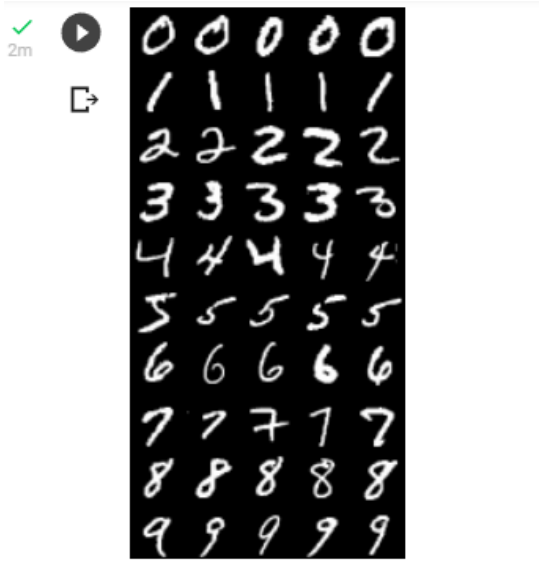
Ans 1 a)

i)

In this part, I am looping through the train labels and creating a dictionary. In the dictionary, I am keeping track of unique classes and their count. As I find a particular class label, I am storing its image matrix in a list. When the count of the class becomes equal to 5, I am not storing that corresponding class image in the list as we only need 5 images of each class. For displaying, I am concatenating the 5 images of each class horizontally.



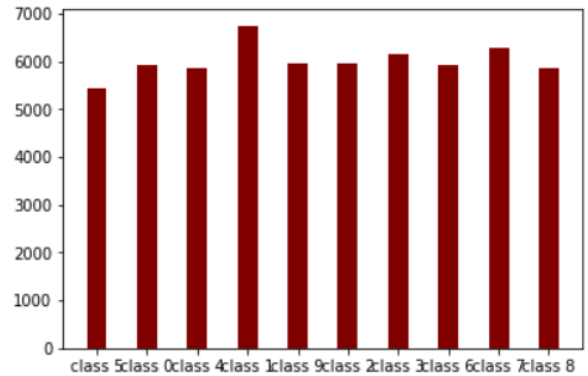
ii)

For EDA, I am looping through all the images and checking their image size. I observed that all the images are of 28x28 dimensions.

After that, I have plotted a bar plot to check class imbalance. For that I have created a dictionary to store the count corresponding to each class out of 10 classes. After that, I am simply plotting the bar plot.

Then, I have normalized all the images. For normalization, I am dividing each cell of the image matrix by 255.0 which is the maximum possible value to get all pixels in 0 to 1 range.

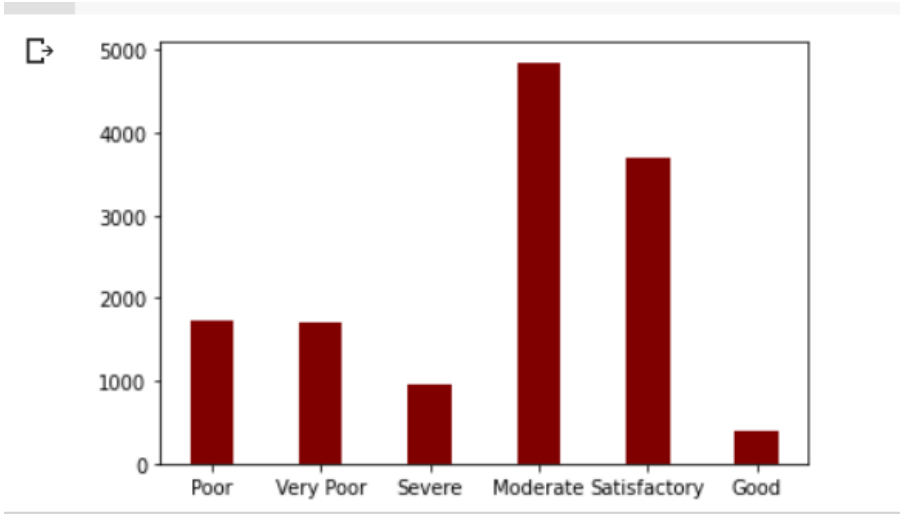
```
Yes, all images are of same size
images dimensions : (28, 28)
```



