# EXPLORING CLASSIFIERS ON IMAGE DATA

Machine Learning and Data Mining: Assignment 1 COMP5318

Jimena Espinoza SID:500052750 Maria P. Mandiola SID:500616851

September 24, 2021

## 1 Introduction

### 1.1 What is the aim of the study?

This assignment aims to perform a supervised classification task by building a high-performance classifier that accurately predicts the class of grey-scale images of size $28 \times 28$ representing an article of clothing within 10 defined categories (classes). Furthermore, it is required that the classifier generalizes well its predictions when new examples are presented, without being computationally expensive in terms of training, prediction time and memory resources needed .
With this purpose in mind, we will implement multiple algorithms using the given data to train the models. The provided data will be explored, cleaned and pre-processed beforehand. Hyper-parameter tuning will be undertaken to find the best parameter for each algorithm while preventing overfitting. Finally, we will evaluate and compare the performance of the models using measures such as Accuracy, Recall, Precision, and F1-score. These measures will guide us in the selection of the best classifier.

### 1.2 Why is the study important?

The importance of this study relies upon three main aspects:

- It explores and compares the performance of the ultimate Machine Learning methods within the Multimedia field, particularly Image Data. Due to the digital technologies advances, the penetration of smartphones and the accessibility of digital photo cameras, this type of data has rapidly increased in the latest years, and it is projected to remain growing in the future [3].

- It applies these Machine Learning methods on a big scale data set. In the context of the Internet and Big Data, handling large data sets is essential. Therefore, a Data Scientist must be able to find and apply thoughtful ways to understand, clean and manage big data sets minimizing the loss of information to ensure the richness of the analysis and the adequateness of the conclusions [2].

- The supervised classification task of multiple categories has enormous potential to be helpful in a variety of fields. Examples are found in health care, financial area, marketing, among others. Large databases might hide valuable information, which can be used for intelligent decision making [1].

  Strategies aimed to classify in an accurate manner new examples can save many resources automatizing tasks, overcoming the performance of the old methods of classification being used. Studying and comparing the different types of classifiers available nowadays through advanced Machine Learning methods, contributes to the general knowledge, providing more insights about which method and/or which parameters should be recommended for specific situations, helping future tasks to be more efficiently achieved.

## 2 Methods

In the following section, a vast explanation of the methodology applied in this project will be described, from data exploration and understanding to the models training.

The hardware and software used to do this analysis, including both the evaluation and comparison of the models are described in Appendix B and C.

The full code has been provided on a separated file (`ipynb` and `pdf` are available) in a Manual-Style script, in favour of the user's correct understanding and implementation of the analysis.

### 2.1 Data Exploration

The data set consists of a training set of 30,000 examples and a test set of 5,000 observations, where only 2,000 are labelled and will be used to evaluate. Each example is described by 784 features, representing the pixels of a $28 \times 28$ image. There are no missing values, and the minimum value is 0 while the maximum value is 1 for both sets. All data points are distributed already within the same scale (0-1). Each example is labelled by one over ten classes. An example of each class is shown in Appendix A.

The input data was provided separated into training and test data. Stratification of both data sets was checked and is shown in Figure 1. It is observed that the partition of the data was done ensuring that the test and the training set have both an approximately even number of each class within them. Henceforth, the train and test set that will be used for training and evaluation with no need of settings on this matter.

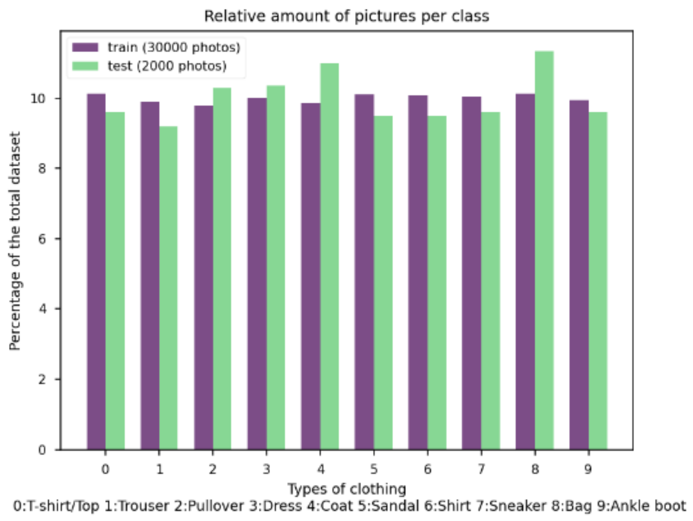


Figure 1: Relative amount of pictures per class in train and test data.

### 2.2 Data Description

As mentioned in the previous section, the provided data set has every feature observation within the range 0 and 1. To further examine the data, a look at the central tendency statistics might be helpful. Regarding the mean and variance shown in Figure 2, it is noticed that the variance across pixels is overall very small. The highest variance observed for a pixel is not more than 0.2. It should be pointed out that the pixels are ordered from pixel 0 to pixel 780 on the "x" axe and that for visualization convenience, not all of them are labelled. The values were connected using a line, however, this might be misleading, as we are dealing with square images of size $28 \times 28$ and the pixel of the next row is appearing as a continuation of the previous row.

