Yeast classification: Localization site of protein

Manel Alba and Alejandro Olvera

June 21, 2016

Abstract

The purpose of the project is apply all the knowledge gained in the **Machine Learning course** so as to face a chosen problem from the UCI machine learning repository. We chose a dataset about **Yeast proteins**, with the goal of **classifying** the different proteins given in the dataset in **10 different classes**. To perform the task, we have tried **3 different pre-processing techniques** and **6 algorithms**; K Nearest Neighbours, Naive Bayes, Linear Discriminant Analysis, Quadratic discriminant Analysis, Random Forest and Neural Networks.

1 Introduction

This is the paper of the final project of Machine learning (**ML**) course, a subject taught in the Master of Innovation and Research in Informatics (**MIRI**) at the Universitat Politcnica de Catalunya (**UPC**). The objective is to apply all the knowledge gained during the course in order to solve a chosen problem from the UCI Machine Learning Repository. In our case, we decided to choose the **Yeast dataset**.
Working with this dataset, our mission was to identify which class of Yeast protein is each individual. The localization site of a protein is primarily determined by its amino-acid sequence, so this is the information that we will use so as to build a **classification model** (supervised learning).The complete information of the dataset is presented below.
Basic description:

- Number of **individuals**: 1484.
- Number of **variables**: 9 + 1 (response variable).
- Number of **modalities** of the response variable: 10.

Description of the variables:

1. **Sequence Name**: Accession number for the SWISS-PROT database.
2. **mcg**: McGeoch's method for signal sequence recognition.

3. **gvh**: von Heijne's method for signal sequence recognition.
4. **alm**: Score of the ALOM membrane spanning region prediction program.
5. **mit**: Score of discriminant analysis of the amino acid content of the N-terminal region (20 residues long) of mitochondrial and non-mitochondrial proteins.
6. **erl**: Presence of "HDEL" substring (thought to act as a signal for retention in the endoplasmic reticulum lumen). Binary attribute.
7. **pox**: Peroxisomal targeting signal in the C-terminus.
8. **vac**: Score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins.
9. **nuc**: Score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins.
10. **class**: the response variable. It describes to which class belongs the individual.

2 Pre-processing

- **Standardize** the data: standardization of datasets is a common approach. Variables that are measured in different scales do not contribute equally to the examples and standarization solves that, obtaining all the variables with a **0 value in mean and 1 in variance**. This is done by centring the data (subtracting the mean) and dividing it by its standard deviation for all the features (columns) of the dataset.
- Standardize the data plus **Principal Component Analysis** (PCA): it is used to decompose a multivariate dataset in a set of successive orthogonal components that explain a **maximum amount of the variance**. This is an unsupervised method. Applied to the yeast dataset, we can see these results:

