Data Exploration

a) Copy/paste runs of your code showing the output

Code:

```
#include <iostream>
#include <conio.h>
#include <string>
#include <vector>
#include <fstream>
#include <algorithm>
#include <cmath>
#include <iomanip>

using namespace std;

// Function for reading file content and printing details of file
void OpeningFileForReading(vector<double> &rm, vector<double> &medv)
{
   fstream inFs;
   string rm_in, medv_in, line;
   // const int MAX_LEN = 1000;

   cout << "Opening file Boston.csv" << endl;
   inFs.open("Boston.csv", ios::in);
   if (inFs.is_open())
   {
      cout << "Boston.csv opened" << endl;
```

```
}
if (!inFs.is_open())
{

   cout << "Could not open file Boston.csv" << endl;
}
cout << "Reading line 1 " << endl;
getline(inFs, line);

cout << "heading: " << line << endl;
int numObservations = 0;
while (inFs.good())
{
   getline(inFs, rm_in, ',');
   getline(inFs, medv_in, '\n');
   rm.push_back(stod(rm_in));
   medv.push_back(stod(medv_in));

   numObservations++;
}

cout << "new length: " << rm.size() << endl;
cout << "Closing file Boston.csv" << endl;
inFs.close();
cout << "Number of records: " << numObservations << endl;
```

```
}

// Function for calculating sum of elements of a vector
double sumOfVector(vector<double> elements)
{
   double sum = 0;
   for (int i = 0; i < elements.size(); i++)
   {
      sum = sum + elements[i];
   }
   return sum;
}

// Function for calculating mean of elements of a vector
double meanOfVector(vector<double> elements)
{
   double sum = sumOfVector(elements);
   int lenghtOfVector = elements.size();
   return sum / lenghtOfVector;
}

// Function for calculating the median of elements of a vector
double medianOfVector(vector<double> elements)
{
   sort(elements.begin(), elements.end());
```

```
    int lengthOfVector = elements.size();
    if (lengthOfVector % 2 != 0)
    {
        return elements[lengthOfVector / 2];
    }

    return (elements[lengthOfVector / 2] + elements[(lengthOfVector / 2) - 1]) / 2.0;
}

// Function for calculating the range of values of vector(maximum and minimum)
vector<double> rangeOfVector(vector<double> elements)
{

    vector<double> range;

    double minValue = elements[0];
    double maxValue = elements[0];

    for (int i = 0; i < elements.size(); i++)
    {
        if (elements[i] < minValue)
        {
            minValue = elements[i];
        }
        if (elements[i] > maxValue)
```

```
        {
            maxValue = elements[i];
        }
    }

    range.push_back(minValue);
    range.push_back(maxValue);
    return range;
}

// Function for calculating the covariance of among two vectors
double covariance(vector<double> firstElements, vector<double>
secondElements)
{
    double intermediateDifferenceAndProduct = 0;
    double length = firstElements.size();
    double meanOfFirst = meanOfVector(firstElements);
    double meanOfSecond = meanOfVector(secondElements);
    for (int i = 0; i < firstElements.size(); i++)
    {
        intermediateDifferenceAndProduct = intermediateDifferenceAndProduct +
                                    (firstElements[i] - meanOfFirst) * (secondElements[i] -
meanOfSecond);
    }
    return intermediateDifferenceAndProduct / length;
}
```

```
// Function for calculating standard deviation
double standardDeviation(vector<double> elements)
{
   double squareOfDeviationFromMean = 0.0;
   double sumOfSquares = 0.0;
   double variance = 0.0;
   double mean = meanOfVector(elements);
   for (int i = 0; i < elements.size(); i++)
   {
      squareOfDeviationFromMean = (elements[i] - mean) * (elements[i] - mean);
      sumOfSquares = sumOfSquares + squareOfDeviationFromMean;
   }
   variance = sumOfSquares / elements.size();
   return sqrt(variance);
}

// Function for calculating correlation among two vectors
double correlation(vector<double> firstElements, vector<double>
secondElements)
{
   double sdOfFirst = standardDeviation(firstElements);
   double sdOfSecond = standardDeviation(secondElements);
   double covarianceOfTwo = covariance(firstElements, secondElements);
   double productOfTwoSd = sdOfFirst * sdOfSecond;
   return (covarianceOfTwo / productOfTwoSd);
```

```
}

// Main function
int main()
{
    vector<double> rm;
    vector<double> medv;
    // calling functions to get the calculated values.
    OpeningFileForReading(rm, medv);
    double sumOfRm = sumOfVector(rm);
    double sumOfmedv = sumOfVector(medv);
    double medianOfRm = medianOfVector(rm);
    double medianOfmedv = medianOfVector(medv);
    double meanOfRm = meanOfVector(rm);
    double meanOfMedv = meanOfVector(medv);
    vector<double> ranges = rangeOfVector(rm);
    vector <double> rangesOfMedv = rangeOfVector(medv);
    double covarianceOfTwo = covariance(rm, medv);
    double correlationOfTwo = correlation(rm, medv);

    // Printing all the calculated values

    cout << "\nStats for rm\n"
         << endl;
    cout << "---------------------------\n";
```

```
    cout << "Sum of Rm: " << setw(20)  << sumOfRm << endl;
    cout << "sum of medv: " << setw(18) << sumOfmedv << endl;
    cout << "Mean of Rm: " << setw(20) << meanOfRm << endl;
    cout << "Mean of Medv: " << setw(17) << meanOfMedv << endl;
    cout << "Median of Rm: " << setw(16) << medianOfRm << endl;
    cout << "Median of Medv: " << setw(12) << medianOfmedv << endl;
    cout << "Range of Rm : " << setw(15) << ranges[0] << " : " << ranges[1] << endl;
    cout << "Range of medv : " << setw(10) << rangesOfMedv[0] << " : " << rangesOfMedv[1] << endl;
    cout << "Covariance: " << setw(19) << covarianceOfTwo << endl;
    cout << "Correlation: " << setw(18) << correlationOfTwo << endl;
    cout << "---------------------------\n";


    return 0;
}
```

