University of Science, VNU-HCM

Mathematics and Computer Science

Pattern Recognition
Lab 01 Report

Group 1

Member

No	Name	ID	Contribution (%)
1	Tran Nguyen Trung Tuan	22280101	18
2	Dang Minh Phuc	22280064	20
3	Le Hong Cat	21110249	14
4	Truong Minh Hoang	22280034	20
5	Nguyen Thanh Kien	22110092	14
6	Le Hoang Nguyen	22280061	14

November 3, 2024

Contents

1 Introduction & Dataset

1.1 Introduction

In this assignment, we aim to classify fashion items from the Fashion-MNIST dataset using machine learning models. Specifically, we will explore the performance of linear models, including logistic regression and Support Vector Machines (SVM), on this dataset. The Fashion-MNIST dataset provides a modern, more complex alternative to the traditional MNIST dataset of handwritten digits, offering images of various clothing items. These items are categorized into 10 distinct classes, and each image consists of 28x28 grayscale pixels, translating to 784 features.

A key focus of this assignment is to examine how the curse of dimensionality influences model performance. The curse of dimensionality refers to the problems that arise when working with high-dimensional feature spaces, where models can become less effective due to sparsity and increased computational requirements. To mitigate this issue, we will apply dimensionality reduction techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), reducing the dataset's feature space and evaluating the impact on model performance.

Our goal is to classify the items effectively and analyze how different models, such as logistic regression and SVM (both linear and RBF kernel versions), perform in various settings, including full-dimensional and reduced-dimensional data. Furthermore, we will compare the computational efficiency and accuracy of these models in order to better understand the trade-offs involved in choosing the right model and dimensionality reduction approach for high-dimensional datasets.

1.2 The Fashion-MNIST dataset

The Fashion-MNIST dataset, introduced by Zalando Research, contains 70,000 grayscale images of fashion products across 10 categories. The dataset is split into a training set of 60,000 images and a test set of 10,000 images. Each image has a resolution of 28x28 pixels, yielding 784 features per image when flattened into a one-dimensional vector. The 10 classes in the dataset represent different clothing items, including:

1	2	3	4	5	6	7	8	9	10
T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot

Table 1: List of items.

Each image is labeled with its corresponding class, and the goal is to classify these fashion items based on their pixel values correctly. As the dataset contains a substantial number of features (784), it presents a challenge for machine learning models, particularly regarding computational efficiency and the potential impact of the curse of dimensionality.

In this assignment, we will perform several preprocessing steps such as data normalization, which ensures that the pixel values range between 0 and 1, making the dataset more suitable for machine learning models. We will also visualize some of the sample images and perform basic exploratory data analysis (EDA) to understand the distribution of the classes. This will allow us to better interpret the model results and identify any imbalances or patterns within the dataset that could affect the performance of the classification models.

2 Model's performance.

2.1 Full datasets.

First, we train with the full dataset with 2 models Logistics Regression and SVM (Linear Kernel & RBF Kernel). We will evaluate the model training by the accuracy metric, and evaluate the testing by the metrics: Accuracy, F1-score, Precision score, and Recall score.

2.1.1 Linear Model - Logistics Regression.

First, we performed the Logistics Regression model for multi-classes and obtained the following results:

	Accuracy	**F1 - Score**	**Precision Score**	**Recall Score**	**Time (s)**
Training	0.876	-	-	-	6.267
Testing	0.852	0.851	0.850	0.852	0.058

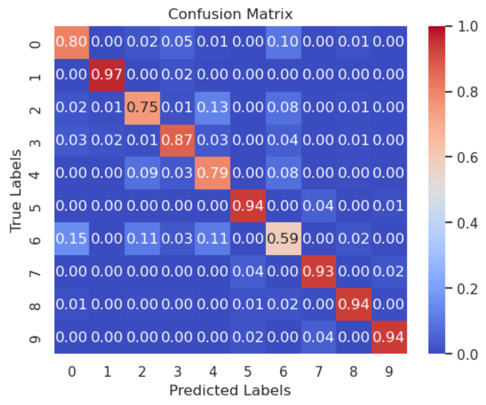


Figure 1: Confusion matrix of Logistics Regression.

