MACHINE LEARNING

Course Work

Student Number: 190533706

Table of Contents

Part 1. Clustering

In this part we are given a dataset with, presumably, employee survey answers. The data contains about 7800 responds and each respond is escribed by eleven variables: sex (binary), age, five questions about the mood and personal condition of the interviewee in everyday life and four questions about his/her feelings at work. The responses to all nine questions were frequencies of a given type of feelings.

	sex	age	q1	q2	q3	q4	q5	w1	w2	w3	w4
0	1.0	63.0	3.0	3.0	3.0	3.0	3.0	2.0	2.0	2.0	2.0
1	2.0	58.0	2.0	3.0	2.0	3.0	2.0	2.0	3.0	2.0	2.0
2	2.0	32.0	2.0	2.0	3.0	2.0	3.0	2.0	2.0	2.0	2.0
3	1.0	35.0	3.0	2.0	2.0	2.0	3.0	2.0	2.0	2.0	2.0
4	2.0	27.0	2.0	2.0	3.0	3.0	2.0	2.0	4.0	2.0	2.0

Table 1. Database

The goal was to divide the respondents into several groups sharing common characteristics using machine-learning (ML) techniques such as dimension reduction and clustering analysis. We show that the data can be divided into three different clusters of respondents.

PCA

We start by applying the principal component analysis or PCA and draw a PCA plot. This technique may be practical when we face a multidimensional data (we have 11 dimensions). A PCA plot converts correlations among all the observations into a 2D graph. Thus, observations that are highly correlated cluster together. This is done by comparing variations among all dimensions and singular value decomposition (SVD). As a result, we get principal components (PC) data-frame with the same number of dimensions as the initial data-frame. The only difference that makes the principal components more descriptive and allows to use them instead the initial variables is that they explain much more variation among the observations. Typically, a scree plot is created first, in order to visualize how much variation is explained by each PC.

PCs are then sorted in the descending order, such that PC1 explains the most variations. Then the PCA plot is plotted using, most commonly, PC1 and PC2.

We apply PCA to raw data (some data modification and achieved the following results:

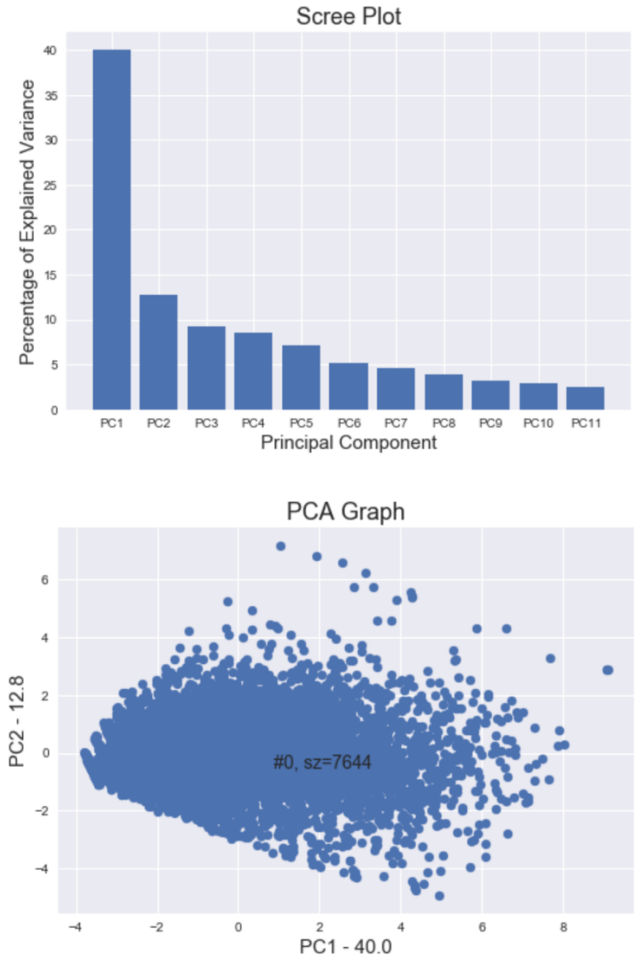

Fig. 1 – Raw Data PCA

As we can see, the first principal component PC1 accounts for 40 percent of the variation. In contrast, PC2 explains just 12.8%, which is not much more than PC3. This leads to poor division. As can be seen on the PCA plot, observations are spread significantly along the PC1 axis, but the variation along the PC2 is negligible. Thus, unfortunately, PCA has not revealed any clusters.