Ans 1 b)

i)

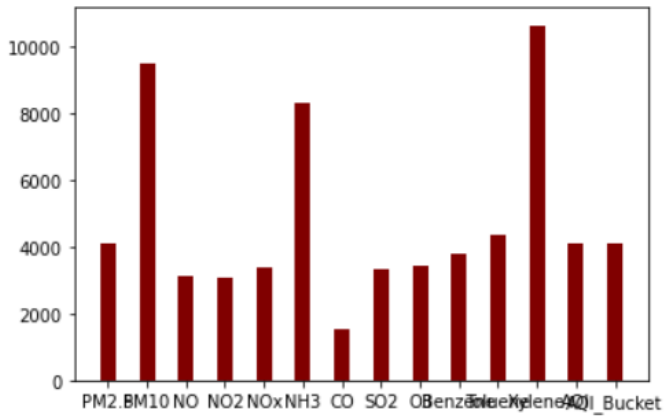
In this part, for EDA I have plotted the bar plot of different classes in aqi_bucket to show the class imbalance. After that, I have done the normalization of features like PM2.5, PM10, etc. I am creating a dictionary for getting the count corresponding to each AQI bucket. For normalization, I am getting the maximum value corresponding to each column and then looping through all the rows and dividing the cells by their corresponding maximum value to get the normalized values from 0 to 1.



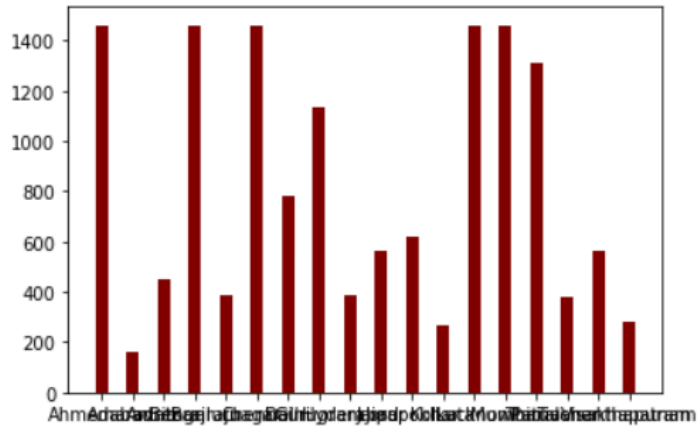
	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene
0	Ahmedabad	2015-01-01	NaN	NaN	0.002355	0.050302	0.036674	NaN	0.005233	0.142577	0.517441	0.000000	0.000044
1	Ahmedabad	2015-01-02	NaN	NaN	0.002483	0.043317	0.035199	NaN	0.005517	0.126638	0.132154	0.008087	0.012092
2	Ahmedabad	2015-01-03	NaN	NaN	0.044538	0.053284	0.063512	NaN	0.098970	0.149954	0.119117	0.014944	0.036056
3	Ahmedabad	2015-01-04	NaN	NaN	0.004351	0.051020	0.038428	NaN	0.009670	0.095894	0.139991	0.009736	0.022293
4	Ahmedabad	2015-01-05	NaN	NaN	0.056568	0.059137	0.080748	NaN	0.125704	0.202878	0.152524	0.015406	0.041530
...	...	...	...	...	...	...	...	...	...	...	...	...	...
29526	Visakhapatnam	2020-06-27	15.02	50.94	7.680000	25.060000	19.540000	12.47	0.470000	8.550000	23.300000	2.240000	12.070000
29527	Visakhapatnam	2020-06-28	24.38	74.09	3.420000	26.060000	16.530000	11.99	0.520000	12.720000	30.140000	0.740000	2.210000
29528	Visakhapatnam	2020-06-29	22.91	65.73	3.450000	29.530000	18.330000	10.71	0.480000	8.420000	30.960000	0.010000	0.010000
29529	Visakhapatnam	2020-06-30	16.64	49.97	4.050000	29.260000							
29530	Visakhapatnam	2020-07-01	15.00	66.00	0.400000	26.850000							

