylabel('MPG');
figure
scatter(mpgarray,horsearray)
title('Horsepower vs MPG')
xlabel('MPG');
ylabel('Horsepower');
figure
scatter(weightarray,horsearray)
title('Weight vs Horsepower')
xlabel('Weight');
ylabel('Horsepower');
%plotting the density graphs
figure
histogram(mpgarray)
title('MPG density graph')
figure
histogram(accelerationarray)
title('Acceleration density graph')
figure
histogram(horsearray)
title('Horsepower density graph')
figure
histogram(weightarray)
title('Weight density graph')
```



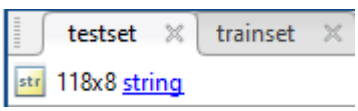


Jamie Godwin 18675053
Big Data - 2021 CMP3749M-2021

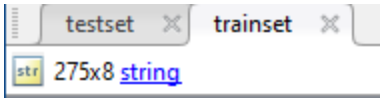




Task 6



Jamie Godwin 18675053
Big Data - 2021 CMP3749M-2021



The test set has 118 by 8 elements in the array compared to the training set which has 275 by 8 elements in the array

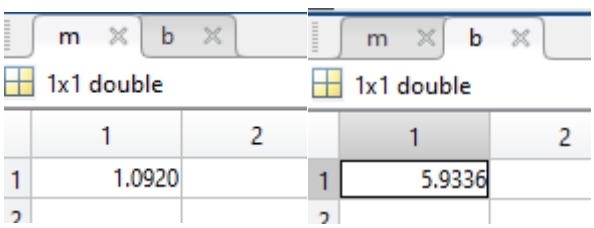
Task 7

```
[m , b] = Regression(trainset, 6,1);
```

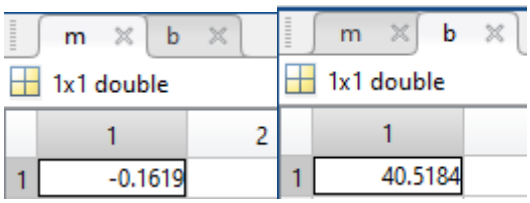
This line calls function regression and sets the columns that will be used for the calculation. In this case 6 and 1 for acceleration and mpg.

```
%regression
function [m,b] = Regression(trainset,q,p)
         %seting the sum vaules and geting x and y vaules from trainset
         x = trainset(2:size(trainset, 1), q);
         y = trainset(2:size(trainset, 1), p);
         sumx = 0;
         sumy = 0;
         sumxy = 0;
         sumx2 = 0;
         %converts string into dobule to be used for caulations
         for i = 1 : size(x,1)
             sumx = sumx+str2double(x(i));
             sumy = sumy+str2double(y(i));
             sumxy = sumxy+str2double(x(i))*str2double(y(i));
             sumx2 = sumx2+str2double(x(i))*str2double(x(i));
         end
         %cacualtes the slope and intercept for simple L regression
         f = (size(x,1)*sumxy)-(sumx*sumy);
         h = (size(x,1)*sumx2)-(sumx*sumx);
         m = f/h;
         b = (sumy-(m*sumx))/size(x,1);
end
```

X and y values are first gotten from the trainset data then all of the sums variables are called. These will be used to calculate regression. For every row the value is gotten by first converting from string to double so that the element in the array is a double not a string this allow it to be used for calculations were all of the values in the column are added this is what sum is used for. Finally the sum values are used to calculate the M which is the slope and b which is the Intercept. B also shows the average least square error which in this case is 5.9336 which is show below along with the value for m.



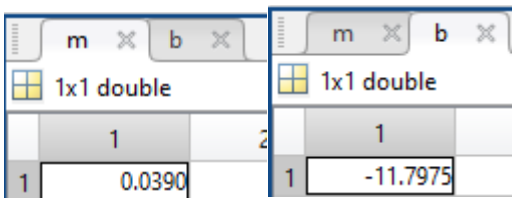
Task 9



The average training least-square error is 40.5184. The regression is calculated the same except for calling the function which uses the line below.

```
[m , b] = Regression(trainset, 4,1);
```

Task 11



The average least square error was -11.7975. The regression is calculated the same except for calling the function which uses the line below.

```
[m , b] = Regression(trainset, 5,4);
```