Project Handwritten Image Recognition

The project aims to implement a classification algorithm to recognize handwritten digits (0-3). The project implements multiple classifiers like Naïve Bayes, Decision Tree, K-nearest and Random Forest with cross-validation for better model selection and accuracy. The project also has an implementation of PCA. With the final classifier as RandomForest and PCA variation 97%, accuracy of 84% was achieved.

This project is based on handwritten digits (0-3) image recognition. The dataset contains 654 labeled images of 28*28 pixel handwritten digits. The dataset is split into 450 training and 204 testing data. There are 4 classes (0-3). Images are stored in respective class sub-folders.

Approach to solve this problem of handwritten digit recognition can be broadly divided into

Step 1 - Training data and Cross validation

- **a. Data Preprocessing**
  - **i) Read image and Pixel conversion**
  - **ii) Standardization of data.**
- **b. Feature reduction-using PCA.**
- **c. Run all algorithm using cross-validation (with range of parameter values)**
  - **i) Naïve Bayes.**
  - **ii) K-nearest Neighbors.**
  - **iii) Decision Tree.**
  - **iv) RandomForest.**
- **d. Run all algorithm with GridSearch (with range of parameter values)**
- **e. Conclusion.**
- **f. Summary**

Step 2- Choosing the best classifier from step -1 and predict the test data

- **a. Summary**

Attachment and References.

Details of each of the above point are as given below.

1) Step 1 - Training data and Cross validation

In step 1, we use only training data, using Stratified cross-validation where we have taken the value of k =5.

Stratified cross-validation randomly splits the data in a way to ensure each class is approximately equally represented across each fold i.e. each fold has the same proportion of observations. Therefore there less chance of bias and variance.

In cross-validation, training data is divided into multiple folds. We iterate on the data 1 fold at a time, where that particular fold is chosen as testing data and the remaining as training data. Therefore model gets a chance to be trained on multiple train-test splits. This way model gets to see unseen data, noise data and can also avoid overfitting of data.

As compared to the above, in the train-test split, data is divided into 80% training and 20% test data. There is, therefore, a chance that data is under fitted or overfit's the model

Therefore stratified cross-validation is the preferred choice for this project.

The classifiers are executed with permutation and a combination of their respective parameters on dataset transformed using different values of PCA variance (0.96 – 0.99).In the end, we have each classifier with its best parameter for a given value of PCA variance. This range of parameter values is further provided to GridSearch to get the best classifier with the highest accuracy score.

a) Data Preprocessing

i) Read image and Pixel conversion

Python's OS module is used to traverse through the directory and sub-directory to read the image files. Once the files are read from respective sub-folders they need to be converted in a format that can be written in the data frame. Therefore image files are first converted into numpy.ndarray of pixel values, a multidimensional array of the same size as of the image (28*28). Thereafter multidimensional array is flattened into series, which is now ready to be populated into the data frame.

The training data frame is this way is populated with rows having 784 columns with each row representing an image. The indexing happens in the order in which the files are read. As these images are been read, a corresponding list of

the labels is populated. The label is nothing but the name of the sub-directory in which that image is present.
At the end of this step, we have training data(X=training data and Y=labels) ready to be used for classification algorithms.

ii) Standardization of data

Although these pixel values can be presented directly to models in their raw format, this can result in challenges during modeling, such as in the slower than expected training of the model.

Instead, there can be a great benefit in preparing the image pixel values before modeling, such as simply scaling pixel values to the range 0-1 to centering and even standardizing the values.

Inputs with large integer values can disrupt or slow down the learning process. As such it is good practice to normalize the pixel values so that each pixel value has a value between 0 and 1.

It is valid for images to have pixel values in the range 0-1 and images can be viewed normally.

In this project, the data was standardized using the StandardScaler method.

We decide to reduce the dimension of the data using PCA (This process is explained in the next section).

The results below summarize the effect of feature reduction based on scaled and unscaled data.