There is a clear pattern of sequential peaks and lows within the mean and variance every 28 pixel approximately, which represents a 'row' in the image. This helps us interpret that the central pixels in a row have a higher mean than the ones near the borders. The former is also valid for the variance, with the caveat that the variance is overall much smaller than the mean, and, in consequence, the fluctuations are also smaller. A drop in mean and variance on the first and last row can be appreciated from the plot. This gives us a general insight into how mean and variance are distributed across the images. However, the spatial distribution is only partially described through this visualization, which is why we complemented the central statistics description with Figure 3.

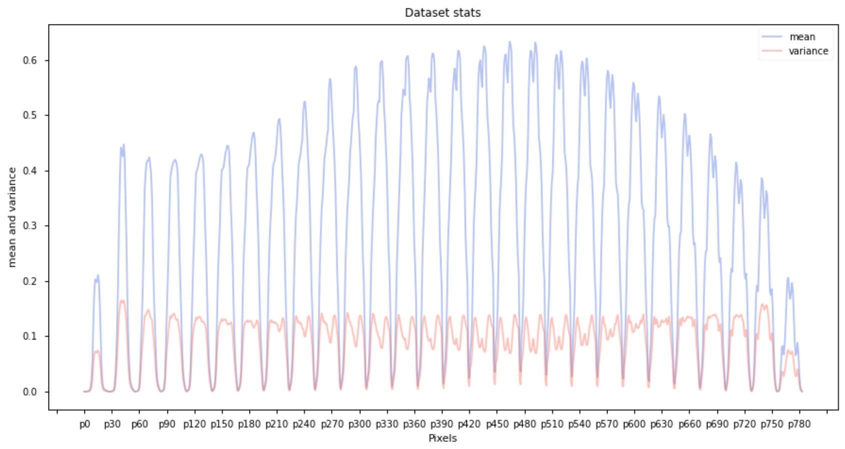


Figure 2: Mean and variance per pixel

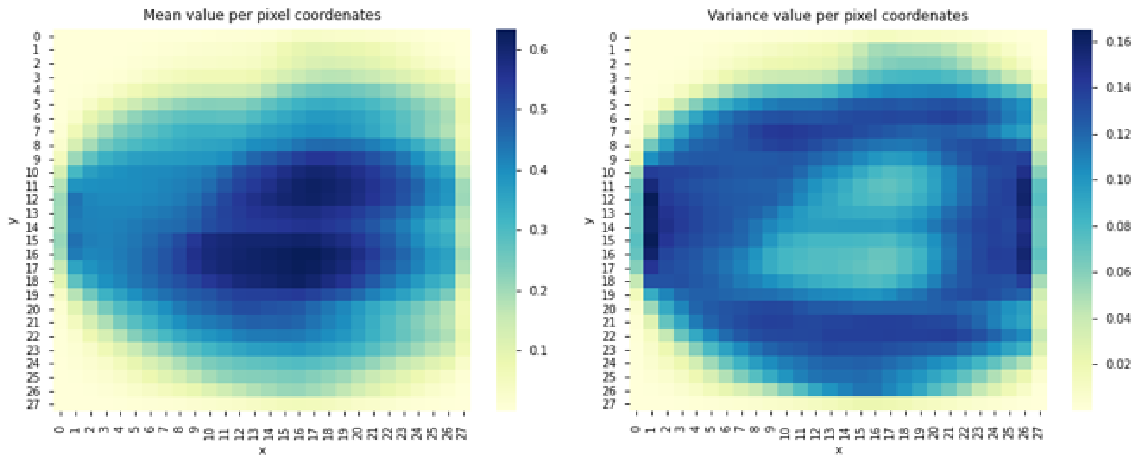


Figure 3: Visualization of Mean *(on the left)* and Variance*(on the right)*

The plots in Figure 3 help locate the mean and variance of each pixel. The former gives a clearer idea of how the mean and variance are distributed per image in their respective positions.

As might be intuited, the corners and borders have a mean and variance close to 0, showing a tendency to be empty. The highest means are located in the centre of the image, while the highest variance is found in what can be thought of as the contour or boundary of the object with the background of the image.

The hypothesis stated at this stage is that each class has a different distribution of values that compose the differentiated shape each of them has. This sets the basis for our classification project, as it means that intelligence can be applied to understand the patterns on these measures, and then be able to predict a class based on its pixels values.

As noticed, some of the pixels have very little information to offer, and a method to select the valuable features is therefore needed. This will also help reduce the current dimension of 784 to a more manageable number of features in terms of computational capacity and interpretability of the data.

## 2.3 Data Pre-processing

After data exploration described in section 2.2, normalization has been considered unnecessary for this project. We have observed that all feature's values are within the range 0 to 1 and because they represent a measure of the grey-scale, we can state they are all on the same scale or unit of measure.

