COWLAR Task Report

Name:

Talha Tahir Khurshid

Applicant:

AI/ML Job

Status:

- Completed Videos:
    1) Git & Github CrashCourse
    2) Resolving Git Merge conflicts
    3) Advanced Git
    4) Python with OOP
    5) The Complete Guide to Python Virtual Environments!
    6) Specialization in deep learning part-1
    7) Specialization in deep learning part-2
    8) Specialization in deep learning part-3
    9) Specialization in deep learning part-4
- Task 1: completed
- Task 2: completed

Problems for task 1:

- **Creating a dataset:** Since I read the dataset in order, it was not giving accurate results in the algorithm so I manually reshuffled the entire dataset to get a randomized training and testing data set.
- **Reading a CSV file correctly:** Had problems accessing the first row of the dataset, as it was taken as an index, so I tried adding a new row but then finally the problem was solved by setting header = NONE when reading the CSV file.
- **Problem in measuring distances using KNN:** Had a problem in implementing the Euclidean distance, but later realized after checking the entire KNN algorithm that I had missed a negative sign in the formula.
- **Problem in using the Mahalanobis distance:** Tried implementing this distance however it posed problems

Results:

- The results shown below implement KNN using 5 and 8 classes.

```
Accuracy: 0.75
time taken with  5 classes : 0.7900283336639404
Accuracy: 0.79
time taken with  5 classes : 0.7800004482269287
Accuracy: 0.76
time taken with  5 classes : 0.7500064373016357
Accuracy: 0.82
time taken with  5 classes : 0.7799897193908691
Accuracy: 0.8
time taken with  5 classes : 0.7799997329711914
Average Accuracy: 0.8
Standard Deviation: 0.0

Process finished with exit code 0
```

```
Accuracy: 0.825
time taken with  8 classes : 1.8656651973724365
Accuracy: 0.84375
time taken with  8 classes : 1.9099974632263184
Accuracy: 0.8125
time taken with  8 classes : 2.0583577156066895
Accuracy: 0.85625
time taken with  8 classes : 2.069998264312744
Accuracy: 0.79375
time taken with  8 classes : 1.9298443794250488
Average Accuracy: 0.79375
Standard Deviation: 0.0
```

- The below shows the result of using different values of ‘K’ in KNN.

```
Accuracy: 0.91
time taken with  1 value in KNN : 2.9497931003570557
Accuracy: 0.87
time taken with  1 value in KNN : 3.279998540878296
Accuracy: 0.93
time taken with  1 value in KNN : 3.11999249458313
Accuracy: 0.905
time taken with  1 value in KNN : 2.9999637603759766
Accuracy: 0.945
time taken with  1 value in KNN : 3.261852264404297
Average Accuracy: 0.945
Standard Deviation: 0.0
Accuracy: 0.8
time taken with  3 value in KNN : 3.3488054275512695
Accuracy: 0.81
time taken with  3 value in KNN : 3.309967279434204
Accuracy: 0.855
time taken with  3 value in KNN : 3.3629472255706787
Accuracy: 0.83
time taken with  3 value in KNN : 3.080256938934326
Accuracy: 0.8
time taken with  3 value in KNN : 3.037954092025757
Average Accuracy: 0.8
Standard Deviation: 0.0
Accuracy: 0.545
time taken with  5 value in KNN : 3.3098690509796143
Accuracy: 0.55
time taken with  5 value in KNN : 3.0399973392486572
Accuracy: 0.61
time taken with  5 value in KNN : 2.9799985885620117
Accuracy: 0.595
time taken with  5 value in KNN : 3.280027151107788
```

```
time taken with  5 value in KNN : 3.0399973392486572
Accuracy: 0.61
time taken with  5 value in KNN : 2.9799985885620117
Accuracy: 0.595
time taken with  5 value in KNN : 3.280027151107788
Accuracy: 0.57
time taken with  5 value in KNN : 2.9933488368988037
Average Accuracy: 0.57
Standard Deviation: 0.0
Accuracy: 0.47
time taken with  7 value in KNN : 3.038177728652954
Accuracy: 0.475
time taken with  7 value in KNN : 4.58436393737793
Accuracy: 0.515
time taken with  7 value in KNN : 2.9500279426574707
Accuracy: 0.445
time taken with  7 value in KNN : 3.0427498817443848
Accuracy: 0.45
time taken with  7 value in KNN : 3.2017061710357666
Average Accuracy: 0.45
Standard Deviation: 0.0
Accuracy: 0.455
time taken with  9 value in KNN : 2.9799983501434326
Accuracy: 0.415
time taken with  9 value in KNN : 3.106342077255249
Accuracy: 0.43
time taken with  9 value in KNN : 3.649998188018799
Accuracy: 0.455
time taken with  9 value in KNN : 3.003096342086792
Accuracy: 0.42
time taken with  9 value in KNN : 3.078568458557129
Average Accuracy: 0.42
```

- The below results show the effect less training images in the KNN classify.

```
Accuracy: 0.6585714285714286
time taken with  100 training and 70 test images in KNN : 6.916828155517578
Accuracy: 0.6585714285714286
time taken with  100 training and 70 test images in KNN : 6.872903823852539
Accuracy: 0.6414285714285715
time taken with  100 training and 70 test images in KNN : 7.1600306034088135
Accuracy: 0.6114285714285714
time taken with  100 training and 70 test images in KNN : 7.208104372024536
Accuracy: 0.62
time taken with  100 training and 70 test images in KNN : 7.150185823440552
Average Accuracy: 0.62
Standard Deviation: 0.0
```

