Project-I by Group Phnom Penh

Christophe Praz
EPFL
christophe.praz@epfl.ch

Daniel Wolfensberger
EPFL
daniel.wolfensberger@epfl.ch

Abstract

In this report, we summarize the results of the first PCML project. In the first section we applied linear regression methods to predict real-valued data. We found that the output variable distribution was clearly trimodal and difficult to fit as a whole, so we separated it into three classes that we analysed and fitted separately. Since the matrix were ill-conditioned, we applied ridge regression to the three intra-class datasets. We noticed that considering the square and cubic power of the features was improving the test-error so we included them in the final model. In the second section we performed binary classification by using logistic regression. It appears that a logistic model that includes all variables without any feature transformation classifies the data relatively well. Non-penalized and penalized logistic regression give very similar prediction variance and error, but the latter one is recommended as it is more stable.

1 Regression

1.1 Data description

The train dataset consists of output variables $\mathbf{y}$ and input variables $\mathbf{X}$, with $N = 2800$ samples. Each input vector $\mathbf{X}_n$ is of dimensionality $D = 73$. Among these 73 input variables, 62 are real valued, 3 are binary and 8 are categorical (3 variables with 3 categories and 5 variables with 4 categories).

A test dataset consisting of $N = 1200$ samples with the same dimensionality as the training set is also provided, although the output vector y is not know in this case. The goal of this first exercise is to produce predictions $\tilde{\mathbf{y}}$ for the test dataset, as well as an approximation of the test-error.

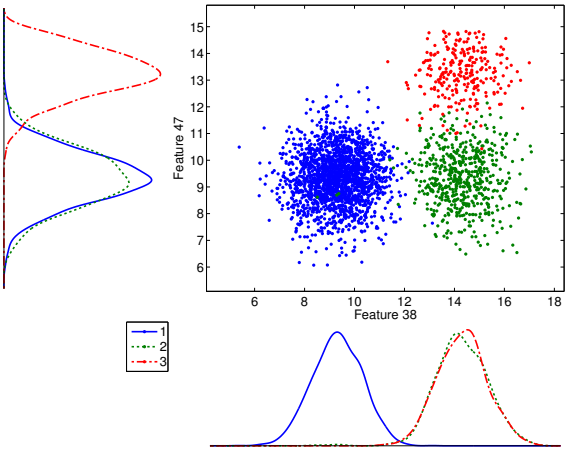
1.2 Data visualization and cleaning

We performed basic exploratory data analysis on our data. We noticed that most of the real valued input variables follow a normal distribution with different mean and standard deviation, so we normalized them. However, two input variables do not follow this rule and seem to be bimodal. These 2 variables (feature 38 and 47) are illustrated in Figure 1(a). By investigating the correlation between input and output variables, we also noticed that these two features were the most correlated with $\mathbf{y}$. Figure 1(b) shows the histogram of the output variable. We clearly see that the $\mathbf{y}$ distribution is trimodal and can be sub-divided into three parts.

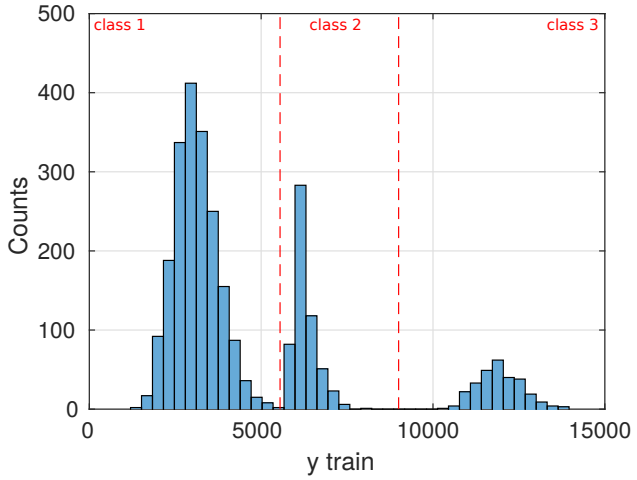
We used dummy encoding for the ordinal/categorical variables, increasing the set of input variables to 86 (62 real, 3 binary, 21 dummy).