K-means Clustering

K-means clustering is a far less complicated and very commonly used technique. The main idea behind this algorithm is to first, choose the number of clusters - k, second, choose k random cluster centers and assign each point to the class with the closest center. Third, repeat the second step until you reach the maximum number of iterations and choose the model for which the variance within each cluster is minimum.

In order to identify the optimal number of clusters we use the elbow method. This method chooses the number of clusters when the inertia is the smallest. Inertia_ is the sum of squared distances of observations to their cluster centers.

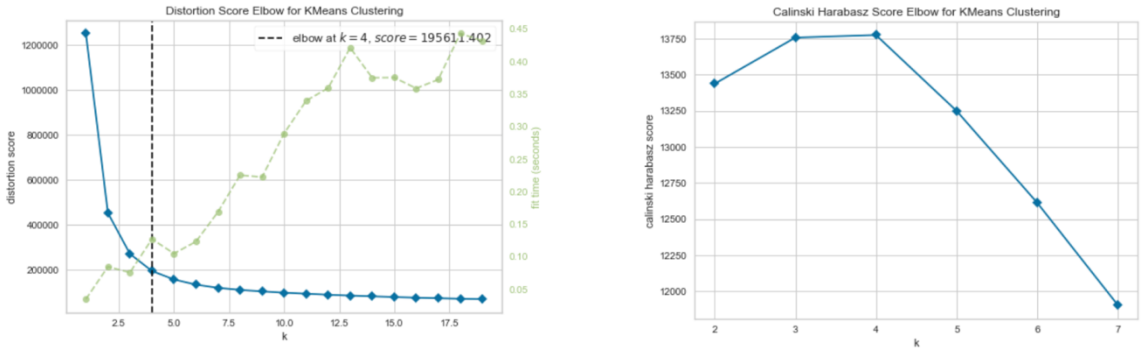

Fig. 2 - Elbow Method

After applying the elbow method, we achieved four clusters as an optimal number of clusters. However, the Calinski Harabazs metric revealed no significant difference between k equal to three and four.

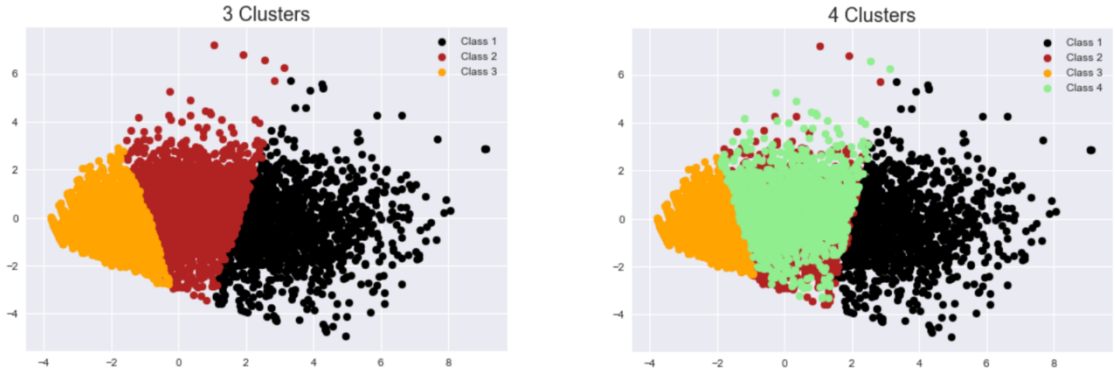

Fig. 3 - Clusters

Moreover, as we plot the four clusters, it is obvious, that two clusters intersect much.

As a result, there are three groups of respondents. PCA applied to this data appeared to be less useful compared to K-means clustering.

Part 2 – Regression

In this part we are given two datasets with information about students from two Portuguese schools. Datasets contain students' grades in two different subjects (Math and Portuguese) as well as their demographic, social and school related features. For convenience I am going to call the dataset with Math and dataset with Portuguese D1 and D2 respectively.

