UNIVERSITÄT
BIELEFELD

Report of the third project

Yasmin Bouhdada, Annalisa Dettori, Andrea Schinoppi

January 2024

1 Introduction

In this project, we will apply three methods of features selection to a real-world data set, extracting the 2 and 6 most important features with a *random forest regressor*, based on the R^2 scores.

Next, we will work with a second dataset, with the purpose to train a *random forest classifier*, a *Naive Bayes classifier*, a *logistic regression classifier* and *k-NN classifier*. After that, we will do feature selection with the *random forest classifier* and select some variables with which we will re-train the models.

Then, we will train the same classifiers on a third dataset, in order to find that one that could provide a *F1 score* of 0.95.

The last part of our work is about clustering; we will cluster images with *k-means* and evaluate the performances.

2 Data Set

2.1 First data set

The first data set (named `real world data`) was a real data set regarding the prices of some houses. It was made of ten columns $X_1, \ldots, X_{10}$ for the independent variables and one column for the dependent variable y. The data set had 1095 numeric samples and it had no NaNs.

We could resume the main information of the variables with this table:

	LotArea	TotalBsmtSF	1stFlrSF	2ndFlrSF	GrLivArea	WoodDeckSF
mean	10722.41	1159.84	0	338.71	1505.13	$91,06$
std dev	11054.40	376.46	34900	432.04	514.24	120.64
min	1300	0	343	0	334	0
max	215245	3206	3228	1872	4676	670

	OpenPorchSF	3SsnPoarch	ScreenPorch	PoolArea	y
mean	47.26	2.78	15.09	2.14	179984.82
std dev	66.79	25.18	56.55	35.79	77610.06
min	0	0	0	0	34900
max	547	407	480	738	755000

2.2 Second data set

The second data set (`dataset1.npz`) was made of ten columns X for the independent variable and one column for the dependent variable y. The data set had 1000 numeric samples, it had no NaNs and in the dependent variable y we had boolean values $\{0, 1\}$ referring to two classes. We didn't know what the variables represented, but we could resume the main information of the variables with this table:

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
mean	0.66	-0.06	0.02	-0.20	0.04	-0.10	0.12	0.06	0.08	0.06
std dev	2.60	5.80	3.50	5.76	2.69	5.75	5.78	5.90	3.31	5.79
min	-7.32	-9.98	-8.44	-9.95	-9.13	-9.98	-9.98	-9.99	-7.91	-9.98
max	7.4	9.97	7.43	9.97	8.92	9.98	9.98	9.97	8.26	9.97

2.3 Third dataset

The third data set was made of thirteen columns X for the independent variable and one column for the dependent variable y. The data set had 110 numeric samples and it had no NaNs. The data set was split in 50 samples for the training set and 60 for the test set. We didn't know what the variables represented, but we could resume the main information of the variables with these tables:

For the training sets X_{train}, y_{train}:

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
mean	-0.28	-0.33	-0.25	0.02	-0.03	-0.31	-0.48	0.08	-0.19	-0.31
std dev	0.92	0.79	0.99	1.06	1.26	1.13	1.00	1.13	1.15	0.8
min	-1.49	-1.23	-3.31	-2.42	-1.95	-2.62	-2.61	-1.84	-2.5	-1.51
max	2.13	2.17	1.79	3.31	4.05	2.44	1.71	2.76	2.84	2.23

	X_{11}	X_{12}	X_{13}	y
mean	0.36	-0.39	-0.34	1.82
std dev	0.90	1.16	0.85	0.38
min	-1.6	-2.88	-1.46	1
max	2.33	2.04	1.96	2

For the test sets X_{test}, y_{test}:

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
mean	-0.06	0.15	0.18	0.19	-0.13	-0.02	0.12	0.14	0.01	-0.10
std dev	1.03	1.05	1.06	0.97	0.79	0.82	0.90	0.92	0.89	1.01
min	-2.16	-1.40	-2.16	-2.18	-1.30	-2.10	-1.69	-1.47	-1.72	-1.80
max	2.04	4.37	3.01	2.87	2.10	1.43	3.52	3.03	3.40	2.78

	X_{11}	X_{12}	X_{13}	y
mean	0.17	0.02	-0.06	1.50
std dev	0.86	1.08	1.03	0.5
min	-1.76	-1.36	-2.16	1.00
max	2.21	2.53	2.04	2.00

3 Foundations

3.1 Methods

3.1.1 Filter methods, wrapper methods and embedded methods

Given the regression (or classification) tasks $(\vec{x}_i, y_i)_{i=1}^n$, we decided to apply three methods to do feature selection: filter methods, wrapper methods and embedded methods.

