# Project 2 - Dimensionality Reduction in Classification

Telmo Cunha (73487)

**Course**: Mathematics for Machine Learning - Instituto Superior Técnico - $2^{nd}$ Semester 21/22

**Introduction** In this project we tackle a classification problem regarding the identification of species of dry beans using features extracted from high-resolution photos. To offer some form of motivation, this is quite an important problem in agriculture since large production volumes require automated methods of classifying seeds and evaluating seed quality. Ultimately, correctly identifying seed species allows for a more efficient segmentation which in turn leads to better quality and more valuable crops. For further information regarding this particular topic see [Murat Koklu, Ilker Ali Ozkan (2020)].

The dataset used in this project can be publicly found at [`https://www.kaggle.com/datasets/muratkokludataset/dry-bean-dataset`]. To offer a brief description of the data, there is a total of 13611 datapoints corresponding to high-resolution photos, from which are extracted 16 features containing pixel counts corresponding to: Area, Perimeter, Major Axis Length, Minor Axis Length, Aspect Ratio, Eccentricity, Convex Area, Equivalent Diameter, Extent, Solidity, Roundness, Compactness, ShapeFactor1, ShapeFactor2, ShapeFactor3, ShapeFactor4 (for a description of these features see [Murat Koklu, Ilker Ali Ozkan (2020)]). Each datapoint is further classified among 7 different species of dry beans given by their Turkish names "Barbunya", "Bombay", "Cali", "Dermason", "Horoz", "Seker" and "Sira".

The project is organized as follows: We begin with a **preliminary data analysis** by identifying the types of available data, performing a summary statistics, identification of missing data, evaluating the presence of outliers and discussing the results and expectations. Next we present methods for **dimensionality reduction**. We begin with **forward feature selection** using two different methods, mutual information with correlation coefficient (**CCMI**) and mutual information based constructive criterion (**MICC**). We then apply **principal component analysis (PCA)** and finally feature selection using the **random forest** importance selection. Next we compare the quality of these dimensionality reduction methods by employing different models on a training set (containing 70% of the data) and studying the performance on a test set (containing the remaining 30% of the data). In particular we shall use **k-Nearest-Neighbours (kNN)** and **Multinomial Logistic Regression (MLR)**. We conclude by discussing the obtained results.

## Preliminary Data Analysis

We follow the guidelines found in chapter 3 of [C. Chatfield and A. J. Collins (1980)], i.e. we start by identifying the types of data present in our dataset, checking if there is missing data and performing an outlier analysis and summary statistics particularly studying correlation between predictors, i.e. the presence of linear relationships between the predictors. We further comment on the issue of transforming variables, in particular why we shall use standardized data.

We note first that all predictors are continuous and that there is no missing data on any predictor. The following table (1), shows the proportion of each class within the dataset.

| No. | Name | Count | Proportion |
|---|---|---|---|
| 1 | Barbunya | 1322 | 9.7% |
| 2 | Bombay | 522 | 3.8% |
| 3 | Cali | 1630 | 12.0% |
| 4 | Dermason | 3546 | 26.1% |
| 5 | Horoz | 1928 | 14.2% |
| 6 | Seker | 2027 | 14.9% |
| 7 | Sira | 2636 | 19.4% |
| **Total** | | 13611 | |

**Table 1: Class distribution**

We note here that the lower representation of Bombay beans might result in lower performance of the models on this particular class. The following table (2), shows a summary statistics of the data.

| No. | Feature | Mean | SD | Min. | Max. |
|---|---|---|---|---|---|
| 1 | Area | 53048.28 | 29324.10 | 20420.00 | 254616.00 |
| 2 | Perimeter | 855.28 | 214.29 | 524.74 | 1985.37 |
| 3 | Major Axis Length | 320.14 | 85.69 | 183.60 | 738.86 |
| 4 | Minor Axis Length | 202.27 | 44.97 | 122.51 | 460.20 |
| 5 | Aspect Ratio | 1.58 | 0.25 | 1.02 | 2.43 |
| 6 | Eccentricity | 0.75 | 0.09 | 0.22 | 0.91 |
| 7 | Convex Area | 53768.20 | 29774.92 | 20684.00 | 263261.00 |
| 8 | Equiv. Diameter | 253.06 | 59.18 | 161.24 | 569.37 |
| 9 | Extent | 0.75 | 0.05 | 0.56 | 0.87 |
| 10 | Solidity | 0.99 | 0.00 | 0.92 | 0.99 |
| 11 | Roundness | 0.87 | 0.06 | 0.49 | 0.99 |
| 12 | Compactness | 0.80 | 0.06 | 0.64 | 0.99 |
| 13 | Shape Factor 1 | 0.01 | 0.00 | 0.00 | 0.01 |
| 14 | Shape Factor 2 | 0.00 | 0.00 | 0.00 | 0.00 |
| 15 | Shape Factor 3 | 0.64 | 0.10 | 0.41 | 0.97 |
| 16 | Shape Factor 4 | 1.00 | 0.00 | 1.00 | 7.00 |

**Table 2: Statistical properties of features (in pixels)**

It is clear from table (2) that some predictors have an absurdly large variance relative to the others, particularly **Area** and **ConvexArea**. This might be an issue later when

performing PCA and when applying kNN which already motivates a transformation of the data.

In appendix, table (14), one can see the correlation matrix for all features. From it, we notice that a few predictors are highly correlated, thus we expect that one of them would be enough for prediction purposes, since the information provided by the remaining ones would be redundant. In particular, the correlations between **Area** and **Convex Area** is 1, up to three decimal places. This suggests that the shape of the beans is approximately convex to begin with. Furthermore the correlations between **Area**, **Convex Area**, **Perimeter** and **Equiv. Diameter** are all higher than 0.95. The correlation of all of these with **Major Axis Length** and **Minor Axis Length** is higher than 0.9. We have a similar situation between a few other predictors. By analyzing the description of the features in [Murat Koklu, Ilker Ali Ozkan (2020)] we know that some features are transformations of the others. For example, all shapefactors are given by relations of **Area** and **MajorAxisLength** and **MinorAxisLength**, **EquivDiameter** is proportional to the square root of the **Area**, etc. From this we expect quite a lot of redundancy between the features.

