
SAPIENZA
UNIVERSITÀ DI ROMA

Homework 1 report

José Manuel Del Valle Delgado 1848580

# Table of Contents



# Introduction

Our goal is to determine the best models for two classification tasks with two different input spaces. The two datasets have the same length in terms of samples (i.e., ~50,000). The only difference is that the datasets have 100 and 1000 features, respectively. The first part is equal across the two datasets since they are very similar. We have no information about the nature of the datasets; some assumptions will be made, in particular I will assume that the datasets are mostly noise-free. Since no information about the nature of the datasets was provided, we cannot make assumptions about the hypothesis space. Thus, we have to choose the best algorithm based on performance, in the first case, and on the internal information that it carries, in the second case.

# Medium size dataset

### Data mining

The first thing that we should do is build intuition from the data. A good way to do this is to visualize the data. We have an input space of 100-dimensional features, so we need a hyperplane large enough to fit all our data. We are limited in how many dimensions we can visualize, so we need to figure out a way to do it in three dimensions or fewer and in a way that the reduction is still significant in a graphical way to gain insights. A popular way to accomplish this task is to use t-SNE; it essentially performs a non-linear transformation of an N-dimensional space into a lower-dimensional space of dimension $D \leq 3$. Applying t-SNE is computationally expensive, meaning that applying it directly to the dataset is slow, with a runtime of approximately ~5 minutes per run on my machine.

Even though t-SNE is widely used for accomplishing this task, a drawback is its great dependence on the 'perplexity' parameter. We would like to test if the previous graph still holds with different configurations of perplexity. However, as previously stated, if every run takes ~5 minutes to complete, we are limited in the way in which we can make fine-tuning.

As explained in the introduction, we have no information about the data, but our goal remains to get a model that best fits future unseen data, regardless of

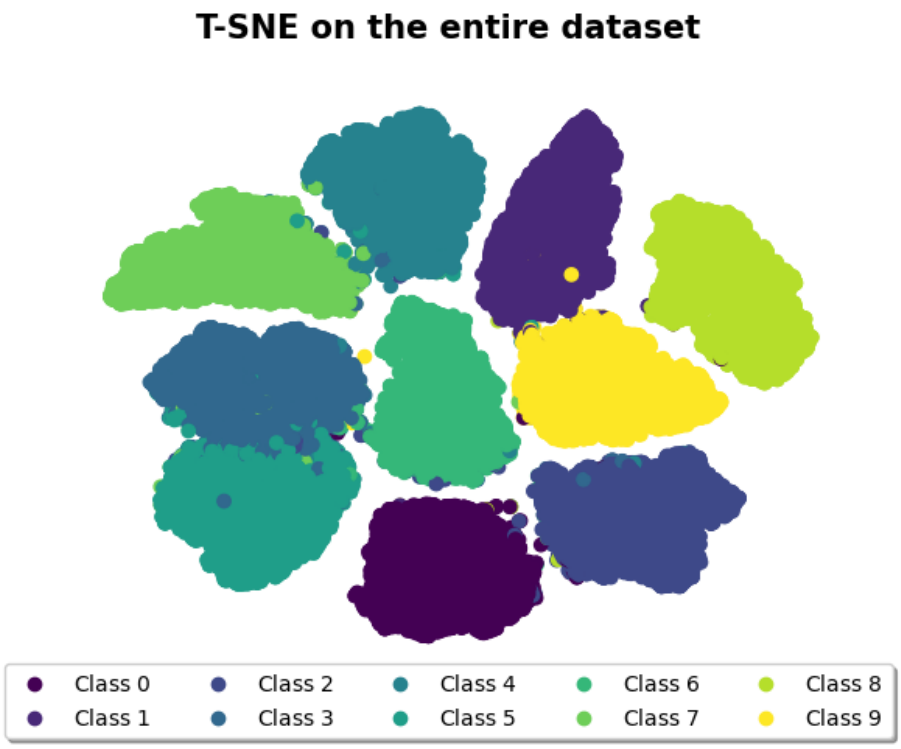
T-SNE on the entire dataset
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9

Figure 1: png

the information that we have. In this context, a reasonable assumption is that the data is generated by the same prior probability distribution. Therefore, in order to reduce the time of t-SNE, we could leverage it and scale our dataset to be smaller but without losing information about the prior probability.