- A filter method consists in assigning a relevance value $r(I)$ for every feature I based on the data only and then take the most relevant features only. In this case we decided to use F measure.

  Given a training set (X, Y), the F measure computes the Pearson correlation of the inputs to the output for each feature.

- A wrapper methods selects features iteratively based on their impact on the chosen classifier/regressor. We decided to be more complete to use both forward (adding a new variable starting from a model with zero variables) selection and backward (starting with the full model and always removing the worst value) selection.

- An embedded method integrates and adapts feature relevance terms within the model itself. We used Lasso that uses the formula $\min_{\vec{w}} NLL + \lambda||\vec{w}||_1$, where NLL refers to the negative data log likelihood.

3.1.2 Random forest classifier

Random forest classifier implements a bagging of decision trees with randomization of the input variable selection.

3.1.3 Naive Bayes classifier

See previous projects.

3.1.4 Logistic regression

See previous projects.

3.1.5 k nearest neighbours

See previous projects.

3.2 Evaluation

3.2.1 Cross validation

See previous projects.

3.2.2 Accuracy measures

As accuracy measures for random forest classifier, naive Bayes classifier, logistic regression and k-NN we used the accuracy score which is the sum of the true positive and true negative over all cases.

4 Experiments

4.1 Set-Up

Our experiments were carried on a jupyter notebook that we converted in a python file. The data sets' extension is `.npz`.

4.2 Analysis and Results

4.2.1 First data set

On the first dataset we did a variable selection with the previous explained methods and using the random forest regressor the *F measure* we selected the following features:

- Two features: `['TotalBsmtSF' 'GrLivArea']`
- Six features: `['2ndFlrSF' 'OpenPorchSF' 'WoodDeckSF' '1stFlrSF' 'TotalBsmtSF' 'GrLivArea']`

With the *forward feature selection* we selected the following features:

- Two features: `['TotalBsmtSF' 'GrLivArea']`
- Six features: `['LotArea' 'TotalBsmtSF' '1stFlrSF' '2ndFlrSF' 'GrLivArea' 'OpenPorchSF']`

With the *backward feature selection* we selected the following features:

- Two features: `['TotalBsmtSF' '2ndFlrSF']`
- Six features: `['LotArea' 'TotalBsmtSF' '1stFlrSF' '2ndFlrSF' 'OpenPorchSF' '3SsnPorch']`

With *Lasso* we selected the following features:

- Two features: `['2ndFlrSF' '1stFlrSF']`
- Six features: `['2ndFlrSF' '1stFlrSF' 'TotalBsmtSF' 'WoodDeckSF' 'OpenPorchSF' 'ScreenPorch']`

As we can see, the variable `'TotalBsmtSF'` had been selected almost always, except for *Lasso* that selected completely different variables. For six variables we always have different solutions, just some of them are recurrent. It's normal to have different solutions, but we should expect some to change not to have completely different results, but so happens with six variables.

First we start considering that our model complete of all variables had $R^2 = 0.51$.

With the selection of two variables we obtained the following results:

Method	R^2
F measure	0.22
Forward selection	0.22
Backward selection	0.20
Lasso	0.18

With the selection of six variables we obtained the following results:

Method	R^2
F measure	0.47
Forward selection	0.46
Backward selection	0.44
Lasso	0.44

We obtained very low R^2 values with the 2 selected features. We can interpret this as an important loss of information that the others features can give to the depending variable (price of houses). For sure these are the best two variables, but these models do not explain enough.

On the other hand, we obtained higher values when we selected 6 features, because we took into account other information given by more features, nevertheless the values are not still very high, but closer to the full model. We can probably assume that the dataset really needs all the variables to get a better performance, because there's not a enough correlation between variables such that some can be removed.

Since we got the best results with the *F measure*, we report the relative plots of the learning curves in Appendix A.

