COMP5318 Machine Learning and Data Mining

Assignment One

Fashion Dataset Classification

SID: 480440172, 430345827

Table of Contents

1. Introduction

1.1 Aim

In contemporary times, machine learning (ML) has seen rapid and widespread utilisation in hopes to provide useful information and to solve real world problems. This study will use several machine learning classifiers on the fashion dataset to classify greyscale images of clothing items.

The aim of this study includes the following:

- Implementation and evaluation of the performance of several supervised machine learning models to classify greyscale images into a set of given categories.
- Hyperparameter tuning and pre-processing to achieve fast and high accuracies
- Comparison and discussion of the classifier's performance on the fashion data.

1.2 Importance of the study

There is an abundance of idle data that has the potential to save lives, solve complex problems and boost knowledge about various subjects. The assistance of machine learning and classifiers can transform this data into useful information. It is crucial that machine learning classifiers perform accurately because the performance of classifiers holds incredible importance in the future of decision-based computing and image classification. It is also important to evaluate the performance of different classifiers on a given dataset because different techniques will have different effects on accuracy and running time on the same dataset (Raschka, 2018). Machine learning algorithms used on a specific dataset need its parameters fine-tuned for that dataset in order to find the desired balance between accuracy and run time.

1.3 Data Description

The dataset consists of fashion item images that are stored as grayscale images with a size of 28x28 pixels. The data used in this study contains a training set of 30,000 examples and a test set of 5,000 examples. The data belongs to 10 different categories/classes with labels from 0-9.

Class names:

- 0 T-shirt/Top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot

Figure 1. Examples of data entries across the data set.

2. Pre-processing

2.1 Normalisation

To use certain supervised learning classifiers the data must be normalised. However, as seen in the colour scale in Figure 2, the data did not need to be normalised as all data points range from 0 to 1.

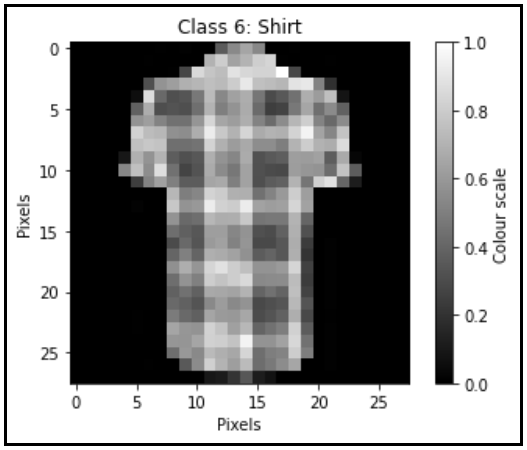

Figure 2 Example of a data entry belonging to class 1: T-shirt/Top.

2.2 PCA dimensionality reduction

Each of the 30,000 examples had 784 features which is very taxing on runtime and therefore dimensionality reduction was required. PCA at 95% variance required 188 of the original 784 features in the dataset. This is equivalent to 23% of the original features. This selection makes sense as we can see in the illustrated data (figure 1) there are a lot of solid spaces of both white and grey where there is little to no detail of importance.