In the appendix, table (4) we show a few box plots as a way to identify outliers in our data. It is clear from the plots that there is quite a large number of outliers at the higher extreme (on those predictors related to seed size). However, we argue that we should not get rid of outliers since, under the assumption that larger seeds produce better quality crops, we would like our models to account for these extremes.

Regarding transformation of variables, even though most of our data is in pixel count, in some instances it gives information regarding a 2-dimensional measure (e.g. **Area**, **Convex Area**), in other instances of a 1-dimensional measure (e.g. **Perimeter**, **Major/Minor Axis Length**) and yet in other instances of an adimensional parameter (e.g. **Aspect Ratio**) which have very different ranges and standard deviations (as can be seen in table 2). Due to this fact we will consider a standardized version of the data. In fact, we will see later that for feature selection standardizing the data is important to apply a (sensible) correcting factor in mutual information. For PCA it is naturally necessary due to the totally different scales of our predictors.

## Dimensionality Reduction

In this section we explain and apply to our data the following methods of dimensionality reduction: forward feature selection using the **CCMI** and **MICC** methods, **principal component analysis (PCA)** and feature importance based on **random forest**.

### Forward Feature Selection

In order to improve the performance of our models, interpretability and, in general, reduce the computational cost, we want to determine, out of the original features, which ones are optimal for the learning process. To do this we employ two filter feature selection methods, the first based on mutual information with correlation coefficient (CCMI), proposed by [Hongfang Zhou, Xiqian Wang and Rourou Zhu (2021)] and the second one, mutual information based constructive criterion (MICC). We begin by introducing a few required concepts.

The Shannon entropy of a random variable $X$ is a measure of the uncertainty within $X$. In mathematical terms the definition is as follows:

**Definition.** ***(Shannon) Entropy***
*Given a discrete random variable $X$, with probability mass function $p$ and support $\mathcal{X} = \{x_1, x_2, ..., x_n\}$, the entropy of $X$ is defined to be,*

$$\mathrm{H}(X) := -\sum_{i=1}^{n} \mathrm{p}(X = x_i) \log \mathrm{p}(X = x_i)$$

The concept of mutual information measures the association between the random variables $X$ and $Y$ capturing, unlike correlation, both linear and non-linear dependencies. The mathematical definition is the following:

**Definition.** ***Mutual Information***
*Given two discrete random variables $X$ and $Y$ with joint probability function $p_{X,Y}(X, Y)$ and marginal probabilities $p_X(X)$, $p_Y(Y)$ we define the mutual information $MI(X, Y)$ as,*

$$MI(X, Y) := \sum_{x \in X} \sum_{y \in Y} p_{X,Y}(x, y) \ln \frac{p_{X,Y}(x, y)}{p_X(x) p_Y(y)}$$

Another useful concept, which will be used as a measure of conditional relevance is that of conditional mutual information.

**Definition.** ***Conditional Mutual Information***
*Given two discrete random variables $X$, $Y$ and $Z$ we define the conditional mutual information $MI(X, Y|Z)$ as,*

$$MI(X, Y|Z) := \sum_{z \in Z} \sum_{y \in Y} \sum_{x \in X} p(x, y, z) \ln \frac{p(z)p(x, y, z)}{p(x, z)p(y, z)}$$

When selecting features the methods take into account both **relevance** and **redundancy**, i.e. we want to select features that are highly related to the class but which contain "different information" in regards to predicting the class. Mathematically, one measures the relevance of feature $X_i$ by considering $MI(X_i, C)$ and redundancy by considering $MI(X_i, X_j)$, however here we use $MI(X_i, C|X_j)$ which captures both relevance and redundancy and describes more accurately the relation between the candidate feature and the class [Hongfang Zhou, Xiqian Wang and Rourou Zhu (2021)].

In the case of (**CCMI**) the measure for feature selection is further weighted by the linear correlation and is given by:

**Definition.** ***Feature selection based on CCMI***
*Given a set of features $F = \{X_1, X_2, ..., X_n\}$ and a subset $S$ of the selected features, so far, the score of feature $X_m$ based on CCMI is given by,*

$$J_{CCMI}(X_m) := \min_{X_s \in S} MI(X_m, C|X_s) - \min_{X_s \in S} |\rho_{X_m, X_s}| MI(X_m, X_s)$$

*where,*

$$\rho_{X_m, X_i} = \frac{Cov(X_m, X_i)}{\sqrt{Var(X_m)}\sqrt{Var(X_i)}}$$

The objective function will then be arg max $J_{CCMI}$ and returns the next best predictor. Briefly, given a set of features $X_m$ we want to select from, and given a set $S$ of selected features, for each $X_m$ we obtain the minimum of the conditional mutual information over $X_s \in S$ (which we want to maximize for selection) and subtract to it a redundancy factor weighted by the correlation factor (which we want to minimize for selection). The algorithm can be seen in appendix, algorithm (1).

For the mutual information based constructive criterion (**MICC**) method, the objective function is given by,

$$\arg\max_{X_i \in F} \left\{ \frac{MI(C, X_i)}{\frac{1}{|S|}\sum_{X_s \in S} \frac{MI(X_i, X_s)}{\min\{H(X_i), H(X_s)\}}} - MI(C, X_i) \right\}$$

where again $S$ is the set of selected features and $F$ is the set of remaining features from where we which to select the next one.