Besides that, it was concluded that it is needed to reduce dimensions for efficiency purposes. Principal Component Analysis (PCA) is used to transform the data to a lower dimensional space [6]. This tool will find the direction where the maximum variation can be found, transforming the features into those new axis. It continues adding orthogonal axis as components until all the variance has been explained.

According to `Scikit-learn` documentation [4], the adjustment of the features to obtain variables with a normalized standard deviation and mean might be crucial when using an algorithm that aims to maximize the variance, which is the case of PCA. However, standardization is recommended when features have a normal distribution. To investigate the effect of standardization when PCA is used in the context of our project, a comparison of the cumulative explained variance by number of components was done between standardized and non standardized data. Figure 4 shows this comparison, through the purple and blue line (respectively). It is noticed that the variance explained when the standardization was applied turns out to be lower in the "elbow" part of the curves. As a result, a higher number of components is needed to explain the same variance on the standardized data. It is intuited that standardization might be loosing small difference in variance between features that might be valuable during the training phase.

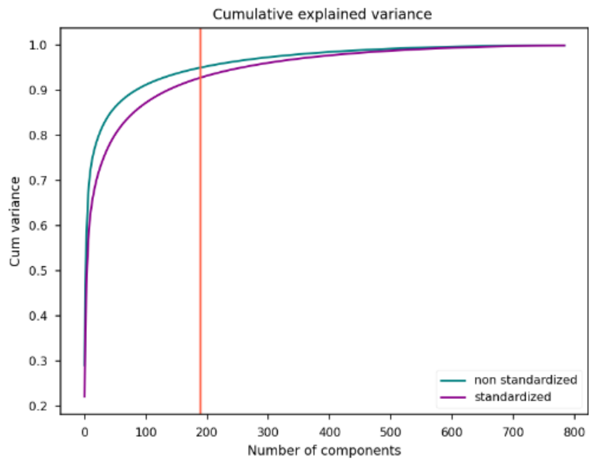


Figure 4: Cumulative Variance Explained per Number of Components

From Figure 4, we can appreciate that the "elbow" of the Cumulative Explained Variance curve starts around 25 components and makes an inflexion when the number of components is approximately 60. Then, it starts decreasing the speed of the explained variance increment per component incorporated. The target that we defined was to keep 95% of the original data set's variance. In the data provided, the preceding means the features are reduced from 784 to 188. The plot shows that this seems to be a reasonable number of components to be chosen, as the explained variance curve gets considerably flattened after this point. A red line represents the point where the number of components is the selected 188. The PCA transformation, fitted to the training data using 188 components, will be used in both training and test data.

A visualization of the reduced data set within the first two components is provided in Figure 5. Surprisingly, some of the classes are relatively compacted and identifiable by the eye in this 2D representation. An example of this is the Class 1:'Trouser'. However, the classes are not separated from each other. It is observed that points from the same category are tightly knitted in the plot. Some of the points belonging to different classes are merged, making the classification task hard to visualize just from the first two components. The first two components explain 46.81% of the total variance observed in the original data set. The remaining 188 components we will consider for model training will help add up to 95% of the variance explained.

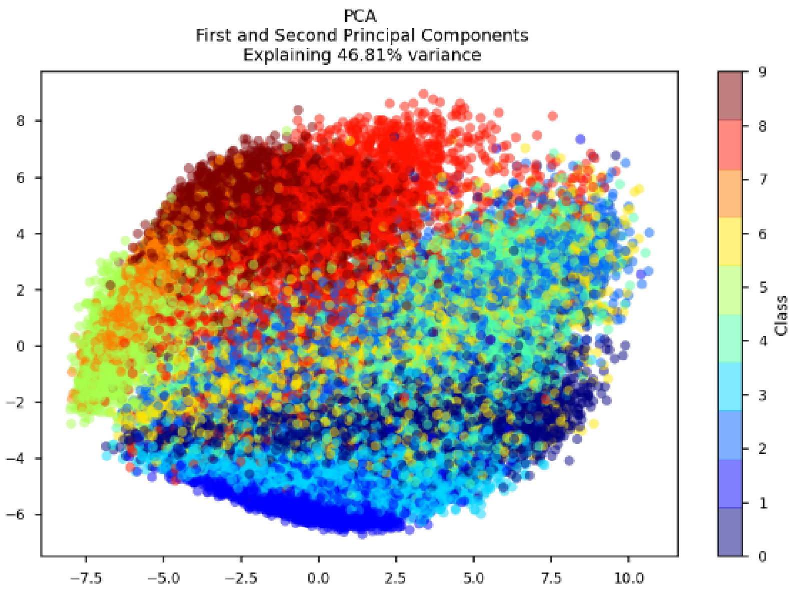


Figure 5: 2D Visualization of classes after PCA

### 2.4 Classifiers implementation

The models selected for training are shown in Table 1.

| Model | Abbreviation |
|---|---|
| K-nearest neighbor | knn |
| Multinomial Logistic Regression | log |
| Naïve Bayes | nb |
| Decision Tree | dt |
| Bagging | bag |
| Random Forest | rfo |
| Gradient Boosting | gb |
| Support Vector Machine Classifier | svc |
| AdaBoost | ada |

Table 1: Classification models

#### 2.4.1 Hyper-parameter tuning

It is required to learn the best parameters that make our models have the closest to ideal performance. However, we also want to reproduce those excellent results in new samples. We will use cross-validation to ensure that our models will generalize well in examples that are not included in the training set. The function `sklearn.model_selection.GridSearchCV` is used in this instance. The parameter `cv` is set to 10 folds, in which the data is divided. The parameter `n_jobs` is set to -1 to run as many jobs in parallel as possible.