Another factor that slows down t-SNE is the presence of many attributes, and not all of them may be useful in terms of the information they convey. We can determine the principal components of this dataset by using PCA. PCA determines the principal components with the insight that higher variance components carry the most information. To reduce the number of total dimensions, we can take the first 'n' components that hold a certain percentage of information. For example, in our dataset, around 9 components carry as much as 98% of the total information about the dataset. In other words, by discarding 91 dimensions, we only lose around 2% of information.

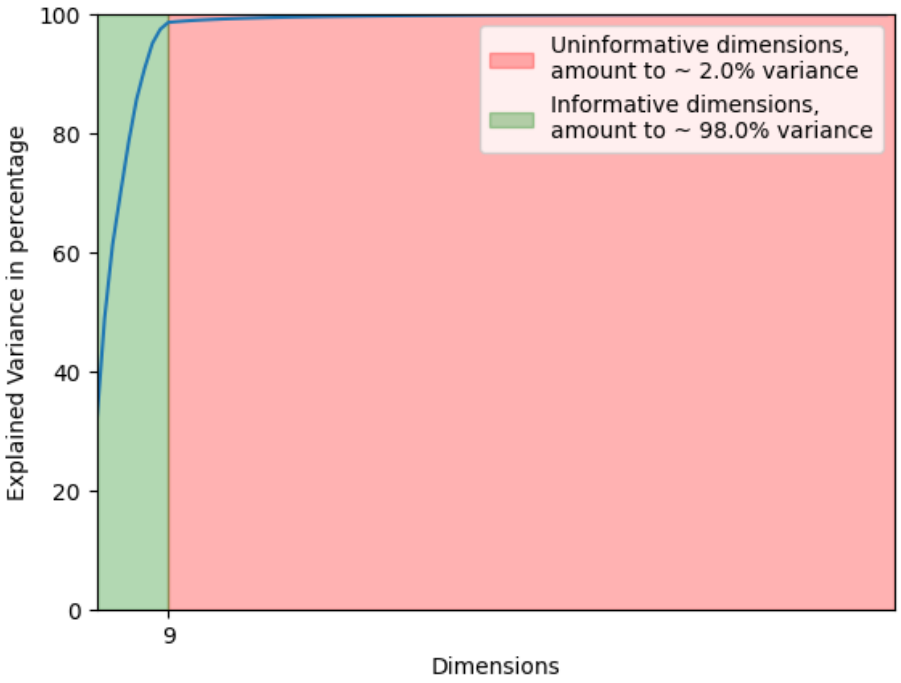


Figure 2: png

After removing 81 dimensions and reducing the dataset to about 10% of the initial size, we can check if the first graph still holds. It seems that it is indeed the case. But before proceeding, one thing I should point out is this: t-SNE makes a non-linear transformation. Therefore, after seeing this graph, we cannot say, for example, that class 8 and 9 are linearly separable because, after a non-linear transformation, we lose information about linearity. What we're interested in

with t-SNE is the fact that it models high-dimensional objects in a way that similar objects are modeled nearby, and different objects are distant. So, we should pay attention to class 3 and 5, but let's not get ahead of ourselves.