Observation made from Fig a, b, c, d (**in the attached excel file –Step-1-Data-Graphs.xlsx, sheet =PCA-**Graph) is that there was no change in the PCA component with scaling and without scaling of the training data. The reason for no change is because standardization is normally done on the dataset with values of different units. In our case, the dataset values are of the same unit. Also, the only possible values of each pixel of each image are either 0 or 255. Hence the scaled value will also correspond to a value of 0 or 255. Thereby it is making no difference in the calculation of PCs.

b) Feature reduction-using PCA

PCA is a dimensionality reduction algorithm, used for both supervised and unsupervised algorithms. Dimensionality reduction concept is transforming existing features to the

meaningful reduced number of features. PCA achieves this by using a combination of the original feature that will give the largest variance, that is, it will cover the maximum variation in the data of the dataset. The first few sets of components explain the maximum variance, based on which the remaining can be discarded as they do not account for a lot of variability in the data.

Each of the PC variances has different counts of components as mentioned below in the tables.

PC Variance	Number of Component's
0.96	215
0.97	233
0.98	257
0.99	294

PC does not have the same density of information. As the number of components increases for a given dataset, there is more chance of overfitting of data e.g. with PC variance 0.99 which has 294 components, with the last component having a variance of 0.0002. This may result in the overfitting of the classifier model because of PCA transformed data. This overrides the use of PCA, which helps in reducing noise.

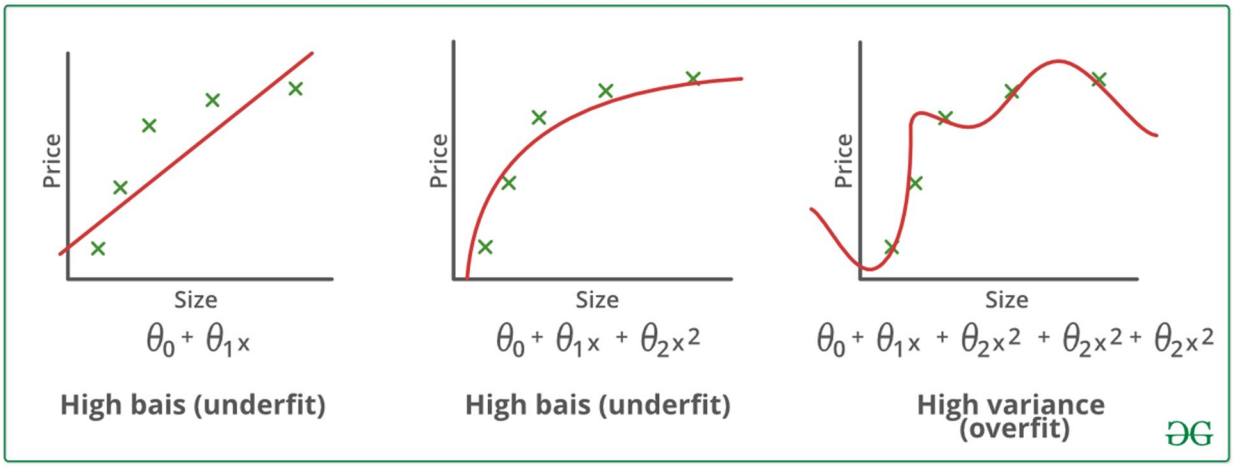
As we see from 0.96 to 0.97, there is an increase in 1% variance but this increases in 18 components where each component has additional negligible variance thereby indicating that these additional components can be insignificant. A similar pattern is seen with higher variance.

Therefore data is transformed with each of the PC variance and accuracy score for multiple classifiers is checked. The process for the same is followed below in the remaining sections.

c) Run all algorithm using cross-validation (with range of parameter values)

Multiple supervised classifiers were used, Gaussian Naïve Bayes, Decision Tree, K-nearest neighbor, and Random Forest to get the possible highest accuracy score.

To have the best training model, we need to ensure that the training model has understood the pattern of the data, also should be low on bias and variance ie that the model should not underfit data or overfit the data. This can be explained from the graph given below, how the model can be biased or have a high variance.



