

# Detailed analysis on performance of decision trees and kNN

Vikram Kalta (2209281)

*MSc Data and Artificial Intelligence*  
*University of Wales Trinity Saint David*  
Swansea, UK

**Abstract**—This paper focuses on evaluating two important classifiers in machine learning with the help of Fashion-MNIST data-set which are kNN and decision tree. In the process some relevant previous research papers are also reviewed in the literature review section to understand different perspectives of the authors on these classifiers. The experiments are carried out with various parameters and confusion matrix is used to evaluate the performance for each of the classifier. The goal is to come up with the best pair of parameters to improve accuracy rate and also to understand the effect of each of the parameters on the classifiers.

**Index Terms**—k-NN, Decision trees, Machine learning, Fashion-MNIST

## I. INTRODUCTION

Machine learning is a branch of computer science that broadly aims to enable computers to "learn" without being directly programmed [1]. It originates from artificial intelligence movement of the 1950s and focuses on prediction and optimization. A machine learning model is a mathematical function that takes input variables and then produces an output. Further adding to it, a predictive modeling in machine learning is the problem of developing a model using historical data to make a prediction on new data. The job of the modeling algorithm is to find the best mapping function we can from input to output variables. Machine takes two basic approaches: supervised learning and unsupervised learning [2]. The main difference is one uses labeled data to help predict outcomes, while the other does not. Supervised learning uses labeled data-sets to train the machine learning model. Using labeled inputs and outputs, the model can measure its accuracy and learn over time. Unsupervised learning uses machine learning algorithms to analyze and cluster unlabeled data-sets. The algorithms try to find the hidden patterns in the data to make a prediction. Supervised learning is further divided into 2 types of problems namely classification predictive modeling and regression predictive modeling. Classification predictive modeling is the task of approximating a mapping function from input variables to discrete output variables. The classification predictive modeling has applications such as spam filters in email, credit card fraud detection etc [3]. Regression predictive modeling is the task of approximating a mapping function from input variables to a continuous output variable. A continuous output variable is a real-value, such as an integer or floating point value. This paper explores 2 machine learning

algorithms namely k Nearest Neighbors and Decision Trees for classification. The experiments carried out uses Fashion-MNIST dataset to train and benchmark the models. The MNIST dataset comprising of 10-class handwritten digits, was first introduced by Yann LeCun in 1998 [4]. MNIST dataset is used for benchmarking machines learning algorithms. MNIST comprises 70,000 grayscale images in the size of 28\*28. Due to its simplicity the Fashion-MNIST is preferred which has same number of images and dimensions. Fashion-MNIST is based on Zalando's website. Every fashion product on Zalando has a set of pictures shot by professional photographers. It uses the front look thumbnail images of 70,000 unique products to build the data. Those products come from different gender groups: men, women, kids and neutral. The ten categories of Fashion-MNIST are as follows:

- T-shirt/top
- Trouser
- Pullover
- Dress
- Coat
- Sandals
- Shirt
- Sneaker
- Bag
- Ankle boots

This paper will test, evaluate and compare the performance of the 2 ML algorithms on the Fashion-MNIST using various metrics which will be explored in the later sections.

## II. LITERATURE REVIEW

Over the past decade, there has been a growing interest in the use of machine learning techniques to improve accuracy and efficiency of currently automated prediction based systems. There has been a lot of classifiers that are in practice and deployed for real world scenarios. The k-NN classifier is one of the computationally feasible and easy to implement [5] classification methods which is sometimes the best choice for machine learning projects. During the 70's, kNN classifier was studied extensively and some of its formal properties were investigated. This section reviews the most relevant studies in this area. A study by the authors of [5] experiments kNN algorithm using MNIST Digits data-set. The authors intend to remove the use of pre-processing and wants to find if