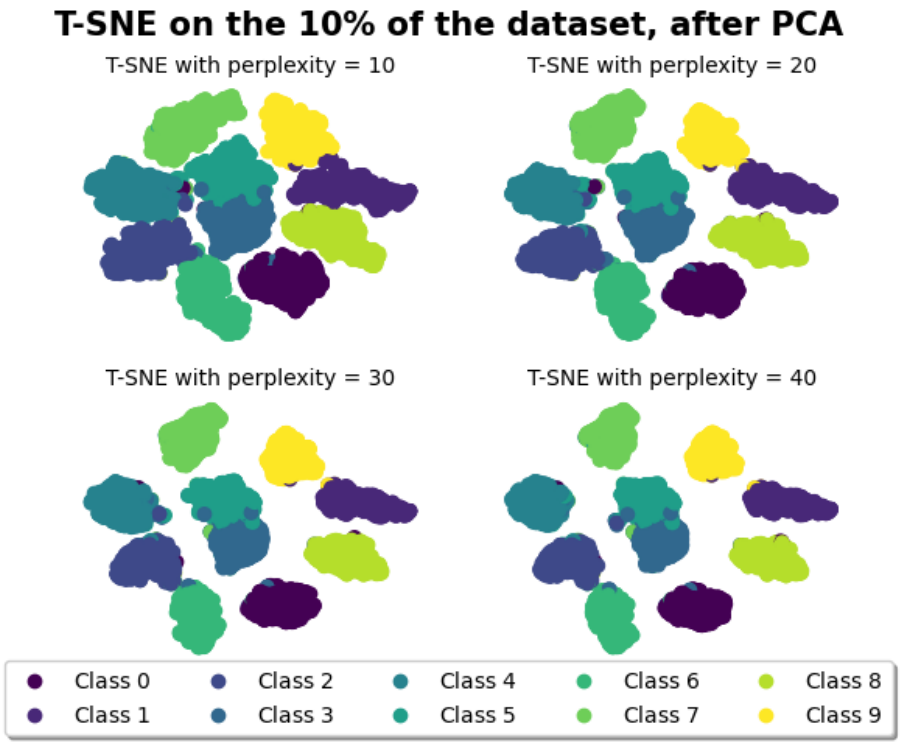


Figure 3: png

Lastly, before moving on, we simply note that the dataset is balanced. This too will be useful later.

### Model Selection

The first thing we can do is start trying different learning agorithms and choose the best based on the performance. We can avoid doing that. Generally speaking multiclass classifiers can be seen as a set of binary classifiers that chooses the class of the prediction using a strategy One vs One or One vs Other, in both cases we have a set of binary classificators. In this regard SVM is the best classifier, it differs from the other classifiers because it not only selects a generic decision boundary, but chooses which maximizes the distance from the closest data points of all classes, called maximum margin decision boundary. By picking the decision boundary with maximum margin we reduce the generalization error, and so our model will perform better on unseen data (intuitively we have more leeway to make mistakes on both sides). We don't know if the data is linearly

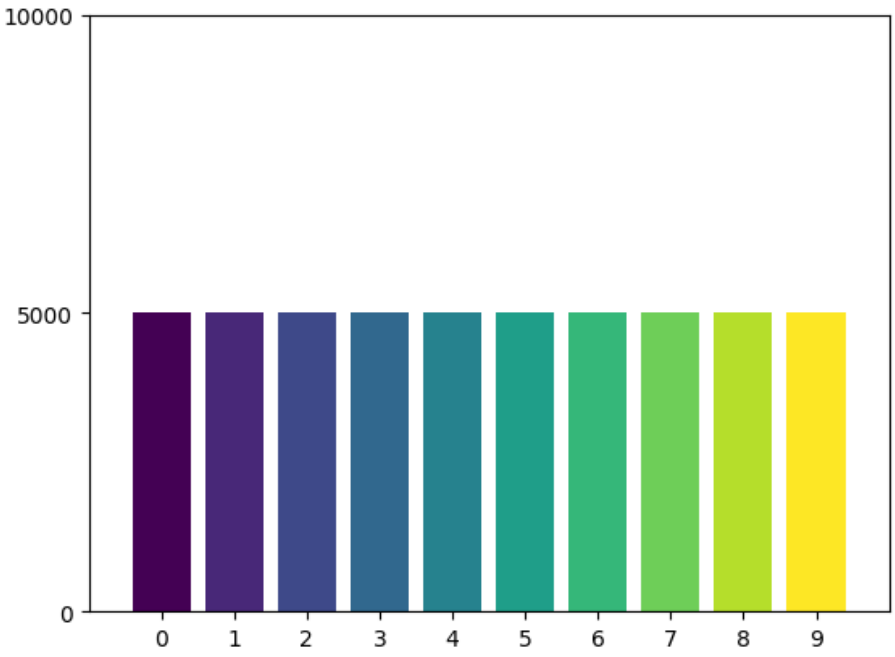


Figure 4: png

separable or not, but by using the kernel trick we could choose a kernel such that the algorithm will still choose a valid decision boundary. But nothing is free and resorting to the kernel trick means that we're locked in the dual problem, thus we have to deal with a time complexity of $O(n^3)$, where $n$ is the number of samples. This dataset is still small enough, and after the PCA we were able to reduce the number of attributes from 100 to 9, so using SVM is still manageable. Since the dataset is balanced and we have no information about the objective that we're trying to maximize the accuracy gives us a great measure of the quality of the model. We can use K-fold cross-validation, in this way we can evaluate the model multiple times and decide do use the model that performs better on average.