The accuracy of the training set and the test set differ very little, indicating no overfitting. Additionally, the high accuracy suggests that underfitting is also not an issue. Most classes have high accuracy, over 0.7, with the sole exception of Class 6, which has an accuracy of 0.59. Class 6 is misclassified as Class 0 in 15% of cases, Class 2 and Class 4 in 11% of cases.

2.1.2 Support Vector Machine - SVM.

Because this is a problem where the number of labels of each class is the same (not being imbalanced) and the amount of data is large, we will use the OVR (One-vs-Rest) method (default in sklearn).

There are 2 types of kernel functions we will use:

- Linear kernel:

$$K(x, x') = x^T x'$$

- RBF (Radial Basis Function) kernel:

$$K(x, x') = e^{-\gamma \|x - x'\|^2}$$

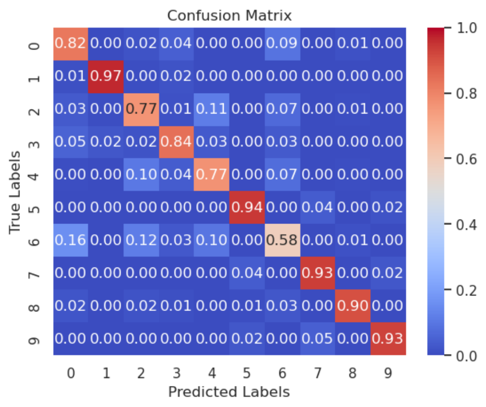
We obtained the following results:

	Accuracy	**F1 - Score**	**Precision Score**	**Recall Score**	**Time (s)**
Training	0.913	-	-	-	572.372
Testing	0.846	0.845	0.846	0.846	65.001

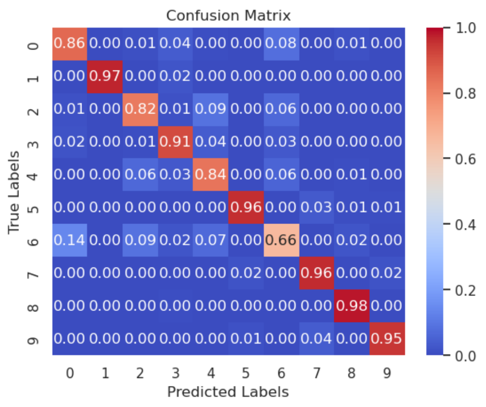
Table 2: SVM (Linear Kernel) performance.

	Accuracy	**F1 - Score**	**Precision Score**	**Recall Score**	**Time (s)**
Training	0.922	-	-	-	232.525
Testing	0.891	0.889	0.889	0.891	128.976

Table 3: SVM (RBF Kernel) performance.



(a) Confusion matrix of SVM (Linear Kernel).



(b) Confusion matrix of SVM (RBF Kernel).

The first observation is that SVM takes a significantly longer time to run compared to Logistic Regression. Additionally, SVM seems to have a slight tendency to overfit, performing better on the training set than Logistic Regression but not performing as well as this model on unseen data.

Since our data is balanced across classes (with 10 classes, 7000 samples per class), we will use accuracy as the main metric to evaluate the models performance.

With the RBF kernel, we observe that SVM performs significantly better on unseen data:

- Our lowest accuracy class, Class 6, improved from 0.56 in Logistic Regression and 0.58 with the linear kernel to 0.66. However, we still see a high level of misclassification of samples in Class 6 as Class 0, which hasn't reduced much (remaining at 0.14%, only a 0.01% improvement compared to Logistic Regression).
- Almost all classes show an increase in accuracy when transitioning from Logistic Regression or SVM with the linear kernel to SVM with the RBF kernel, except for Class 1. This class remained unchanged across all three models, maintaining a high accuracy of 0.97. Furthermore, with the RBF kernel, Class 1 is rarely confused when predicting samples from other classes, indicating that Class 1 still benefits in performance compared to the previous two models.

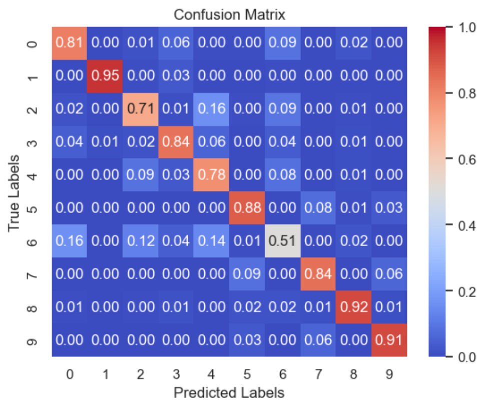
2.2 Reduced datasets.