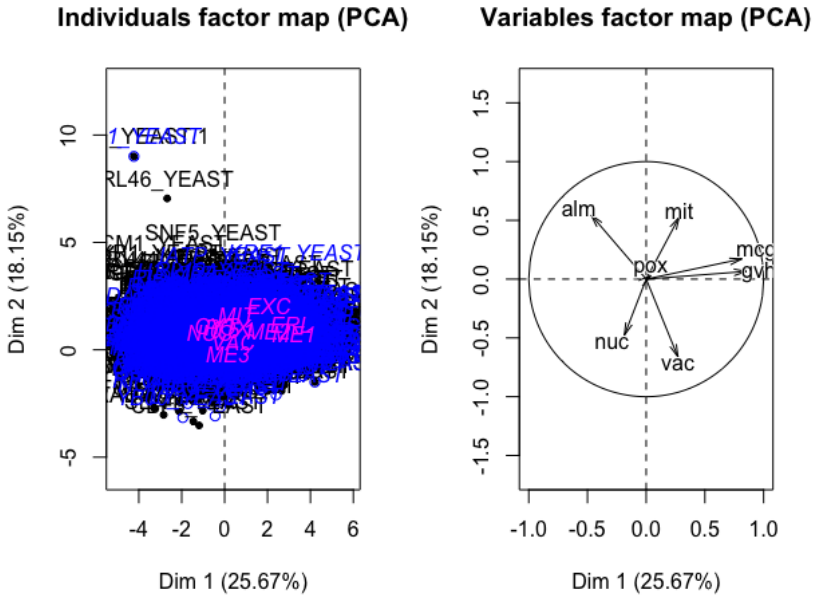


Figure 1: Result of executing PCA on yeast data

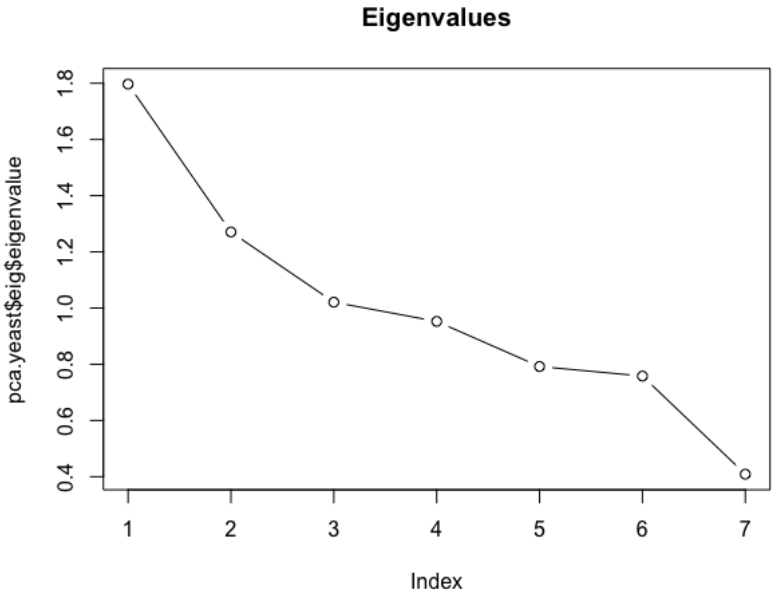


Figure 2: Eigenvalues of applying PCA on yeast data

	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	1.80	25.67	25.67
comp 2	1.27	18.15	43.82
comp 3	1.02	14.58	58.41
comp 4	0.95	13.61	72.02
comp 5	0.79	11.31	83.33
comp 6	0.76	10.83	94.16
comp 7	0.41	5.84	100.00

Table 1: Eigenvalues of applying PCA on yeast data

Applying the **Kaiser rule**, which said that at we should keep up to 80% of the total variance explained, we decide to work with **5 principal components**.

- Standardize the data plus **Feature and Individual Selection**: also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. This is commonly used for three reasons:
  - Simplification of models

– Shorter training times.

– Reduction of variance.

In our case, we decide to apply this pre-processing method after seeing how are the classes distributed among the individuals and the features associated to this classes (notice that for each class, we have an equivalent feature).

	V1
CYT	463
ERL	5
EXC	35
ME1	44
ME2	51
ME3	163
MIT	244
NUC	429
POX	20
VAC	30

Table 2: Distribution of the yeast classes

As we can see, the distribution is quite unbalanced. For example, only 5 individuals are class ERL out of 1484. After doing several analysis with the pre-processing techniques described above, we realize that classes like **ERL or VAC** were always predicted with a 100% error, so we decide to try to analyze the dataset without both all the **individuals of class VAC and ERL** and without the **feature erl** (which the 99% of the cases has the same value).

3 Methods used

To evaluate the performance of the classification methods used, we have three steps:

1. Divide the data randomly in **training data** (1/3 of the total data) and **test data** (2/3 of the total data).

2. Perform a **10x10 Cross-Validation** of the train data, so as to check the correct **behavior** of the models and identify possible cases of **over-fitting/under-fitting**.

3. Optimize a model as much as possible **tuning its parameters**. This was used in Neural Networks, K Nearest Neighbours and Linear Discriminant Analysis. A more detailed explanation will be shown below.

4. Check again the behaviour of the models once optimized using the Cross-Validation approach.

5. Train the best algorithms with all the data so as to check their performance in the problem when more data is provided.

Once defined how an algorithm will be evaluated, let's see the selected methods that we are going to use in the analysis:

1. **K Nearest Neighbours** (KNN): is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple **majority vote** of the nearest neighbours of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.
   In order to optimize this method, we searched for the **optimal number of neighbours**.

2. **Naive Bayes** (NB): based on applying Bayes theorem with the naive assumption of independence between every pair of features. Both Linear Discriminant Analysis and Quadratic Discriminant Analysis are also of the Naive Bayes classifier family but differ mainly by the assumptions they make regarding the distribution of $P(x_i| y)$.

3. **Linear discriminant Analysis** (LDA): NB family classifier with the difference that LDA assumes that all classes share the same covariance matrix. In order to optimize this method, we tried to provide it the prior probabilities beforehand (estimating them from the training data).

4. **Quadratic Discriminant Analysis** (QDA): NB family classifier with the difference that QDA estimates each class covariance matrix instead of making the assumptions of NB or LDA.