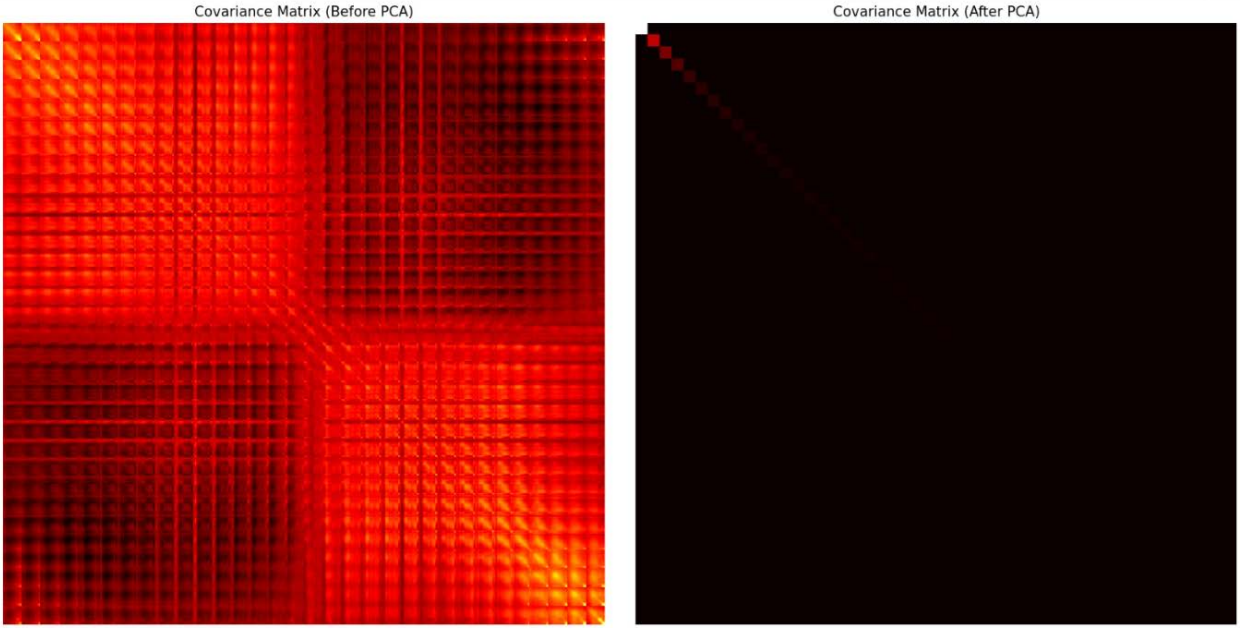
```
Accuracy: 0.3775
time taken with  50 training and 120 test images in KNN : 6.547948598861694
Accuracy: 0.4066666666666667
time taken with  50 training and 120 test images in KNN : 6.43002724647522
Accuracy: 0.38083333333333336
time taken with  50 training and 120 test images in KNN : 6.466084718704224
Accuracy: 0.3975
time taken with  50 training and 120 test images in KNN : 6.315119504928589
Accuracy: 0.39416666666666667
time taken with  50 training and 120 test images in KNN : 6.458536863327026
Average Accuracy: 0.39416666666666667
Standard Deviation: 0.0

Process finished with exit code 0
```

- These results show the effect using the dimension reduction technique using principle component analysis (PCA). The PCA is set to 50.
  The time is reduced roughly by 0.5 seconds.

```
Accuracy: 0.99
time taken before PCA : 3.114166021347046
Accuracy: 0.995
time taken before PCA : 3.0679965019226074
Accuracy: 0.995
time taken before PCA : 3.768148183822632
Accuracy: 0.985
time taken before PCA : 3.357408285140991
Accuracy: 0.995
time taken before PCA : 3.1399972438812256
Average Accuracy: 0.992
Standard Deviation: 0.0040000000000000036
```

```
time taken after PCA : 2.5699636936187744
Accuracy: 0.995
time taken after PCA : 2.3799962997436523
Accuracy: 1.0
time taken after PCA : 2.316422700881958
Accuracy: 0.985
time taken after PCA : 2.5144784450531006
Accuracy: 0.995
time taken after PCA : 2.616790294647217
Average Accuracy: 0.993
Standard Deviation: 0.00509901951359279
```

- Covariance matrix visualization:

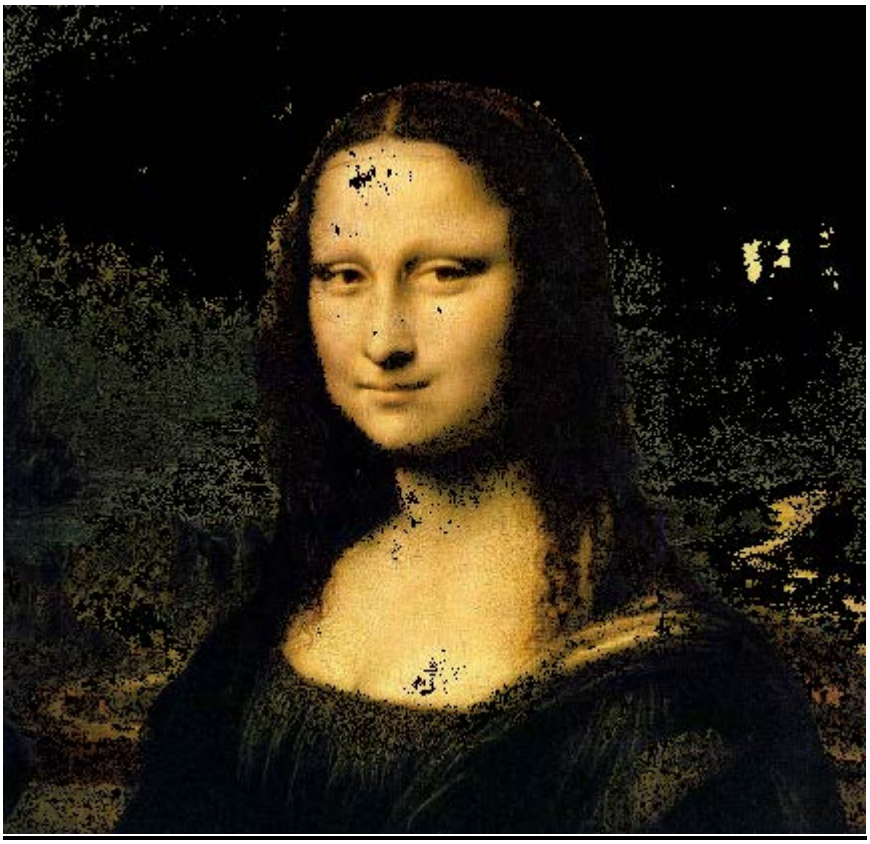


Problems for task 2:

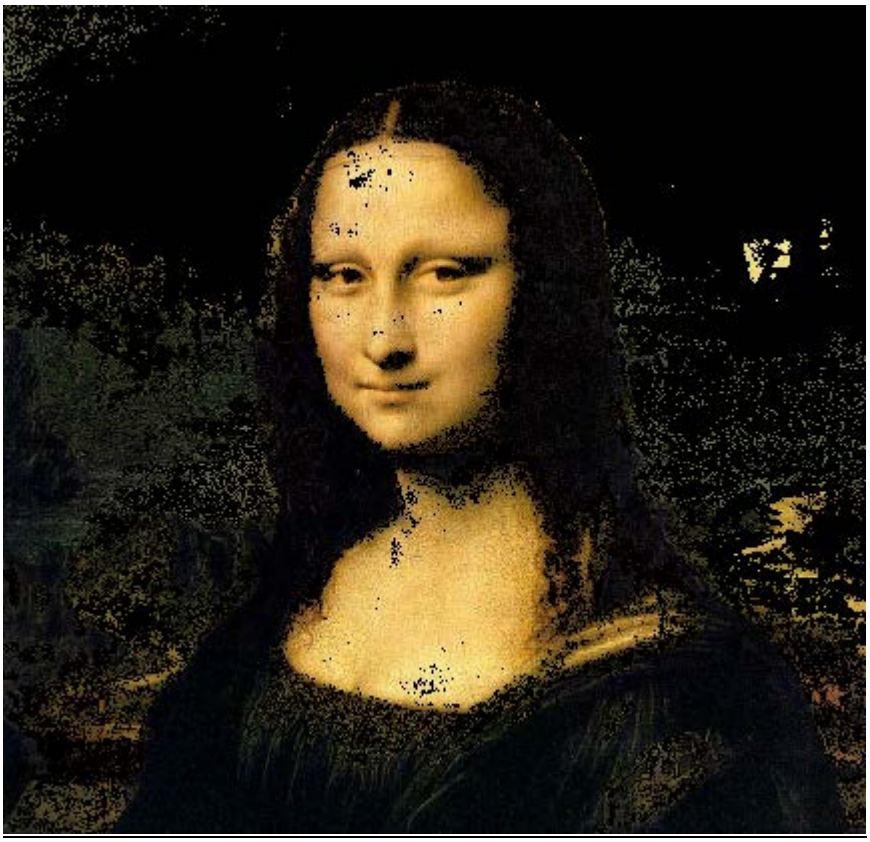
- **Reading the seed image:** The seed image looked like they contained only pure red or blue color however , the blue seed pixel had a very small component of red color in them which caused me to change the code so that only values from the first and last channel of the array are taken in order to properly read the seed pixels
- **Van gogh image:** Since the van gogh image was a 4 channel it didn't work with my code at first so I added the code where if the array is a 4 channel array, then the last channel is dropped since it contained no important information about the image . This is was done by doing python slicing.
- **Had difficulty implementing the manual Kmeans :** At first I was getting error on running the manual kmeans on given images , but later by changing my approach I was able to correctly implement the Kmeans algorithm manual.

Results:

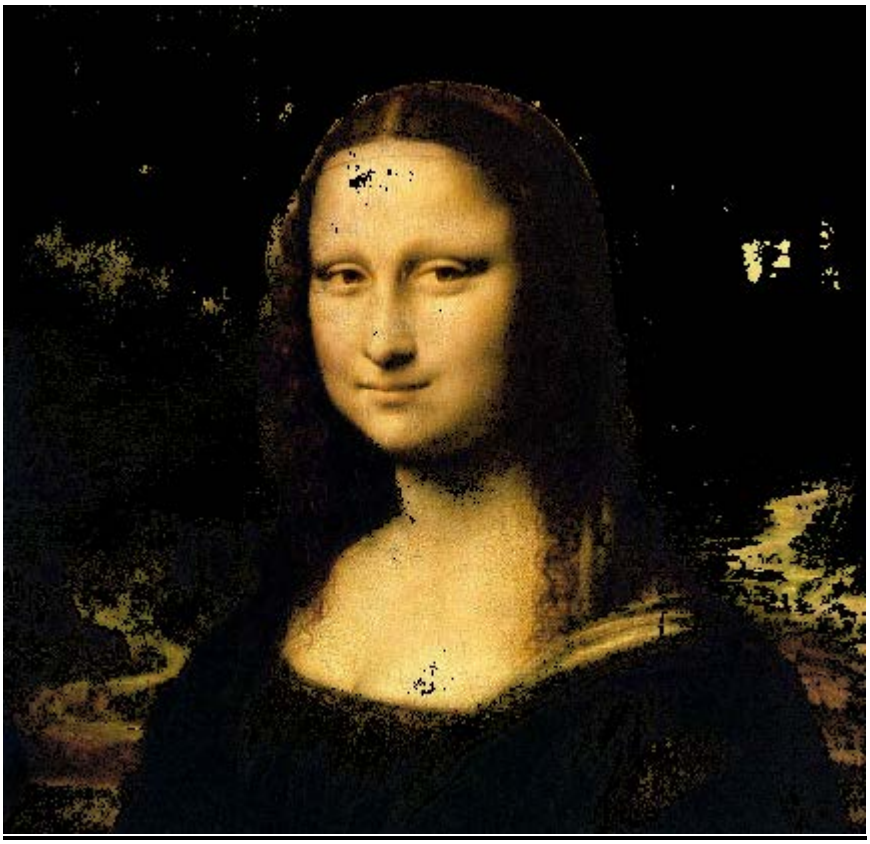
These are the results with using KMeans=64 cluster and Mona_lisa stroke 1



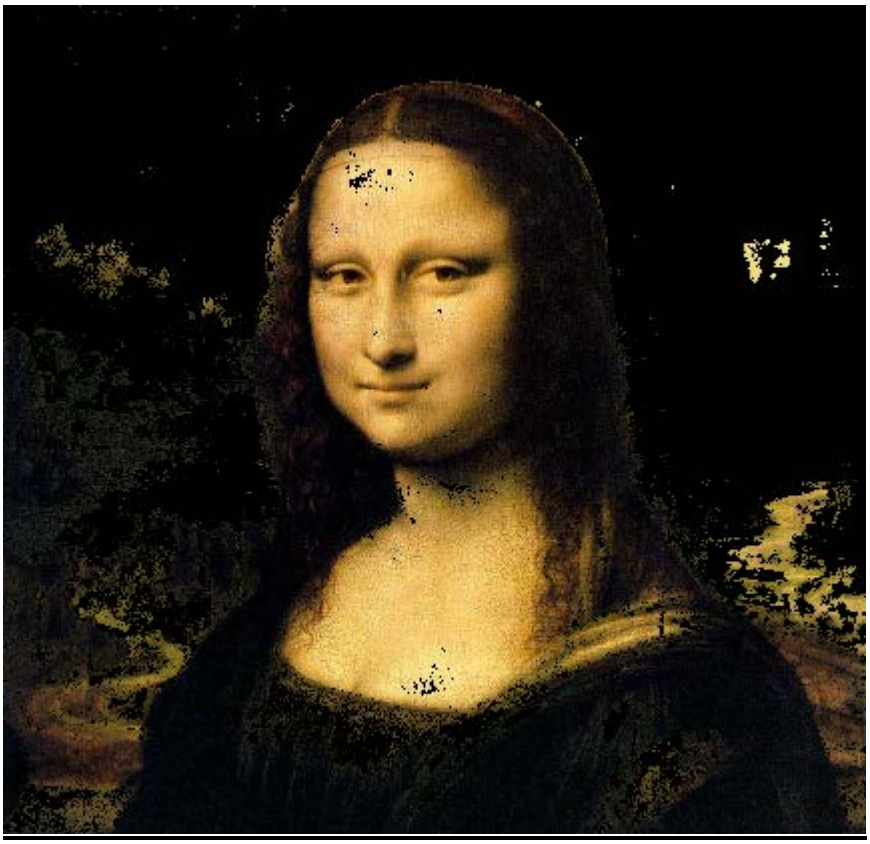
The above image is by using manual K means =64 with **Mona_lisa stroke 1**



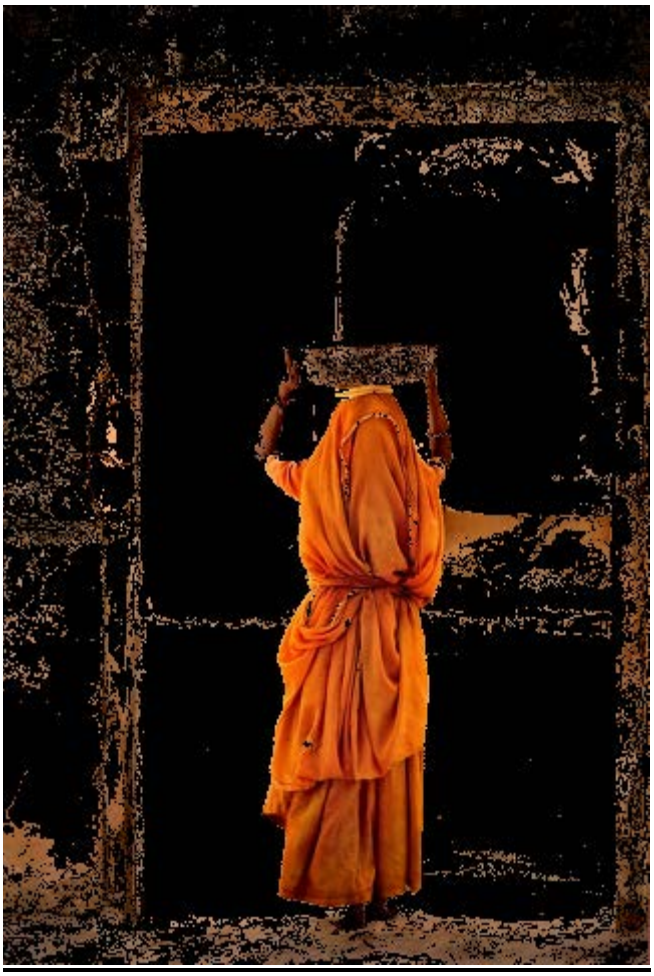
This image is calculated using in inbuilt KMeans=64 with **Mona_lisa stroke 1**