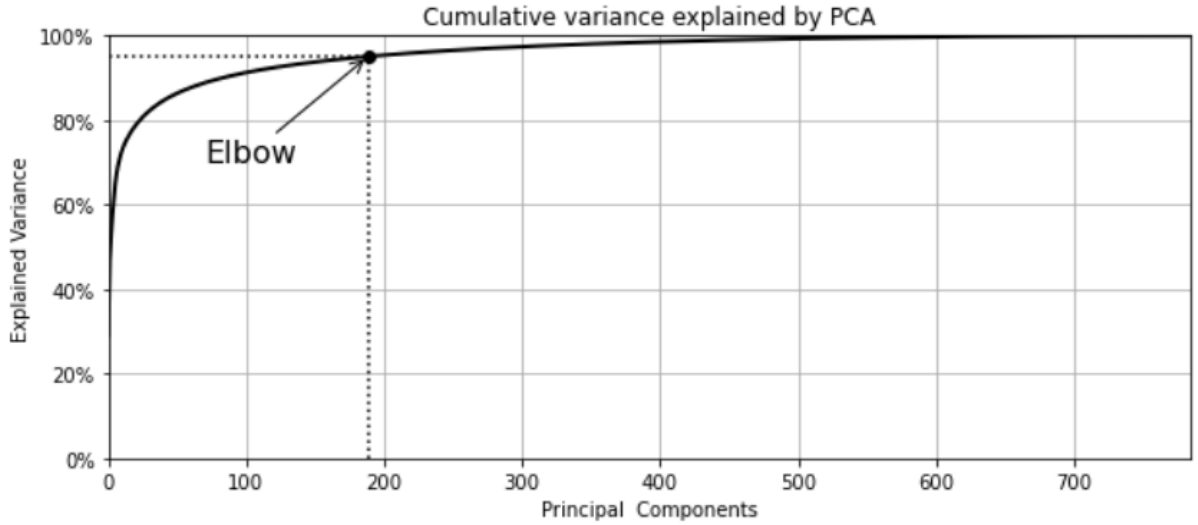

Figure 3. Cumulative variance explained by PCA. The dotted line marks 95% variance explained by 188 principal components.

We performed an initial classifier test with KNN (k=3) to check the effect of PCA on accuracy and running time. PCA had very little effect on accuracy; 84.0% on the original data compared to 89.3% on the reduced data, but a large effect on running time; the original data took 1 minute 28 seconds compared to 6.62 seconds with the reduced data. Therefore, we used the PCA reduced data to build the classifiers.

3. Classifiers and Methodology of Parameter Tuning

3.1 K-Nearest Neighbours (KNN)

The KNN classifier is a simple supervised learning classifier that is used for classification and regression problems (Tran, kNN, August 16, 2020). It works by classifying new examples based on proximity to a specified number of nearest examples, i.e. k neighbours. KNN requires normalisation and with large data, although it can often be very accurate, predictions are slow. It also can be affected by the curse of dimensionality, which means the error increases with the number of features. The solution for this is dimensionality reduction, for which PCA was utilised as mentioned previously. In this study, to determine the best hyperparameters, parameter tuning is necessary. To fine tune the number of neighbours k and power parameters of KNN, a list of predefined values of hyperparameters is passed into the GridSearchCV function. GridSearchCV runs all combinations of the values passed as parameters, creates a model and evaluates it using 10-fold cross-validation. KNN was tested with nearest neighbours of values of 1, 3, 5, 7, 9, 11 using the p parameter as 1 (Manhattan distance) and 2 (Euclidean distance).

3.2 Decision Tree (DT)

Decision Trees build classification models in the form of a tree structure where it splits the data into smaller subsets, until a class leaf node is reached where all class values are the same (Tran, Decision Tree, September 6, 2020). The homogeneity (also referred to as purity) is calculated for each node and provides the basis as to which attribute to split on. It is worth noting that decision trees are unstable in that a small change in data can lead to a large change in the structure of the tree. There are several parameters that can be tested using GridSearchCV including the splitting methods such as “gini” for Gini impurity and “entropy” for information gain. The exhaustive GridSearch includes max_depth parameters because we ideally want to pre-prune the DTs to avoid overfitting and extensive computation. DT was tested with maximum depths of 10, 15, 17, 20, and 22, and split criteria either “gini” or “entropy”. The best parameters against the validation set were maximum depth of 10 and entropy.

3.3 Bagging on DT

A bagging ensemble (also known as bootstrap aggregation) is a method of manipulating the training data by fitting the base classifiers on random subsets with replacement from the original data, and then either voting or averaging the predictions from all individual classifiers to form a final prediction (Tran, Ensemble, September 6, 2020). The bagging classifier helps reduce the variance of each individual base classifier because of the bootstrap sampling and voting techniques it employs. Bagging generally performs better than a single base classifier, so it was carried out in order to improve on the accuracy of the single base DT. In our model, we used the bagging ensemble on DT with the best parameters for DT. The best parameters found in the Grid Search for bagging are max_samples=0.5 and n_estimators=100. However, the final model used the default n_estimators because the constraints of this study need to have the classifier run under 10 minutes.

