Results of MNIST dataset training and tuning the hyperparameters

Model description

Initially, I have use previous XGBoost model and modified it so that it works for multiple classes. Then I calculated the accuaracy and F1 scores for the XGBoost.

Then I have done the same with KNN model. By comparing the two models, KNN ran much faster and also given good prediction with much better accuaracy and precision.

So I switched to KNN to model the MNIST dataset. I have normalized features in X and tuned the number of components in PCA and also added an Ensemble method Bagging. I have also used weighted distance metric for KNN for better performance.

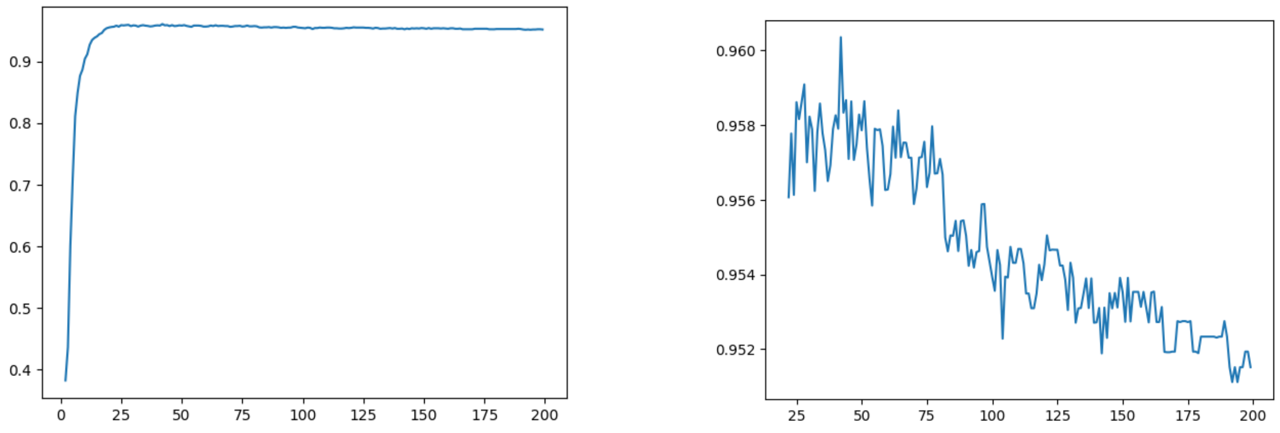
Tuning of Hyperparameters

Since bagging gives a noisy output each time, I tuned the hyperparameters using only normal KNN with PCA and then used them in Bagged KNN.

As the model have two simple numeric hyperparameters k value and number of components, I decided to run a for-loop and plot the results in the graph

Tuning of number of PCA components:

Below is the resulting graph for tuning number of PCA components



(a) full graph

(b) closeup

Figure 1: F1 scores for different number of PCA components.

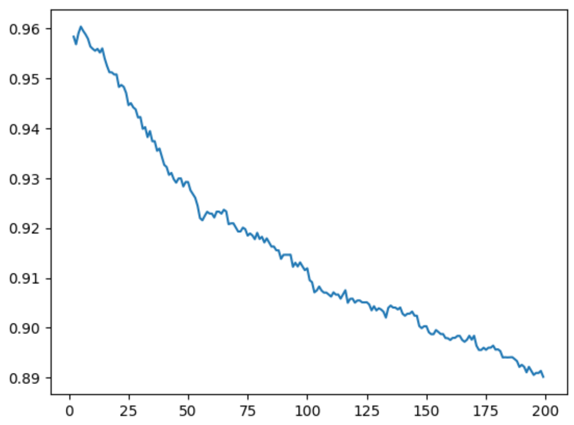
If we look at the graph we can see that model performs the best for **42 PCA components**

Tuning of k value = Neighbours

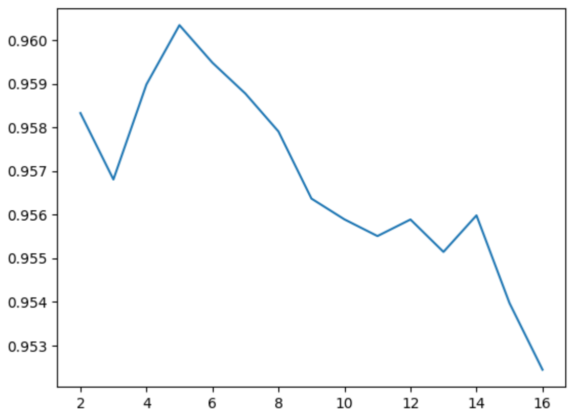
Similar to the above cases, as the k value are discrete numbers, we can write a for-loop for them to calculate optimal k value

Below are graphs for different k values between 2 to 200

So from the graph we can say that the optimal value for k for best F1 score is **k = 5**