Although our features are all continuously distributed random variables, in order to apply these results we use a discretized version of the data, which is why we only show the theoretical results for discretized random variables. The discretization is performed by partitioning each predictor $X_i$ in $k$ bins, with $k = \sqrt{N}$ where $N = 13611$ is the number of data points. We discretize using equal widths and this results in $\Delta X_i = \frac{\max X_i - \min X_i}{k}$. When calculating the mutual information we need to take into consideration the following correction:

$$\begin{aligned} &H(X_i^{\Delta_1}) + \ln \Delta_1 + H(X_j^{\Delta_2}) + \ln \Delta_2 \\ &-H(X_i^{\Delta_3}, X_j^{\Delta_4}) - \ln \Delta_3 - \ln \Delta_4 \\ \rightarrow \text{MI}(X_i, X_j) \ \text{ as } \ &\Delta_k \rightarrow 0 \ \ , k = 1, 2, 3, 4 \end{aligned}$$

Where for the multivariate case we consider $k = N^{1/4}$.

When calculating the mutual information using the original data we verified that the correction term was dominating the entropy values. This was mainly due to the fact that the bins width in certain cases were very large and certainly not going to zero as required. To overcome this problem, as mentioned in the beginning, we decided to standardize the data. This transformation resulted in mutual information values (computed using the *infotheo* package function *mutinformation()*) being a lot closer to our computed corrected values.

The feature ordering (from most to least informative) obtained using the (**CCMI**) algorithm was the following (the number to feature correspondence can be seen in table 2):

$$(2, 9, 14, 10, 13, 12, 4, 5, 8, 15, 1, 6, 7, 16, 3, 11)$$

In the case of (**MICC**) (again from most to least informative) we obtained:

$$(2, 6, 13, 11, 14, 16, 8, 3, 4, 9, 15, 10, 1, 5, 7, 12)$$

**Remark**: When calculating the entropy we used the Shrink method since it was the one showing best convergence properties in [M. Rosário Oliveira (2022)].

**Code**: In this part, besides the main (file "main.R"), we employed auxiliary functions which can be found in the files "CCMI.R","MICC.R" and "MI.R". The first and second one compute the $K$ best features for prediction using respectively the CCMI and MICC objective functions. The last one calculates the corrected mutual information of input variables $X_i$, $X_j$ by performing their discretization and using the entropy values from the *entropy()* function in the *infotheo* library (it is necessary for the first two).

**Principal Component Analysis (PCA)** In order to reduce dimensionality of the problem one can also use principal component analysis. Compared to feature selection, we now select new predictors $Z_i = \sum_{j=1}^{p} \alpha_{ij} X_j$, i.e. linear combinations of the original predictors $X_j$, instead of a subset of the original predictors. These new variables are the principal components. Considering our data is standardized, let $\Sigma$ be the variance-covariance matrix of the original predictors. The components $\alpha_{ij}$ for the vector $u_1$ which explains the highest portion of variability in the data is given by the following optimization problem:

$$\begin{aligned} &\arg\max_{u_1} \ u_1^T \Sigma u_1 \\ &s.t. \ \|u_1\| = 1 \end{aligned}$$

The solution satisfies $\Sigma u = \lambda_1 u$, i.e. $u$ is an eigenvector of $\Sigma$ with corresponding eigenvalue $\lambda_1$, where $\lambda_1$ is the portion of total variance explained by the first principal component. The components of $u$, i.e. the scores, represent the portion of each original predictor present in the principal component. The remaining principal components are obtained similarly with the extra condition that $Cov(Z_k, Z_i) = 0 \Leftrightarrow u_k^T u_i = 0$, for all previously obtained $u_i$.

Application wise, we simply use the *R* function *prcomp()* on the entire (standardized) dataset. The implementation

can be found in "main.R", the section with title "Principal Component Analysis".

The following figure (1), shows the obtained scree plot, i.e. the variance explained by each principal component. The eigenvalues obtained were the following,

$$\Lambda = \{8.875, 4.229, 1.281, 0.818, 0.438, 0.184, 0.112, 0.052, 0.008, 0.001, 0.001, 0.000, 0.000, 0.000, 0.000, 0.000\}$$

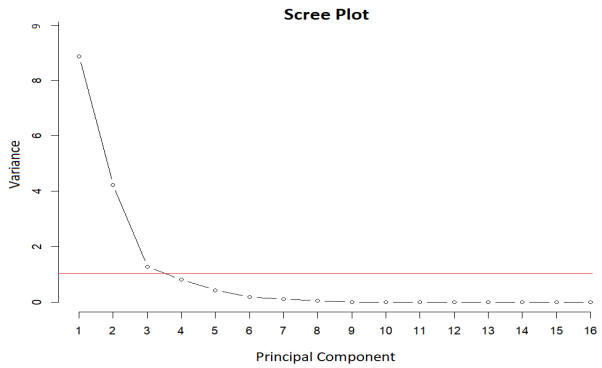


**Figure 1: Scree Plot**

We have $\sum_{\lambda_i \in \Lambda} \lambda_i = 15.999$, using the elbow criteria based on the scree plot, we use the first three principal components, the explained variance is given by $\frac{8.875+4.229+1.281}{15.999} = 89.91\%$. This value is higher than $80\%$, furthermore all the chosen eigenvalues are larger than 1 (we are considering standardized data) which would be two other different criteria for choosing the number of principal components. The proportion of variance explained for the first three principal components is given by $\frac{\lambda_1}{\sum_i \lambda_i} = 0.555$, $\frac{\lambda_2}{\sum_i \lambda_i} = 0.264$ and $\frac{\lambda_3}{\sum_i \lambda_i} = 0.080$. The loadings of the first three principal components can be seen in appendix, table (15).

The next plot shows a projection of the datapoints on the first two principal components with classes identified by colors, fig. (2).