As seen from the graph, the first figure on the left is a line of a linear equation. The equation of the line is made in such a way to reduce the SSE. Therefore a line may or may not pass the data points. Thereby the algorithm is not able to learn the data well therefore they underfit the data or have high bias

Similarly, the figure in the middle is the quadratic equation where the curve may or may not pass through the data points resulting in a high bias .When compared with linear equation it is low.

Similarly, a more complex curve will pass through all data points. Thereby results in low bias but high variance or overfitting the data. The algorithm sometimes overgeneralizes in an away that they tend to induce high variance ie the model will also learn noise data.

Therefore the algorithm should be normally low bias and low variance as far as possible.

It should be able to predict data which it has never seen before, for this the complete data must pass through the model at least once. To achieve we k fold cross-validation with k=5.

This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in the validation set. Interchanging the training and test sets also adds to the effectiveness of this method

Whatever accuracy results are mentioned below in the subsection is on training data with cross-validation.

i) Naïve Bayes

Gaussian Naïve Bayes' is a supervised, conditional, probabilistic algorithm based on the assumption that features are independent of each other. It is applied based on

Bayes' theorem. Since we are using Gaussian we assume data to be normally distributed.

Accuracy score was 0.788, when run with Training data.

ii) K-nearest-Neighbors

KNN is a lazy learning algorithm, used to label a single sample of data based on similar known labeled examples of data. It computes the distance between the new data point with every data point of the training sample. For computing the distance it uses Euclidean distance (Other distance's measure are Hamming or Manhattan distance). Model pick 'K' entries from the training data point distance calculation which is close to the new data point. Then it does the majority vote i.e. the most common class of labels among the 'K' entries will be the label for the new data point.

Values of 'K' are preferred to be odd to avoid tie as it uses a voting technique to choose a label.

The range of knn values from (3-30, at an interval of 2) was tested.

Accuracy score of Knn =5 for PCA variance =97% was 0.751

iii) Decision Tree

The general motive of using Decision Tree is to create a training model that is used to predict the class or value of target variables by learning decision rules inferred from prior data (training data). There are multiple algorithms used to build a decision tree, but they all follow the principle of 'greediness', that is the algorithm tries to search for a variable that gives maximum information gain or divides the data in a most homogenous way in terms of class or label.

It uses either Gini or Entropy with the Information gain method.

Entropy is a measure of variability or randomness. It is used to calculate the homogeneity of the sample. If the sample is completely homogeneous the entropy is 0 and if the sample is equally divided then it has an entropy of 1.

Gini index, if we select 2 items from the population at random then they must be of the same class and the probability for this is 1 if the population is pure.

Information gain measures of how much information a feature gives us about the class. Features that perfectly partition should give maximum information. It measures the reduction in Entropy or Gini.

Given below is an explanation with an example and decision tree diagram to understand gini, entropy and information gain.