Output:

```
Opening file Boston.csv
Boston.csv opened
Reading line 1
heading: rm,medv
new length: 506
Closing file Boston.csv
Number of records: 506


Stats for rm


---------------------------
```

Sum of Rm: 3180.03

sum of medv: 11401.6

Mean of Rm: 6.28463

Mean of Medv: 22.5328

Median of Rm: 6.2085

Median of Medv: 21.2

Range of Rm : 3.561 : 8.78

Range of medv : 5 : 50

Covariance: 4.48457

Correlation: 0.69536

```
Opening file Boston.csv
Boston.csv opened
Reading line 1
Heading: rm,medv
new length: 506
Closing file Boston.csv
Number of records: 506

Stats for rm

----------------------------
Sum of Rm:              3180.03
sum of medv:            11401.6
Mean of Rm:              6.28463
Mean of Medv:           22.5328
Median of Rm:           6.2085
Median of Medv:         21.2
Range of Rm :           3.561 : 8.78
Range of medv :          5 : 50
Covariance:             4.48457
Correlation:            0.69536
----------------------------

Process returned 0 (0x0)   execution time : 0.056 s
Press any key to continue.
```

b) Describe your experience using build-in functions in R versus coding your own functions in C++.

Using build-in function in R was easier and faster than coding my own function in C++. C++ is the language which I have not used for a year and more so, I was struggling in syntax for a while but was able to catch up easily. Obviously, coding in C++, took some more time than using R build in functions.

c) Describe the descriptive statistical measures mean, median and range and how these values might be useful in data exploration prior to Machine Learning.

Mean: Mean is the sum of all the value divided by total number of elements. Mathematically, it can be written as follows:
Mean = sum/number of elements

Median: Median is the middle element in the collection of sorted numbers either in ascending order or in descending order. It is more preferred than mean when there is outlier in the data.

Range: It is the lower and upper bound in the collection of values.

These metrics are very important in data exploration because data exploration consists of understanding of variables, detecting outliers and detecting patterns. These metrics can be helpful for detecting outliers which can later help us to create robust machine learning models.

d) Describe the covariance and correlation statistics and what information they give about two attributes. How might this information be useful in machine learning?

Covariance explains the relationship between two variables. It only gives sigh positive and negative. If the value is positive, then it means that there is direct relationship between two variables while negative means inverse relationship.

Correlation explains the intensity of change of one variable with change in another variable and it ranges from -1 to 1. 1 means strong positive relation and -1 means weak relationship while 0 means no linear relationship at all.

Correlation and covariance can be useful in feature engineering and dimension reduction which can later help us to create robust machine learning models with higher accuracy.