without any pre-processing the accuracy can be increased over the normal application of kNN. The authors have relied on euclidean distance metrics to get the distance between 2 points on a plane. The authors are concerned with the performance degradation of the model due to the distance between 2 alike digits being greater than the 2 non-alike digits since similar digits can have various shapes and orientation in the data-set. Thus the pre-processing the data can help to mitigate such classification errors and consequently improve the models performance. The authors have come up with a smarter solution of calculating the distance between test and training data points i.e [60000\*10000] before we train the model and store them in a sorted list. This strategy can help tremendously improve the run times. Authors have conveyed through graphs that the distance between 2 alike digits is less than 2 non-alike digits but when it comes to digits like 8 and 0, the distance between 2 non-alike digits can be small. The authors have chosen confusion matrix as their evaluation metrics. Further they use k-fold cross-validation to obtain the best value of k. The authors experiments with values of k-fold from 1 to 10, and for each of the k-fold value kNN model is run for values from 1 to 10. The authors then finds 3 as the best value for k as it shows nearly 97% accuracy using confusion matrix. The problem with this approach is that it is a expensive approach when it comes to time complexity since cross validation is run 10 times and kNN is run 10 times again. Cross validation is in itself a very expensive operation depending on the value of k-fold. The decision tree classifier is used for addressing challenges of classification and regression. The model creates a tree from learning the given data-set to classify the unknown class. It is a hierarchical exemplification of knowledge relationships that contain nodes and connections [6]. Where relations/edges are used to classify and nodes represents purposes. There are several types of decision tree algorithms such as: Iterative Dichotomies 3 (ID3), Classification and Regression Tree (CART) etc. Entropy is employed to measure a data-set's impurity. The values of entropy always lies between 0 and 1. Closer the value to 0 the better.

TABLE I  
BENEFITS & DRAWBACKS IN DECISION TREE

| <i>Benefits</i>                                    | <i>Drawbacks</i>                               |
|----------------------------------------------------|------------------------------------------------|
| Simple to comprehend                               | Can lead to overfitting                        |
| Classifies both categorical and numerical outcomes | Complexity increases for more training samples |

Table 1.

The authors in [7] experimented decision trees on mnist digit dataset and got the accuracy of around 90% which is found using a confusion matrix. In [8] author uses handwritten digit data-set from kaggle to evaluate the performance of decision trees. The author splits handwritten digit data-set into 21000 each. One for training and one for test. Author conducted the experiment and is able to get the results to a satisfied rate. The accuracy for the classifier for most of the digits is around

83%, for 1 it's 93.73% and for 8 is 84.12%. Overall accuracy was 83.4%.

### III. EXPERIMENTS

This paper works with 2 ML classification models which are kNN and decision trees. For experimenting the script uses python programming language. Fashion-MNIST data-set which is categorized into 10 different types is used for training the models and then testing the models. The confusion matrix will be later used to evaluate the results of both the experiments.

#### A. Training kNN Classifier

It starts with experimenting kNN algorithm. The data-set is downloaded and stored locally from kaggle's website. The data-set for training and testing is downloaded separately and contains 60000 and 10000 rows/samples for each respectively. The script uses pandas library to load the dataset into the memory for the program. Pandas is a library in python which has utility functions for reading a csv file. It then converts the csv to a dataframe. Next, as part of cleanup phase duplicates are removed. The script then gets an insight into the dataset by exploring the unique labels. As part of experimenting with kNN classifier the script uses sklearn package and imports the following modules:

- Pipeline - Pipeline [9] is a function in python which accepts tranformer functions and then expects the estimator as a final argument. The pipeline executes the each of the steps passed as a argument to the function in the same sequence as they are passed.
- MinMaxScaler - MinMaxScaler [9] is used for normalizing the data which takes the column as an input and scales each feature individually mostly between 0 and 1.
- Cross validate - Cross validate plays a major role in training the model. It's explained in depth in the later part.
- KNeighborsClassifier - KNeighborsClassifier is the classifier itself. For selecting an optimal value of k the script uses a range of 1 to 10 and validates the training data against each of value of k. In each of the iteration the test error rate and train error rate is stored in a data structure. Then this stored data is then used to plot a elbow graph to check the lowest error rate in the values of k and hence selecting the one with lowest error rate.