Linear SVM

Radial Basis Function SVM

Cross-validation splitting strategy

10.000000

10.000000

Accuracy mean

0.987400

0.988100

Accuracy standard deviation

0.002757

0.002558

The SVM with a Radial Basis Function is the winner, as it outperforms (even if only slightly) SVM with a linear kernel. Even if the standard deviation is not too high (all iterations yielded an accuracy scorre really close to the mean), the mean accuracy seems to be too high, indicating a potential issue of overfitting the data, implying low generalization power. Fortunately, during the K-fold validation, we kept a portion of the dataset for testing. Using this test set, we obtain an accuracy of ~98.7%, which aligns with the data collected during the K-fold phase.

This means that, rather than overfitting, we're predicting the real function $f$ that generated the data very well. Due to several reasons such as the unknown nature of $f$, lack of information about the hypothesis space, the potential presence of noise in the data, and the finite size of the dataset, reaching 100 accuracy is not feasible anytime soon. While this is normal, it also implies that somewhere we're making mistakes. These mistakes take the form of misclassifications, and in order to visualize where we're making mistakes, it comes in handy to use something called a confusion matrix.

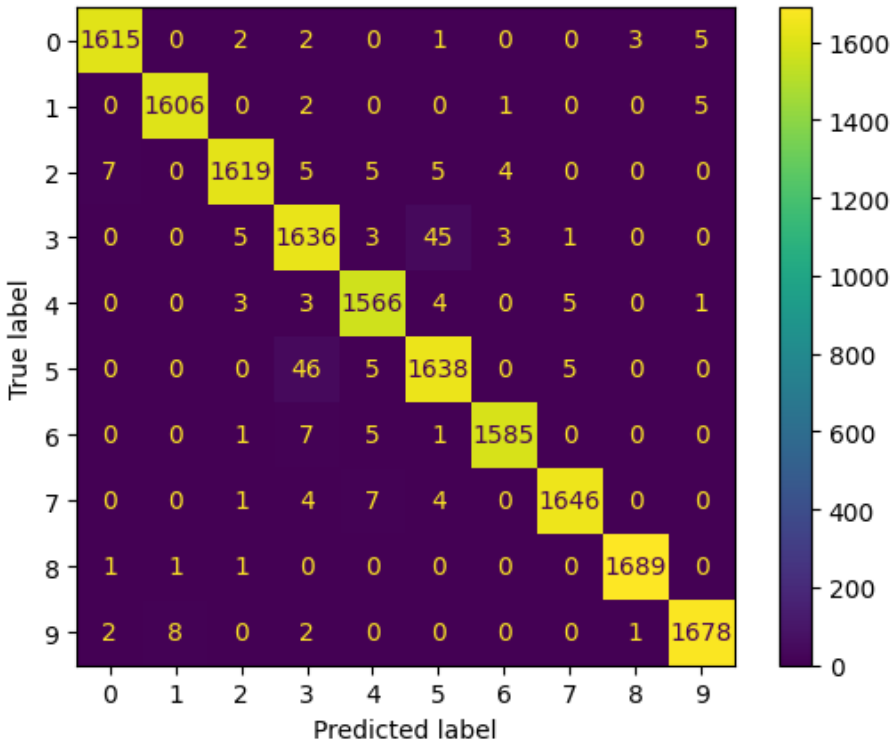
True label
Predicted label
0 1615 0 2 2 0 1 0 0 3 5
1 0 1606 0 2 0 0 1 0 0 5
2 7 0 1619 5 5 5 4 0 0 0
3 0 0 5 1636 3 45 3 1 0 0
4 0 0 3 3 1566 4 0 5 0 1
5 0 0 0 46 5 1638 0 5 0 0
6 0 0 1 7 5 1 1585 0 0 0
7 0 0 1 4 7 4 0 1646 0 0
8 1 1 1 0 0 0 0 0 1689 0
9 2 8 0 2 0 0 0 0 1 1678
0 1 2 3 4 5 6 7 8 9
1600
1400
1200
1000
800
600
400
200
0

Figure 5: png

Based on the confusion matrix, we observe that we tend to misclassify mostly classes 3 and 5, as previously noted. Let's revisit the t-SNE plot with perplexity 30 and focus only on those two classes.

