Dimensionality reductions for classifiers

Ilmari Vahteristo (1107891), Yahya Makhlouf (1107658)

May 2, 2023

Contents

1 Introduction

In this project, the goal was to create a classifier on data about music, and see how different dimensionality reduction methods compare. The dataset used was the *Turkish*

music emotion dataset available at https://archive.ics.uci.edu/ml/datasets/Turkish+Music+Emotion+
The data consists of 400 pieces of music (as 30 second samples), with 51 numeric features for each sample. All the samples were labeled as 'relax', 'happy', angry' or 'sad'. There labels were even, and there was 100 samples of each class.
We use Python to do the analysis and classifier training. First, we perform rudimentary data-analysis, to identify problems, then we perform random grid search to compare different classifiers, dimensionality reductions, and parameters and store the results of the testing to a file. We then analyze the results, and compare the RF and KNN classifiers, and the different dimensionality reductions.

1.1 Preliminary data-analysis

To analyse the data, we checked for missing values and anomalies in the data by doing pairplots. We did not remove any outliers from the dataset.

To gain an understanding, we calculated the relevance of each feature in the dataset wrt. to all other variables, by calculating the mutual information between them (explained later). The mutual information matrix of the dataset is shown in figre 1.

1.2 Dimensionality reduction

Dimensionality reduction is a class of methods to condense the original information in data, to a smaller set, but still contain relevant information. Feature selection, is a subclass of dimensionality reductions, where a subset of the original features are chosen. Another common way is dimensionality reduction through a projection, where the chosen variables are not the same as in the original dataset.

1.2.1 Singular value decomposition

Singular Value Decomposition (SVD) is a matrix factorization method that decomposes a matrix A (n, m) into three matrices:

$$A = U\Sigma V^T$$

where A is the matrix to be decomposed, U is an orthogonal matrix of size (n,n), Σ is a diagonal matrix of size $n \times m$ with non-negative real numbers on the main diagonal, and V^T is an orthogonal matrix of size $m \times m$.

1.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a widely used method for dimensionality reduction, data compression, and feature extraction.
Let X be an $n \times m$ data matrix with n observations and m features. The goal of PCA is to find k orthogonal variables, $Y_1, Y_2, ..., Y_k$, that capture the maximum amount of variance in the data. These variables are defined as linear combinations of