It is noticed that all combinations of parameters will be tested using the `GridSearchCV` function. The cross-validation using 10 folds with a large data set might be computationally expensive. However, this evaluation method provides us with the possibility to train the models with the whole training data-set provided (the 30,000 examples) without sacrificing a subset for validation purpose and ensuring there will be no no overlap between the validation and training sets.

Table 2 shows the parameters included in the grid search per model. The best parameters found are displayed in a separated column, at the right. The `GridSearchCV` implemented for the Gradient Boost classifier turned out to be significantly slower. For that reason, we excluded this classifier from the hyper-parameter tuning process. We will be comparing the performance of this classifier in terms of accuracy, time and memory used. However, it is acknowledged that the performance comparison will not be fair in the case of this model since the parameters used in the training model has not been validated. It is observed that the smoothing parameter used in the nb is not relevant in terms of performance.

According to the Scikit-Learn documentation[5], when the parameter `return_train_score` is defined as 'True', there is a computational cost. Even though it is useful because it provides the score obtained in the training set, the decision made was to set this parameter as 'False' as it was needed to help the speed of parameter tuning cross-validation and evaluate the best model afterwards, using cross-validation only in the best parameters found and not in all the combinations.

| Model | Grid Parameters | Best Parameters |
|---|---|---|
| knn | number of neighbours: 5, 9 | 5 |
| | p : 1, 2 | 1 |
| | algorithm: kd tree, ball tree | kd tree |
| | metric: minkowski | minkowski |
| | weights: uniform, distance | distance |
| log | solver: saga, sag, newton-cg | saga |
| | penalty: l1, l2, elasticnet, none | l1 |
| | warm: True, False | False |
| nb | var smoothing : np.logspace(0,-10, num=100) | 2.310e-04 |
| dt | criterion: gini, entropy | entropy |
| | splitter: best, random | best |
| | max depth: 12, 13, 14 | 12 |
| bag | max depth: 1, 2 ,5 ,8 ,10 | 10 |
| | max samples: 0.05, 0.1, 0.2 | 0.2 |
| | max features: 0.05, 0.1, 0.2 | 0.2 |
| rfo | criterion: gini, entropy | entropy |
| | max depth: 1, 5, 10 | 10 |
| | max features: sqrt, log2 | sqrt |
| | n estimators: 100, 500 | 500 |
| svc | kernel: poly, rbf | rbf |
| | C: 1, 10, 100 | 10 |
| | degree: 1, 2, 3, 10 | 1 |
| | gamma: scale | scale |
| ada | learning rate: 0.01, 0.1, 0.3, 1 | 0.01 |

Table 2: Hyper-Parameter Tuning

#### 2.4.2 Model training

After tuning our hyper-parameters, the ones that were proven to have the best performance per model during Cross-Validation were used to train each classifier. These selected parameters are Table 2, in column "Best Parameters".

## 3 Experiment and Results

### 3.1 Performance measures

In the following section, the performance of the models trained using the best parameters will be assessed in terms of the results and their accuracy, and the resources in terms of time and memory used in the process.

#### 3.1.1 Confusion matrix

Although the confusion matrix is not a performance measure itself, it allows us to glance at how well the true labels match the predicted labels. From the confusion matrix performance measurements can be computed: Accuracy, Precision, Recall, and F1-score. However, `Scikit-learn` provide us with a complete report using the function `metrics.classification_report` that will be used in the following analysis.

For simplicity purpose, three models have been selected for their confusion matrix to be displayed in Figure 6, namely: knn, log, and svc. This visualization allows us to learn how many clothing items are being correctly or incorrectly classified and for what other clothing pieces are being confused. Yellow squares are showing the higher values and purple the lowest. Overall, the higher values are allocated in the diagonal, showing most items are being classified in a correct manner. We can see, for example, that all three models are good at categorizing 'Trouser'.

On the other hand, the three models seem to be struggling with the classification of 'Shirt'. They commonly classify objects supposed to be 'Shirt' as 'T-shirt/Top', 'Pullover' and 'Coat'. But, interestingly, on the other way around, when the true label actually corresponds to this second group ('T-shirt/Top', 'Pullover' and 'Coat'), they are many times they are wrongly classified as 'Shirt'. Some misclassifications on 'Ankle boot' and 'Sneaker' classes are also spotted. The similarities between these classes can be appreciated by the human eye, though it is acknowledged that distinguishing between these categories might be challenging for machines too.