We apply both Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) for reducing dimension. We choose `n_components = 100` which remains about 84% of information for PCA, and `n_components = 9` which is the minimum of number of features and number of classes for LDA.

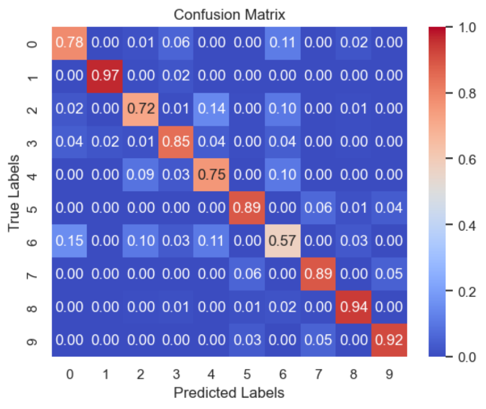
2.2.1 Linear Model - Logistics Regression.

After reducing, training and evaluation, we have a evaluation table below, and confusion matrices:

	Training Time (s)	**Accuracy**	**F1 - Score**	**Precision Score**	**Recall Score**
PCA	0.281	0.812	0.811	0.811	0.812
LDA	0.214	0.827	0.826	0.826	0.828



(a) Confusion matrix of LR applied PCA.



(b) Confusion matrix of LR applied LDA.

2.2.2 Support Vector Machine - SVM.

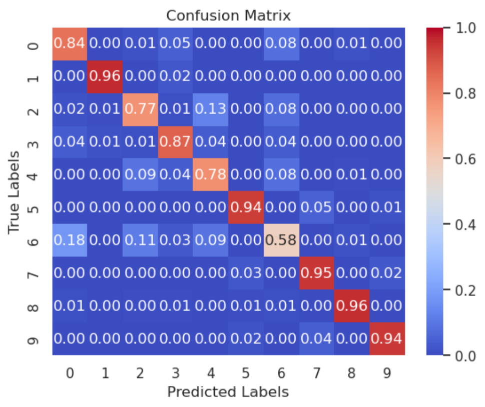
First, we do it with Linear Kernel. We obtained the following results:

	Accuracy	F1 - Score	Precision Score	Recall Score	Time (s)
Training	0.869	-	-	-	193.676
Testing	0.858	0.856	0.856	0.858	11.179

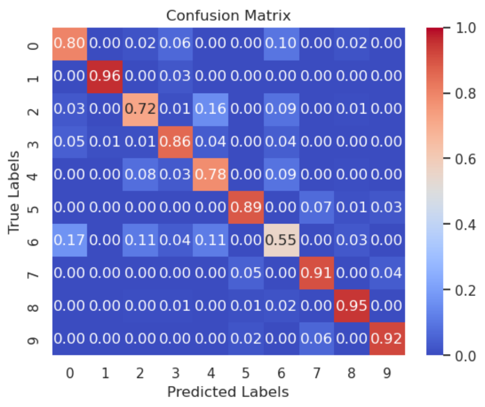
Table 4: SVM (Linear Kernel) applied PCA.

	Accuracy	F1 - Score	Precision Score	Recall Score	Time (s)
Training	0.843	-	-	-	15.793
Testing	0.833	0.832	0.831	0.833	5.109

Table 5: SVM (Linear Kernel) applied LDA.



(a) Confusion matrix of SVM (Linear Kernel) applied PCA.



(b) Confusion matrix of SVM (Linera Kernel) applied LDA.

