Assignment 5

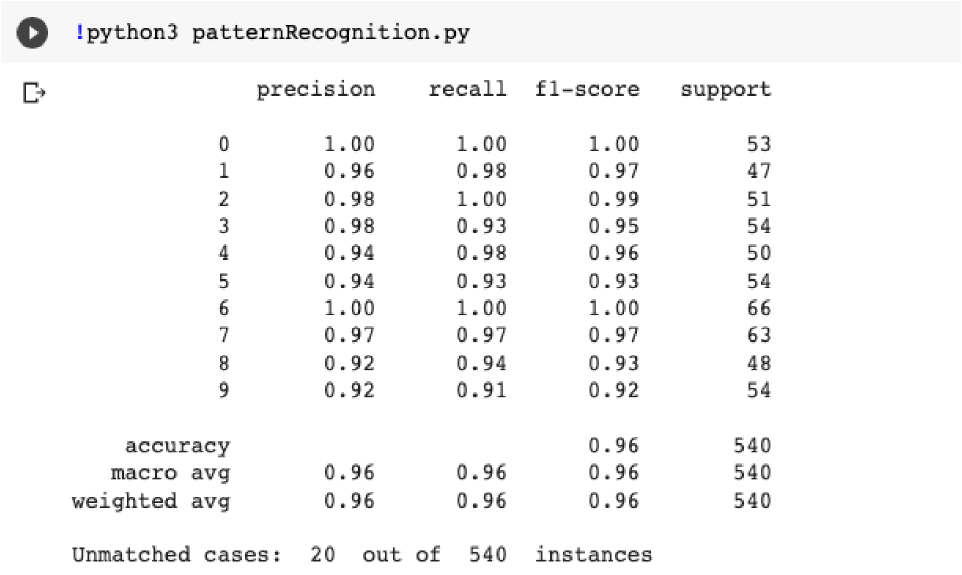
Name: Diwakar Yalpi

UIN: 01127089

Github link: https://github.com/diwakaryalpi/AI-Assignment5

The 32 * 32 pixel format has been transformed into 1024 using 1 vector and each pixel denotes each feature of a particular instance of digit. We have the class of each instance, and the original digit dataset is already given. I have divided the given dataset into test and train datasets, I have considered 30% as the test dataset and rest of the percentage of data as train dataset. I have trained the dataset so that the output can be predicted for the test dataset. I have used the neural network as the classifier, and I have used the python3 library sklearn for this purpose. I have used sklearn library as it is the standard library for machine learning.

I have divided the assignment looking at results without using Principal Component Analysis (PCA) in the 1st part and in the 2nd part used PCA. I have considered number of hidden nodes in the neural network as 1000. I am getting the below confusion matrix.



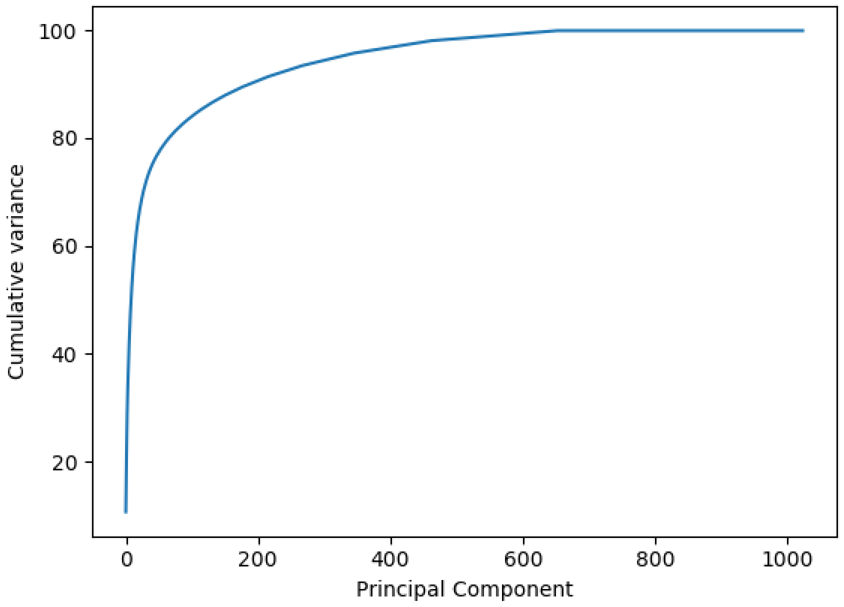
```
!python3 patternRecognition.py

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        53
           1       0.96      0.98      0.97        47
           2       0.98      1.00      0.99        51
           3       0.98      0.93      0.95        54
           4       0.94      0.98      0.96        50
           5       0.94      0.93      0.93        54
           6       1.00      1.00      1.00        66
           7       0.97      0.97      0.97        63
           8       0.92      0.94      0.93        48
           9       0.92      0.91      0.92        54

    accuracy                           0.96       540
   macro avg       0.96      0.96      0.96       540
weighted avg       0.96      0.96      0.96       540

Unmatched cases:  20  out of  540  instances
```

The average values of Precision, recall and f1-score are 96%. We must train the data furthermore so that the number of unmatched cases can be reduced and to increase the percentages of other parameters.