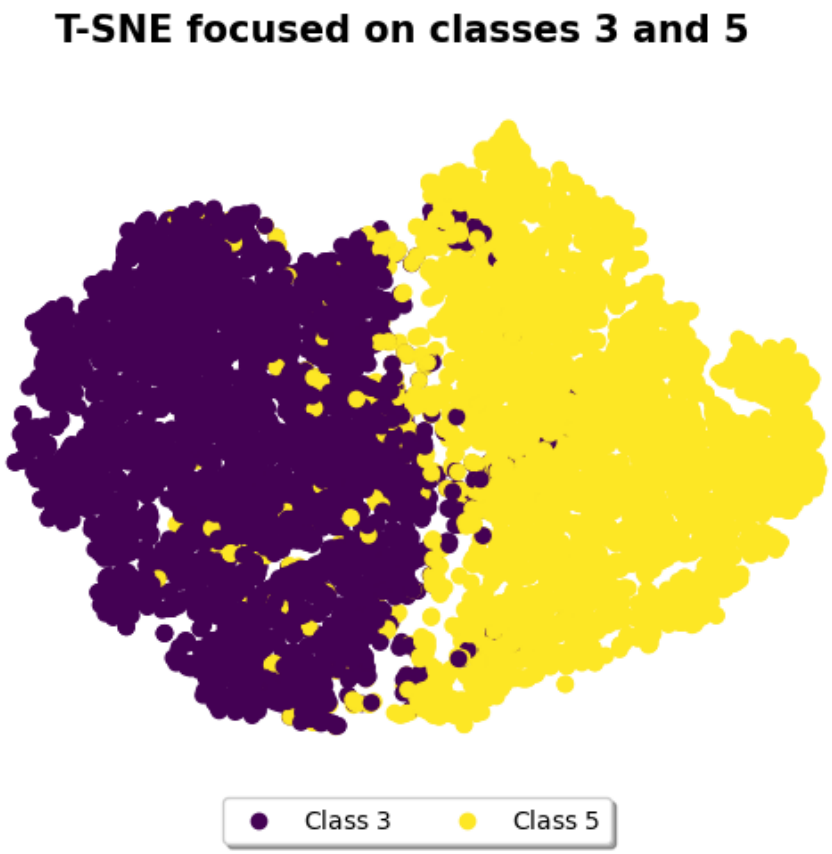


Figure 6: png

In the t-SNE graph of those two classes, we observe that they seem to almost merge. We didn't explore this idea before, but let's do it now. In t-SNE, clusters contain elements that are closer in the original space while trying to keep well-separated from relatively distant points. In this case, the classes merge, meaning that in the original space, these elements were really close to each other.

We can take a step further in this analysis and note that this notion of distance can be seen as a measure of similarities. Thus, some elements of class 3 and 5 are very similar, and as a result, we have some errors classifying them.

# Large size dataset

### Data Mining again

As discussed in the introduction the datasets are very similar, so the datamining phase is mostly the same with some numbers changed.