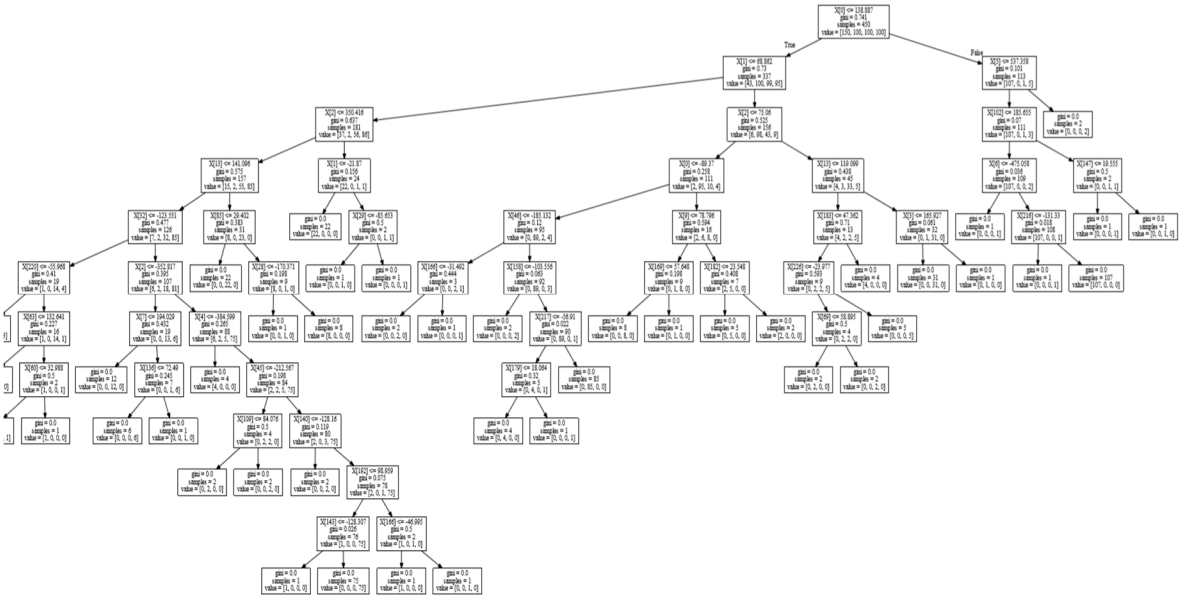


Fig – 1

To find out which attribute will be the root node, the decision tree uses either the Entropy or gini index along with Information gain. The purity or impurity of a given attribute is always concerning class or label.

In the gini index, we calculate the weighted child entropy of the attribute. Like Entropy, we calculate information gain by subtracting parent entropy - weighted child entropy. Attribute with the highest values is the information gain for Entropy. One that has the lowest entropy is the information gain for gini. Both methods give the same result.

The difference between the gini index and entropy is in their way of calculation. The calculation of the gini index is simple and faster. Entropy, on the other hand, makes use of log, therefore slower and resource consuming in the calculation of entropy.

Entropy has an edge in some data cases involving high imbalance. Since entropy is using log, values of lower probability are getting scaled up.

For example, if you have 2 events, one .01 probability and other .99 probability.

In the Gini index, the probability square will be .01^2+.99^2, .0001 + .9801 means lower probability does not play any role as everything is governed by majority probability.

Now in case of entropy .01*log (.01)+.99*log(.99)= .01*(-2)+ .99*(-.00436) = -.02-.00432 now in this case clearly seen lower probabilities are given better weight-age.

Entropy = $-\sum pk \log 2\ pk$

Gini Index = $1 - \sum pk2$

Weighted child entropy is calculated by adding proportionally branch entropy of a given attribute to get total entropy for that attribute.

In the decision tree, nodes are split based on information gain that is the difference between parent and child weighted entropy.

Splitting is always based on the value within the range of that attribute value.

Based on the above concept of building the decision tree, the below steps are carried out to for Entropy

- Initially, parent entropy is calculated for every distinct class of the output attribute.
- For every attribute, we calculated the entropy of every unique value with respect to the class.
- The entropy is weighted based on the number of records associated with the value of that attribute. This becomes are weighted child entropy for the attribute.
- Information gain is calculated by subtracting Parent entropy – weighted child entropy. Higher the value more the information gain.
- This way information is gain is calculated for all attributes.

The attribute which has the highest information gain other words attribute that will give purest nodes becomes the root node for the decision tree. As shown in the above diagram above **Fig -1**, X [0] columns is chosen as the root node with entropy (Information Gain) =0.741.

Every split of the tree follows the same process as above. This split process ends when the leaf node that's the last node is in pure form having one class or we have reached the max depth of the tree. As seen from the diagram **Fig -1**, each node has decreasing impurity meaning the leaf node of that branch will be pure or have a majority value of 1 class. E.g. At the end of the tree, column X [179] has entropy 0.32 with 5 samples, it breaks in 2 leaf nodes with both leaf nodes having entropy =0.0, purest form where the node is ruled by only 1 class. Similarly, as you can see in the diagram **Fig -1** many leaf nodes have entropy 0.0, where nodes have a value of the same class

For Gini Index, apart from calculating the entropy, we follow the same process as mentioned above.

The project used various parameters to tune the classifier so it has to have a maximum accuracy score. Multiple permutation and combination were used for tuning the model.