I have tried to analyze the results by considering the 1024 features for every instance without using PCA. Below are few observations:

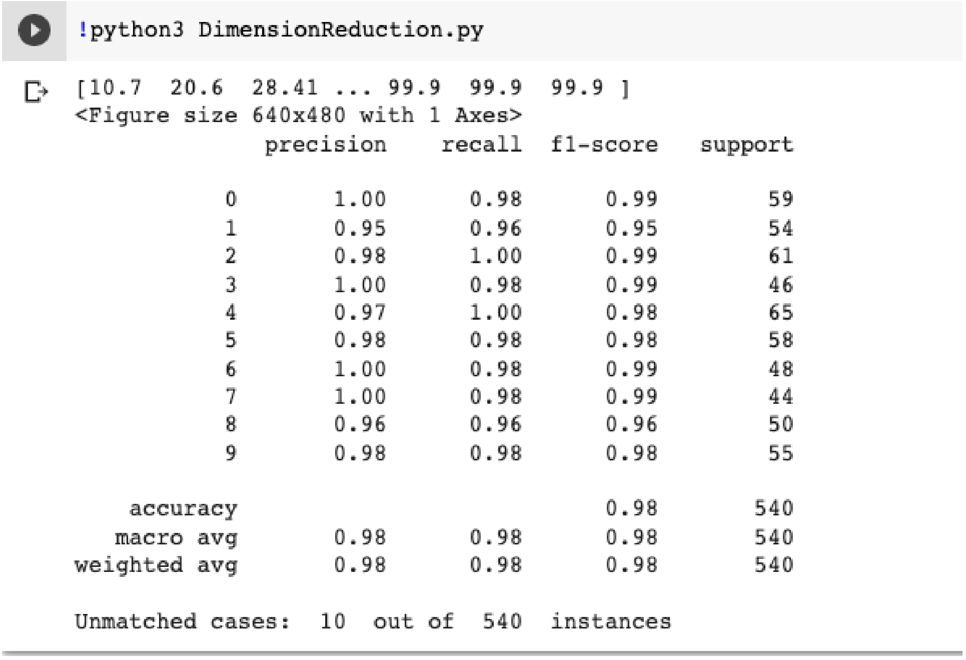
- Those are the number of hidden nodes from the above execution. I try to increase the number of nodes from 10 to 100 then the precision also increases from 87% to 96%. This lets us conclude that the selection of the number of nodes is highly crucial.
- Now I start increasing the number of nodes from 100 to 1000, the value of precision doesn't change drastically. The precision increases only by 1%.
- After particular point the change observed in the model is static even though I try to change number of nodes. There are other ways to increase the precision and accuracy such as removing outliers, scaling etc., changing the number of nodes is not the only way to increase the precision and accuracy.

Because there are ways to increase the precision percentage by reducing the number of features from 1024. Since considering all the 1024 features might not train the data well and also to decrease the number of unmatched instances. I use what we call in machine learning, the technique of dimensionality reduction. For this I am using the method Principal Component Analysis (PCA) out of many methods. PCA method changes the features into difference axes so that the variance is highest. I have considered the 1024 components from the above data and tried to implement PCA on the data. The covariance matrix has been plotted using matplotlib library that is being plotted with Principal component on x-axis (I.e., the range of values of each pixel) and cumulative variance on y-axis.


Cumulative variance ratio

Initially the cumulative variance value increase drastically until 100 on the principal component axis. By looking at the above graph I can conclude that the cumulative variance becomes constant once the number of nodes is increased from 100.

Below is the confusion matrix generated for after doing the dimension reduction using PCA method. I have used 1000 nodes for working on the given data.



```
!python3 DimensionReduction.py
```

```
[10.7  20.6  28.41 ... 99.9  99.9  99.9 ]
<Figure size 640x480 with 1 Axes>
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        59
           1       0.95      0.96      0.95        54
           2       0.98      1.00      0.99        61
           3       1.00      0.98      0.99        46
           4       0.97      1.00      0.98        65
           5       0.98      0.98      0.98        58
           6       1.00      0.98      0.99        48
           7       1.00      0.98      0.99        44
           8       0.96      0.96      0.96        50
           9       0.98      0.98      0.98        55

    accuracy                           0.98       540
   macro avg       0.98      0.98      0.98       540
weighted avg       0.98      0.98      0.98       540

Unmatched cases:  10  out of  540  instances
```

- From the above image I can say that implementing the PCA method on the given data, the precision has increased from 96% to 98%.
- The number of unmatched instances has reduced from 20 to 10.
- The other values like recall and f1-score values also have increased from 96% to 98%
- All the values had improvement when I used PCA method for dimension reduction. I am considering only components that have important weightage on classification.