The model also uses cross validate method and its working is explained below. For example, if a model is trained and evaluated using 10-fold cross-validation, the train\_score would be calculated as follows:

- The training data is divided into 10 folds.
- The model is trained on 9 of the folds and evaluated on the remaining fold (iteration 1) which is nothing but the validation data-set. The model's accuracy on the training data is recorded.

- This process is repeated until the model has been trained and evaluated on all 10 folds. Each time the validation set is different from the previous one.
- The train\_score is calculated as the mean accuracy of the model on the training data across all 10 iterations.

An elbow graph for this part of experiment displaying test set error rate captured from cross validate method is shown in figure 2.

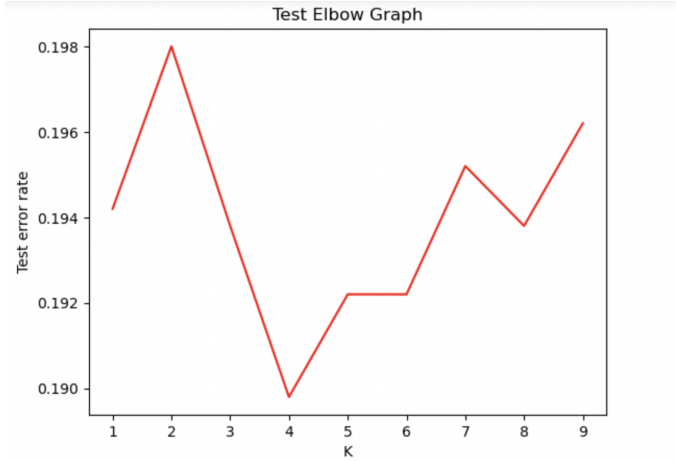


Fig. 1. Elbow graph for test error rate

It is evident that the  $k=4$  has the lowest error rate. Hence the final model is trained with  $k=4$  and again tested against training data to test the performance. The matrix is used to visualise the results of the confusion matrix. The similarities between train error rate and the test error rate says that the model is generalised well.

Finally, it predicts the value for the test data-set and again using the confusion matrix the result is displayed. An image for displaying the confusion matrix is given below:

### B. Training Decision Tree Classifier

Next, this paper explores decision tree classifier performance on the same dataset. The data loading and cleanup process remains same. The script uses pyplot module to display the first 25 figures from the training data-set. A test is first executed to choose the right value for max depth from 1 to 15, to check which value for max depth has the lowest training error rate using cross validate method. The figure 3 shows elbow graph from 7 to 12 as some values were later decided to let go off to make the execution quicker.

Next the most optimal value 11 for max depth is chosen. Keeping in mind the number of features in the data-set, it calls for dimensionality reduction to reduce the complexity of the model and also the overall time complexity. For dimensionality reduction purpose the script uses principal component analysis (PCA) [10]. With the use of PCA and deciding to keep 95% variance the columns gets reduced to 178 columns. Which is a