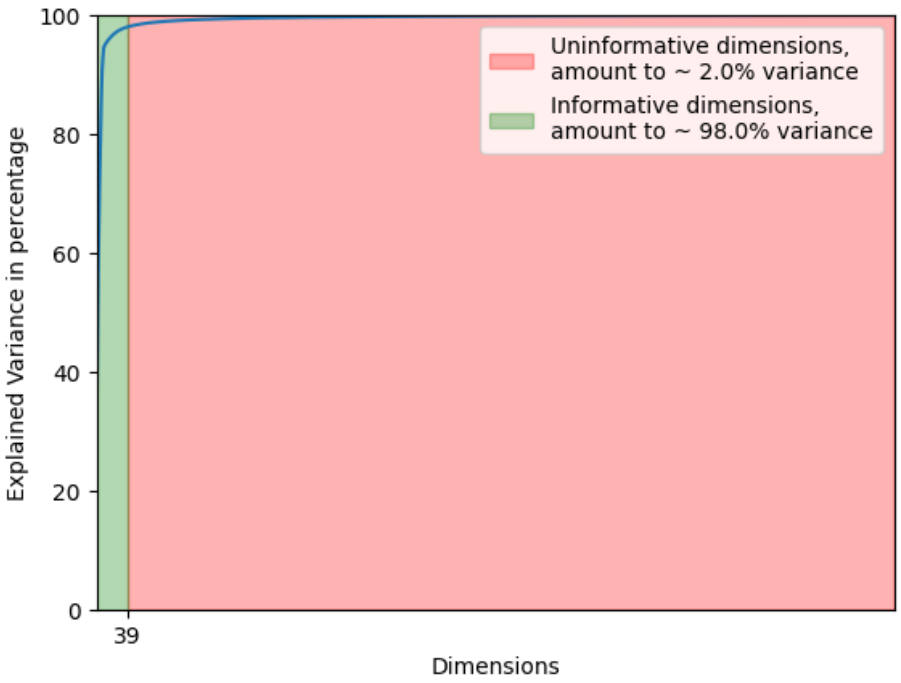


Figure 7: png

Given what we've discussed in the medium dataset we can already say that we will have some troubles differentiating class 3,4,5,6 as the y seem to have created a cluster, but we will come to it later.

Lastly befome moving to model selection we check if the data is balanced.

### Model selection again

What was previously discussed about SVM still holds, but the classification task is easy enough, and we can consider choosing another model. In particular, we'll keep an eye on random forests.

Random forests are an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the

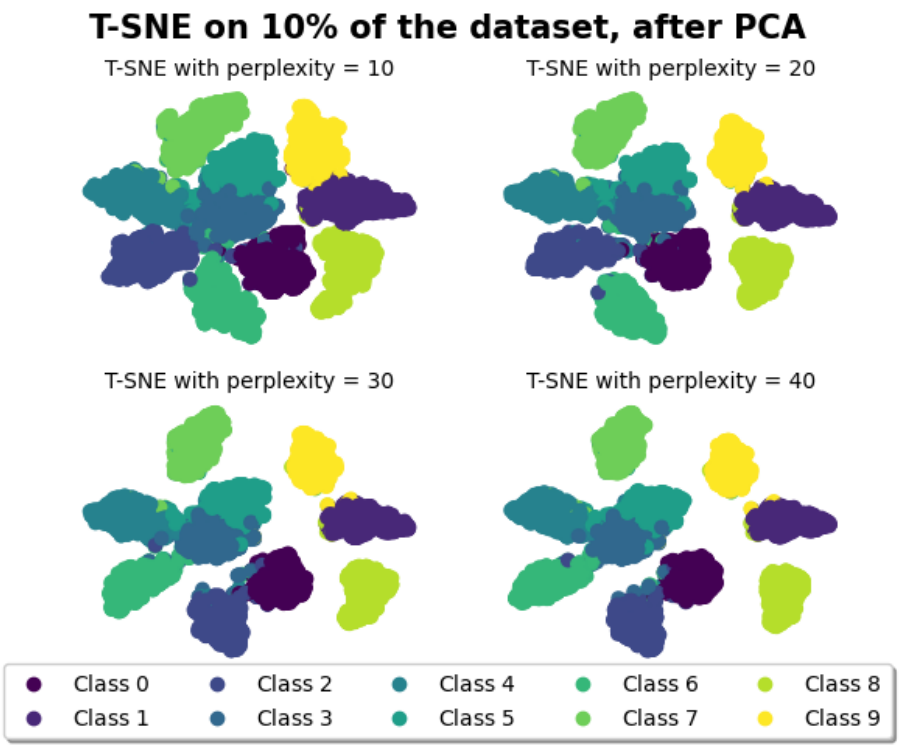
T-SNE on 10% of the dataset, after PCA
T-SNE with perplexity = 10
T-SNE with perplexity = 20
T-SNE with perplexity = 30
T-SNE with perplexity = 40
Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9