Attached file (at the end of the document) contains a table with permutation and combination of classifier parameter with a range of PCA variance

Parameter's that gave the best accuracy was

Criterion="gini",max_depth=12,min_samples_split=15

PCA variance =0.96

Accuracy score was 0. 751

iv) RandomForest

RandomForest is based on the concept of 'Ensemble' or group. In RandomForest multiple decision trees are generated (based on n_estimator values). This group of weak models is combined to form a powerful model and it unifies their result to obtain a good prediction. The approach of a voting mechanism from the aggregated prediction of all the trees gives a better chance of good and fairer predication. Also, rows and columns in each of these decision trees are randomly picked from the original dataset and a new dataset is formed, this technique is called bagging. This approach takes care of dimension reduction, helps take of noise data, and helps to reduce variance and overfitting of data. Thereby this approach can handle larger datasets with high dimensionality. Each tree gives a prediction. This is why the same result is not accomplished with the Decision tree.

Multiple decision trees are made using either gini index or entropy.

The project used various parameters to tune the classifier so it has to have a maximum accuracy score. Multiple permutations and the combination were used for tuning the model.

Attached file (Ste-1-Data-Graphs.xlsx at the end of the document) contains a table with permutation and combination of classifier parameters with ranges of PCA variance

Parameter's that gave the best accuracy was

N_estimator = 200 criterion="gini", max_depth=12, min_samples_split=11

PCA variance =0.97

Accuracy score was 0. 822

d) Run all algorithm with GridSearch (with range of parameter values)

Grid searching is a process of performing parameter tuning and determining optimal values for a given model. GridSearch can be applied across multiple classifiers to get the best classifier model. Each classifier is passed with multiple parameters with a range of values. It iterates through parameter combinations and values for each given classifier and stores the model for each combination. Finally, the classifier model with tuned parameters and optimal value giving the best accuracy score is determined.

GridSearch algorithm becomes slower in execution due to this but gives an accurate classifier model with a tuned parameter and optimal values at the end.

Based on this approach, all classifier algorithms used in this project were passed through the Gridsearch with multiple parameters and a range of values. The range of values was obtained while running the classifier algorithm individually with k-fold cross-validation

This process was performed across all PCA variance (0.96 – 0.99).

The best classifier model was generated with the RandomForest algorithm.

N_estimator = 200 criterion="gini", max_depth=12, min_samples_split=11

PCA variance =0.97

Accuracy score was 0. 835

e) Conclusion

As seen from above we have executed various classifier algorithms both individually and through gridsearch.

All classifier algorithms used in this project were run through stratifiedkfold =5 with different permutation and combination of parameter and range of values(Check the attached file for data and graph for each of the classifiers). Below table shows the accuracy score achieved by each of the classifiers across PCA variance (96% - 99%)

The best classifier model was RandomForest with an accuracy score of 83.5% for PCA variance 97%

Stratifiedkfold=5				
PCA	**0.96**	**0.97**	**0.98**	**0.99**
Naïve Bayes	0.788			
Decision Tree	0.751			
Random Forest		***0.835***		
K-nn	0.751			

A similar exercise was carried with CV=5 using gridsearch across all PCA variance (96% - 99%). The below table shows the best classifier model with tuned parameters and optimal values.

CV=5 with GridSearch					**Parameters of classifier's**
PCA	**0.96**	**0.97**	**0.98**	**0.99**	
Random Forest	0.831				Max_depth=10,min_sample_split=11,criterion= 'entropy,n-estimator=350
		0.835			***Max_depth=12,min_sample_split=11,criterion ='gini, n_estimator=200***
			0.835		Max_depth=14,min_sample_split=10,criterion= 'gini, n_estimator=350

				0.826	Max_depth=11,min_sample_split=10,criterion='gini, n_estimator=2000

As seen from the above observations RandomForest was chosen as the best model with the highest accuracy score of 83.5% for PCA variance 97 %.

Therefore by both the methods RandomForest classifier is the preferred choice for this project.