1.3 Data clustering : a three classes regression model

We applied least-squares and ridge regression to the whole dataset but could not reduce the RMSE to a satisfactory level. Preliminary results and $\tilde{\mathbf{y}} - \mathbf{y}$ scatterplot encouraged us to divide the dataset into three classes, as illustrated on figure 1(b). The problem is thereby divided into two distinct parts: 1) Predict values for $\tilde{\mathbf{y}}$ within each class using a regression method. 2) Classify datapoints into one of the three classes based on the input variables. Both steps will include different prediction errors and we would have therefore to analyse the contribution of each of these to the final prediction error.



(a) 2d histogram of feature 38 against feature 47. The three classes are displayed with different colours.



(b) Histogram of $\mathbf{y}$. Dashed lines indicate where we split the training set (y=5650 and y=9000) to form the three classes.

Figure 1:

1.4 Intra-class feature transformations and ridge regression

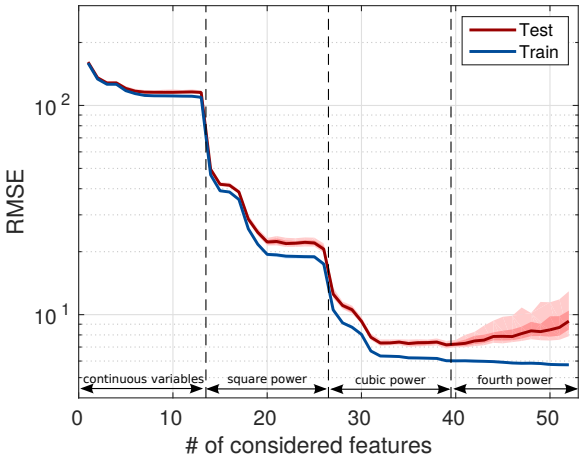
Since we applied the exact same methodology to the three classes, we will describe and illustrate here only the procedure for class 2. Table 1 summarizes the results obtained within the two other classes. For the rest of this section, we will always refer to the dataset belonging to class 2, composed of $N_{c2} = 564$ datapoints $(5650 < y < 9000)$.

In a first step, we applied simple least-squares to the dataset in order to select which features to include in our regression model. For sake of simplicity, we first discarded the binary and dummy variables. A quick look into the distribution of $\mathbf{y}_{c2}$ with respect to the real valued input variables suggested us to consider elevating the input variables to the powers 2 and 3. In order to choose only the most relevant features among the 62 available, we first sorted them in decreasing order of correlation with $\mathbf{y}_{c2}$. We then inserted them one by one into the least-squares regression model, and consecutively added the square and the cube power of these features. For each considered number of features, we performed 1000 iterations of 5-fold cross-validation and computed the train and test error. The errors were then averaged over all splits given by one instance of cross-validation, resulting in 1000 errors for each model configuration. Following a trial and error procedure, we found that only 13 real valued input features were improving the test error, so we decided to discard the rest of them. In a similar way, we confirmed that adding the square and cubic power to these variables was improving the average test error, although adding more powers (4, 5, ...) was increasing both the average test error and its variance. Figure 2(a) illustrates our best model as well as this one-by-one iterative process.

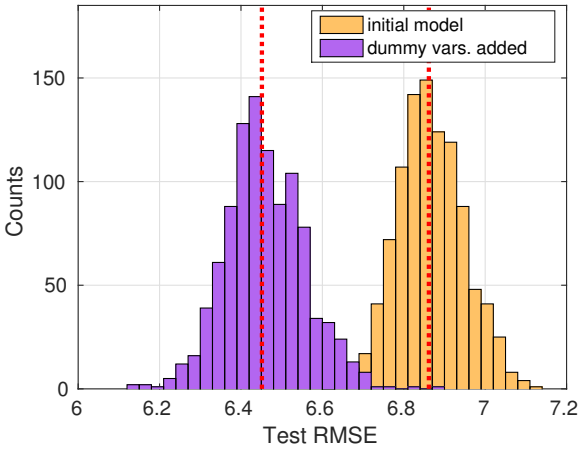
We then tried to add the binary and dummy variables to the model. We found that we get a small improvement both in the train and test error. Figure 2(b) illustrates the improvement in the test error.

Note that we picked 5-fold cross-validation because it seemed to us to be the best trade-off considering the size of our data samples N_{c1}, N_{c2} and N_{c3}, as displayed on Table 1.