This image is calculated using in inbuilt KMeans=64 with **Mona_lisa stroke 2**

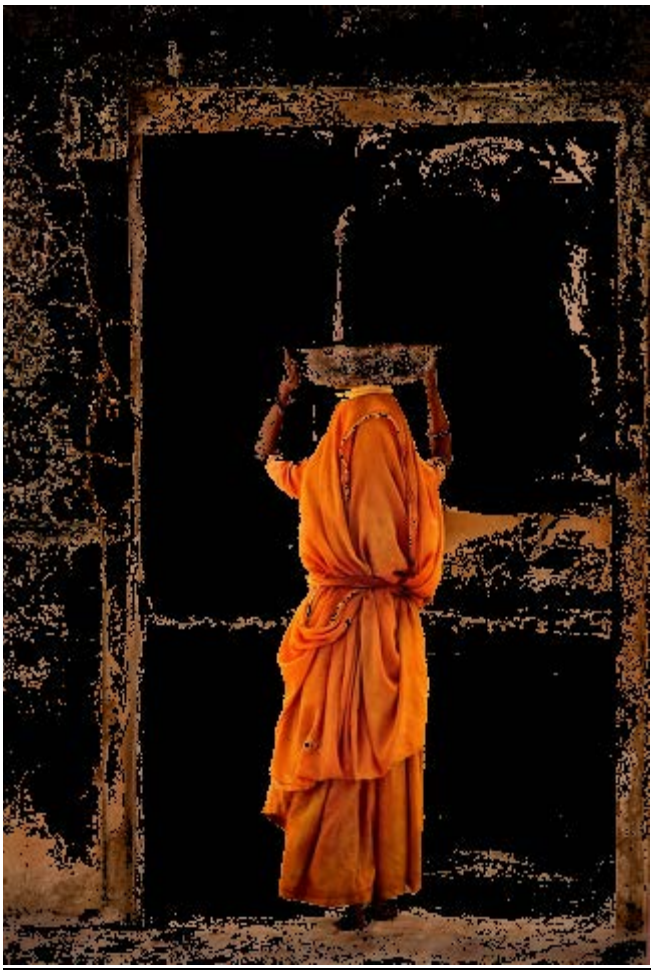


This image is calculated using in inbuilt KMeans=64 with **Mona_lisa stroke 2**



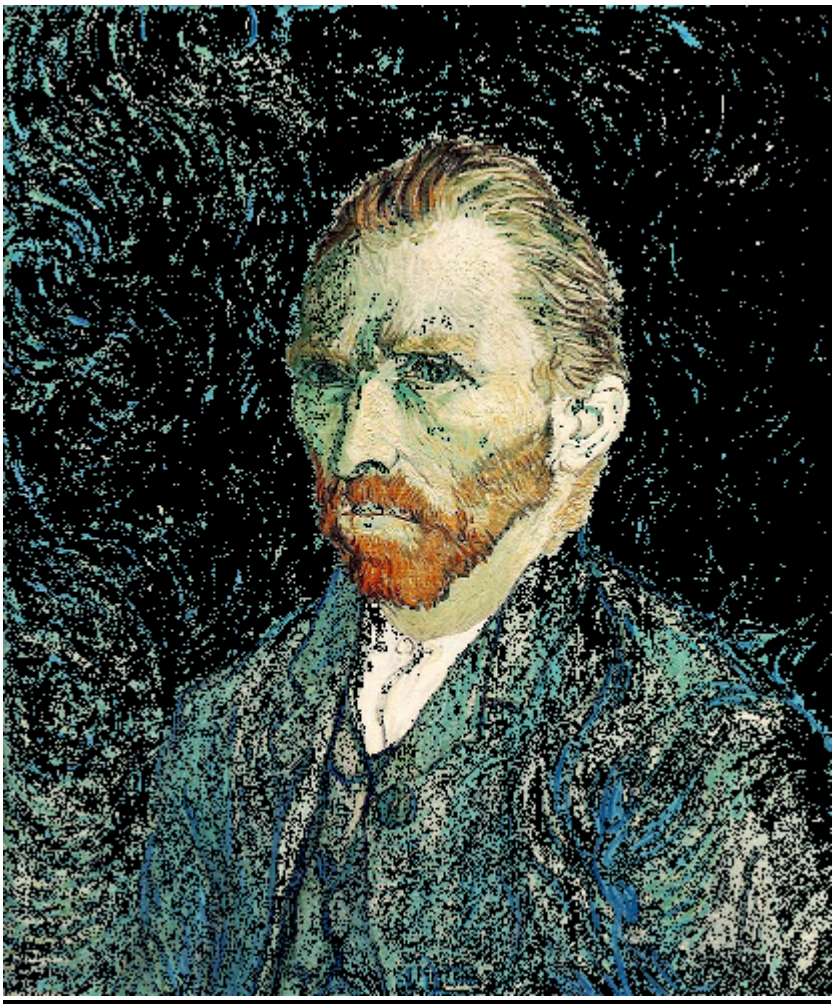
This image is calculated using in manual KMeans=64 with **Lady stroke 1**