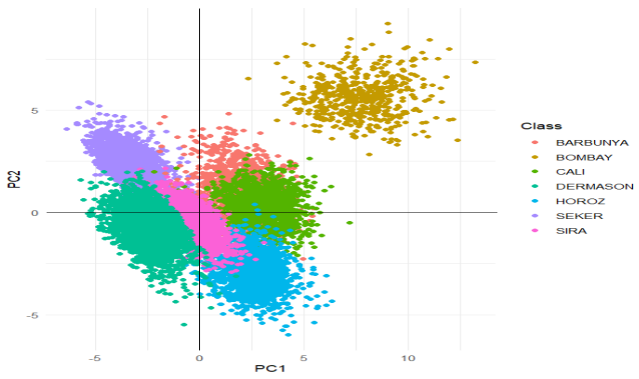


**Figure 2: PC1 vs PC2**

We can clearly see a cluster for each seed species and a clear separation of the "Bombay" class from the others. This reduces our concern over the lower presence of "Bombay" type seeds identified at the beginning since they appear to have very distinctive features when compared to the other six seed species. However, we can also observe quite an overlap over the remaining six species in the center of the graph. These might be more complicated to distinguish but still, this is dependent on the separation provided by the third principal component.

**Random Forest** As another alternative for dimensionality reduction, we briefly describe the random forest method of variable selection following [Gilles Louppe, Louis Wehenkel, Antonio Sutera and Pierre Geurts (2013)] and chapter 6 of [Gilles Louppe (2014)] (the last one given here mostly for future reference). Random forest classification relies on building an ensemble (forest) of decision trees, where each tree works as a classification model, and defining the resulting class by a majority vote. From this process of classification, it is possible to determine which predictors were most useful during the decision process.

Let $X = (X_1, ..., X_p)$ be our set of predictors with support $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_p$. Each tree of the random forest consists of a root node, $\mathcal{X}$, and, at each node $t$ in the tree we select a rule, based on a certain predictor $X_j$, which splits the previous set into two subsets, e.g. choosing $X_j < c$, $c \in \mathbb{R}$. The tree ends when we reach the terminal nodes, which are labeled as the best guess, by majority, from the subset at that node. The following figure, (3) attempts to somewhat illustrate the idea.

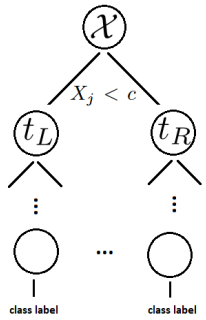


**Figure 3: Decision tree**

Starting from the sample size $N = 13611$, at each node $t$ the chosen split $s_t = s^*$, which partitions $N$ into $t_L$ and $t_R$, satisfies the following relation:

$$s^* = \arg\max_{s_t} \Delta i(s_t) = i(t) - p_L i(t) - p_R i(t)$$

where $p_L = N_{t_L}/N$, $p_R = N_{t_R}/N$ and where $i(t)$ denotes some impurity measure which can, in particular, be the Shannon entropy which was defined above. However, we shall consider the **Gini index**, since the measure provided

by the function *randomForest()*, part of the *randomForest* library, is the **Mean Decrease Gini**. The definition of importance is the following:

**Definition.** ***Feature importance***
*The importance of a predictor $X_j$ for predicting a class $Y$ over $N_T$ trees (with $T$ the number of nodes on each tree) is given by:*

$$Imp(X_j) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = X_j} p(t) \Delta i(s, t)$$

where $p(t) = N_t/N$, i.e. the proportion of samples that are left at node $t$ and $v(s_t)$ is the chosen variable for the split at node $t$. In other words, the importance of a predictor is greater if it maximizes the variation of impurity measure (which in this case is the Gini index, but could also be the entropy) over the largest number of samples and over all trees.

Using then the *randomForest* library, we apply random forest to our whole (standardized) dataset, since we are only interested in determining feature importance and not on classification over a training set. The results can be seen in appendix, table (16). Considering the mean decrease Gini score, we have the following ordering of our features (in decreasing importance and based on table 16):

$$(2, 15, 12, 13, 4, 3, 7, 6, 5, 8, 1, 11, 14, 16, 10, 9)$$

We should note there is quite some variance over the resulting mean decrease Gini score at each run of the algorithm, the results shown in table (16) are an average over five runs of the algorithm.

## Classification models

On the following classification models (kNN and MLR) we train the models on a training set comprising $70\%$ of the data and test them on a test set with the remaining $30\%$. To partition the data we constructed the function *train_test_split()* which can be found in *train_test_split.R*.

In order to evaluate the performance of the models we use following metrics: Accuracy, Macro Recall, Macro Precision and Macro $F_1$ which we define next.

**Definition.** ***Accuracy (Acc.)***

$$Acc. = \frac{\textit{Number of correct predictions}}{\textit{Total number of predictions}}$$

**Definition.** ***Macro Recall (MR)***
*The Macro Recall for a single class $C_i$ is given by:*

$$MR(C_i) = \frac{\textit{Correctly assigned observations of class i}}{\textit{Total observations of class i}}$$

*The total Macro Recall is the arithmetic mean of all classes Macro Recall, i.e. $MR = \frac{\sum_{i=1}^{k} MR(C_i)}{k}$, where $k$ is the number of classes.*

**Definition.** ***Macro Precision (MP)***
*The Macro Precision for a single class $C_i$ is given by:*

$$MP(C_i) = \frac{\textit{Correctly assigned observations of class i}}{\textit{Total assignments of class i}}$$

*The total Macro Precision is the arithmetic mean of all classes Macro Precision, i.e. $MP = \frac{\sum_{i=1}^{k} MP(C_i)}{k}$, where $k$ is the number of classes.*

**Definition.** ***Macro $F_1$ ($MF_1$)***
*The Macro $F_1$ for a single class $C_i$ is given by:*

$$MF_1(C_i) = 2\frac{MR(C_i) \cdot MP(C_i)}{MR(C_i) + MP(C_i)}$$

*The total Macro $F_1$ is the arithmetic mean of all classes Macro $F_1$, i.e. $MF_1 = \frac{\sum_{i=1}^{k} MF_1(C_i)}{k}$, where $k$ is the number of classes.*

**Code**: The code for this section uses the following auxiliary functions, "accuracy.R", "macro_recall.R", "macro_precision.R", "macro_f1.R" and "print_results.R".