Despite of the valuable inferences we can make by observing and comparing the confusion matrices, it is not possible to state which classifier is better only by looking at them. The following sections will provide more insights to develop models' performance comparison further.

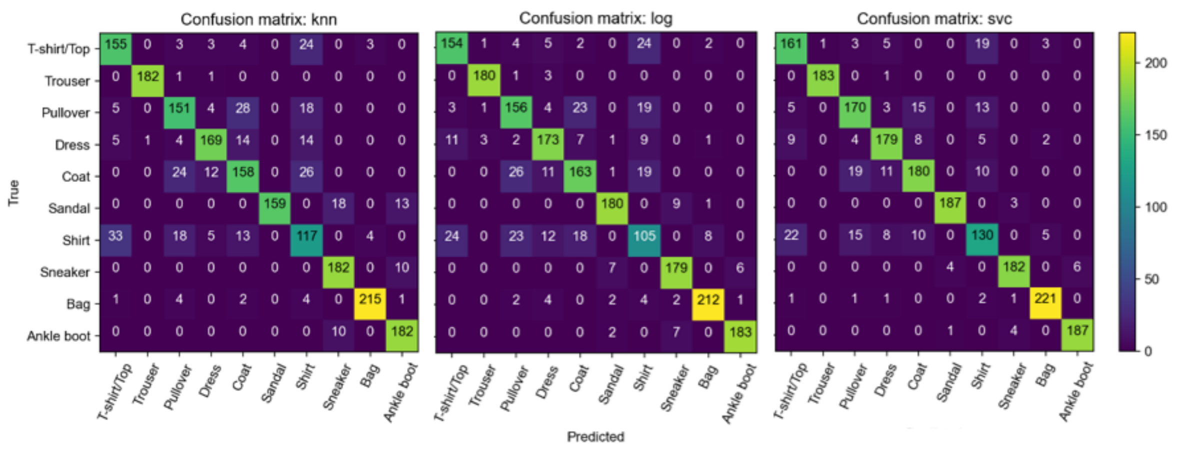


Figure 6: Confusion Matrix: knn *(left)* , *(middle)* log, and *(right)* svc.

### 3.1.2 Accuracy

Accuracy rate is shown in Figure 7. The results shown by the bars correspond to the percentage of correctly labelled items measured in the 2,000 samples Test Data provided. The stars show the accuracy rate achieved on the Training Data, making it easy to compare both data-sets performances. The gap between the bar and the star of each model gives us insights into how well each model is generalizing or over-fitting.

In general, it can be stated that the best Accuracy rate measured in test data is seen in the svc model, followed by the rfo, log, bag and knn models, in order.

On the Accuracy figure on the left, it is noted that there are cases where the Accuracy measured in Train Data (the stars) is very high, being 100% in the case of ada and knn. This shows that there is over-fitting occurring, meaning the model is completely adjusted to match the training data with its respective labels, but not generalizing well when new samples are assessed by the model.

Similarly, on the right plot, the Accuracy rate measured using Cross-Validation is been provided. In this case, the data was split into 10 folds and each fold was also partitioned into 10 parts, using 9 to train the model, and 1 part to test. The 10 different values of accuracy are averaged and shown in the figure. This procedure is what makes the CV Accuracy method to be considered the most reliable one, and might be the one taken into account to make conclusions about which model performs better.

This confirms that the best model found according to the CV-Accuracy rate is the Support Vector Machine (SVC).

The superior performance of SVC model in this data-set is reasonable, as SVM models are known to be powerful classifiers, that can manage multi-class problems of both linear and non linear data, through the maximization of the Margin hyper-plane.

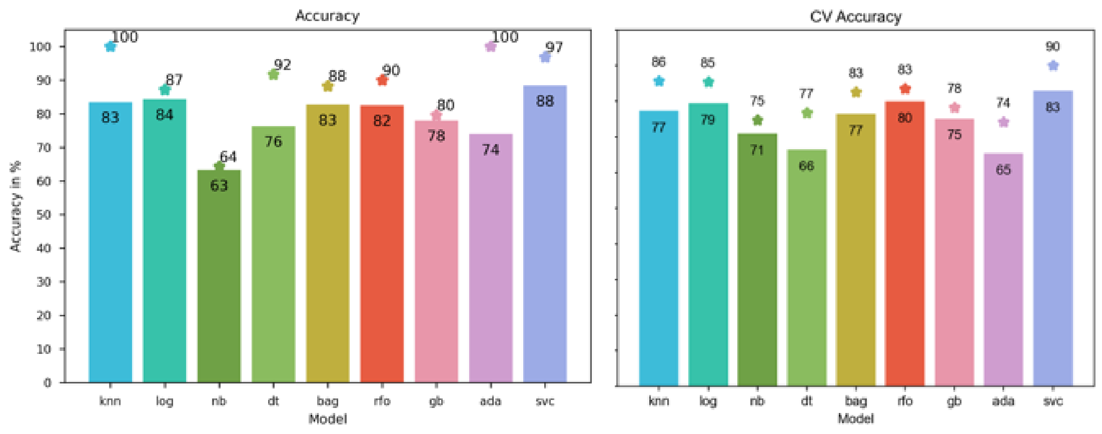


Figure 7: Single Accuracy *(on the left)* and Cross Validated Accuracy *(on the right)* obtained per model