✓ 2m 54s completed at 3:21 PM

ii) In this part, I am firstly plotting the count of null values present in each columns. In the second plot, I am plotting the count of null values corresponding to each city. For the second plot, even if one of the feature is null, I am incrementing the count and plotting it. Finally, I am displaying the data after imputation. In this data, For the float valued columns, I have replaced the null data with 0 whereas in the AQI_Bucket column I have replaced the null data with the maximum occuring AQI which is moderate.



[29531 rows x 16 columns]



	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene
0	Ahmedabad	2015-01-01	0.00	0.00	0.002355	0.050302	0.036674	0.00	0.005233	0.142577	0.517441	0.000000	0.000044
1	Ahmedabad	2015-01-02	0.00	0.00	0.002483	0.043317	0.035199	0.00	0.005517	0.126638	0.132154	0.008087	0.012092
2	Ahmedabad	2015-01-03	0.00	0.00	0.044538	0.053284	0.063512	0.00	0.098970	0.149954	0.119117	0.014944	0.036056
3	Ahmedabad	2015-01-04	0.00	0.00	0.004351	0.051020	0.038428	0.00	0.009670	0.095894	0.139991	0.009736	0.022293
4	Ahmedabad	2015-01-05	0.00	0.00	0.056568	0.059137	0.080748	0.00	0.125704	0.202878	0.152524	0.015406	0.041530
...	...	...	...	...	...	...	...	...	...	...	...	...	...
29526	Visakhapatnam	2020-06-27	15.02	50.94	7.680000	25.060000	19.540000	12.47	0.470000	8.550000	23.300000	2.240000	12.070000
29527	Visakhapatnam	2020-06-28	24.38	74.09	3.420000	26.060000	16.530000	11.99	0.520000	12.720000	30.140000	0.740000	2.210000
29528	Visakhapatnam	2020-06-29	22.91	65.73	3.450000	29.530000	18.330000	10.71	0.480000	8.420000	30.960000	0.010000	0.010000
29529	Visakhapatnam	2020-06-30	16.64	49.97	4.050000	29.260000	18.800000	10.03	0.520000	9.840000	28.300000	0.000000	0.000000
29530	Visakhapatnam	2020-07-01	15.00	66.00	0.400000	26.850000	14.050000	5.20	0.590000	2.100000	17.050000	0.000000	0.000000

Some imputation methods include :

1) Complete Case Analysis : In this method we simply remove the row if there is missing data in that row.
   Pros : Implementation is easy, we're not manipulating the data.
   Cons : loss of information

2) Arbitrary Value Imputation : In this method we replace the missing data with arbitrary values.
   Pros : Implementation is easy, we still have knowledge of rows where there was missing data.
   Cons : Manipulates the data randomly.
3) Frequent Category Imputation : In this method we replace the missing data with the frequently occurring element in that column.
   Pros : not much data manipulation as we have used frequently occurring elements.
   Cons : Implementation is difficult as compared to above methods.

Ans 3 c)

For linear regression, I have separated the data into 90% training and 10% testing. I am using the mean squared error loss function for gradient descent. I have taken 5000 epochs with a learning rate of 0.03. In the gradient descent function, I am updating the learning parameters. The equation of line would be y = t0 + t1x1 + t2x2 + …. + t10x10. Parameters t0…t10 are learned using mean squared error loss function.

The output of code written from scratch comes closer to the inbuilt output as I increase the epochs. However due to time and resource constraints I am keeping epochs to 5000.