4.2.2 Second data set

We worked on the first dataset which was a classification task using a *Random forest classifier, Naive Bayes, Logistic regression and kNN*.
With all the models we used cross validation with five folds. The results we obtained for the accuracy are the following:

Method	Accuracy
Random forest classifier	0.91
Naive Bayes classifier	0.90
Logistic regression classifier	0.54
k-nearest neighbors classifier	0.65

Then we decided to use forward selection. We selected the model with the highest value for the R^2 on the test set which was the model complete of the second, third, sixth and nineth variable to obtain the following results:

Method	Accuracy
Random forest classifier	0.91
Naive Bayes classifier	0.90
Logistic regression classifier	0.51
k-nearest neighbors classifier	0.86

Finally, we plotted all the combinations of two features, as required by the exercise.

4.2.3 Third data set

The task to accomplish with this dataset was a challenge: we had to apply the learned concepts of the lecture to obtain a *F1 score* of at least 0.95 for the test set, without using it for training.

First of all we decided to train all the models we learned about to see their *F1-scores*:

Model	F1-score
Random Forest Classifier	0.91
Naive Bayes Classifier	0.95
Logistic regression Classifier	0.90
k-nearest neighbors Classifier ($k = 5$)	0.80

We only got 0.95 in *Naive Bayes classifier*, so we thought about standardizing the data using a `StandardScaler`, to see if we could improve the performance. The following table reports the results of the training on the standardized data which allowed us to get a F1-score of 0.965 using a k-NN with:

Model	F1-score
Random Forest Classifier	0.80
Naive Bayes Classifier	0.90
Logistic regression Classifier	0.85
k-nearest neighbors Classifier ($k = 5$)	**0.97**

The standardization performs badly on *random forest classifier, Naive Bayes classifier* (that without the standardization reached 0.95) and *logistic regression*. On the other hand *k-NN classifier* reaches the value of 0.97

5 Bonus task

This task required to load the three given pictures (see Appendix B) and compress them by clustering the pixels of each picture according to their color values and then to represent each pixel of a cluster in the mean color of the cluster. Finally we had to visualize the results for different reasonable numbers of clusters using at least one evaluation score.

We clustered the images using k-means with different number of clusters and then calculated the silhouette score for each one.
Each image has been compressed with three different numbers of clusters.
The following table presents the silhouette score for each different number of clusters and for each image.

Clusters / Image	CITEC	UBI-X	UHG
2	0.6		
3		0.44	
4	0.61		0.54
5		0.52	
6			0.55
7		0.52	
8	0.59		
9			0.53

We obtained a good result, but we can observe that the number of clusters does not affect significantly the performance.

The compressed images are available in Appendix C.