The plot also brings out Random Forest (rfo) as a high performance model in this case of study, in contrast with Decision Tree (dt) and Ada Boost (ada), which are the lowest in terms of Accuracy. For dt, this can be explained by the fact that Random Forest are an ensemble of various single Decision Trees which often outperforms versus each of them performance on testing. In the case of ada. which is a Bagging method, the difference is expected, as we know that Random Forest is considered an improved version of Bagging, adding to the process the usage of a subset of features per each classifier.

Naive Bayes is also a model that has not shown a brilliant performance. This might be related to the fact that assumptions made when defining this model are not completely true. Even though we now Naive Bayes has sometimes surprisingly good results, it is acknowledged that in this case, features are definitely not independent to each other, as they represents positions, or points in a map, forming figures through the connection of one point to the other. That means they are correlated, and assuming they are not is a too big oversight in this case. However, it is worth mentioning that our worst model shows an Accuracy rate considerately better than a random guess (62% vs 10%).

The performance of the Multinomial Logistic Regression (log) is worth to mention, as it occupies the third place in the best Accuracy score ranking. This is an interesting result, taking in consideration this is a classical probabilistic model competing with much more sophisticated models

### 3.1.3 Precision, Recall and F1-score

Figure 8 displays rich information about our models' performances. It includes F1 score, Recall and Precision, providing valuable granularity, as it offers information opened by class. This gives us more details about how can the performance of each model be explained. Values have been highlighted using a gradual colour scale where darker means higher and lighter means lower values.

From this figure, we can confirm what was inferred before by the confusion matrices in Figure 6, that Shirt is the class that has lower performance in the classification task. This is consistently shown on the three different measures across all the models. It is noted that Sandals have a perfect precision in the model knn, meaning all the Sandals are being detected by this classifier. However, as the recall is 0.84, some of the examples classified as Sandals are something else.

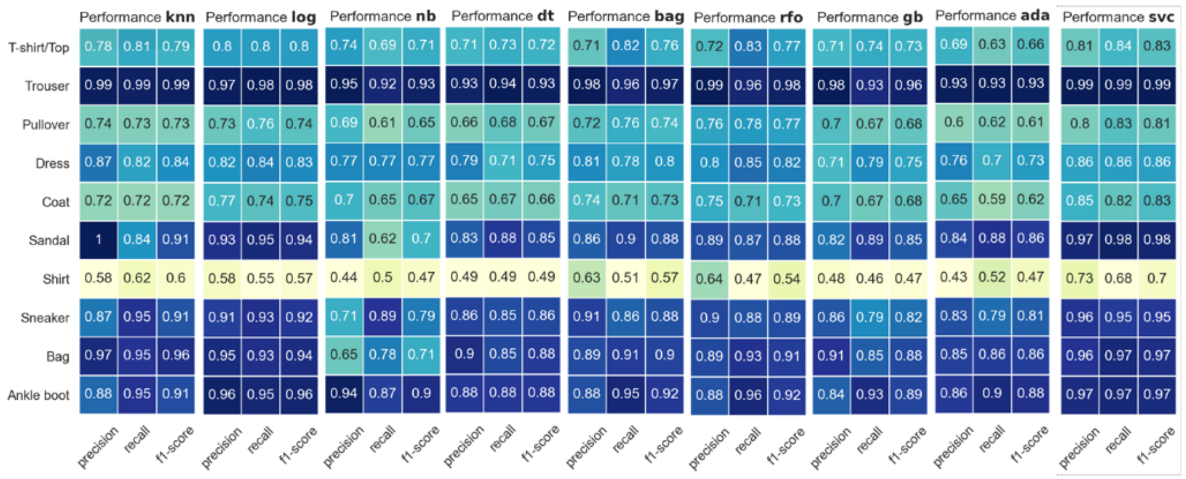


Figure 8: Performance measures across models.

### 3.1.4 Time

Figure 9 on the left, shows the time that it takes for each model to be trained using the 30, 000 examples on the training set. On the right, the figure displays the time that it takes to predict the labels on the 2, 000 test examples.

It is noticed that the scale of the plot on the left is much larger than the one on the right. It is reasonable to spend more time training the models, considering the calculation involved and the length of the training set. The gb model deserves special recognition, being by far the slower model in the training phase. This was perceived during the tuning hyper-parameter stage, in which the extensive grid search was aborted, having a not validated model being compared using random parameters.

On the other hand, it seems reasonable that almost no time was needed to train the knn model since the values are stored to perform distance calculations on each new example in the later prediction stage. This is observed on the right plot, where knn is the second most costly model during prediction. The svc also does not have much time during the training phase. Therefore, it is the most expensive classifier when it goes on prediction time. Overall, the training time is not excessive considering the task. Therefore svc model shoud not be disregarded.