5. **Random Forest** (RF): meta estimator that fits a number of **decision tree** classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. This is called an **ensemble method**.

6. **Neural Networks** (NN): Non-linear method, based on finding **adaptive regressors** to a problem. The implementation used by us is the **Multi-layer Perceptron**.
   In order to optimize this method, we searched for the **optimal decay for the weights**, so as to regularize the network.

If we had to classify our algorithms from most complex to less complex, the ranking will be: NB, KNN, LDA, QDA, RF and NN. As more complex is a method, more caution one must be in order to not over-fit the data, using a proper regularization approach.

4 Results

In this section we are going to show all the results obtained during the project (notice that due to strong correlations, QDA can only be executed with the PCA pre-processing.)
For **NN and KNN** we have **tuned their parameters beforehand**, so as to optimize them as much as possible. Here we can see the results found:

1. With data standardized

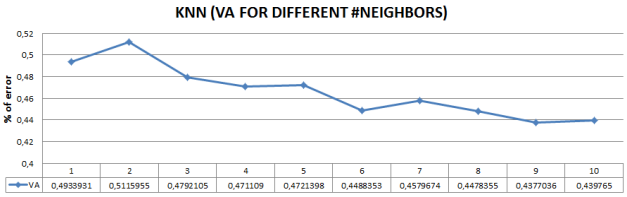


Figure 3: Calculating the ideal number of K neighbours for KNN

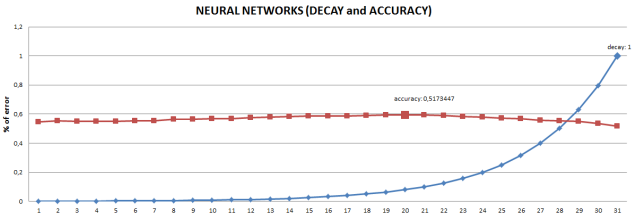


Figure 4: Calculating the optimal decay for the NN

The best value corresponds to a decay of 0.079.

2. With PCA

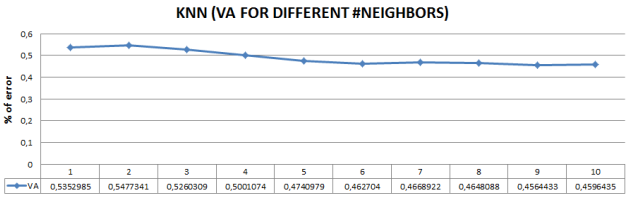


Figure 5: Calculating the ideal number of K neighbours for KNN

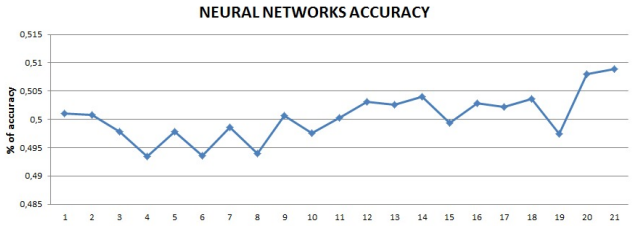


Figure 6: Calculating the optimal decay for the NN

The best value corresponds to a decay of 0.001.

3. With Feature selection

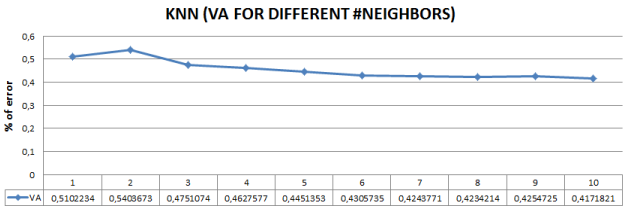


Figure 7: Calculating the ideal number of K neighbours for KNN

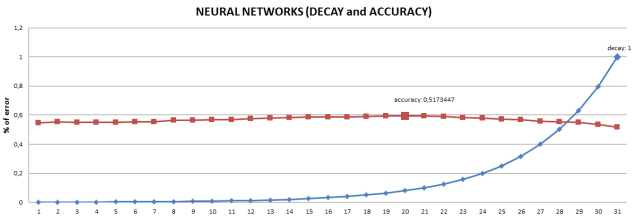
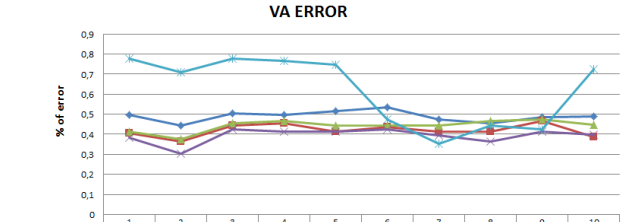


Figure 8: Calculating the optimal decay for the NN

The best value corresponds to a decay of 0.079.