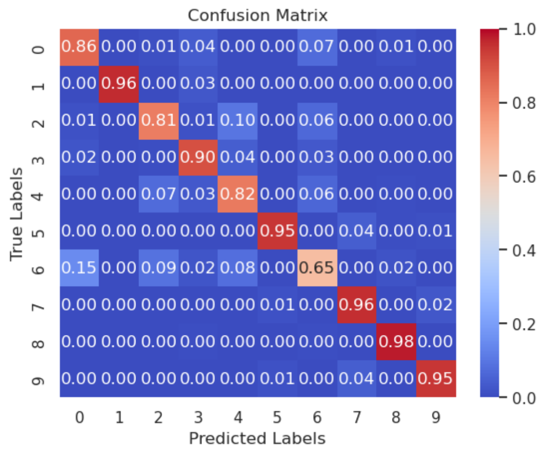
Next, we try with RBF Kernel. We obtained the following results:

	Accuracy	F1 - Score	Precision Score	Recall Score	Time (s)
Training	0.902	-	-	-	55.155
Testing	0.885	0.884	0.884	0.885	20.125

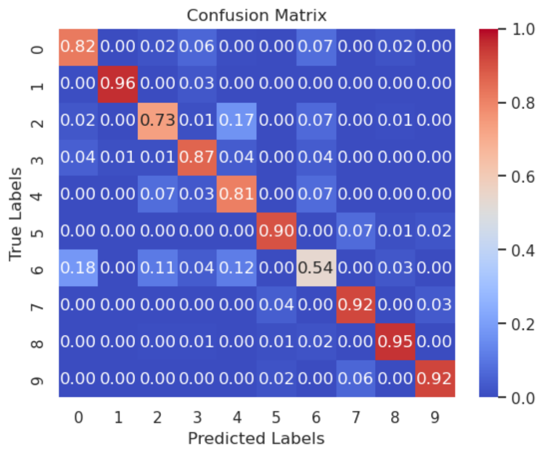
Table 6: SVM (RBF Kernel) applied PCA.

	Accuracy	F1 - Score	Precision Score	Recall Score	Time (s)
Training	0.852	-	-	-	13.756
Testing	0.842	0.840	0.840	0.842	8.501

Table 7: SVM (RBF Kernel) applied LDA.



(a) Confusion matrix of SVM (RBF Kernel) applied PCA.



(b) Confusion matrix of SVM (RBF Kernel) applied LDA.

3 Dimensionality reduction & The curse of dimensionality.

In Machine Learning, Dimension Reduction transforms data from a high-dimensional space to a lower-dimensional one while preserving key characteristics. This reduces issues like data sparsity and processing time caused by the curse of dimensionality. Common in fields with large datasets and many features—such as signal processing, speech recognition, neuroinformatics, and bioinformatics—dimension reduction methods are divided into linear and non-linear types, as well as feature selection and feature extraction approaches. It aids in noise reduction, data visualization, cluster analysis, and as a step in more complex analyses.

3.1 The impact of Dimensionality reduction.

Dimensionality reduction is an essential technique in data analysis and machine learning, designed to retain only the most important features, streamlining data to optimize performance. Some of the effects include:

- **Improving Model Performance**: By reducing data dimensions, algorithms concentrate on key features, avoiding unnecessary or noisy data and improving pattern recognition, accuracy, and predictive capabilities.
- **Reducing Computational Cost**: High-dimensional data demands extensive resources. Dimensionality reduction simplifies data structure, saving processing time and memory—especially beneficial for large datasets and real-time applications.
- **Enhancing Interpretability**: High-dimensional data can obscure feature relationships. Dimensionality reduction techniques like PCA and t-SNE allow for visualization in 2D or 3D, revealing patterns and clusters.

3.2 Mitigates the curse of dimensionality

Dimensionality reduction is essential for overcoming the "curse of dimensionality"—a phenomenon where model performance degrades as the number of dimensions increases:

- **Reducing Overfitting Risks**: High-dimensional models tend to capture noise rather than true patterns. Lowering dimensions removes uninformative features, reducing noise and overfitting, helping models focus on meaningful data structures.

- **Preserving Essential Data Structure**: Techniques like PCA retain maximum variance, ensuring the main structure and properties of the data are maintained even with fewer dimensions, preserving data integrity in analysis and modeling.

4 Comparison & some conclusion

4.1 Logistic Regression vs SVM time run on fully dimensionality dataset

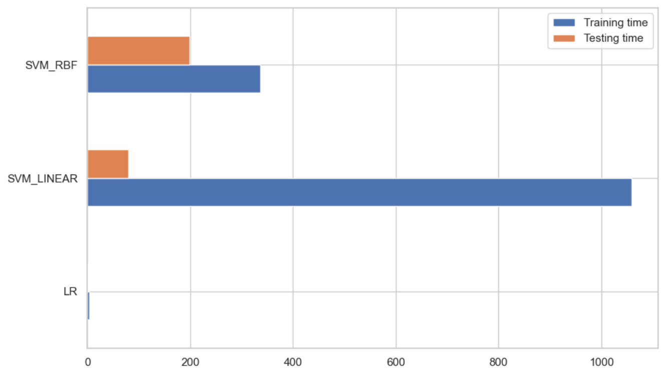


Figure 6: Time run of Models.