As part of the conclusion for step -1 below is probably justification and explanation for accuracy score achieved by each of the classifier models.

i) Naïve Bayes

The Naive Bayes algorithm has an underlying assumption that the data set we are working has attributes independent of each other GIVEN the output attribute. In this case, we are performing PCA on the original data set which transforms the data into components that are orthogonal to each other (linearly independent of each other) which may not be the same as the CONDITIONAL independence that Naive Bayes works with. This might be one of the reasons why the accuracy is low because the transformed data might NOT be really conditionally independent as more components are getting introduced (by increasing the accuracy) more might be the deviation from the requirement of conditional independence.

The other aspect which we have assumed is that the PCA transformed attributes are assumed to be following a normal distribution since we have used Gaussian Naive Bayes for the calculation of the conditional probabilities. The non-conforming normal distribution mean and standard deviation be therefore not resulting in accurate probabilities.

ii) K-nearest Neighbors

As seen from the attached file (Step-1-Data-Graphs, sheet = knn Data & Graph), knn was run with a range of (3, 30,2) at an interval of 2 e.g. 3,5,7.27, etc.

Best accuracy score is for knn=5, PCA=0.97 score=0.751.

As seen from the runs of the kNN with different values of k on different k on data which was transformed with different levels of variation. After k =5, it is observed that there is a decrease in accuracy as the k increases when we decide to retain either 96%, 97%, 98% or 99% variation in the original data. Another interesting

observation is that for 98% and 99% variation in the data, there is a decrease in the accuracy for the same value of k. Therefore the conclusion can be made that we decide to incorporate more variation in the original data, the Euclidean distance calculation sees an effect to the extent that it affects the classification slightly. Thereby it indicates that the components at the end, although comparatively small in value have affected to infer from the relevant neighbors.

The dropping accuracy for increasing k value is a clear indication of a possible underfitting where looking at more neighbors is inconclusive in making a decision. Although we get the same accuracy of 0.751 for 96% variation with k = 3 and 97% variation with k=5, to avoid the possibility of overfitting by including outliers, we have chosen k =5 with retaining 97% variation in the data

iii) Decision Tree

Analysis of the decision tree was done with and without tuning parameter. Analysis done without parameters with PCA variance = 0.96 gave accuracy score of 0.49, where max_depth was 12 and min_sample_splilt = default (2). This low score can be the reason for the leaf node being pure with very few samples, thereby implying that the model is learning noisy data. These parameters were taken as the benchmark and further pruning was done.

As seen in the attached excel, a range of PCA variance (0.96 – 0.99) were taken. For each variance, multiple permutations and combination with 3 parameters a) max_depth b) min_sample_split c) criterion was used to get the best accuracy score. Excel sheet contains the data and graph for the combination mentioned above.

PCA Variance = 0.97

PCA variance = 0.97 give the best accuracy score of 0.742 with max_depth =10, criterion = 'entropy' and min_sample_split = 17.This combination of parameters was chosen considering that the original tree with entire data included was of max_depth = 12 and min_sample_split =2.

The min_sample_split chosen indicates that we were still able to obtain high accuracy, indicating that nodes with the number of samples < 17 were probably having a majority of 1 particular class only. Thereby it indicates that the number of misclassification was negligible.

A similar good score of 0.751 was achieved with max_depth =12, min_sample_split =15, PCA = 0.96, criterion = Entropy. It was observed in the tree diagram **Fig -1** that

the nodes at the level 11 and 12 were split based on the previous level node at depth=10 which already had a high majority of one class value. If we would go further splitting beyond this point can be an indication of overfitting of the model with leaf nodes having 1 or 2 samples as seen in Fig-1

Similar score of 0.747 with max_depth =8 and min_sample_split = 17, PCA = 0.96 was obtained. Tree depth is less for the model to learn data therefore; this can result in the underfitting of the model.

A score of 0.747 with max_depth = 6 and min_sample_split = 3 was also observed but this could result in the model being under fitted.

Though min_sample_split =17-20 with PCA = 0.96 gave accuracy of approx. avg of 0.749, however for this score the model can be under fitted will max_depth is < 10.