**k-Nearest Neighbours (kNN)** The kNN method simply assigns to an unclassified data point $x^{(i)}$ the highest represented class of the $k$ nearest neighbours, based on Euclidean distance in this particular case. To train the kNN classifier we use the *knn()* function which is part of the *class* library in R. Since we are using an Euclidean distance, we compare the model's performance using the original data, normalized data using the function *normalize()* (*normalize.R*) and using standardized data. We expect normalization, and possibly standardization, to give better results since otherwise the distance is dominated by terms with much larger values such as **Area** and **Convex Area**.

The next table (3) shows the result of kNN with $k = 3$ considering the original (i.e. not standardized or normalized) data with all predictors. All values of accuracy, macro recall, macro precision and macro $F_1$, in all the following tables, are shown in percentage.

| k | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|
| 3 | 72.23 | 71.83 | 72.93 | 72.20 |

**Table 3: kNN results with original data (all predictors)**

The next table (4) shows the result of kNN using all predictors with normalized data for several values of $k$.

| k | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|
| 3 | 91.33 | 92.59 | 92.90 | 92.72 |
| 5 | 91.35 | 92.64 | 93.02 | 92.81 |
| 15 | 91.69 | 92.82 | 93.30 | 93.02 |
| 20 | 91.52 | 92.75 | 93.25 | 92.96 |

**Table 4: kNN results with normalized data (all predictors)**

As expected there is a clear improvement when compared to the original non-transformed data. Furthermore, changing the value of $k$ does not seem to impact significantly the model quality parameters. We also tried this for standardized data and we observed the same performance as normalized data (which we omit for brevity). We shall always consider standardized data for consistency with our choices for feature selection.

**kNN with feature selection**

We now perform the same evaluation but now using only a few selected predictors obtained using the feature selection methods. We use one eighth (2 predictors), one third (5 predictors) and one half (8 predictors) of the selected features. The following table (5), shows the results for the selection using the (**CCMI**) method, where $p$ represents the number of used predictors.

| k,p | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|
| 3,2 | 64.01 | 64.47 | 64.46 | 64.42 |
| 3,5 | 88.59 | 89.30 | 89.45 | 89.36 |
| 3,8 | 88.27 | 89.07 | 89.21 | 89.12 |

**Table 5: kNN results using CCMI**

The results using the predictors from (**MICC**) are given in the following table (6).

| k | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|
| 3,2 | 86.06 | 86.43 | 86.65 | 86.52 |
| 3,5 | 90.00 | 91.28 | 91.26 | 91.27 |
| 3,8 | 91.33 | 92.61 | 92.58 | 92.59 |

**Table 6: kNN results using MICC**

Clearly, in the case of k-Nearest Neighbours, (**MICC**) outperforms (**CCMI**). With 8 predictors the performance is essentially the same, for (**MICC**), as when considering all 16 predictors. This shows there is a lot of redundancy between the predictors which is to be expected since, as noted during the preliminary data analysis, a lot of the features are just functions of other features.

**kNN with PCA**

We now show the results obtained when performing kNN using the features resulting from PCA. To do this we first project each (standardized) data point $\hat{x}^{(i)} \in \mathbb{R}^{16}$ on the first three principal components and build a new *dataframe* with these three columns plus the class labels. Then, we extract a training and test set and we just employ kNN as before. The results are given in the following tables (7) for $k = 3$ and $k = 15$, where the last one, as seen before, shows slightly better results.

| k | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|
| 3 | 85.97 | 86.16 | 86.29 | 86.18 |
| 15 | 87.68 | 87.53 | 88.29 | 87.72 |

**Table 7: kNN results with PCA**

**kNN with random forest importance selection**

When using kNN with the features selected using the random forest importance selection method we obtained the following results, table (8).

| p | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|
| 2 | 86.70 | 87.16 | 87.34 | 87.22 |
| 5 | 89.99 | 91.36 | 91.53 | 91.43 |
| 8 | 90.00 | 91.23 | 91.38 | 91.30 |

**Table 8: kNN results using random forest**

**Multinomial Logistic Regression (MLR)** A generalization of logistic regression to handle multiple classes, MLR models the probability of a class given the predictors. Considering the softmax formulation following [Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani (2013)] and given $k = 1, ..., K$ classes and $p$ predictors we have:

$$Pr(Y = k|X = x) = \frac{e^{\theta_{k0}+\theta_{k1}x_1+...+\theta_{kp}x_p}}{\sum_{l=1}^{K} e^{\theta_{l0}+\theta_{l1}x_1+...+\theta_{lp}x_p}}$$

In order to obtain the parameters $\theta_{ij}$ in the expression above one usually minimizes a cost function $J$ relative to the training data $(x^{(i)}, y^{(i)})$, where $x^{(i)} \in \mathbb{R}^p$ and $y^{(i)}$ is the corresponding class. The cost function is given by:

$$J(\theta) = -\left[\sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}(y^{(i)} = k) \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})}\right]$$

where each $\theta^{(i)} = (\theta_{i1}, \theta_{i2}, ..., \theta_{ip})$ for $i = 1, ..., K$. The minimization of $J$ is usually carried out by an iterative optimization algorithm, typically gradient descent.