3.4 Random Forest (RF)

RF utilises DT as its base classifier where each DT in the ensemble only uses a subset of the total features, i.e. bagging (Tran, Ensemble, September 6, 2020). It therefore creates several subsets on the training data and builds a classifier for each. The predictions of each classifier are combined with a majority vote to produce the final prediction. A benefit of RF is in bagging which generates diversity and reduces correlation between the base classifier trees. RF also generally performs better than a single DT and is robust to overfitting. In our model classifier, RF was run with “entropy” split criteria, which was found to be the best in DT. There were two parameters of note which we tested on for GridSearchCV on RandomForrestClassifier with the bootstrap variable set to true. The first variable was max_depth which was set to 8, 10, 14, and 20, and the second was n_estimators which was set to 50,75,100, and 400. The best parameters against the validation set on the second grid search were max_depth = 25 and n_estimators = 1000 which dramatically increased running time.

3.5 Support Vector Machines (SVM)

Support Vector Machines (SVM) are one of the more robust supervised learning algorithms (Tran, SVM, September 13, 2020). SVM finds a hyperplane to separate examples of different classes in the n-dimensional space, where n is the number of features. SVM attempts to maximise the margin between the closest support vectors and can use the kernel trick to perform non-linear classifications. For generalisation purposes, SVM uses a soft margin which allows some examples to be misclassified. A soft margin SVM strives to solve an optimisation problem with the aim of increasing the distance of the decision boundary to support vectors and maximise the number of points that are correctly classified in the training set. The SVM classifier can be tuned according to the parameters of C and gamma. The C parameter adds a penalty for each misclassified example. Therefore, if C is large the penalty is higher and a decision boundary with a smaller margin is implemented. However, if C is small the penalty for the misclassified examples is low and a decision boundary with a large margin is implemented. The gamma parameter can be tuned when using the radial basis function (RBF) as the kernel. In RBF, the gamma parameter contrasts the distance of influence that a single example can have. Therefore, it should be noted that models with large gamma values are prone to overfitting.

In our model, SVC was first tested without cross-validation to find the best type of kernel. This was to reduce running time during grid search. Then using the best kernel, SVC was run with the C parameter at 0.001, 0.01, 0.1, 1, 10, and 100, and the gamma parameter at 0.001, 0.01, 0.1, 1, 10, and 100.

4. Results

4.1 Final Classifiers Performance Results

Model	Hyperparameter tuned	Accuracy Training	Accuracy Testing	Precision on Testing	Recall on Testing	F1 on Testing	Training time (seconds)
KNN	k-nearest neigbours and distance measure	86.0%	84.1%	84.5%	84.1%	84.1%	17.4
DT	Criterion, max_depth, n_estimators	77.0%	77.6%	77.0%	76.9%	76.9%	24.7
Bagging ensemble of DT	Bootstrap, DT base classifier	84.0%	81.1%	81.0%	81.0%	80.9%	57
RF	Max_depth, n_estimators	87.0%	85.1%	84.6%	84.9%	84.6%	726
SVM	Keneral, C, gamma	89.0%	88.9%	88.8%	88.9%	88.8%	64

Figure 4: Results Table

The aim of this study was to implement several supervised machine learning models to classify greyscale image into a set of given categories. We successfully implemented KNN, DT, Bagging, RF, and SVM classifiers. The second goal was to evaluate these classifiers to achieve the highest accuracy by tuning hyperparameters. The results table in Figure 4 presents all the performance score of the classifiers on the test data. Overall, the results were decent however it is important to note the machines that were used to calculate the training time for reproducibility. This table can be found in the appendix.