This image is calculated using in inbuilt KMeans=64 with **Lady stroke 1**

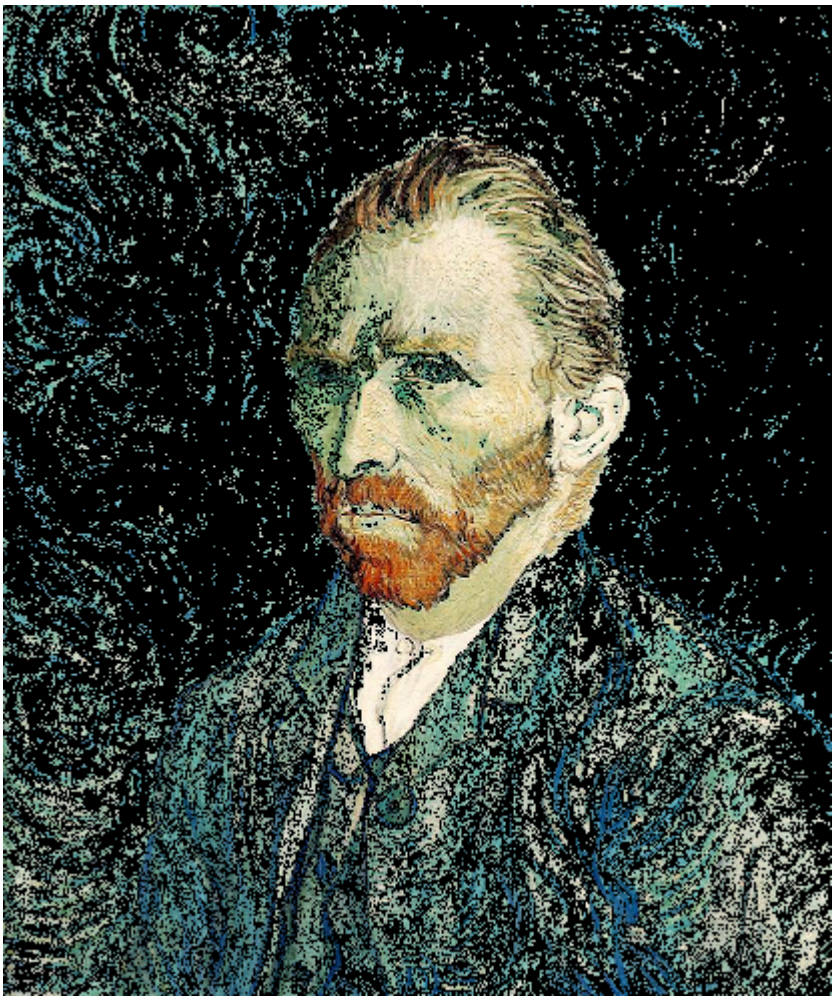


This image is calculated using in manual KMeans=64 with **Lady stroke 2**

This image is calculated using in inbuilt KMeans=64 with **Lady stroke 2**



This image is calculated using in manually KMeans=64 with **Van gogh stroke1**



This image is calculated using in inbuilt KMeans=64 with **Van gogh stroke 1**

Now I will use my chosen cluster number=1 which I took as a test and got some interesting results

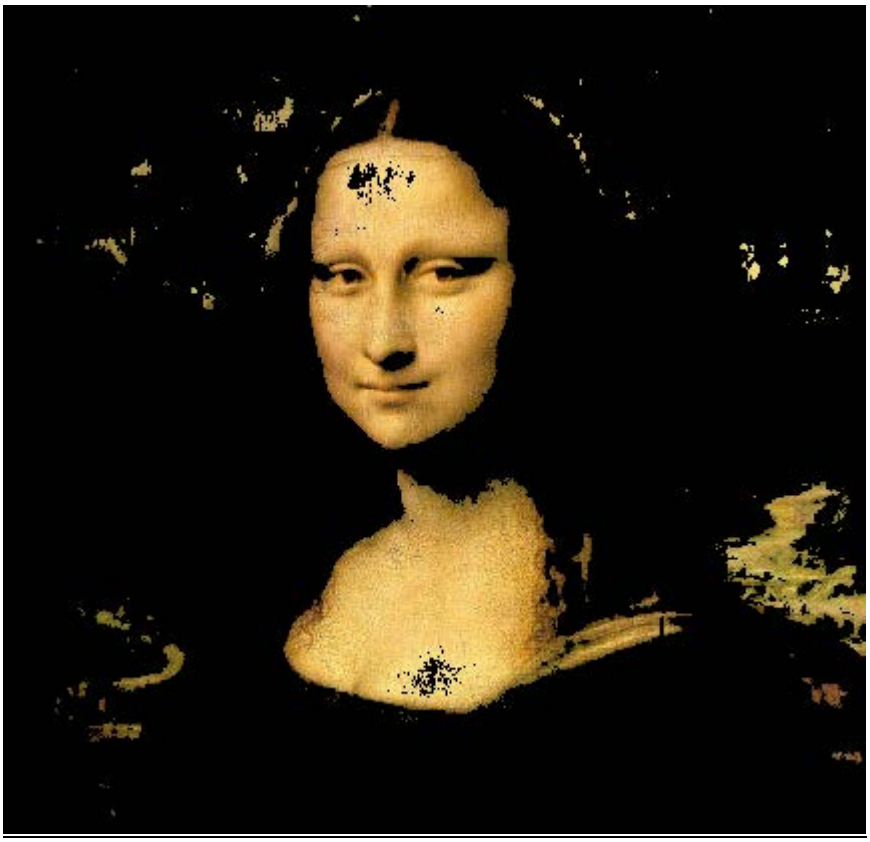
These are the results with using KMeans=1 cluster and Mona_lisa stroke 1