To implement this we use the *nnet* package in *R* and call the function *multinom()*. The results obtained by using all features are given in the following table (9).

| Set | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|
| Train | 93.01 | 94.05 | 94.25 | 94.15 |
| Test | 91.65 | 92.97 | 93.14 | 93.04 |

**Table 9: MLR results using all features**

The results obtained using the predictors from (**CCMI**) are given in the following table (10).

| Set | p | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|---|
| Train | 2 | 65.19 | 63.98 | 62.68 | 61.53 |
| Train | 5 | 91.58 | 92.52 | 92.83 | 92.65 |
| Train | 8 | 92.24 | 93.28 | 93.53 | 93.40 |
| Test | 2 | 65.52 | 64.48 | 62.88 | 61.85 |
| Test | 5 | 91.04 | 92.02 | 92.37 | 92.16 |
| Test | 8 | 91.60 | 92.78 | 92.95 | 92.86 |

**Table 10: MLR results using (CCMI)**

The results using the predictors from (**MICC**) are given in the following table (11).

| Set | p | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|---|
| Train | 2 | 88.52 | 88.68 | 89.13 | 88.81 |
| Train | 5 | 91.74 | 92.83 | 93.04 | 92.91 |
| Train | 8 | 92.70 | 93.76 | 93.96 | 93.86 |
| Test | 2 | 88.42 | 88.65 | 89.11 | 88.80 |
| Test | 5 | 91.21 | 92.38 | 92.52 | 92.43 |
| Test | 8 | 91.74 | 93.04 | 93.14 | 93.08 |

**Table 11: MLR results using (MICC)**

The results obtained using the three principal components resulting from **PCA** are given in the following table (12).

| Set | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|
| Train | 88.00 | 87.67 | 88.10 | 87.81 |
| Test | 87.37 | 87.14 | 87.70 | 87.34 |

**Table 12: MLR results from PCA**

The results obtained using the predictors selected by random forest are given in the following table (13).

| Set | p | Accuracy | Macro Recall | Macro Precision | Macro F1 |
|---|---|---|---|---|---|
| Train | 2 | 89.07 | 89.28 | 89.65 | 89.40 |
| Train | 5 | 91.89 | 92.96 | 93.22 | 93.08 |
| Train | 8 | 92.24 | 93.28 | 93.53 | 93.40 |
| Test | 2 | 88.81 | 89.14 | 89.51 | 89.27 |
| Test | 5 | 91.31 | 92.51 | 92.72 | 92.60 |
| Test | 8 | 91.60 | 92.78 | 92.95 | 92.86 |

**Table 13: MLR results using random forest feature selection**

**Discussion** We begin by discussing the obtained model metrics comparing the two methods of forward feature selection, CCMI and MICC. In this particular case we observe that MICC clearly outperforms CCMI. This can be seen particularly in the context of kNN (tables 5 and 6) where the first two predictors using MICC have comparable performance to the best 5 or 8 predictors selected using CCMI and much better performance than the best two selected by CCMI. We note also that, in the case of CCMI, we actually see a slight decrease in performance when using kNN with 8 features instead of 5, i.e. the model is selecting features which are actually impairing the classification after some point. It is not easy to justify this difference in performance but we suspect it might have to do with the correlation coefficient used in the objective function of the CCMI model. Since it is factoring the redundancy ($MI(X_m, X_s)$) by a correlation factor it is essentially neglecting somewhat the possible non-linear redundancy between the variables giving higher weight to linear redundancy. Furthermore, having noted at the beginning that several features are non-linear transformations of the others this might be an issue for this particular method.

In the case of MICC vs PCA we feel the comparison is a bit unjustified. While MICC is selecting features based on their relevance to predict the class, PCA just looks at the highest variance among the predictors which might not correlate well with explainability in regards to classification. Nonetheless, the performance obtained using three principal components with kNN and MLR (tables 7 and 12) is actually decent (performance measures around 86 88%). In regards to the selected features, the first principal component relates mainly with **Perimeter**, **MajorAxisLength** and **ShapeFactor2** but also quite strongly with quite a few other predictors such as **Area**, **ConvexArea** and **EquivDiameter**. The second principal component picks **MinorAxisLength** most strongly and quite a mix of the other features while the third one clearly relates mainly with **Solidity**. The first 5 features selected by MICC were **Perimeter**, **Eccentricity**, **ShapeFactor1**, **Roundness** and **ShapeFactor2**. Besides **Perimeter** and possibly **ShapeFactor2** we do not see a lot of features in common with the three main principal components and selected features of MICC. However this might very well

have to do with the fact that MICC avoids non-linear redundancy between features which is not captured by PCA which, nonetheless, identified quite a large number of predictors which are approximately linear combinations of other features. These can be seen by the eight last eigenvalues which are very close to 0.

The five most important features identified using the random forest method were (in decreasing order of importance): **Perimeter**, **ShapeFactor3**, **Compactness**, **ShapeFactor1** and **MinorAxisLength**. Similarly to PCA and MICC, **Perimeter** is identified as the best predictor. The remaining ones are difficult to compare to MICC since the importance method does not account for redundancy between features and simply in their ability to better separate classes. The performance of random forest selection was inferior to MICC but better than PCA, particularly with kNN. With MLR the results were pretty similar to MICC and better than PCA.

Comparing the two classifiers, kNN and MLR, when using all features the performance is pretty much the same. When using dimensionality reduction we see MLR come out on top but typically by a small margin of around 1$\sim$2%. We further conclude that the performance is high over all classes since both macro recall and macro precision values are around 93%.

To finalize, we further tried to apply PCA to the best 5 predictors selected by (**MICC**) in hopes of seeing a clearer alignment of principal components with original predictors for better interpretability. The results were however much like the ones we had before and we saw no benefit in providing them here.

## References


C. Chatfield and A. J. Collins. (1980). *Introduction to multivariate analysis*. Chapman & Hall.

Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. (2013). *An introduction to statistical learning*. Springer.

Gilles Louppe. (2014). Understanding random forests: From theory to practice. *PhD thesis*.

Gilles Louppe, Louis Wehenkel, Antonio Sutera and Pierre Geurts. (2013). Understanding variable importances in forests of randomized trees. *Advances in Neural Information Processing Systems 26 (NIPS 2013)*.

Hongfang Zhou, Xiqian Wang and Rourou Zhu. (2021). Feature selection based on mutual information with correlation coefficient. *Applied Intelligence*.

M. Rosário Oliveira. (2022). *Information theory and filter feature selection*. Class notes: Mathematics for Machine Learning (IST).