Now that the model is defined, we need to train it in order to get the most accurate regression coefficients $\boldsymbol{\beta}_{c2}$. Since the matrix is ill-conditioned, least-squares results can not be trusted so we decided to apply gradient descent and ridge regression. Ridge regression was computationally faster and giving slightly better preliminary results so we decided to focus on that method. We varied the value of λ from 10^{-2} to 10^{2}. Similarly to what we did to define the model, we ran 1000 iterations of 5-fold cross-validation for each value of λ and computed the associated train and test errors. Figure 3(a) shows the RMSE as a function of λ obtained for class 2. We can see that there is a small improvement obtained for some values of λ with a minimum test RMSE at $\lambda_{c2} = 12.4$. Based on this, we built the $\boldsymbol{\beta}_{c2}$ vector by applying ridge regression to the whole intra-class training set N_{c2} using λ_{c2}. Table 1 summarizes the main characteristics of the three intra-class models. The reported test error is the average test error obtained over the 1000 iterations of 5-fold cross-validation using the best λ.



(a) RMSE as a function of the considered input variables inserted in the model. The double arrows indicates the type of features added in the intervals delimited by the dashed vertical lines. The last quarter illustrates the effect of adding higher order powers into the model, which does not improve the regression performance. Dark areas correspond to the 25-75 percentile range whereas light areas correspond to the 10-90 percentile range. The bold line is the median.



(b) Comparison of least-squares regression performance in the absence or presence of binary and dummy variables. Each count in the histograms is the averaged RMSE obtained from a 5-fold cross-validation. 1000 iterations have been made to generate the histograms.

Figure 2:

Subset	N	# real feat. incl.	feat. transf.	bin. incl.	λ_{best}	$\text{RMSE}_{\text{test}}$
Class 1	1952	62 (all)	$\mathbf{X}^2, \mathbf{X}^3$	yes	47.8	5.70
Class 2	564	13	$\mathbf{X}^2, \mathbf{X}^3$	yes	12.4	6.32
Class 3	284	33	$\mathbf{X}^2, \mathbf{X}^3$	no	0.68	15.60

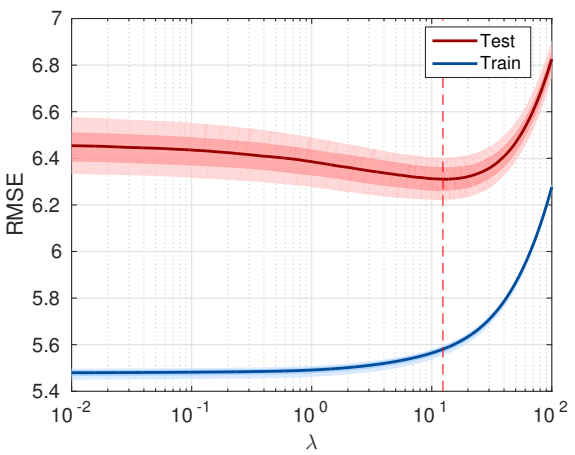
Table 1: From left to right : class ID, intra-class dataset size N, number of real valued features included in the model, feature transformations included in the model, whether binary and dummy variables were included in the model, value of best λ for ridge regression, intra-class test RMSE.

We can see that globally the intra-class test RMSE is quite satisfactory, knowing that $\mathbf{y}$ spans between $\sim$1500 and 14000. It must be noted that in term of train and test RMSE, the improvement brought using ridge regression over least-squares was modest.

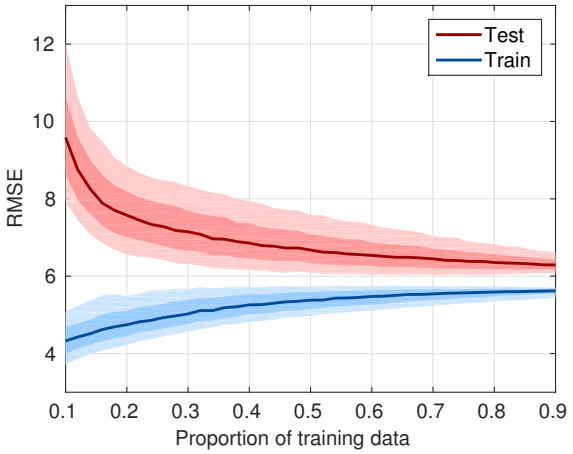
Finally, we computed a learning curve for each class. We first held out 20% of the data that we kept as a fixed test sample. Then, we iteratively increased the training sample from 10% of the rest of the dataset to 90%. For each proportion of the training data, we did 1000 iterations of random training data sampling in order to get insight on the variance of the error. We used ridge regression with $\lambda = \lambda_{\text{best}}$ for training. Figure 3(b) shows the learning curve thus obtained for class 2. We see that the train and test errors seem to converge, with variance of both RMSE decreasing as we increase the training data sample. The curves look also very smooth, as expected after averaging 1000 iterations. The gap between the two curves could suggest that we are slightly overfitting our data, although it is reasonably small and might also be explained by the limited amount of data at disposal ($N_2 = 564$).