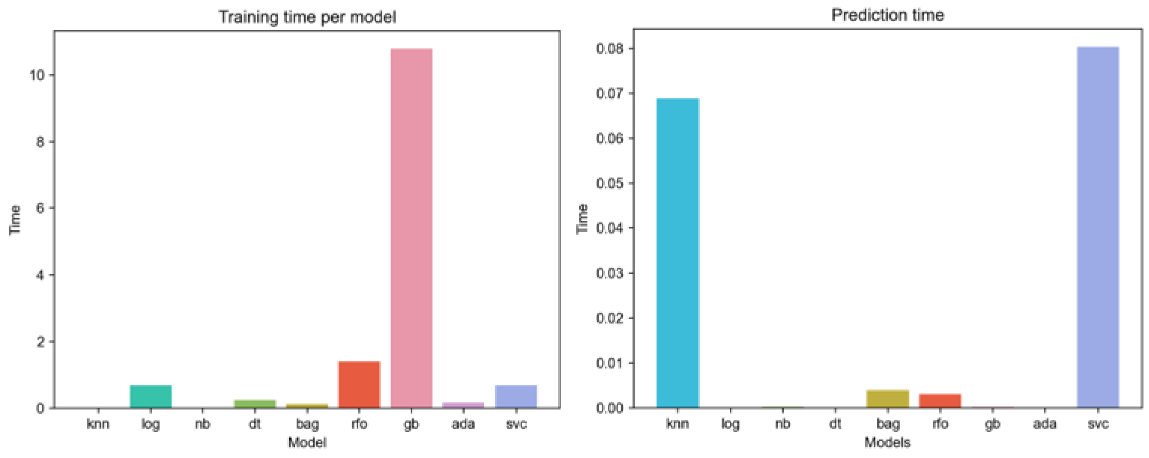


Figure 9: Training time *(left)*, and *(right)* prediction time per model.

### 3.1.5 Memory

Figure 10 shows the peak size of memory used during the training stage on the left, and the size of memory used by the end of training on the right. Consistently with the observation made for time, the knn model is the second more in need of memory. This is because at the training stage the knn algorithm is memorizing all the observations. At the end of the training phase, the same amount of memory resources remain occupied. The complexity in terms of memory is the highest for the svc model. However, this might be due to calculations required to build the model because after the training phase is finished fewer memory resources remain used.

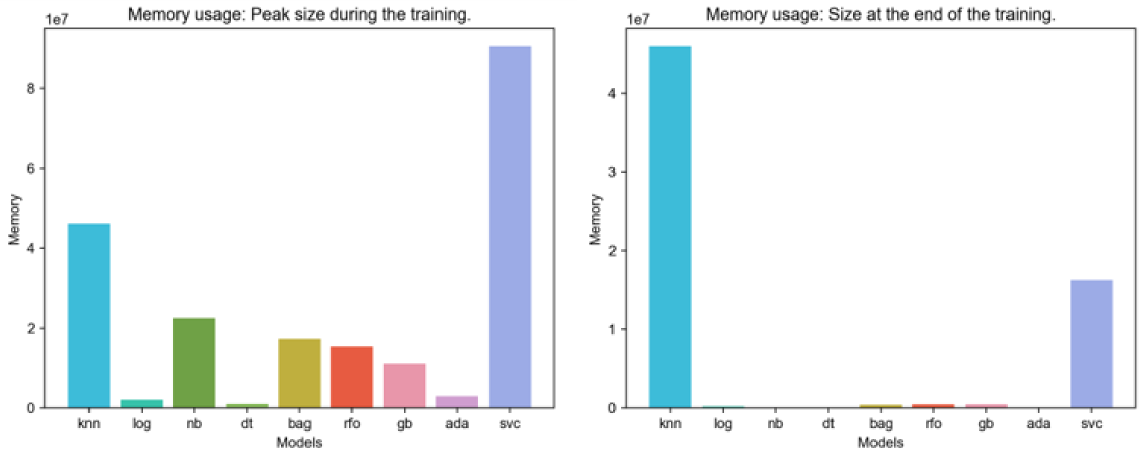


Figure 10: Memory usage: *(left)* Peak size during the training, and *(right)* Size at the end of the training.

## 4 Conclusion

Our research concludes that the best model for this specific classification task is the Support Vector Machine, using the Radial Basis Function kernel. The svc model reached the highest results under Cross-Validation evaluation, which is evidence enough to state it is the most robust clothing class predictor.

The process to get these results was very challenging, and therefore, nourished our learning process in a very concrete manner. One relevant take out of this project was the confirmation of the importance of Dimensionality Reduction. Without PCA, we would not have been able to apply the models to the data sets with the available computing capacity. This unsupervised method turned out to be very helpful when handling high dimensional data. It helps to drop the data that is not explaining much variation and not providing incrementally significant information. As a result, we end up sacrificing a small percentage of variance that will not be explained by the remaining transformed data set, but gaining important improvement in running time and computational need.