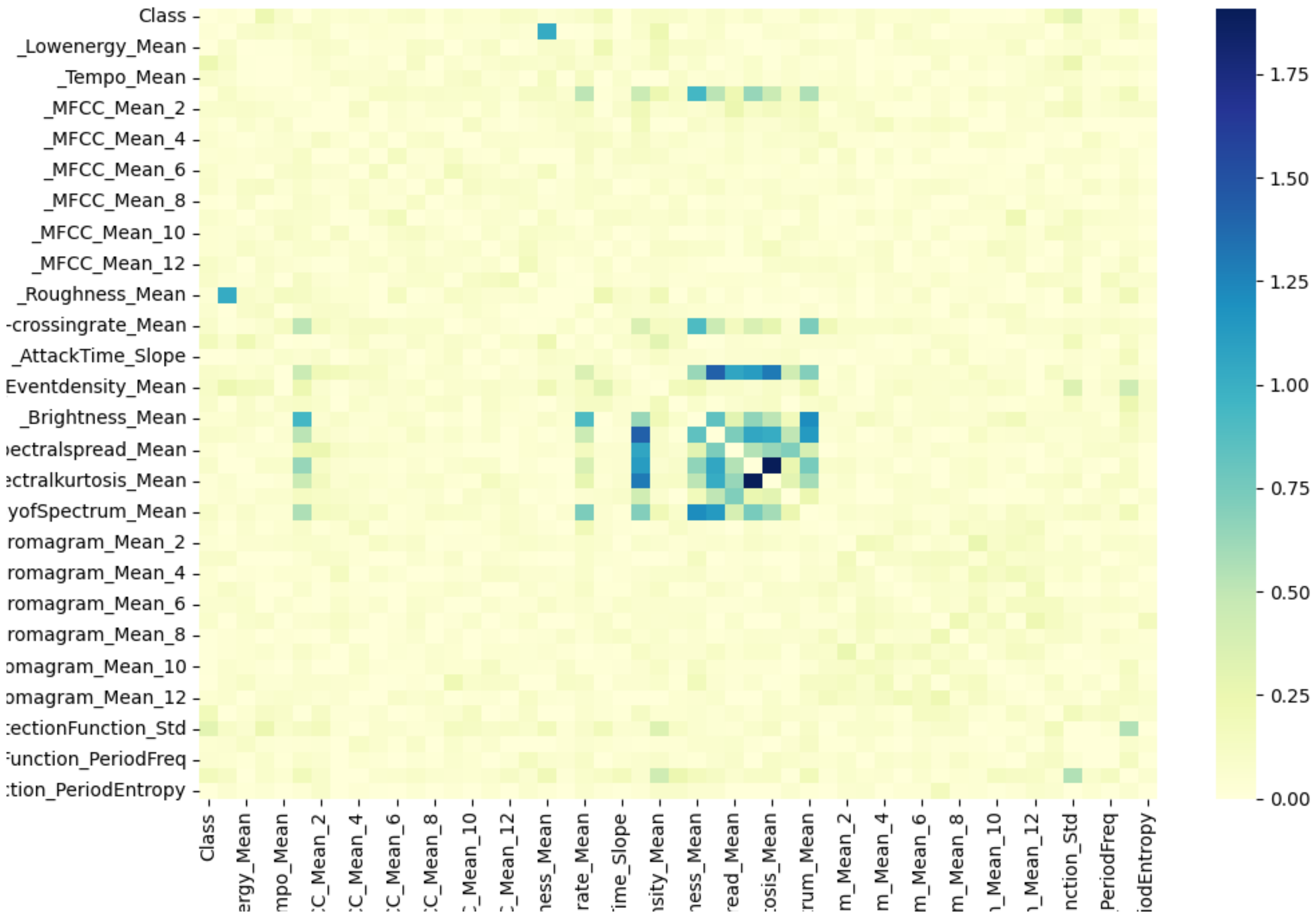

Figure 1: Mutual information between each column in the dataset, including label. The diagonal (features' entropies) are not considered in the heat map. NOTE: Only every second feature is named, due to obvious size constraints.