The test accuracy score is 81.2

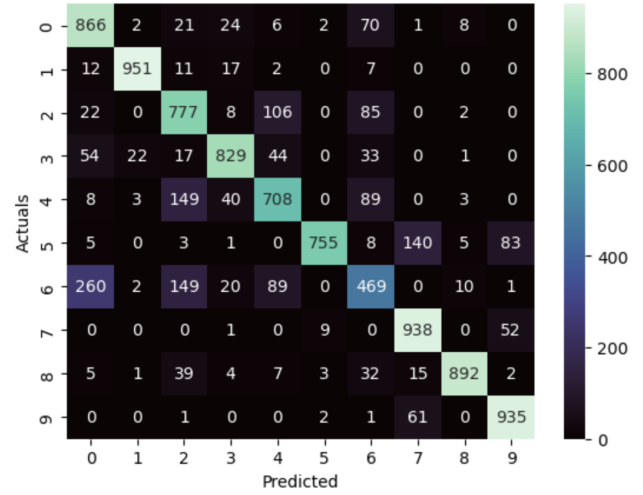


Fig. 2. Confusion matrix for kNN

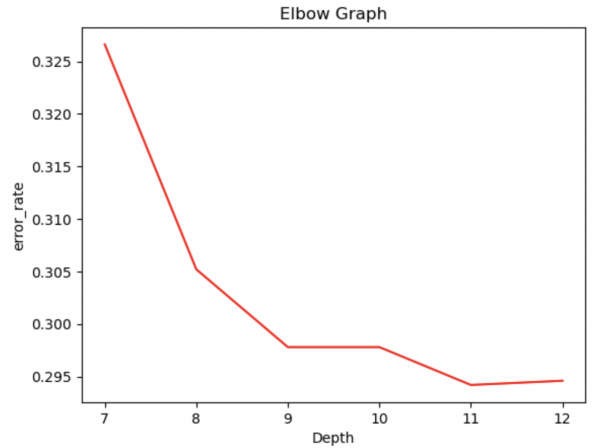


Fig. 3. Confusion matrix for kNN

huge difference from the original 784 columns. Upon calling the predict method on test data-set a accuracy of 71.73% is achieved. The confusion matrix for which is shown in figure 4.

To improve the accuracy the experiment is again continued with same parameters but this time without including PCA. For which the confusion matrix is displayed in figure 5.

As evident without reducing the dimensions the training data-set has improved the accuracy. This was found to be still better after the variance for PCA was selected as 99%.

## IV. RESULTS AND DISCUSSIONS

In this paper a couple of machine learning algorithms were applied to fashion-MNIST data-set. The maximum accuracy which was achieved was 81.2% with decision tree classifier model as shown in figure 2. The time complexity for kNN as seen was  $O(n \cdot \log(n) \cdot m + k)$ , where  $n$  is the total training