PCA Variances = 0.96, 0.98, 0.99

As seen from the data and graph [sheets = DT-Data and DT-Graph] in the excel file attached, for max_depth =12, 14 there is a low accuracy score. Tree depth may be leading to overfitting. The min_sample_split of 2 and 3 gave an approx. accuracy score in the range of 0.702 – 0.744, can be a result model learning noisy data or overfitting of the model.

max_depth=14 and min_sample_split = 17 gives a score of 0.740 for PCA variance 0.99, but it takes 294 features to give a little good accuracy. This combination of parameters could also be potentially used for the final model however we did not go ahead with this because the tree would take more time to train with the increase in the number of the features**.**

iv) RandomForest

Attached file (sheets = RF-Data and RF-Graph) contains permutation and combination of various parameters used for tuning the classifier. As seen the best score is

N_estimator = 200, criterion="gini", max_depth=12, min_samples_split=10, PCA variance =0.97, Accuracy score=0.822.

This shows that a combination of these parameters gave a good score and leaf nodes may be pure or have majority values of one particular class.

Though an accuracy score of 0.836 was achievable with (n-estimator =150 max_depth =12, min_sample_split =3, criterion ='gini', PCA =0.96) this would result in overfit of the model, that model would also learn noise data.

At the same time accuracy score of 0.836 was obtained with (n_estimator = 350, max_depth =14 and min_sample_split =10, criterion = 'gini') but this would result in more processing time with increase in number of tree and depth of tree.

Though the accuracy score with n_estimator =150 is better the project will go ahead with n_estimator =200. The bagging approach of Random Forest is the reason for n_estimator=200.

With a lesser number of trees as in the case of n_estimator =150, the number of subsets of data formed with different combinations of row and column will also be lesser. Therefore the spread of variation of data is also less. This increases the chance of overfitting of data. When compared with n_estimator =200, with more trees there is a better chance of spread in the variation of data. This also decreases the chance of overfitting data. Therefore more chances that the final prediction is unbiased and fairer because of the variation in the trees formed by these subsets of data.

At the same time accuracy score of 0.836 was obtained with (n_estimator = 350, max_depth =14 and min_sample_split =10, criterion = 'Gini') this would result in more training time with an increase in the number of trees and depth of the trees. But since we need to balance out between accuracy score and training time, training time was given preference in this case since the gain in accuracy wasn't very significant.

f) Summary

As seen from above we have executed various classifier algorithms both individually and through gridsearch.

By both the methods Random Forest was the preferred choice achieving the highest accuracy score for PCA variance 97%.

The principle approach of the algorithm makes it the preferred choice for the dataset used in this project.

Random Forest is an ensemble learning method which generates multiple weak models and unifies their result to obtain a good prediction. The approach of using a voting

mechanism from the aggregated prediction of all the trees gives a better chance of good and fairer prediction.

Since this model uses multiple decision trees it becomes an ideal choice for the dataset with high dimensionality as in our case.

The Bagging approach of the classifier reduces variance which is quite high in the kind of dataset project is using. This also means it helps to reduce the chance of overfitting of the data.

To summarize, RandomForest is the best model for this project. With this approach, it is further used in step-2 with an increased size of training data and predicted for test data.

2) Step 2- Choosing the best classifier from step -1 and predict the test data

a) Summary

RandomForest is the best classifier as seen above. With this classifier, the final execution will be carried out. Training data now consist of 450 images of size 28*28 pixels with digits ranging from (0, 1, 2, and 3). Test data consist of 204 images of size 28*28 pixel with digit ranging from (0, 1, 2, and 3).

Both training and testing data are transformed with a PCA variance of 97%.

Random Forest algorithm is executed with below parameters to generate a training model

(max_depth=12, min_sample_split=11, criterion='gini', n-estimator=200)

Post this using the training model is used to predict the accuracy of the validation data.

Accuracy score of 85.7% was achieved post execution.

Attachment and References

Handwritten-Supervised.xlsx

http://cs229.stanford.edu/proj2010/Kim-ClassificationofHandwrittenDigitsbytheSetofPartialLinearandQuadraticModels.pdf

https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/

https://www.ritchieng.com/machine-learning-dimensionality-reduction-feature-transform/