1.5 Global model and error estimation

What we did so far was to define a regression model for each class, resulting in three regression coefficients vectors $\boldsymbol{\beta}_{c1}$, $\boldsymbol{\beta}_{c2}$ and $\boldsymbol{\beta}_{c3}$. As explained in section 1.3, the next step is now to classify the data into one of the categories based on the input features. This has been done considering the features 38 and 47 only. As one can see on Figure 1(a), these two features regroup the datapoints into three distinct clusters which are actually our three classes. We assumed that each cluster follows a bivariate normal distribution and that these two distributions are independent (ie. $\boldsymbol{\Sigma} = \text{diag}(\sigma_{c_i,38}, \sigma_{c_i,47})$), and we computed its centroid $\boldsymbol{\mu}_{c_i}$ and standard deviation $\boldsymbol{\Sigma}_{c_i}$ using the whole datasample. For a given datapoint (y_n,$\mathbf{X}_n$), its corresponding class is determined by computing the minimal Euclidean distance



(a) Ridge regression applied on class 2. λ_{best} is indicated by a the red dashed line.



(b) Learning curve for class 2.

Figure 3:

with respect to the clusters' centroid weighted by the cluster standard deviation. Mathematically, this could be written as

$$\mathrm{C}(y_n, \mathbf{X}_n) = \min_{c_i} \left(\sqrt{(x_{38} - \mu_{c_i,38})^2/\sigma^2_{c_i,38} + (x_{47} - \mu_{c_i 47})^2/\sigma^2_{c_i,47}} \right) \tag{1}$$

This simple model was able to classify the whole dataset into the three classes defined on Figure 1(b) with an accuracy of 98.8%. However, we noticed that for the 1.2% that are misclassified, the use of the incorrect regression model was leading to very large errors, ending up with relative errors between prediction and real value of y as large as 100%. This last point has to be emphasized since this couple of "outliers" will have a deep negative impact on the global RMSE, much stronger than the intra-class RMSE.

An alternative solution to reduce the impact of the misclassified datapoints on the RMSE is to consider a probabilistic approach in which we compute the predictive value of a datapoint using the three regression models, and then weight them according to the probability that this datapoint belongs to that class (using the same multivariate normal distribution hypothesis). If p_{c1}, p_{c2} and p_{c3} are these probabilities, prediction $\tilde{y_n}$ for a datapoint described by $\mathbf{X}_n$ is given by

$$\tilde{y_n} = p_{c1}\tilde{\mathbf{X}}^t_{n,c1}\boldsymbol{\beta}_{c1} + p_{c2}\tilde{\mathbf{X}}^t_{n,c2}\boldsymbol{\beta}_{c2} + p_{c3}\tilde{\mathbf{X}}^t_{n,c3}\boldsymbol{\beta}_{c3} \tag{2}$$

As expected, the weightings attenuate the error associated with the misclassified points. On the other hand, it will slightly increase the error of the correctly classified points, particularly if these points are located at the vicinity of another cluster. Because the RMSE is more penalized by a couple of large errors than many small ones, we decided to keep the probabilistic approach in our final model. In real life, this choice would have to be made depending on the context of the problem, whether we want to privilege a more accurate model which has a $\sim$2% chance of failing badly or a slightly less accurate one which mitigates the error in case of failure.

Finally, we computed the global test RMSE averaging 1000 iterations of 5-fold cross-validation applied on the whole dataset. Predictions were calculated using (2). We obtained an average test RMSE of **431.18** $\pm$ **75.78**, where the $\pm$ interval corresponds to the standard deviation computed over all cross-validation instances (ie. 1000$\times$5 RMSE). It gives us a larger, more conservative uncertainty range that if we had computed the standard deviation over the average RMSE per cross-validation.