D1 contains 395 observations, while D2 – 649. Each observation in both datasets contains 30 (without G1 and G2) features. The target in both datasets is the student's final grade G3.

Preprocessing

Most of the variables are represented by a yes/no value. Such form is inconvenient for most of the models, so we convert them into binary 1/0. There are several variables with nominal values. We transform all of them into dummy variables, because nominal values are inconvenient and just replacing them with numbers creates a non-existing distance between some of the values.

We split the data into train and test sets. We are going to compare the performance of each model by two metrics: R-squared (R2) and mean square error (MSE) in the test set. We also conduct bootstrap testing in order to achieve a more accurate error estimation.

OLS

We begin by fitting an Ordinary Least Squares linear regression OLS. First, we fit the model to the whole D1 using all the predictors and got such results: R2 = 0.349 and

MSE = 21.6. This is poor compared to 0.861 and 4.25 achieved by the model using G1 and G2 as predictors of G3. Using D2, the same models returned 0.402 and 9.52 and 0.859 and 1.76 respectively.

Then we tried to change the predictor set. This time only the predictors with correlation with the target higher than 0.2 were used. The number of predictors in D1 shrunk from 30 to 2 – years of mother's education and number of failures. This model explained the variance in the data even worse - R2 fell to 0.169. However, the test MSE improved from 21.6 to 16.04, which, we believe, is the most important. Using the same approach on D2 we have achieved the best R2 = 0.927. MSE, on the other hand, rose from 9.52 to 12.42.

Since there is a big number of predictors, it is important to choose only the correct ones. For this reason, we tried the Backward Elimination algorithm which combined the models until all the p-values of the features were less than 0.05. p-values account for the probability, that a predictor is really significant in predicting the given target. This method worked slightly worse on D1 than the previous one. On D2 such method decreased MSE to 9.07, although R2 was just 0.364.

It can be noticed that many features depend on each other. For example, mother's and father's education, which is logical because usually people tend to marry equally educated people. We have found all 30 pairs of predictors with correlation greater or equal to 0.3 and created a model of them and their products. While this approach has not made any improvements in D1, in D2 there was the lowest for OLS MSE = 5.93 and a moderate R2 = 0.427.

Finally, we regressed G3 on failures only and achieved good accuracy in both datasets. For D1 MSE = 13.98, R2 = 0.139 and for D2 – MSE = 6.2, R2 = 0.183

Lasso & Ridge

We decided to use Lasso and Ridge regression because these algorithms filter the predictors well. For example, Lasso applied to D2 returned such feature importance graph:

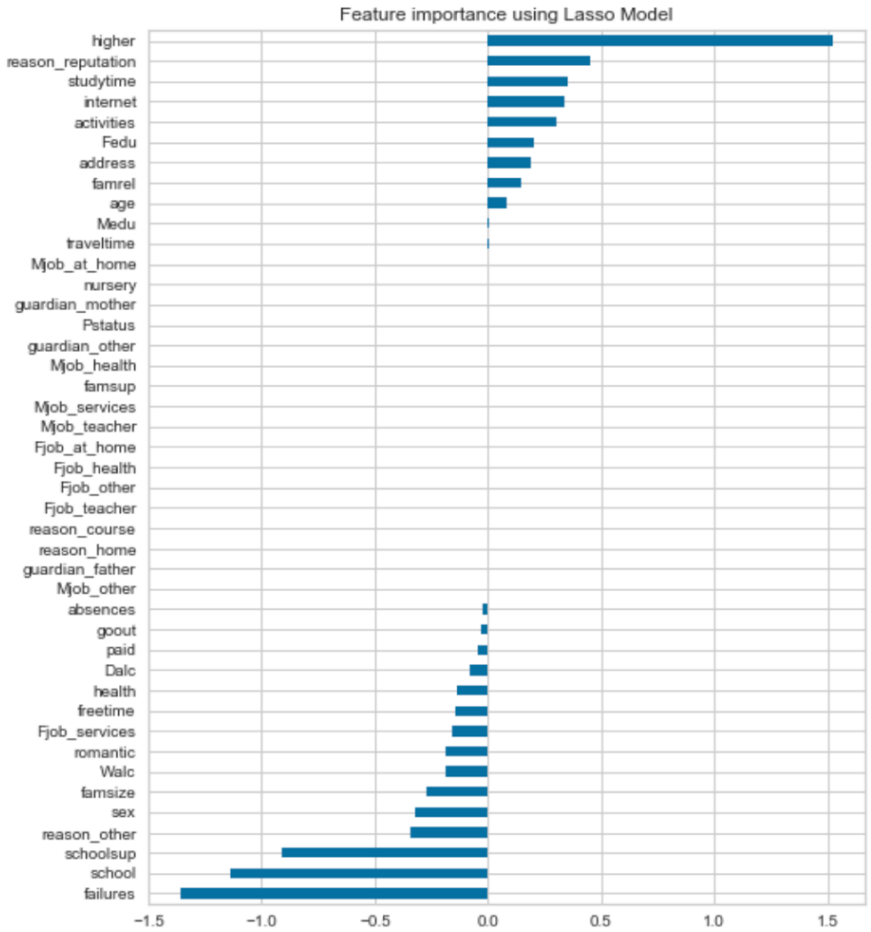

Fig. 4 - Feature Importance Lasso D2

It has revealed that the final grade of a student, G3, does not depend on the parents' job, type of guardian and others. After fitting this model to D2 we have achieved a good prediction and moderate explanatory power. MSE was equal to 6.9 and R2 – 0.337.

D1 was better processed using Ridge than using Lasso. Here the choice of variables was different, as can be seen on Fig. 5. The MSE and R2 were 16.2 and 0.223 respectively.

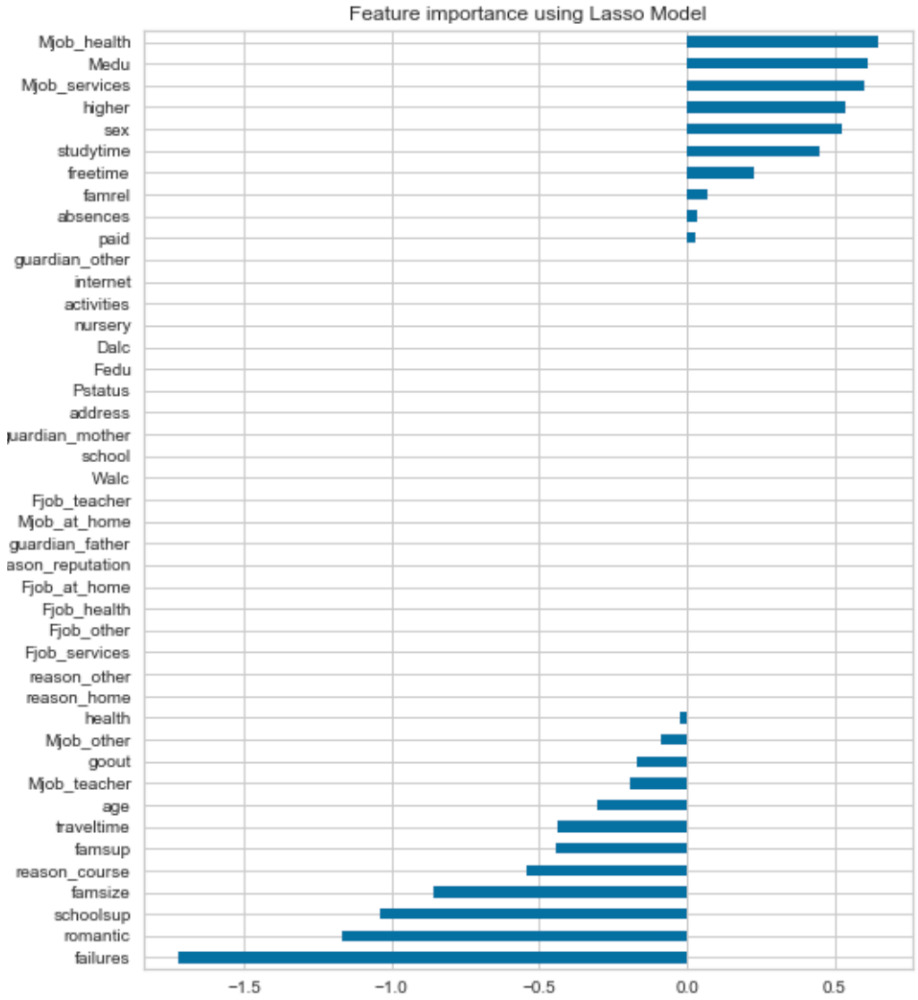

Fig. 5 – feature Importance Ridge D1

Trees

We decided to fit regression trees and achieved moderate results on D1 and worse results on D2 compared to OLS methods.