To inspect the results, we have made a comparison of the methods using the same **10x10 Cross-Validation** (same folds are used in each algorithm), where both the training error and the validation error have been collected of the execution of each fold.Here we can see the results found:

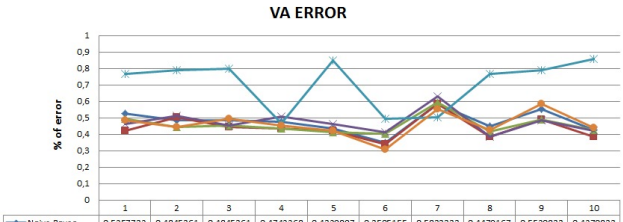
1. With data standardized



	1	2	3	4	5	6	7	8	9	10
Naive Bayes	0,4949495	0,4444444	0,5050505	0,4949495	0,5151515	0,5353535	0,4747475	0,4545455	0,4848485	0,4897959
LDA	0,4040404	0,3636364	0,4444444	0,4545455	0,4141414	0,4343434	0,4141414	0,4141414	0,4646465	0,3877551
KNN	0,4141414	0,3737374	0,4545455	0,4646465	0,4444444	0,4444444	0,4444444	0,4646465	0,4747475	0,4489796
Random Forest	0,3838384	0,3030303	0,4242424	0,4141414	0,4141414	0,4242424	0,3939394	0,3636364	0,4141414	0,3979592
Neutral Networks	0,7777778	0,7070707	0,7777778	0,7676768	0,7474747	0,4747475	0,3535354	0,4444444	0,4242424	0,7244898

Figure 9: Comparing the behavior of the different algorithms in the 10x10 CV

2. With PCA

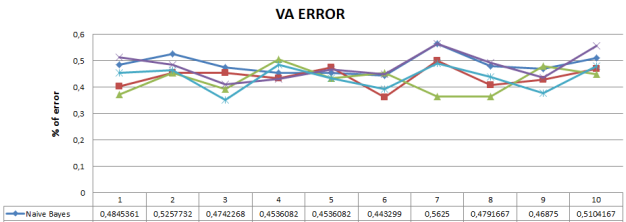


	1	2	3	4	5	6	7	8	9	10
Naive Bayes	0,5257732	0,4845361	0,4845361	0,4742268	0,4329897	0,3505155	0,5833333	0,4479167	0,5520833	0,4270833
LDA	0,4226804	0,5051546	0,443299	0,4329897	0,4226804	0,3402062	0,5833333	0,3854167	0,4895833	0,3854167
KNN	0,4948454	0,443299	0,4536082	0,4329897	0,4123711	0,4020619	0,59375	0,4166667	0,4895833	0,4270833
Random Forest	0,4598605	0,511427	0,453643	0,5091405	0,4632471	0,4142277	0,6301871	0,3836401	0,4842747	0,4207808
Neural Networks	0,7676768	0,7878788	0,7979798	0,4646465	0,8484848	0,4949495	0,5050505	0,7676768	0,7878788	0,8571429
QDA	0,4845361	0,443299	0,4948454	0,4536082	0,4226804	0,3092784	0,5520833	0,4270833	0,5833333	0,4375

Figure 10: Comparing the behavior of the different algorithms in the 10x10 CV

We can clearly appreciate from this plot that, in this case, the NN **overfits** the data.

3. With Feature selection



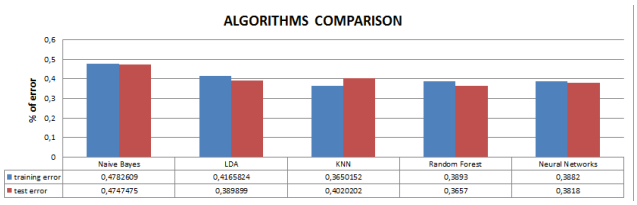
	1	2	3	4	5	6	7	8	9	10
Naive Bayes	0,4845361	0,5257732	0,4742268	0,4536082	0,4536082	0,443299	0,5625	0,4791667	0,46875	0,5104167
LDA	0,4020619	0,4536082	0,4536082	0,4329897	0,4742268	0,3608247	0,5	0,40625	0,4270833	0,46875
KNN	0,371134	0,4536082	0,3917526	0,5051546	0,4329897	0,4536082	0,3645833	0,3645833	0,4791667	0,4479167
Random Forest	0,5132153	0,4839989	0,4099927	0,4294552	0,4654223	0,446869	0,5620033	0,491017	0,4349132	0,5542641
Neutral Networks	0,4536082	0,4639175	0,3505155	0,4845361	0,4329897	0,3917526	0,4895833	0,4375	0,375	0,4791667

Figure 11: Comparing the behavior of the different algorithms in the 10x10 CV

We can clearly appreciate from this plot that, in this case, the NN **overfits** the data.