Murat Koklu, Ilker Ali Ozkan. (2020). Multiclass classification of dry beans using computer vision and machine learning techniques. *Computers and Electronics in Agriculture - ELSEVIER*.

# Appendix

## Correlation Matrix

| | Area | Perimeter | MajorAxisLength | MinorAxisLength | AspectRation | Eccentricity | ConvexArea | EquivDiameter | Extent | Solidity | roundness | Compactness | ShapeFactor1 | ShapeFactor2 | ShapeFactor3 | ShapeFactor4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Area | | 0.967 | 0.932 | 0.952 | 0.242 | 0.267 | 1.000 | 0.985 | 0.054 | -0.197 | -0.358 | -0.268 | -0.848 | -0.639 | -0.272 | -0.356 |
| Perimeter | 0.967 | | 0.977 | 0.913 | 0.385 | 0.391 | 0.968 | 0.991 | -0.021 | -0.304 | -0.548 | -0.407 | -0.865 | -0.768 | -0.408 | -0.429 |
| MajorAxisLength | 0.932 | 0.977 | | 0.826 | 0.550 | 0.542 | 0.933 | 0.962 | -0.078 | -0.284 | -0.596 | -0.568 | -0.774 | -0.859 | -0.568 | -0.483 |
| MinorAxisLength | 0.952 | 0.913 | 0.826 | | -0.009 | 0.020 | 0.951 | 0.949 | 0.146 | -0.156 | -0.210 | -0.015 | -0.947 | -0.471 | -0.019 | -0.264 |
| AspectRation | 0.242 | 0.385 | 0.550 | -0.009 | | 0.924 | 0.243 | 0.304 | -0.370 | -0.268 | -0.767 | -0.988 | 0.025 | -0.838 | -0.979 | -0.449 |
| Eccentricity | 0.267 | 0.391 | 0.542 | 0.020 | 0.924 | | 0.269 | 0.319 | -0.319 | -0.298 | -0.722 | -0.970 | 0.020 | -0.860 | -0.981 | -0.449 |
| ConvexArea | 1.000 | 0.968 | 0.933 | 0.951 | 0.243 | 0.269 | | 0.985 | 0.053 | -0.206 | -0.362 | -0.270 | -0.848 | -0.641 | -0.274 | -0.362 |
| EquivDiameter | 0.985 | 0.991 | 0.962 | 0.949 | 0.304 | 0.319 | 0.985 | | 0.028 | -0.232 | -0.436 | -0.328 | -0.893 | -0.713 | -0.330 | -0.393 |
| Extent | 0.054 | -0.021 | -0.078 | 0.146 | -0.370 | -0.319 | 0.053 | 0.028 | | 0.191 | 0.344 | 0.354 | -0.142 | 0.238 | 0.348 | 0.149 |
| Solidity | -0.197 | -0.304 | -0.284 | -0.156 | -0.268 | -0.298 | -0.206 | -0.232 | 0.191 | | 0.607 | 0.304 | 0.153 | 0.344 | 0.308 | 0.702 |
| roundness | -0.358 | -0.548 | -0.596 | -0.210 | -0.767 | -0.722 | -0.362 | -0.436 | 0.344 | 0.607 | | 0.768 | 0.230 | 0.783 | 0.763 | 0.472 |
| Compactness | -0.268 | -0.407 | -0.568 | -0.015 | -0.988 | -0.970 | -0.270 | -0.328 | 0.354 | 0.304 | 0.768 | | -0.009 | 0.869 | 0.999 | 0.484 |
| ShapeFactor1 | -0.848 | -0.865 | -0.774 | -0.947 | 0.025 | 0.020 | -0.848 | -0.893 | -0.142 | 0.153 | 0.230 | -0.009 | | 0.469 | -0.008 | 0.249 |
| ShapeFactor2 | -0.639 | -0.768 | -0.859 | -0.471 | -0.838 | -0.860 | -0.641 | -0.713 | 0.238 | 0.344 | 0.783 | 0.869 | 0.469 | | 0.873 | 0.530 |
| ShapeFactor3 | -0.272 | -0.408 | -0.568 | -0.019 | -0.979 | -0.981 | -0.274 | -0.330 | 0.348 | 0.308 | 0.763 | 0.999 | -0.008 | 0.873 | | 0.484 |
| ShapeFactor4 | -0.356 | -0.429 | -0.483 | -0.264 | -0.449 | -0.449 | -0.362 | -0.393 | 0.149 | 0.702 | 0.472 | 0.484 | 0.249 | 0.530 | 0.484 | |

**Table 14: Correlation matrix**

## Outliers (boxplots)

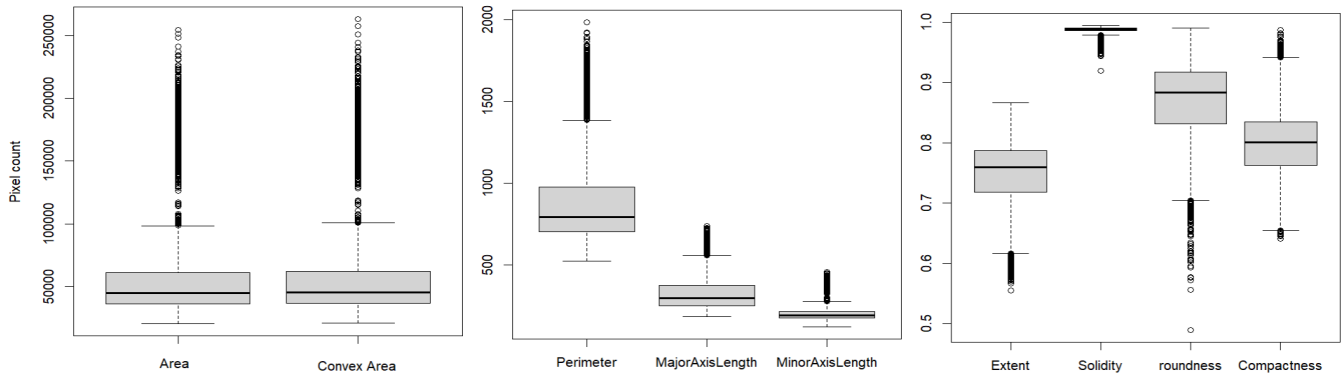


**Figure 4: Outliers for a few predictors**

## CCMI Algorithm

**Algorithm 1** CCMI

**Input:** Training data with features $F = \{X_1, X_2, ..., X_n\}$; The class labels of our training data $C$; Number of features to extract $K$;
**Output:** A set $S$ with $K$ ordered selected features;
**Initialization:** $S = \emptyset$; $A, B, C = NA$ (as $n \times n$ matrices) ; $J$ (n-dimensional column vector); count=1;