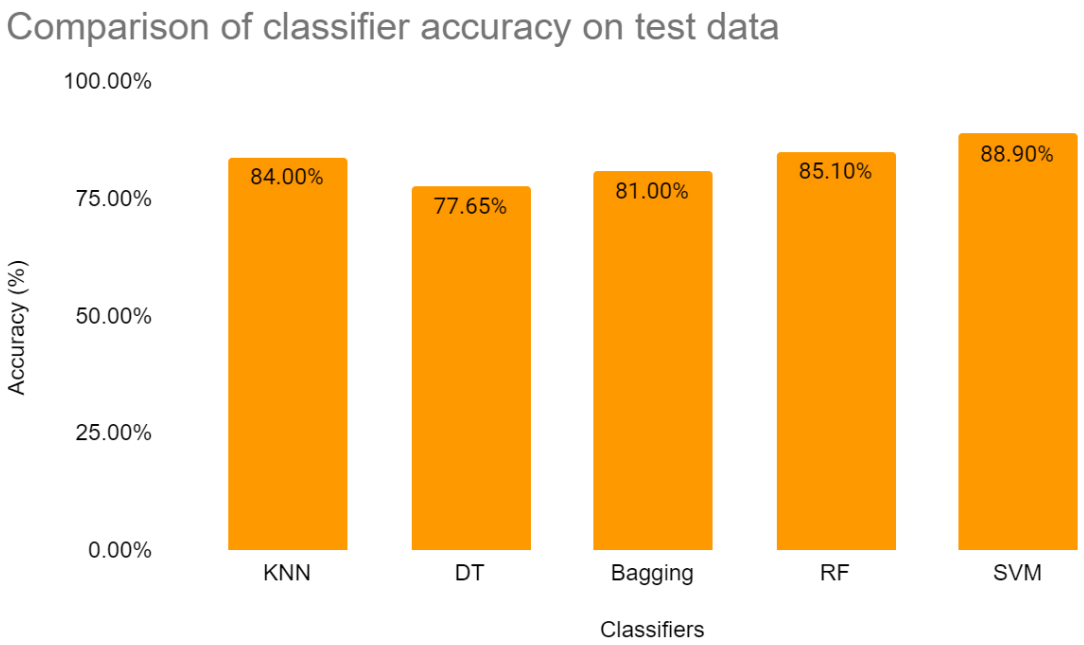

Figure 5: Comparison of classifier accuracy on test data

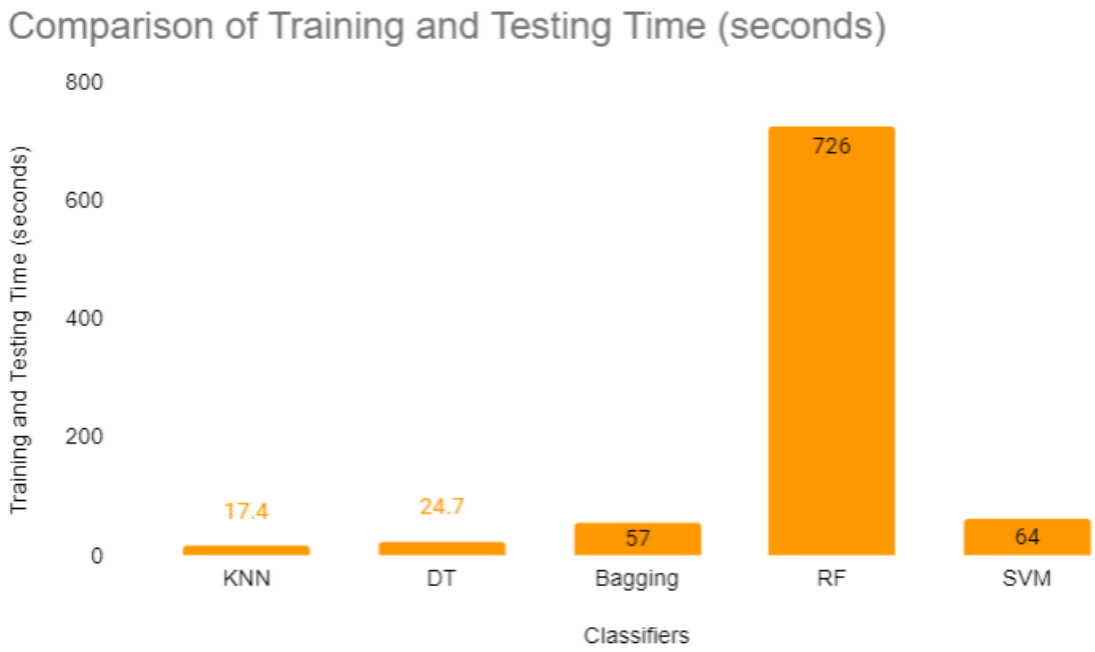

Figure 6: Comparison of classifiers execution time

5. Discussion

The first classifier we implemented was KNN. KNN is known to be often quite accurate, which we observed with an accuracy of 86.0% on training data and 84.1% on testing data with a running time of 17.4 seconds. KNN is generally subject to the curse of dimensionality in that it is not ideal for high-dimensional data. The dataset utilised has 30,000 data entries each with 188 features after PCA. Despite this fact, in this study KNN performed quite well both in accuracy and running time. It was among the highest in accuracy (3rd), and fastest in running time. This could be due to the simplicity of the classifier when it comes to classification. We also used a relatively high number of nearest neighbours in the best KNN model which contributes to its robustness to noisy examples. Grid search performed on KNN revealed the best parameters to be 7 nearest neighbours using the Manhattan distance measure (p=1). These parameters provided a precision score of 84.5% which is quite high, recall and the F1 score are 84.1%. These results are consistent with the accuracy score, therefore KNN with the best parameters performed well on the test data.

GridSearchCV on the DT classifier showed that the best parameters were entropy for split criteria and a maximum depth of 10. DT yielded the lowest accuracy with 77.0% on training data and 77.5% on testing data. The F1 score of the DT model was consistent with the accuracy and resulted with 76.9%. In terms of time, DT performed second fastest at 24.7 seconds. DT are advantageous in that they have high interpretability; however the low accuracy is due to its instability. Small changes in the data often lead to large changes in the final DT. When fully grown the DT had a maximum depth of 22. Thus, grid search showing that the best maximum depth of 10 means that pre-pruning helped to prevent overfitting. This prevents the tree from becoming too specific and helps it to be more generalisable on testing data. However, this was not enough for DT to outperform the other classifiers.