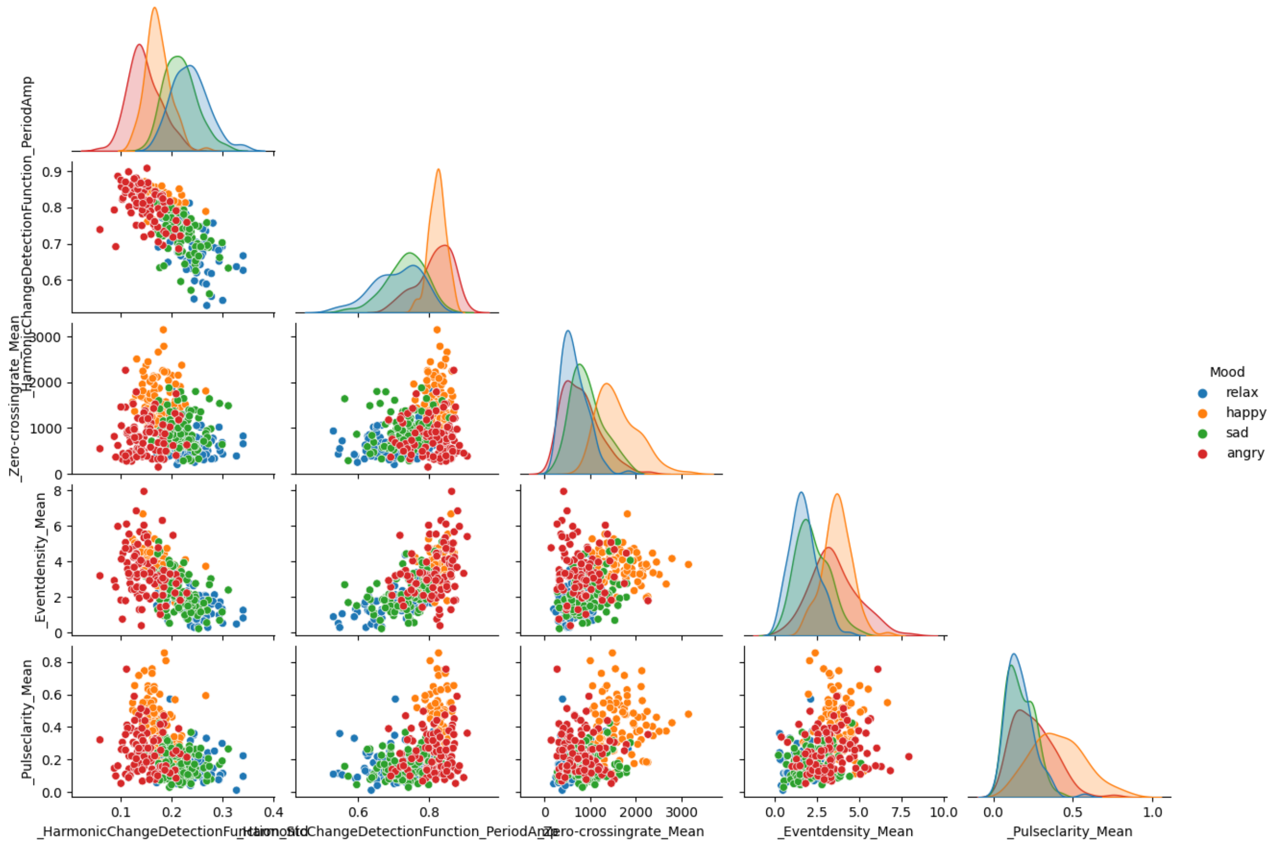

Figure 2: Plot of the distribution of variables colored by the label to which they belong to.

the original variables in X, such that the first principal component Y_1 has the largest possible variance, the second principal component Y_2 has the second largest possible variance, and so on. When using PCA, we can decide the number of features to retain by looking at the percentage of variance explained by each principal component. A common rule of thumb is to retain enough principal components to explain a certain percentage of the total variance, such as 80%. Other techniques for determining the optimal number of principal components include cross-validation and information criteria such as the BIC or AIC. In this case the 80% rule already gives almost the best results possible in a short amount of time.

1.2.3 Forward feature selection with mutual information maximization

Forward feature selection is a feature selection method that aims to select a subset of features that are most relevant to the target variable. The method starts with an empty set of features and then iteratively adds the feature that fits best. The relevance of a feature is measured using mutual information, which quantifies the dependence between two random variables. In the context of forward feature selection, mutual information is calculated between each feature and the target variable, given the already selected features. The feature with the highest mutual information is then added to the set of selected features.

1.2.4 Forward feature selection based on conditional mutual information

CMIM is a feature selection method that follows a forward-selection approach based on an approximation of a criterion. This approximation considers families of features M, where each family contains a unique feature that has already been selected. A feature X can be eliminated if there is another feature X in the selected set that makes X and Y conditionally independent given X. This condition can be expressed as the existence of a k such that $I(Y; X|X_{\nu(k)}) = 0$.

Since mutual information is always positive, this can be reformulated as minimizing $min(I(Y; X|X_{\nu(k)}) = 0$. On the other hand, the higher the value of $(I(Y; X|X_{\nu(k)})$, the more relevant the feature X is. Therefore, the remaining features are ranked based on this criterion, and the one with the highest value is selected.

1.3 Classifiers

1.4 k-Nearest Neighbor

The k-nearest neighbor algorithm (KNN) is a type of model where the function is only approximated locally and all computation is deferred until classification.
Given a new observation $\boldsymbol{x}_0$, the k-NN algorithm first identifies the k points in the training data that are closest to $\boldsymbol{x}_0$, represented by $\mathcal{N}_0$. It then estimates the probability for each class, that is, the fraction of points in $\mathcal{N}_0$ with a given class label. Finally, KNN classifies $\boldsymbol{x}_0$ to the class with the highest estimated probability.