(a) full graph



(b) closeup

Figure 2: F1 scores for different k values

Table 1: Hyperparameter Tuning for KNN with PCA

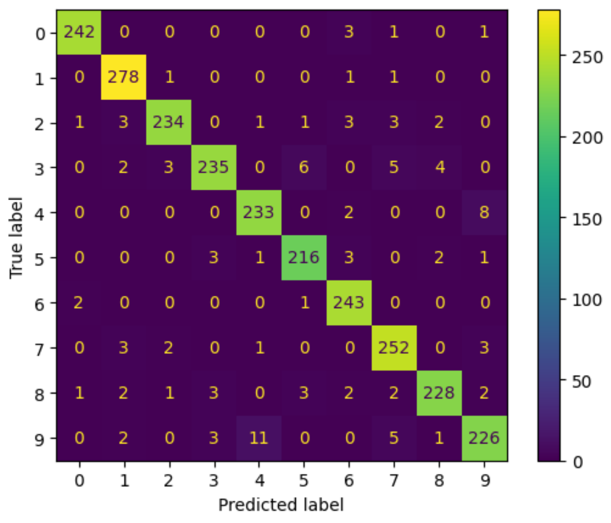
Number of PCA components	k value
42	5

Evaluation Results

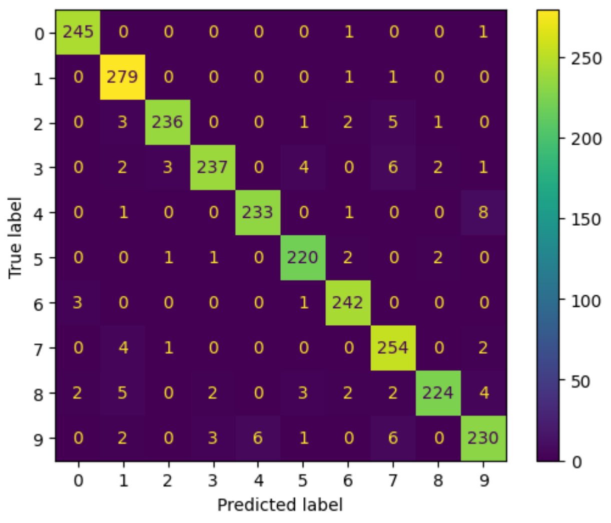
The accuracy and F1 scores of normal KNN with PCA but with random hyperparameters is 0.9551, 0.9548

The accuracy and F1 scores of tuned KNN with PCA is 0.9603, 0.9603

The corresponding Confusion Matrices are



(a) Confusion Matrix for KNN with no tuned parameters

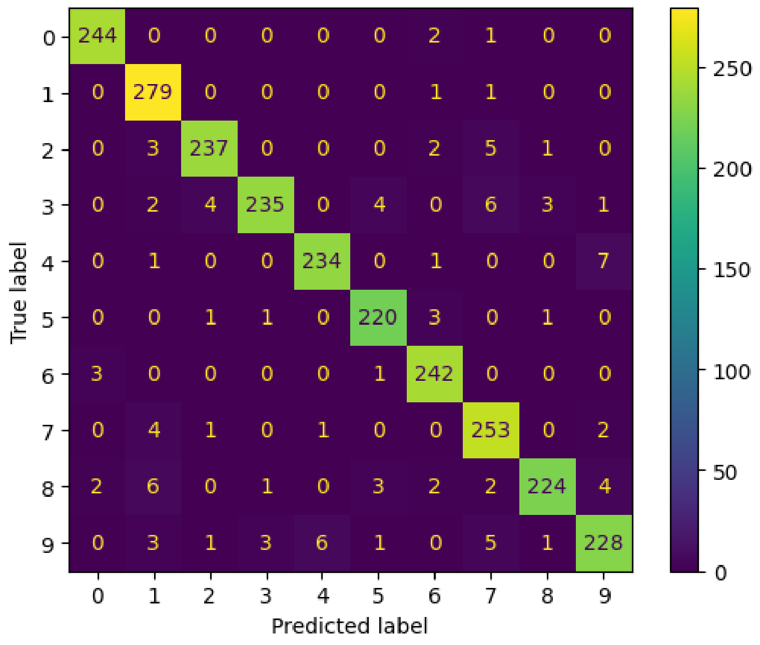


(b) Confusion Matrix for KNN with tuned parameters

Figure 3: F1 scores for different k values

The accuracy and F1 scores of Bagged KNN with tuned parameters are 0.9587, 0.9587 The corresponding Confusion Matrix is (figure 4)

So what I think about this is that even with bagging there is no particular use for KNN and bagging will not improve its accuracy much. This is because the KNN algorithm is already fast enough to run and bagging will not add much benefit to it.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	244	0	0	0	0	0	2	1	0	0
1	0	279	0	0	0	0	1	1	0	0
2	0	3	237	0	0	0	2	5	1	0
3	0	2	4	235	0	4	0	6	3	1
4	0	1	0	0	234	0	1	0	0	7
5	0	0	1	1	0	220	3	0	1	0
6	3	0	0	0	0	1	242	0	0	0
7	0	4	1	0	1	0	0	253	0	2
8	2	6	0	1	0	3	2	2	224	4
9	0	3	1	3	6	1	0	5	1	228

Figure 4: Confusion Matrix for Bagged KNN.

We can also see that more number of PCA does not improve the accuracy and very low number of PCA components also do not improve it. Only some particular values depending on the dataset will improve the accuracy