Accuracy: 0.7173  
71.73

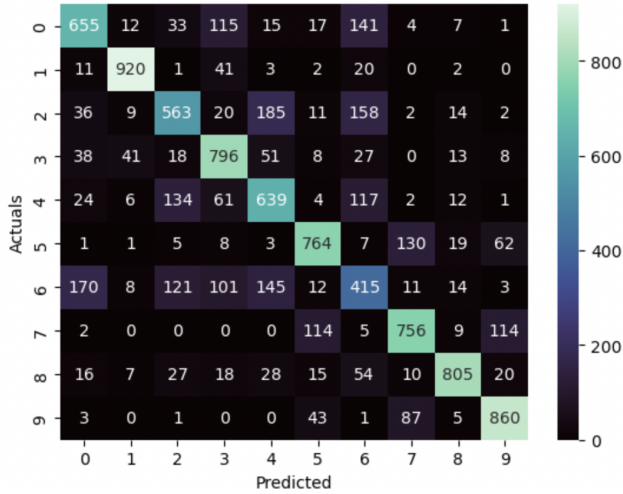


Fig. 4. Confusion matrix for Decision Tree with PCA

Accuracy: 0.8162  
81.62

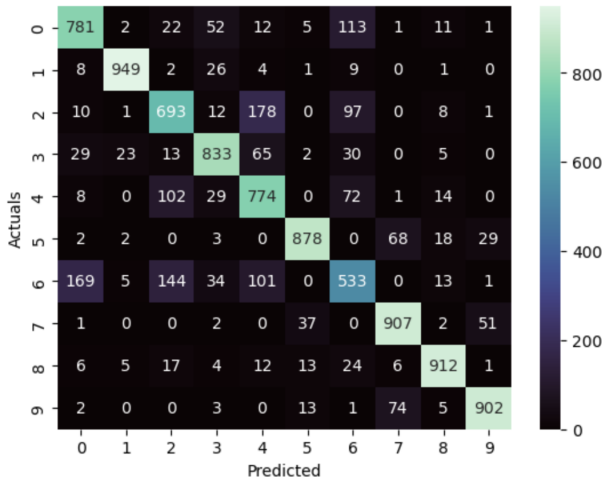


Fig. 5. Confusion matrix for Decision Tree without PCA

data-set samples,  $O(n * \log(n))$  is the time taken for sorting the training data-set which can be done using quick sort or merge sort like algorithms having worst case  $n * \log(n)$  time complexity;  $m$  is the test data-set sample count which can be equal to greater than 1 depending on the size of the test data-set; and finally  $k$  is the value of  $k$  selected. Overall it makes it a really expensive operation. In the literature review as discussed above the time complexity can be further brought down by pre-calculating the distances between each of test and training samples. Although this comes at a cost of space complexity which is  $O(n * m)$ . The elbow graph for kNN showed that between 1 and 10, 4 was the most optimal value for  $k$ . Confusion matrix shows few of the pullovers were negatively classified as shirt, this may look obvious because pullovers and shirts looks same and the distance between non-

alike digits here is lesser than the alike digits. Similarly, the model struggles to identify the shirts with tees and makes 260 wrong predictions with tees, 149 with pullovers and 89 with coat.

In the case of decision tree classifier as evident an accuracy of 81.62% was achieved when the step where PCA was introduced had to be removed. The first experiment conducted was with PCA which gave about 71.73% accuracy rate. The script runs a loop from 1 to 10 to check the most optimal value of max depth, which was found to be 11. The model was initially trained with PCA with 95% variance and then later 99% variance, the accuracy rate did not touch the accuracy rate of the experiment without the PCA involved. But PCA definitely gives a performance boost since it removes the less relevant columns. Confusion matrix again showed similar stats as kNN. 169 shirts were negatively identified as tees. Next was shirt being identified as pullover 144 times. The model continues to struggle to differentiate between the shirts, tees, pullovers and coats. Further the time complexity for prediction in decision tree classifier is always  $O(\log(n))$ ,  $n$  being the total nodes and  $\log(n)$  is the max depth/height in the worst case.

## V. CONCLUSIONS

This study showed around 80% accuracy for both the experiments conducted using kNN and decision tree. As evident the models struggled to correctly identify tees, shirts, pullovers and coats. A various parameters were experimented with to come to the best accuracy rate possible. These algorithms can be a powerful tool for image classification if tuned as per the industrial needs. Although there were several limitations found such as similar looking products were sometimes miss-classified. Since both of the trained models can be tuned to be highly efficient in terms of time complexity, it can also be integrated with real world applications where latency is directly proportional to the user experience. Further research is required to identify use-cases where users can leverage the power of such models. Future research could focus on increasing the accuracy of the models by using appropriate methods to classify the images well. Further work is needed to understand the ethical and regulatory implications of using machines learning in industries to develop guidelines and best practices for its use.

## VI. APPENDICES

### A. Appendix

- [Code for kNN classifier](#)
- [Code for decision tree classifier](#)

### B. Appendix

[Link for downloading kaggle's data-set](#)

### C. Appendix

Additional Resources:

- [Video link to explaining running of kNN classifier's code in python](#)
- [Video link to explaining running of decision tree classifier's code in python](#)

## REFERENCES

- [1] Qifang Bi, Katherine E Goodman, Joshua Kaminsky, Justin Lessler, "What is Machine Learning? A Primer for the Epidemiologist" 2019
- [2] Julianna Delua, "<https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>", 2021
- [3] Jenna Burrell "How the machine 'thinks': Understanding opacity in machine learning algorithms", 2015
- [4] Han Xiao, Kashif Rasul, Roland Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms", 2017
- [5] Divas Grover, Behrad Toghi, Yaser P. Fallah, "MNIST Dataset Classification Utilizing k-NN Classifier with Modified Sliding-window Metric", 2019
- [6] Bahzad Taha Jijo, Adnan Mohsin Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning", 2021
- [7] Birjit Gope, "Handwritten Digits Identification Using Mnist Database Via Machine Learning Models" 2021
- [8] Tsehay Admassu Assegie, Pramod Sekharan Nair, "Handwritten digits recognition with decision tree classification: a machine learning approach", 2019
- [9] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., "Scikit-learn: Machine Learning in Python", 2011
- [10] Fereshteh Sadeghi, "Dimensionality Reduction"