With Logistic Regression, since this is a linear model, the running time of both training and testing is extremely fast, almost negligible.

With SVM, we observe that the Linear kernel has a significantly longer training time but faster prediction time than the RBF kernel:

- **Training time**
  - Because our data has many features, using the Linear kernel requires calculations based on the original data features, which takes a lot of time.
  - With the RBF kernel, we do not need to compute each feature individually but rely on the distance matrix, so the training time is faster than the prediction time.
- **Testing time**
  - With the Linear kernel, the prediction process is simpler because it only requires using support vectors to calculate distances in the original linear space.
  - With the RBF kernel, we need to calculate the distance between the new data point and the support vectors in the RBF space, which makes the prediction time longer.

4.2 Logistc Regression vs SVM on all dimensionality versions

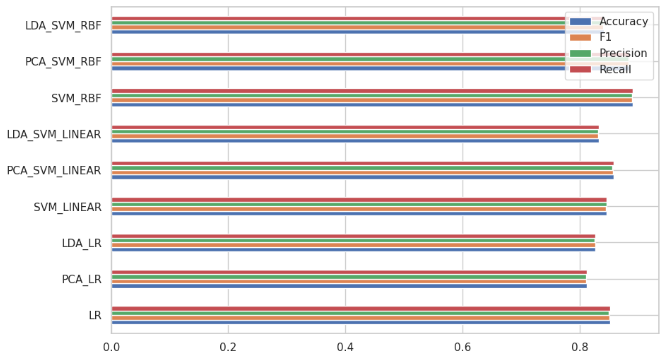


Figure 7: Evaluation of Models.

There are some fantasy thing from plot:

- The bar plot shows the highest evaluation belong to Support Vector Machine with RBF kernel.
- Since the lower dimensionality version has less information than the orginal one, it's performance usually is worse than the fully dimension. However, Linear Regression applied LDA and SVM that use Linear kernel and is applied PCA have better performance than their fully dataset.

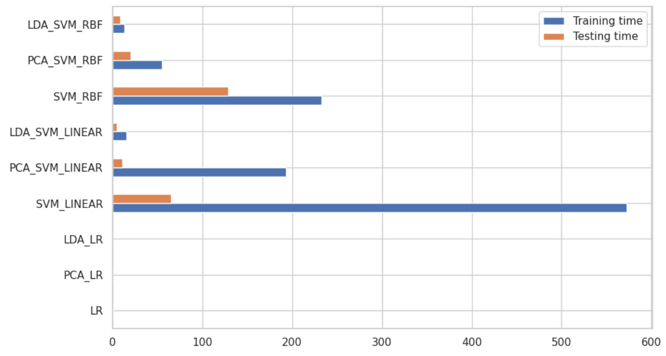


Figure 8: Time run of Models.

As mentioned before, running time of Linear Regression is very fast, while SVM consumes lots of time for training and predition. In constrast, the lower dimension version of SVM takes less time than the fully one. The LDA version spends least time as new dimesion is 9, and RBF kernel runs faster than Linear kernel.

4.3 Pros and Cons

To get to this point, we will compare them using the following table.

	Logistic Regression	**SVM with Linear Kernel.**	**SVM with RBF Kernel.**
Pros	- Easily to understand how the model work - The lower version (LDA) shows a better performance - Fast computation time.	- The only model that improves performance when reducing the dimensionality of the data (using PCA). - Better performance compared to Logistic Regression in full dimension.	- The model with the best performance on both full and reduced-dimensional data - When combined with PCA, it can create a model that is both powerful and has reasonable computation time
Cons	- The performance is not stable as it can not early convergence with low max_iter.	- Highest computation time among the three models. - Tends to overfit.	- Computation time is still long compared to Logistic Regression, but we can still consider this a reasonable trade-off for higher performance.

Table 8: Pros and Cons of Each Model