The next classifier is a bagging ensemble on DT. The best parameters found in the Grid Search for bagging are max_samples = 0.5 and n_estimators = 100. However, the final model used the default n_estimators because the constraints of this study need to have the classifier run under 10 minutes. utilised in the ensembles. Bagging is known to perform better than a single base classifier. We observed this trait with ~5% increased accuracy on both the testing and training data. The accuracy on the training set was 84.0%, which took 57 seconds, and 81.0%% accuracy on the testing data. The running time also increased compared to single DT, however this is expected of ensembles.

RF showed an improvement on accuracy compared to DT. The initial grid search revealed the best parameters were a maximum of 400 estimators and maximum tree depth of 20, which were the highest parameters tested. A second grid search was thus performed with higher parameter values. The best maximum depth was 25 and the highest estimator value of 1000 was chosen. 87.0% accuracy was achieved on the training data, 85.1% accuracy on the testing data, with a training time of 726 seconds (12 minutes and 6 seconds). RF was the slowest to run out of all classifiers. This is likely due to the large number of base classifiers with n_estimators=1000 and the bagging step which samples features with replacement from training data. This step was performed 1000 times for the classifier with the best accuracy. The RF classifier achieved a F1 score of 84.6% which is less than the accuracy. Although RF showed an increase in accuracy, it was only an increase of ~1% compared to other non-ensemble classifiers such as KNN. Therefore, when considering the trade-off between accuracy and running time between KNN and RF, in this case KNN will be preferred. The biggest limitation of RF could be that large number of trees makes the algorithm too slow and ineffective against the test data.

SVM kernels were first tested with default parameters to find the best one. Linear, poly, and RBF gave accuracies of 84%, 85%, and 86%, respectively on the testing data. Therefore, grid search was performed with the RBF kernel. Out of 36 possible combinations (see methods section for details), a gamma value of 0.01 and C parameter of 10 were identified to be the best. These parameters also achieved the best accuracy of all classifiers at 89.0% on training data and 88.9% on testing data sets with a running time of 64 seconds on the testing data. The recall, precision and F1 scores are consistent with the accuracy and SVM performed as the best classifier in this study. The low gamma value creates

a smoother decision boundary that is less sensitive to each data point. C is a regularisation parameter. A relatively large value of C allows the model to be less restrictive where each point has a bigger influence on the final prediction.