2 Classification

2.1 Data description

The train dataset consists of output variables $\mathbf{y}$ and input variables $\mathbf{X}$, with $N = 1500$ samples. Each input vector $\mathbf{X}_n$ is of dimensionality $D = 30$. Among the input variables, 1 is binary, 3 are categorical with respectively 4, 4 and 3 categories. The output variable $\mathbf{y}$ is binary with possible values of -1 and 1. In the training set 823 samples have a value of $y = -1$ and 677 a value of $y = 1$.

The test dataset consists of $N = 1500$ samples with the same dimensionality as the training set ($D = 30$). However the output vector $\mathbf{y}$ is not available. The goal of this second exercise is to classify the test dataset and to provide an approximation of the test classification error.

2.2 Data visualization and cleaning

We performed basic exploratory data analysis on our data. For sake of brevity, we will not present all the feature distributions but it is important to notice that most continuous distributions are strongly assymetric, with sometimes several modes and cover widely different ranges. This is not the case for categorical variables however, as their distributions are almost uniform with no dominant category.

2.3 Feature transformations

Since the input variables cover a wide variety of ranges, we normalized them (using z-scores) before applying the logistic regression. We tried to remove outliers using a standard method (remove everything below $\mathrm{Q}_{25} - k \cdot \mathrm{IQR}$ or above $\mathrm{Q}_{75} + k \cdot \mathrm{IQR}$). Using $k = 1.5$, a value often used in practice, led to the removal of more than 80% of the rows[1], whereas $k = 3$ still led to the removal of more than half of the rows. In the end, we decided to keep the data as it is without removing any possible outlier, in order to avoid possible overfitting.

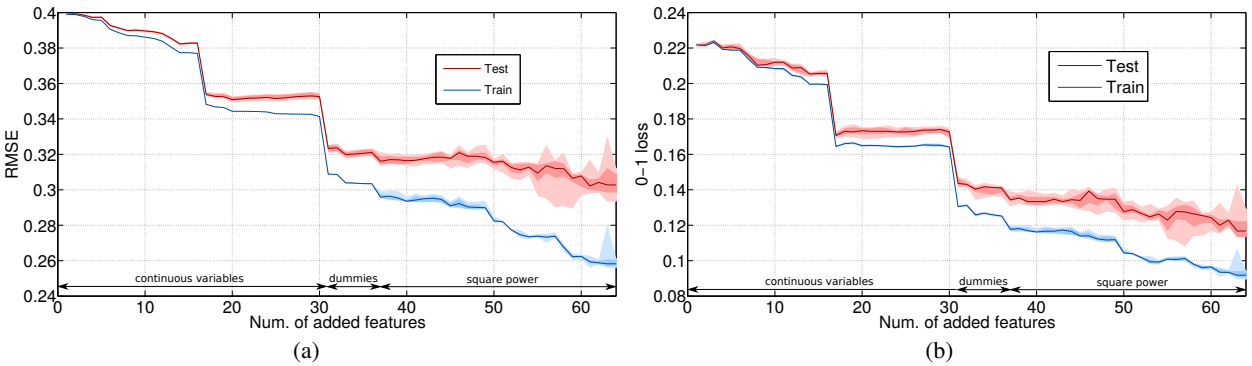
The output variable was converted to a binary 0-1 variable by assigning values of 0 to all samples with a value of $y = -1$.

2.4 Model selection

In a first step, we selected the appropriate features to include in our logistic model. Since, with 36 features, the possible combinations of features is nearly infinite, we decided to consider only the dummy variables and the continuous variables raised at powers 1, 2 and 3. In order to choose the best number of features, we added the features one by one to the logistic model by ordering them by decreasing correlation with the output variable. For every considered number of features, we performed 10 iterations of 10-fold cross-validation using normal logistic regression to fit a set of parameters. The train and test errors were then averaged over all splits given by the 10-fold cross-validation. This gives thus 10 errors for every model configuration. It would have been preferable to perform more iterations but since the problem needs to be solved numerically, this would have taken a long time. The distributions of test and train errors for every considered model are shown in Figure 4.

It can be clearly seen on the figure that there is no clear improvement in considering powers larger than 1 because the error bias does not really decrease whereas the error variance increases significantly. Dummy variables however seem to be useful for classifying the output.