Finally, due to the fact that the errors are very similar, we will compute the final training and test error of each method **training it with all the training** data available (not using Cross-validation), so as to get a glimpse of how much affect the increasing of training data in the behaviour of our models. Here we can see the results found:

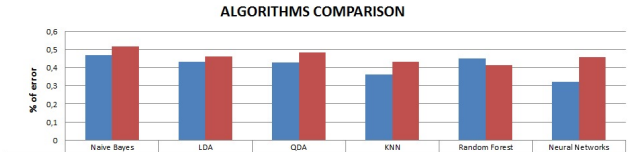
1. With data standarized



	Naive Bayes	LDA	KNN	Random Forest	Neural Networks
training error	0,4782609	0,4165824	0,3650152	0,3893	0,3882
test error	0,4747475	0,389899	0,4020202	0,3657	0,3818

Figure 12: Training and test error of all the algorithms

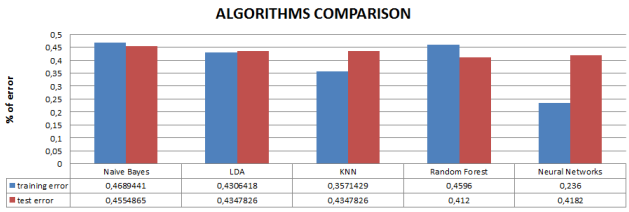
2. With PCA



	Naive Bayes	LDA	QDA	KNN	Random Forest	Neural Networks
training error	0,4668737	0,4327122	0,4275362	0,3623188	0,4482	0,3215
test error	0,515528	0,4616977	0,4824017	0,4306418	0,412	0,4565

Figure 13: Training and test error of all the algorithms

3. With Feature selection



	Naive Bayes	LDA	KNN	Random Forest	Neural Networks
training error	0,4689441	0,4306418	0,3571429	0,4596	0,236
test error	0,4554865	0,4347826	0,4347826	0,412	0,4182

Figure 14: Training and test error of all the algorithms

5 Conclusions and further development

Looking at the different results, we can conclude that are the best methods that we have found are **RF**(36% error), **KNN**(43% error) and **KNN**(around 43%

error, very complicated to decide which method is better), in their corresponding situations (after standardize, PCA and Feature Selection).
As we have seen during our paper, the final test error obtained is about 36% with RR, improving the result of the paper followed [2] by near a 10%. However, this is not an accurate result in order to face the problem of yeast classification, not only due to the fact that the average error is high but also, during the calculation the test error for each class, we found that it is very high for some classes, in some cases achieving an error of 100% (the class ERL, which has very few individuals).
A possible solution that could improve our classification, which is different from applying our techniques used for pre-processing, could be building an **embedded classifier**, based on grouping similar classes in bigger groups to firstly differentiate between these new groups and later separate the groups in the initial classes. We discard this option because of our lack of background in the field of the project, but with the help of an expert in proteins and Yeast this could be tried.
Another possible solution could be make the analyses but instead of using the whole dataset, a **sampling** of it, where the proportion of classes is more equally distribute among the individuals. **Replicating individuals or subsampling** could be forms of implementing this option.
In addition, we only used 6 different methods and 3 ways of pre-processing the data from the wide world of Machine Learning, so **more methods and pre-processing techniques** should be taken into account so as to check if the prediction could be improved.
In the end, we could affirm that working with a dataset of these characteristics (protein features) without a learned **background** in the field and with little experience in Machine Learning it is not the best idea, but it was worth to do it because we used this chance to **gain experience and knowledge** that will be very useful in the future.

References

[1] Yeast Data Set, *From UCI machine learning repository*, `http://archive.ics.uci.edu/ml/datasets/Yeast`.

[2] Paul Horton and Kenta Nakai, *A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins.*

[3] Christopheer M. Bishop, *Pattern Recognition and Machine Learning*, second edition.

[4] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, *An introduction to Statistical Learning*, sixth edition.

[5] Skicit-learn web, *Machine Learning in Python*, `http://scikit-learn.org/stable/`.