1.4.1 Random Forest

Random Forest is an ensemble learning method for classification and regression. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of class (classification) or mean prediction (regression) of the individual trees. Random Forest corrects for decision trees' habit of overfitting to their training set by creating multiple trees and using them as votes.

In Random Forest, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training data. When splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

1.5 Performance metrics

To compare the performance of a model, we calculated four metrics from using the model on the test data. The used metrics were accuracy, recall, precision, and F1.

1.5.1 Recall(macro)

The formula for macro-recall is given by:

$$\text{Macro-recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i}$$

where n is the number of classes.

1.5.2 Precision(macro)

The formula for macro-precision is given by:

$$\text{Macro-Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i}$$

where n is the number of classes.

1.6 F1(macro)

The formula for macro-F1 is given by:

$$\text{Macro-F1} = \frac{1}{n}\sum_{i=1}^{n}\frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

where n is the number of classes, Precision_i is the precision for class$i$, and Recall_i is the recall for class i.

2 Methodology

A Python script was written to perform and compare the different factorization algorithms. The dataset was split into a training and test dataset with a 75/25 split. A random grid search over classifiers parameters was performed, and the chosen parameters and results were tracked. For each set of parameters, 5, 15, and 25 features were selected/created and all dimensionality reductions compared. A total of 2000 different methods to build a classifier were tested.
The results were then analyzed, and the performance of different dimensionality reductions and classifier parameters were compared.

3 Results

Here we present the results from trying multiple different classifiers and options. For each dimensionality reduction technique, we show the best results for k in $5, 15, 25$ with both classifiers and their parameters.

3.1 Classifier performance on different dimensionality reductions

3.1.1 KNN

Here are the results with the highest accuracy for the KNN classifier, for each dimensionality reduction method.

Reduction	features	standardize	KNN	Weights	Accuracy	Precision	Recall	F1
CMIM	25	True	4	distance	0.50	0.52	0.50	0.49
MIM	5	True	4	uniform	0.78	0.78	0.80	0.78
None	50	True	16	uniform	0.66	0.68	0.68	0.66
PCA	15	True	1	uniform	0.69	0.70	0.70	0.68
SVD	15	True	1	uniform	0.72	0.73	0.72	0.72

Table 1: The best results for each dimensionality reduction method with the KNN classifier.

MIM (mutual information maximization) gives the best results.

3.1.2 Random Forest

Here are the results with the highest accuracy for the Random Forest classifier, for each dimensionality reduction method. Some non-default random forest parameters are not shown here to fit.

Reduction	features	standardize	Trees	Accuracy	Precision	Recall	F1
CMIM	25	True	20	0.65	0.65	0.65	0.64
MIM	15	True	100	0.83	0.83	0.84	0.83
None	50	True	20	0.82	0.83	0.82	0.83
PCA	5	True	100	0.74	0.74	0.76	0.73
SVD	5	True	200	0.74	0.74	0.75	0.73

Table 2: The best results for each dimensionality reduction method with the Random Forest classifier.

As with KNN, MIM gives best results.

3.2 Relevance of features

Relevance of single features was be measured with mutual information, which can capture relationships between the distributions of variables. It is very useful, since it is not directly constrained by specific functions, such as linear correlation measures.

In this case, the 5 most relevant features selected using MIM, ordered by descending MIM scoring are :

1. Harmonic Change Detection Function Standard Deviation
2. Harmonic Change Detection Function Period Amplitude
3. Zero-crossing rate Mean
4. Entropy of Spectrum Mean
5. Event Density Mean

4 Conclusion

In this assignment we used different dimensionality reductions and compared the performance of different classifiers used with the dimensionality reductions.
On the Turkish Music dataset, it seems that selecting features using MIM showed its potency in this task, despite known weaknesses, of possibly selecting redundant features. Surprisingly, CMIM was the worst of the reductions.
From the two classifiers, Random forest was better than KNN, but not significantly.
As seen in tables 1 and 2, the highest classification accuracy for random forest was 83% and 78% for KNN.