$f_{new} \leftarrow \underset{X_i}{\text{argmax}}\, MI(X_i, C)$
$S \leftarrow S \cup f_{new}$
$F \leftarrow F - f_{new}$
**while** *count*<$K$ **do**
  **for** $m \in \{$*indices of* $F\}$ **do**
    **for** $i \in \{$*indices of* $S\}$ **do**
      $A(m,i) \leftarrow MI(X_m, C|X_i)$
      $B(m,i) \leftarrow \rho_{X_m, X_i}$
      $C(m,i) \leftarrow MI(X_m, X_i)$
    $J(m) \leftarrow J_{CCMI}(X_m)$
  $f_{new} \leftarrow \underset{X_m \in F}{\text{argmax}}\, J_{CCMI}(X_m)$
  $count = count + 1$
  $S \leftarrow S \cup f_{new}$
  $F \leftarrow F - f_{new}$

The implementation in R can be found in the attached file "CCMI.R".

### Loadings of PC1, PC2 and PC3

| | PC1 | PC2 | PC3 |
|---|---|---|---|
| Area | 0.841 | 0.506 | -0.070 |
| Perimeter | 0.926 | 0.369 | -0.021 |
| MajorAxisLength | 0.971 | 0.207 | -0.096 |
| MinorAxisLength | 0.704 | 0.706 | 0.008 |
| AspectRation | 0.683 | -0.680 | -0.191 |
| Eccentricity | 0.690 | -0.657 | -0.185 |
| ConvexArea | 0.844 | 0.503 | -0.061 |
| EquivDiameter | 0.886 | 0.458 | -0.056 |
| Extent | -0.178 | 0.454 | -0.096 |
| Solidity | -0.426 | 0.212 | -0.836 |
| roundness | -0.739 | 0.442 | -0.185 |
| Compactness | -0.710 | 0.676 | 0.169 |
| ShapeFactor1 | -0.659 | -0.684 | -0.037 |
| ShapeFactor2 | -0.937 | 0.266 | 0.136 |
| ShapeFactor3 | -0.712 | 0.674 | 0.169 |
| ShapeFactor4 | -0.590 | 0.206 | -0.608 |

**Table 15: Loadings of the first three principal components**

### Importance results from Random Forest

| | BARBUNYA | BOMBAY | CALI | DERMASON | HOROZ | SEKER | SIRA | MeanDecAcc. | MeanDecGini |
|---|---|---|---|---|---|---|---|---|---|
| Area | 0.102 | 0.285 | 0.192 | 0.159 | 0.044 | 0.033 | 0.119 | 0.120 | 644.321 |
| Perimeter | 0.247 | 0.286 | 0.240 | 0.178 | 0.079 | 0.065 | 0.256 | 0.180 | 1084.329 |
| MajorAxisLength | 0.147 | 0.248 | 0.255 | 0.158 | 0.132 | 0.123 | 0.114 | 0.155 | 834.168 |
| MinorAxisLength | 0.090 | 0.291 | 0.190 | 0.187 | 0.076 | 0.091 | 0.088 | 0.133 | 916.116 |
| AspectRation | 0.043 | 0.008 | 0.070 | 0.028 | 0.168 | 0.153 | 0.076 | 0.081 | 658.223 |
| Eccentricity | 0.036 | 0.005 | 0.076 | 0.026 | 0.139 | 0.149 | 0.076 | 0.076 | 676.755 |
| ConvexArea | 0.125 | 0.292 | 0.225 | 0.194 | 0.055 | 0.043 | 0.142 | 0.143 | 824.046 |
| EquivDiameter | 0.084 | 0.271 | 0.186 | 0.130 | 0.038 | 0.029 | 0.118 | 0.107 | 652.630 |
| Extent | 0.002 | 0.000 | 0.006 | 0.004 | 0.002 | 0.012 | 0.006 | 0.005 | 129.717 |
| Solidity | 0.092 | 0.000 | 0.015 | 0.014 | 0.005 | 0.016 | 0.016 | 0.021 | 208.780 |
| roundness | 0.248 | 0.005 | 0.076 | 0.034 | 0.049 | 0.020 | 0.172 | 0.085 | 624.050 |
| Compactness | 0.078 | 0.010 | 0.150 | 0.032 | 0.267 | 0.245 | 0.116 | 0.131 | 1062.890 |
| ShapeFactor1 | 0.120 | 0.302 | 0.213 | 0.185 | 0.106 | 0.136 | 0.119 | 0.156 | 1014.192 |
| ShapeFactor2 | 0.053 | 0.067 | 0.167 | 0.068 | 0.092 | 0.119 | 0.078 | 0.091 | 492.799 |
| ShapeFactor3 | 0.074 | 0.010 | 0.155 | 0.035 | 0.238 | 0.260 | 0.117 | 0.130 | 1082.941 |
| ShapeFactor4 | 0.014 | 0.000 | 0.213 | 0.014 | 0.025 | 0.093 | 0.028 | 0.053 | 350.729 |

**Table 16: Importance results obtained from the Random Forest algorithm (averaged over 5 runs)**