Figure 8: png

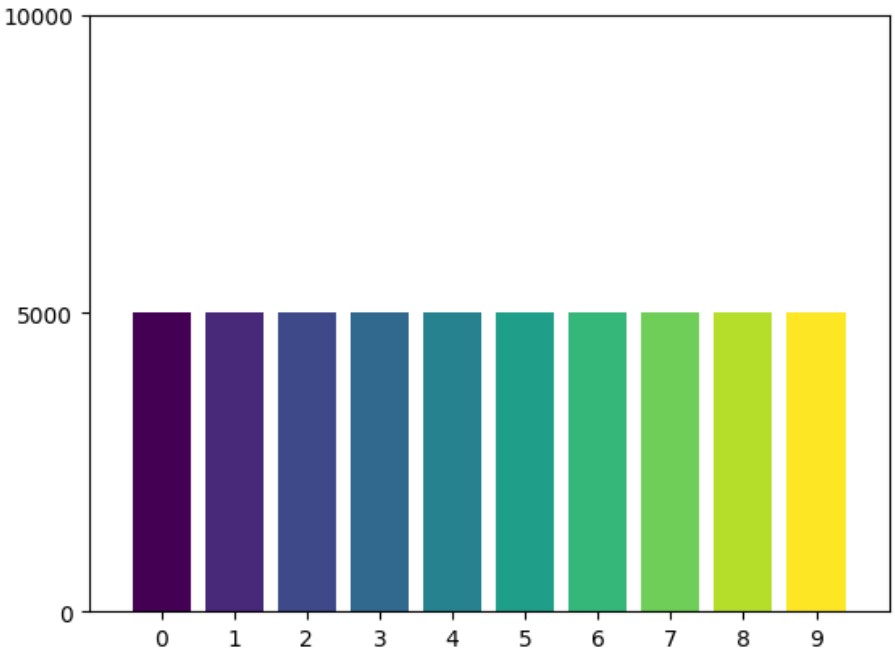


Figure 9: png

individual trees. They offer the advantage of being less prone to overfitting and providing a good balance between bias and variance.

Given the characteristics of our dataset, including its reduced size after PCA and the balanced nature of the classes, random forests might offer a more computationally efficient alternative with the potential to capture complex relationships within the data, and direct informations about the decision process (more on this later).

As before, we can use K-fold cross-validation to evaluate the performance of the random forest model and select the configuration that performs the best on average across the folds. This will help us make an informed decision about which model to choose for our classification task.

Random Forest

Radial Basis Function SVM

Cross-validation splitting strategy

10.000000

10.000000

Accuracy mean

0.972500

0.973900

Accuracy standard deviation

0.003308

0.003315

Technically speaking, SVM is slightly better than random forests, but we still reach a 96.9% accuracy, so we're not losing much. However, we gain the fact that random forest makes decisions about the classification result by using the classification that has the highest probability of being true.

If we visualize the confusion matrix...

From the confusion matrix, we gather that, as before, there is a bit of confusion between 3 and 5, and they tend to be swapped. But if we take a good look at the matrix, it doesn't really describe what we expect.

By looking at the t-SNE graph, we expected higher misclassifications among classes 3, 4, 5, and 6. So, let's take another look at the t-SNE graph and just focus on those classes.

We'll also take two particular points that will highlight why this peculiarity happens.