We had the constraint of computational power. Using more resources, we could perform a more exhaustive grid search, and it could be that better parameters could be found. Furthermore, an interesting future work would be to do a comparison between performance using different parameters, in terms of both performance and efficiency (time, memory), and to see, for instance, the difference in terms of how much resources we need to have 1% more of accuracy. Another future work opportunity would be to explore the effect of transformations applied to the images, such as size reduction, contrast adjustment or the use of filters have on the performance of the models.

The expectation is our best model will perform even better in a larger test set as it is failing mainly in one over ten classes ('Shirts'). If the complete test data of 5,000 observations are correctly stratified, one class will represent a 10% of a larger total number.

## References

[1] Julie M David Balakrishnan et al. Significance of classification techniques in prediction of learning disabilities. *arXiv preprint arXiv:1011.0628*, 2010.

[2] H. Bowne-Anderson. *Harvard Business Review*, what data scientists really do, according to 35 data scientists. https://hbr.org/2018/08/what-data-scientists-really-do-according-to-35-data-scientists, 2018. Accessed September 22, 2021.

[3] N. Pantic. How many photos will be taken in 2021? https://blog.mylio.com/how-many-photos-will-be-taken-in-2021-stats/. Accessed September 22, 2021.

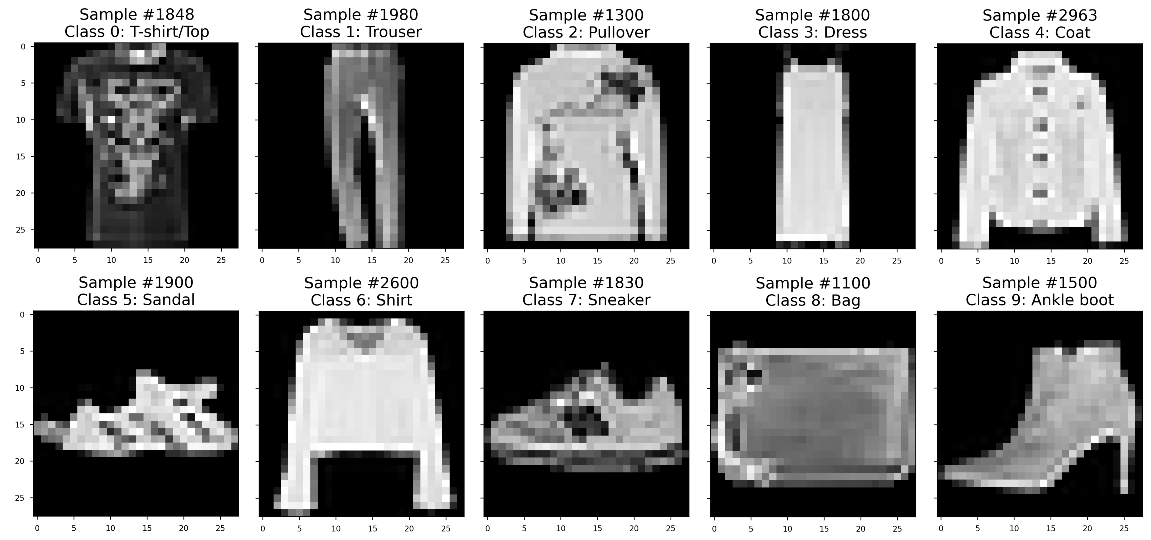
[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Importance of feature scaling. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html, 2011. Accessed September 21, 2021.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, R. Prettenhofer, P.and Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Pytho. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, 2011. Accessed September 21, 2021.

[6] Dr Nguyen Hoang Tran. Dimensionality reduction, week 6b. *COMP5318 Machine Learning and Data Mining*, September 2021. University of Sydney.

# Appendices

## A Classes



## B Hardware

| Hardware Overview | | |
|---|---|---|
| Model Name | MacBook Pro | HP 14ck14000 |
| Processor Name | Quad-Core Intel Core i5 | Intel® Core(TM) i3-7 |
| Processor Speed | 2 GHz | 2.30GHz |
| Number of Processors | 1 | 1 |
| Number of Cores | 4 | 2 |
| L2 Cache (per Core) | 512 KB | 512 KB |
| L3 Cache | 6 MB | 3 MB |
| Memory | 16 GB | 16 GB |

## C Software

| Software Overview | |
|---|---|
| Name | Version |
| Windows HSL | 10, 64 bits |
| macOS BigSur | 11.4 bits |
| Python | 3.8.10 |
| Jupyter core | 4.7.1 |
| Jupyter Notebook | 6.4.0 |
| tqdm | 4.61.1 |
| seaborn | 0.11.1 |
| scikit-learn | 0.22.1 |
| pandas | 1.2.5 |
| numpy | 1.19.2 |
| more-itertools | 8.8.0 |
| matplotlib | 3.3.4 |
| h5py | 2.10.0 |