Based on this knowledge, we decided to consider only the normalized input variables as well as the dummy variables while ignoring powers larger than one.



(a) (b)

Figure 4: Distributions of the RMSE (a) and 0-1 loss (b) as a function of the number of considered variables. The power 3 is not displayed since the hessian is often ill-conditioned and the general trend is similar to the power 2.

[1] A row is removed if at least the value of one feature is considered as an outlier

2.5 Simple (non-penalized) logistic regression

To minimize the cost of the logistic regression problem we used the Newton method which showed better convergence properties than the gradient descent method. To get an estimate of the test errors on the test dataset, we performed 100 iterations of 10-fold cross-validation, fitting every time a new vector of $\boldsymbol{\beta}$ and averaged all the resulting test errors (RMSE, 0-1 loss and logarithmic loss) to get the mean errors displayed in Table 2. As in the regression part, to be more conservative, the standard deviations were computed over all cross-validation instances (100×10 error values).

Finally, we fitted a $\boldsymbol{\beta}$ vector on all data from the training dataset and used it to classify the samples from the test dataset.

Method	$\text{RMSE}_{\text{test}}$	$\text{0-1-loss}_{\text{test}}$	$\text{logloss}_{\text{test}}$
Simple logistic	$0.3214\ (5.3863 \cdot 10^{-4})$	$0.1396\ (7.4188 \cdot 10^{-4})$	$0.357\ (4.3 \cdot 10^{-3})$
Penalized logistic	$0.3213\ (5.4230 \cdot 10^{-4})$	$0.1399\ (7.3132 \cdot 10^{-4})$	$0.3551\ (3.8 \cdot 10^{-3})$

Table 2: Estimates of test errors and their standard deviations (in parenthesis) for the simple logistic regression and the penalized logistic regression (Section 2.6)

2.6 Penalized logistic regression

In order to choose the best λ parameter for penalized logistic regression, we performed 10 iterations of 10-fold cross-validation for 100 different values of logarithm of λ ranging from -2 to 3. The corresponding train and test errors are shown in Figure 5. It can be seen that values of λ below 10^2 barely affect the train and test errors. Beyond this value however both the test and train RMSE increase strongly. This result seems to suggest that for penalized logistic regression any λ below 10^-2 could be choosen. There is however a very small mimimum in the test error around λ = 0.85, so this is the value that we used for the penalized logistic regresison. It is also worth noticing that in general the number of iterations until convergence increased with λ.

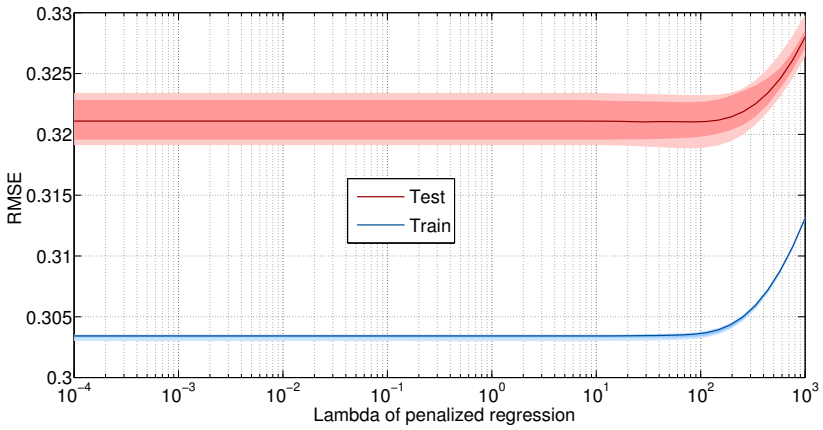


Figure 5: Distributions of the RMSE for the test and train datasets as a function of the λ parameter of the penalized logistic regression

To get an estimate of the test errors we performed similarly as in the simple logistic regression, by performing 100 iterations of 10-fold validations. The average errors are shown in Table 2.

Finally, we can observe that classification gives satisfying results with relatively small errors. Non-penalized and penalized logistic regression give very similar results but the later method has the advantage of being more stable and allows convergence even for perfect classification.

Acknowledgments

All the work presented in this report was done by Christophe and Daniel. Christophe worked mostly on the regression part whereas Daniel worked mostly on the classification part, but we were constantly in interaction. We would like to thank Emti for the report sample which served as a model for the structure of our report.