The above image is by using manual K means =1 **Mona_lisa stroke 1**

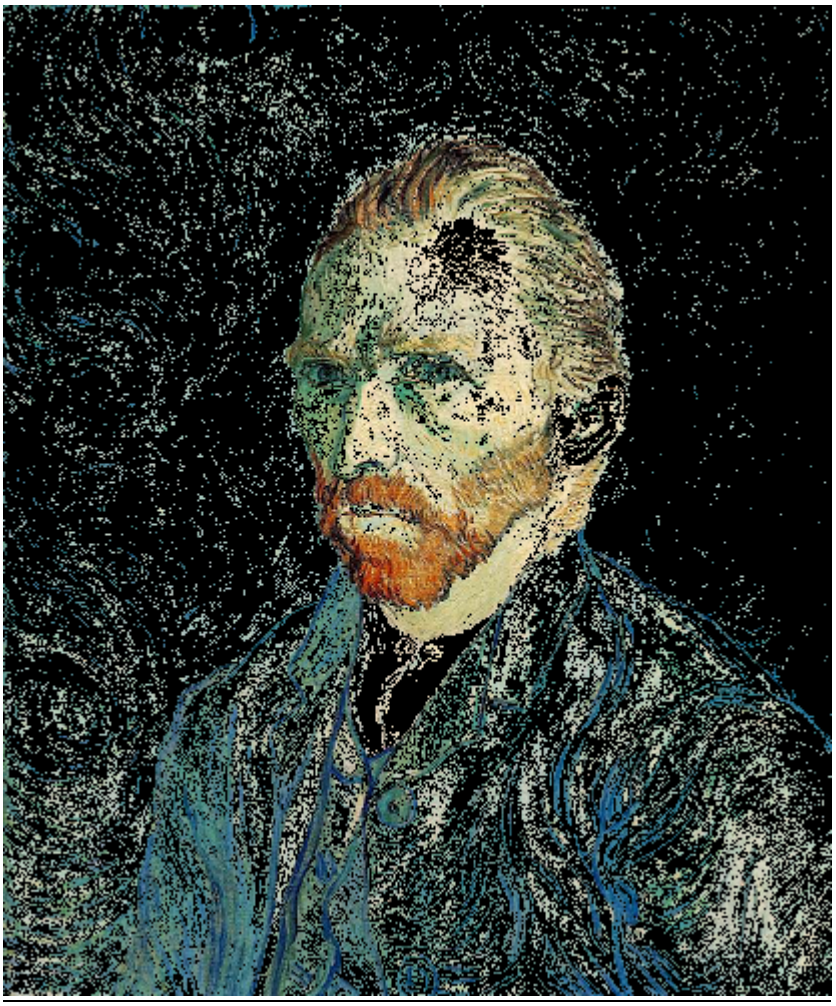
This image is calculated using in inbuilt KMeans=1 with **Mona_lisa stroke 1**

These are the results with using KMeans=1 cluster and Mona_lisa stroke 2

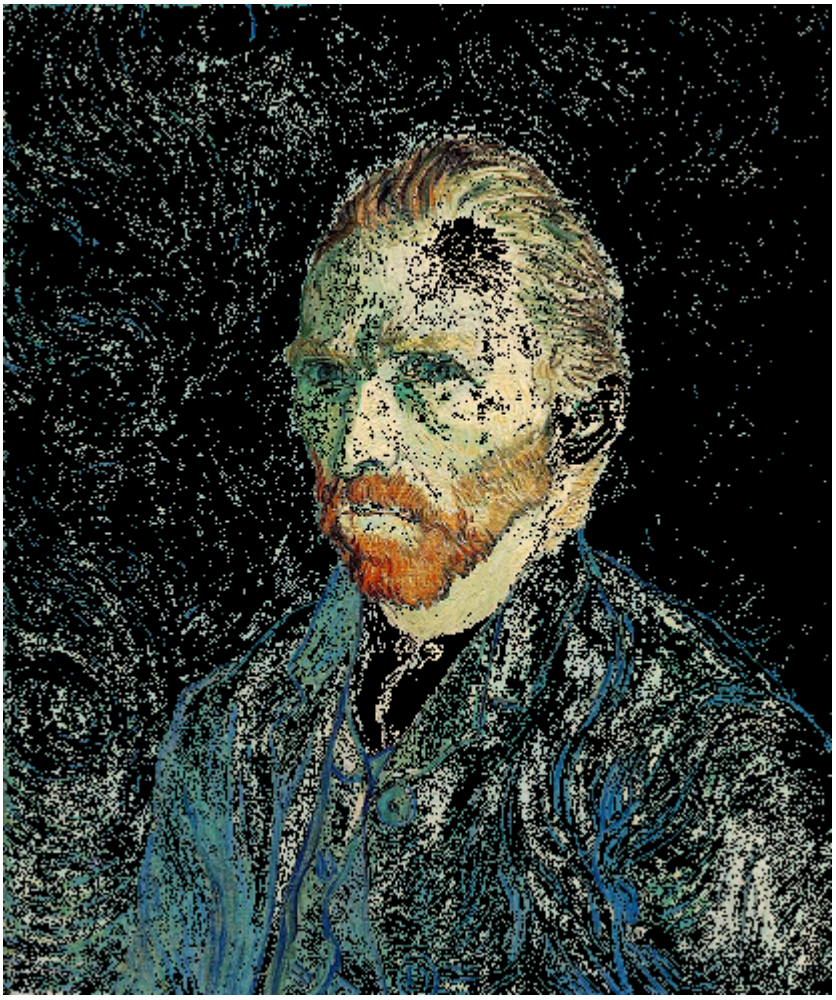
The above image is by using manual K means =1 with **Mona_lisa stroke 2**