Output :

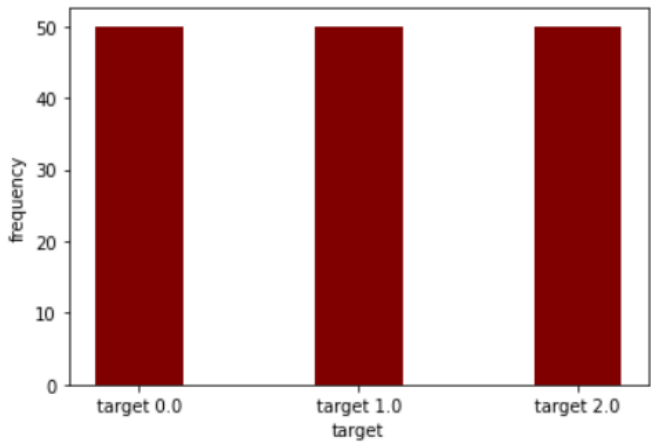
```
scratch predicted [185.12895322900988, 157.64998861808436,
100.30286695231545, 171.4775691898705, 210.0884602265496,
168.8004876340456, 221.07007639961043, 62.46502714771048,
173.078992493428, 194.06851072956863, 180.76341297503825,
164.81743682342167, 130.8665871801135, 210.2383058884834,
126.76026906206332, 192.25962217781637, 171.4123058084549,
199.71503540596018, 143.63902064244573, 128.28963585233942,
96.2522448248852, 157.97677285710225, 195.5138308142104,
186.97098998123448, 169.91332489081364, 171.68484633075164,
92.20556172708794, 186.74993498212243, 133.74518444350196,
244.54791156009682, 113.82739591323187, 132.69284599631777,
130.29370269162212, 208.80644871799632, 88.60457560838434,
139.74838702311803, 125.66673313891519, 68.26432014051447,
183.16299981768492, 143.08038778831855, 140.59066963504685,
183.718443232128, 80.41596621618652]

inbuilt [188.07247897 185.55957945  90.37526191 152.17268368 250.669681
 198.29412469 280.59643037  51.0774851  176.96995559 201.88392495
 172.33337822 155.46180594 152.2660552  234.46360623 123.0999278
 165.98980815 174.976899   226.33969244 152.66327917 101.26180257
  83.99410255 143.86990982 192.06373833 195.77168679 153.43116132
 172.81321567 111.80961387 163.74102065 131.5665228  258.29175959
 100.86830795 117.82315286 123.01401147 219.84967296  62.92523948
 133.26807984 121.21476491  53.61407284 191.35028281 105.1108812
 123.4818946  210.06896662  54.98629244]
```

Ans 4

In this problem, initially for EDA, I am checking the class imbalance. As seen from the plot, there is no class imbalance. Then, I am applying normalization on each of the 4 features to get data in the 0 to 1 range. Then for each feature, I am dividing it into 4 quarters i.e. quarter 1 would be from [0,0.25), then quarter 2 from [0.25, 0.5) and so on. Then I am calculating the conditional probabilities of occurrence of each quarter given it belongs to class 1. Then the probability of each quarter given it belongs to class 2. Then the probability of each quarter given it belongs to class 3. I am doing this for all the 4 features. After getting all the conditional probabilities, I am now predicting whether the test data features belong to class 1, 2 or 3. For that I am simply obtaining P(class=1 | features), P(class=2 | features) and P(class=3 | features). Now the probability that is the highest would be the classified class for those features.

I got 100% for having 90% train data and 10% test data. For the same data I got 93.33% accuracy using sklearn.




```
accuracy for code written from scratch (in precent): 100.0
accuracy for inbuilt (in percent) : 93.33333333333333
```

Ans 5

In this problem, I am basically splitting data into 90% train and 10% test. After that, for each test data age, I am obtaining the euclidean distance from the train data ages. Then I am obtaining the minimum “k” distances and corresponding to those distances we’ve the BMD and I am taking the mean of that BMD to obtain the predicted BMD of the test data. In this method I am following for K ranging from 1, len(train data). Then corresponding to each “K”, I have obtained the R2 score. It was observed that the R2 score came out to be maximum for K = 150.

```
Optimum K : 150
```