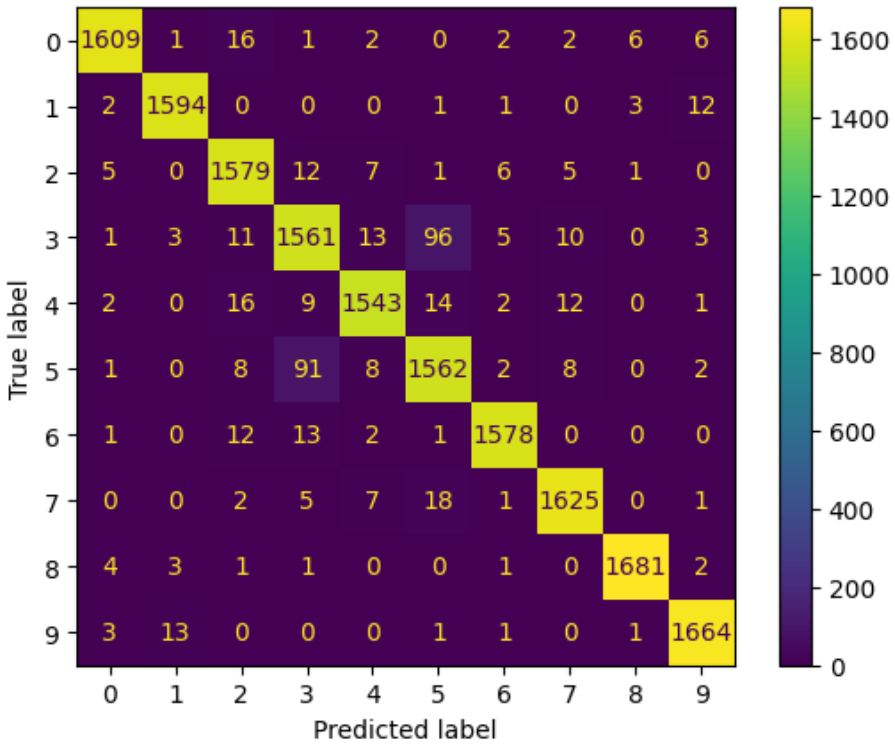


Figure 10: png

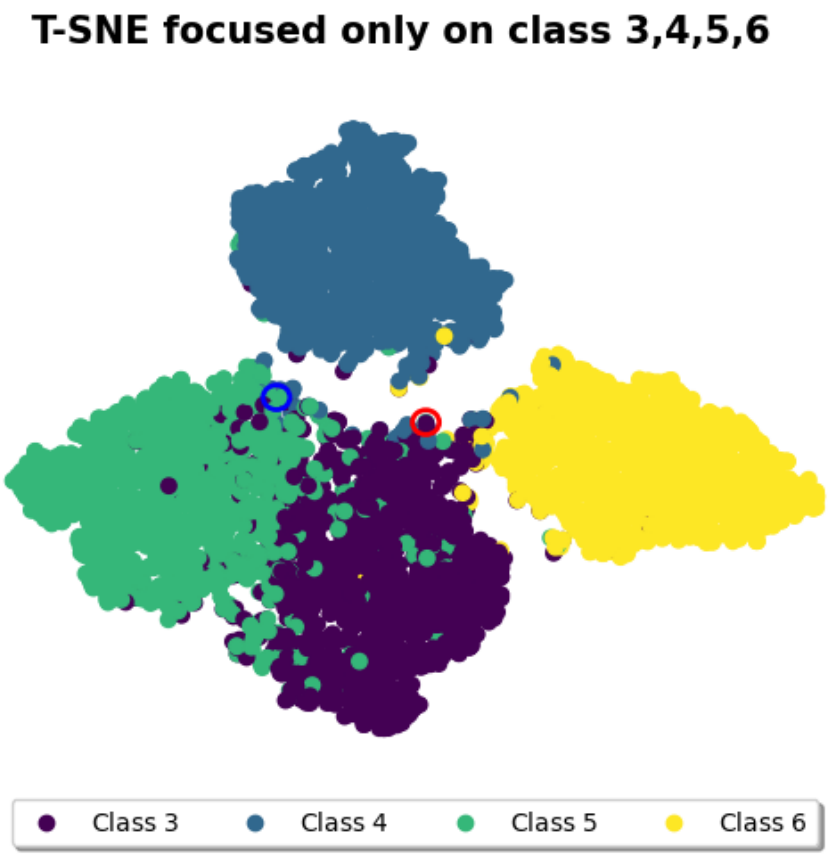


Figure 11: png

What's interesting about these two points is that the blue one has been correctly classified, while the red one has not. However, since we're using decision trees, we can query the probability of being part of a given class. Technically, this can also be done with SVM, but decision trees are more suited for this kind of thing. This results in a ten-dimensional array (there are ten possible classes) in which every element gives us the information of being classified as being of the class equal to the index.

Probabilities of Red Circle

Probabilities of Blue Circle

Probability of being of class 0

1.0

0.0

Probability of being of class 1

1.0

0.0

Probability of being of class 2

1.0

0.0

Probability of being of class 3

32.0

20.0

Probability of being of class 4

22.0

32.0

Probability of being of class 5

29.0

48.0

Probability of being of class 6

1.0

0.0

Probability of being of class 7

3.0

0.0

Probability of being of class 8

3.0

0.0

Probability of being of class 9

7.0

0.0

Real value

5.0

5.0

Prediction

3.0

5.0

What is remarkable is that these two particular points have high uncertainty around being classified as class 3, 4, or 5, which is in line with what we saw in the t-SNE graph. Another interesting aspect is the fact that if the subset on which we had done the training would change, these points might get interchanged (the red one being correctly classified while the blue one is not). This sensitivity to the training subset highlights the complex relationships and potential overlaps between classes in our dataset.

## Resources

Is normalization always good ?

How to visualize features in case of classification problem?

How to yse t-SNE effectively

Why is PCA often used before t-SNE

t-SNE clearly explained

SVM why look for maxim margin

Scikit documentation about t-SNE

What is Principal Component Analysis and how it is used

Hands on Machine Learning with Scikit-Learn, Keras and TensorFlow, 2nd Edition Hands on Machine Learning with Scikit-Learn, Keras and TensorFlow, 2nd Edition - github repo Bishop Pattern Recognition and Machine Learning Machine Learning, Tom Mitchell