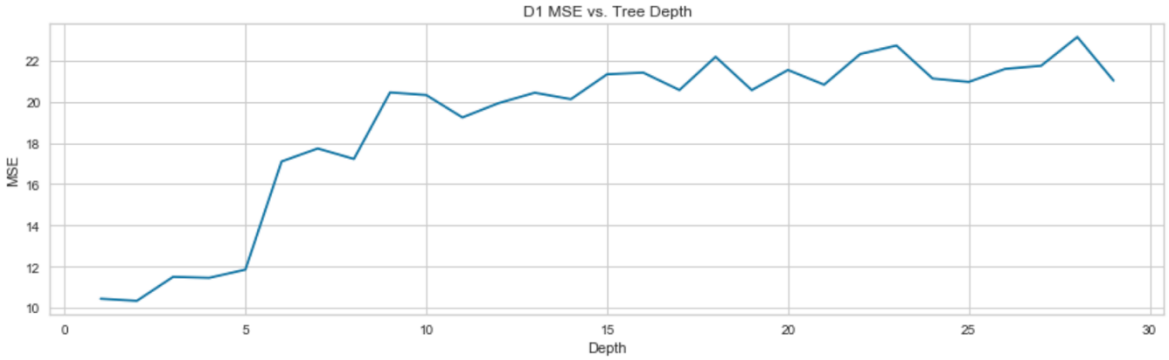

Fig. 6 – D2 MSE vs. Tree Depth

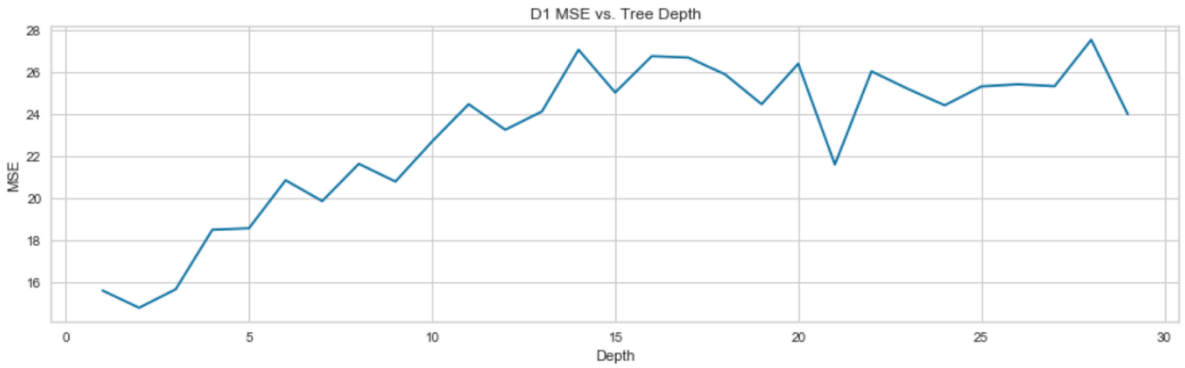

Fig. 7 – D1 MSE vs. Tree Depth

Both models behaved similarly when tree depth was changed, as can be seen on figured 6 and 7. Best MSE for D1 was 14.76 with tree depth equal to 2. Best MSE for D2 was 10.33 with tree depth also 2.

Other models

We also fit AdaBoost and Random Forest. Both of these methods tend to slightly improve prediction compared to Regression Trees, however, these results are not worth to be described here.

Part 3. Classification

In this part we are dealing with a bank dataset. Each observation represents a call to a potential client of the bank with a commercial offer to create a deposit. The target variable is a binary response to the offer – 1 if the client was attracted ant 0 otherwise.