The above image is by using inbuilt K means =1 with **Mona_lisa stroke 2**

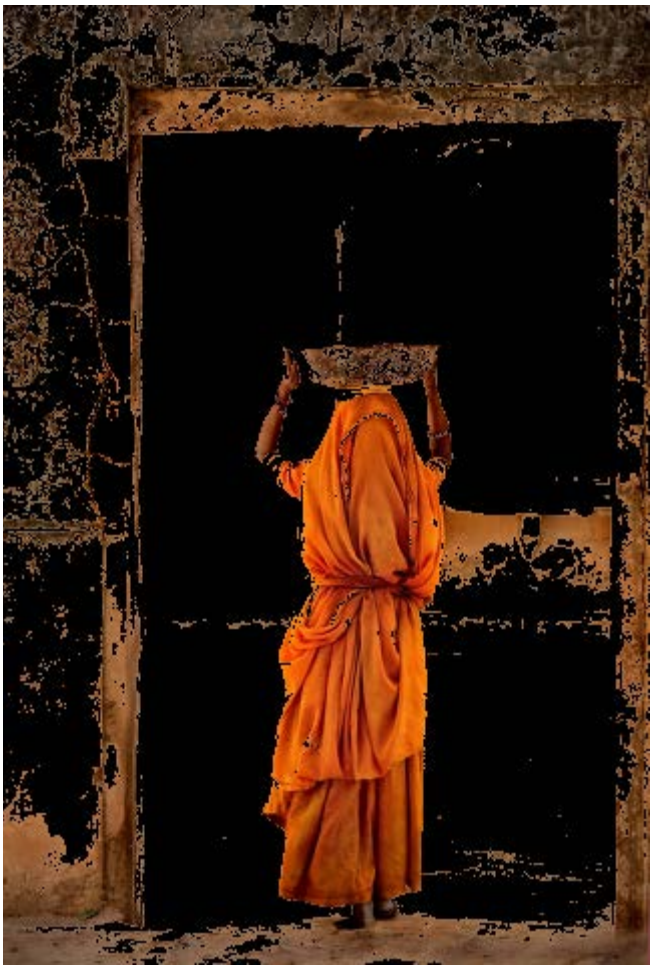


The above image is by using manually K means =2 with **Van gogh stroke 1**



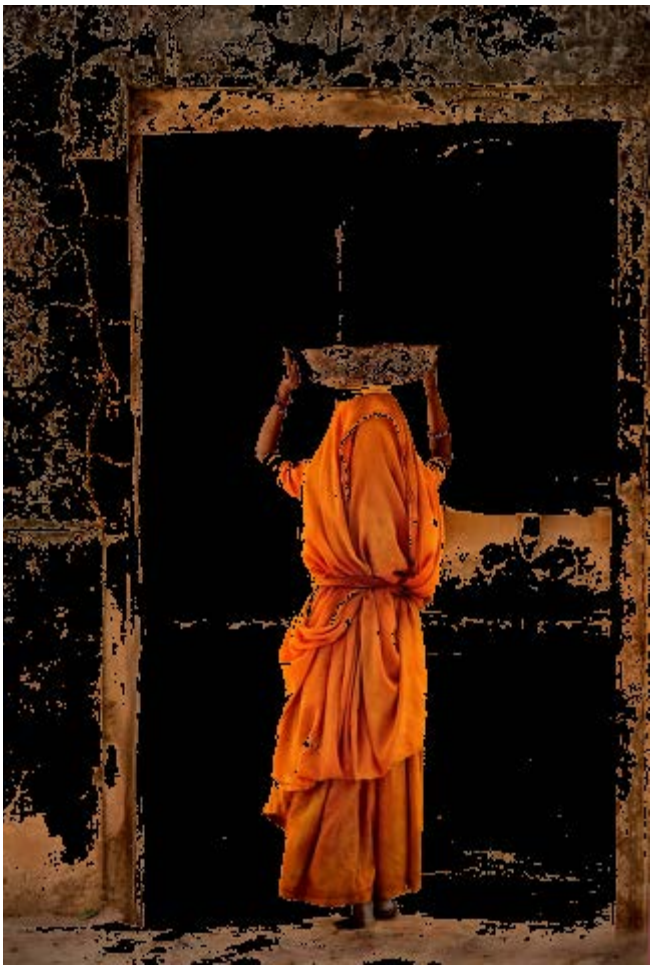
The above image is by using manually K means =1 with **Van gogh stroke 1**

The above image is by using manually K means =1 with **lady stroke 1**



The above image is by using inbuilt K means =1 with **Lady stroke 1**

The above image is by using manual   K means =1 with **Lady stroke 2**



The above image is by using inbuilt K means =1 with **Lady stroke 1**