6. Conclusion

We found that SVM with an RBF kernel and hyperparameters gamma value at 0.01 and C at value 10 resulted in the highest accuracy on the testing data at 88.9% with a decent running time of 64 seconds. RF came second best at 85.1% accuracy on testing data but proved to be the longest to run at 726 seconds (12 minutes and 6 seconds). Other classifiers such as bagging, DT and KNN were faster than SVM to run but were ~4-11% worse in accuracy.

It should also be noted that the full fashion dataset contains 5000 testing samples but only 2000 were used here for testing. Testing on the full 5000 testing set will give a better insight into the true accuracy of the classifiers as that is how the dataset was designed to be used (Schapire,1997). Additionally, the variance used for PCA could be varied to test if 95% variance is the best for representing the dataset in dimensionality reduction.

In future, multinomial logistic regression (LG), gradient boosting (GB), and Adaboost classifiers could be implemented. They were attempted for this study but due to time constraints could not be substantially explored to be reported on and hence were left out. LG would be advantageous in that it not only outputs the predicted class but also the probability of belonging to that class (Grover, 2021). However, one problem that may present itself is that for high dimensional datasets such as this, LG is prone to overfitting and if the data is not linearly separable, transformations are needed to capture the complex relationships. GB, on the other hand, has more flexibility and no preprocessing is required with good results on categorical data. The downside is that it is also prone to overfitting and is computationally expensive which is bad for high dimensional data. Finally, Adaboost works well if the base classifiers are simple (Tran, Ensemble, September 13, 2020). This was tested with DT but the best maximum depth of 10 proved to make the base classifier too complex and the algorithm was stopped after a long running time. Exploration with Adaboost could prove to be even better than SVM with its best parameters.

7. References

Kurama, V., 2020. *Gradient Boosting In Classification.* https://blog.paperspace.com/gradient-boosting-for-classification/

Li, Y, Cao, L., Zhu, J., & Luo, J. *Mining Fashion Outfit Composition Using an End-to-End Deep Learning Approach on Set Data*. IEEE Transactions on Multimedia, vol. 19, no. 8, pp. 1946-1955, Aug. 2017, doi: 10.1109/TMM.2017.2690144.

Schapire, R. E. (1997, July). Using output codes to boost multiclass learning problems. In *ICML* (Vol. 97, pp. 313-321).

Tran, N.H. (2020, August 16) *kNN* [Lecture notes]. Canvas@USYD. https://canvas.sydney.edu.au

Tran, N.H. (2020, September 6) *Decision Tree* [Lecture notes]. Canvas@USYD. https://canvas.sydney.edu.au

Tran, N.H. (2020, September 6) *Ensemble* [Lecture notes]. Canvas@USYD. https://canvas.sydney.edu.au

Tran, N.H. (2020, September 6) *SVM* [Lecture notes]. Canvas@USYD. https://canvas.sydney.edu.au
Grover, R. 2021. *Advantages and Disadvantages of Logistic Regression.* https://iq.opengenus.org/advantages-and-disadvantages-of-logistic-regression/

Weerts, H. J., Mueller, A. C., & Vanschoren, J. (2020). Importance of tuning hyperparameters of machine learning algorithms. arXiv preprint arXiv:2007.07588.

8. Appendix

8.1 How to run the code:

1. Import libraries (1st cell)
2. Import testing and training data (2nd and 3rd cells)
3. Run PCA (cell under "Cumulative variance explained by PCA" graph)
4. All grid searches and single classifiers can now be run in any order.
5. Note: To run the “Results and “Comparisons” section, the following must have been run:
    a) Points 1-3 in this list,
    b) The five Subsections titled “Evaluate the best parameters from GridSearch on test data”, and
6. Note: To output the results of the best classifier (section 5), run:
    a) Points 1-3 in this list
    b) The two cells in section 5.

8.2 Hardware and software specifications

Hardware	OS and Software
Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 2.00 GHz 16.0 GB RAM, 64-bit operating system, x64-based processor	Windows 10, Anaconda Distribution Jupiter Notebooks, Python 2.7.17
Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.59 GHz 8.0 GB RAM, 64-bit operating system, x64-based processor	Windows 10, Anaconda Distribution Jupiter Notebooks, Python 2.7.17

Figure 7: Hardware and software specifications used for processing