There are about 4500 observations each containing 16 features describing the potential clients' social information and data considering previous attempts of the bank to attract him. There are missing values and the whole dataset is highly imbalanced.

Most of the variables are binary, but with nominal yes/no values, so we convert them into 1/0.

We start by revealing the data dependencies.

Fig. 9 Data Dependency

As we can see form the plots above, there are some values of the variables for which a positive response of the potential client is more likely. Thus, we can create a new data frame where these values are hot-encoded. For example, instead of keeping all 3 categories of the ‘poutcome’ variable, it is practical to keep 1 if the outcome was successful and 0 otherwise. We do the same for ‘marital’, ‘education’ and ‘job’ columns.

Since we are dealing with a highly imbalances data, it is incorrect to use simple accuracy as the goodness of prediction metric. We use F1 and RocAuc scores.

F1 = 2 * (precision * recall) / (precision + recall)
precision = TP / (TP + FP)
recall = TP / (TP + FN)

Another method that we use is under-sampling for training procedure. This allows to fit models on balanced samples.

Logistic Regression

First, we randomly divide all observations into train and test sets, regardless, that the data is imbalanced. As a result, we get the following confusion matrix, ROC Score = 0.76 and F1 = 0.38, which accounts for a bad balance of prediction. Namely, in this case recall is low and which decreases the numerator of F1.

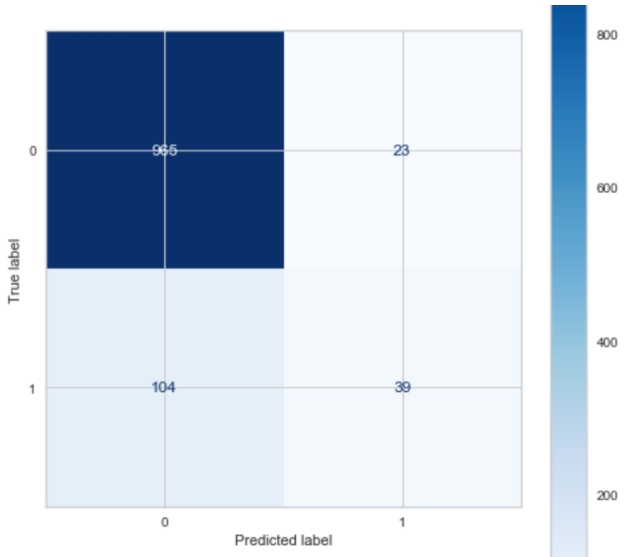

Fig. 10 Confusion Matrix

Then, we used the under-sampling technique and significantly improve performance of the model.

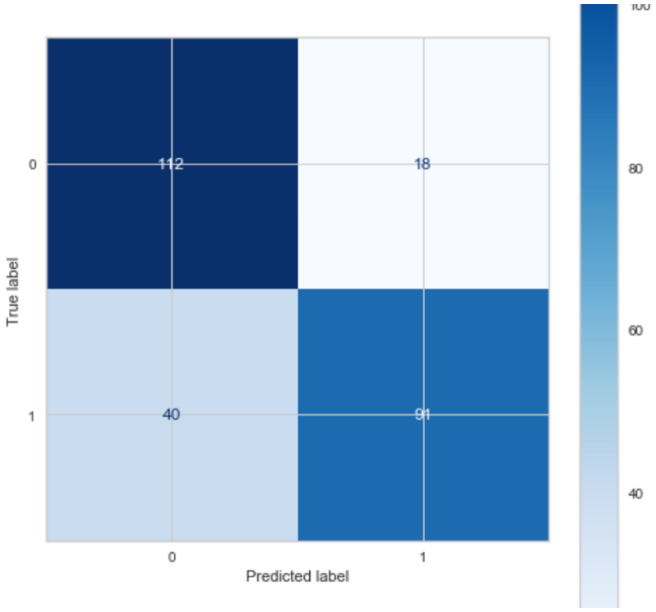

Fig. 10 Confusion Matrix

This time ROC coefficient rose to 0.68 and F1 – increased to 0.6, which is a good estimation.

We also tried other models, i.e. Random Forest and AdaBoost, and XGBoost, however little improvement was achieved to describe these models here.