A Learning curves

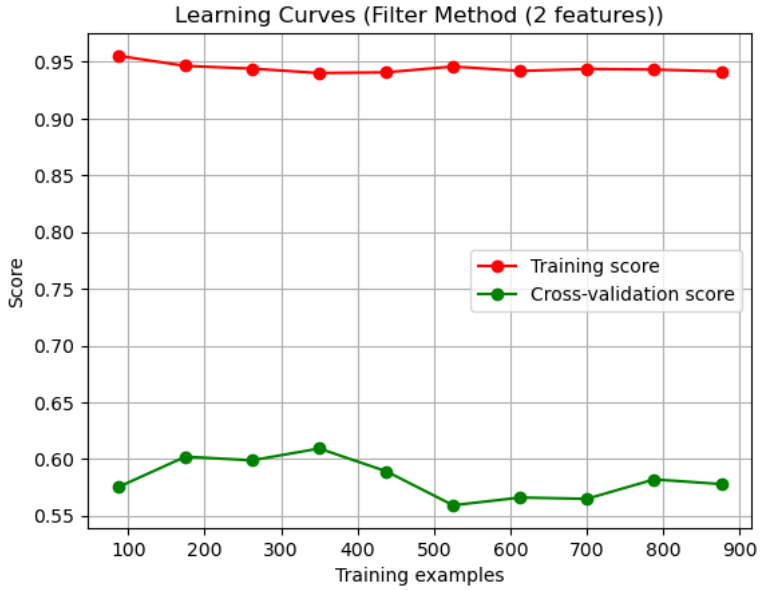


Figure A.1: Leaning curve for F measure and forward selection with two variables

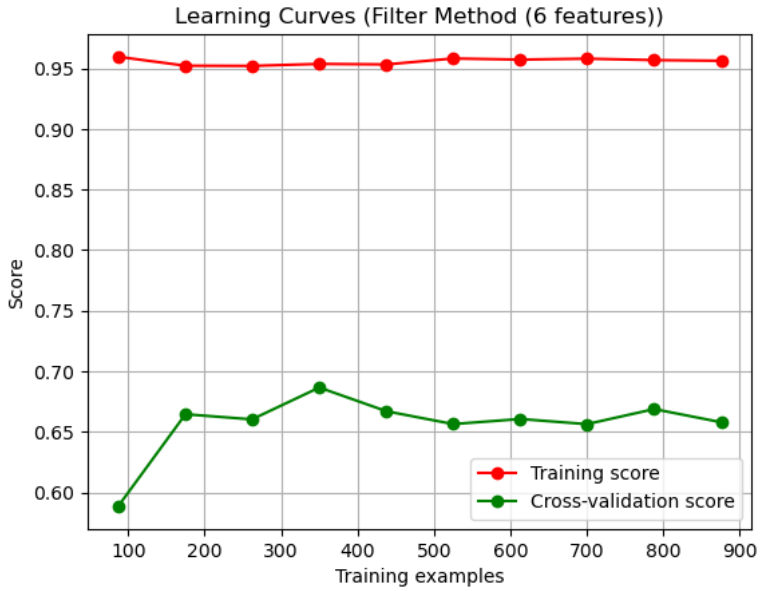


Figure A.2: Leaning curve for F measure with six variables

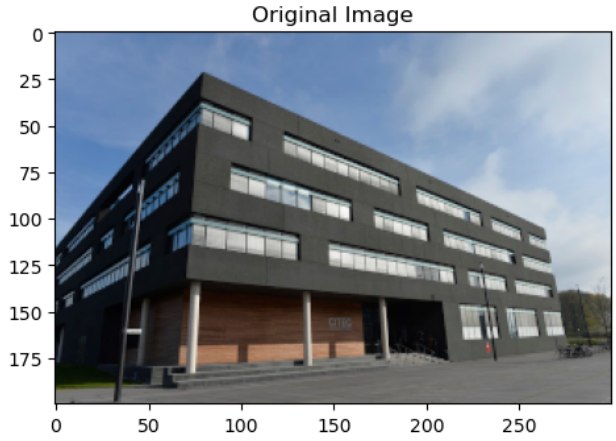
B Given pictures for bonus task

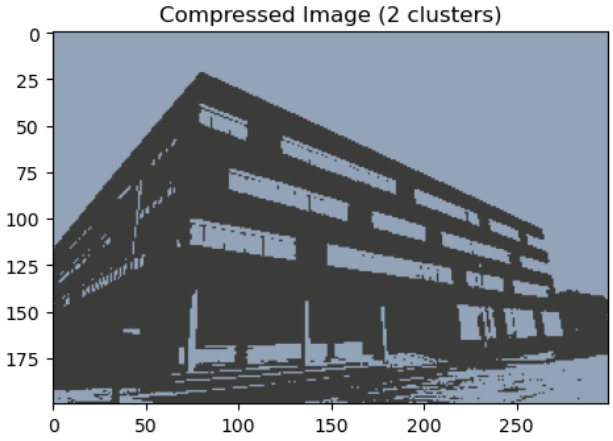
Figure B.1: CITEC image

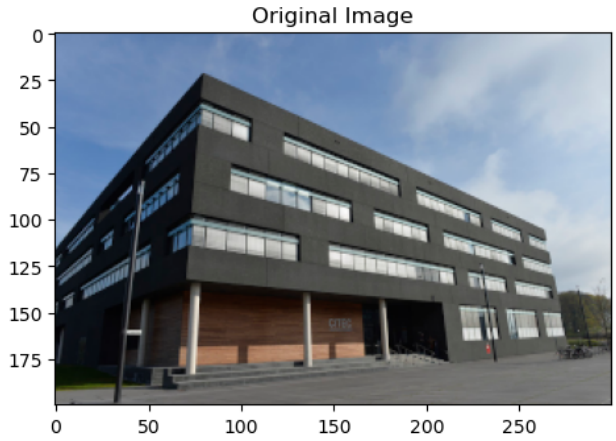
Figure B.2: UBI-X image

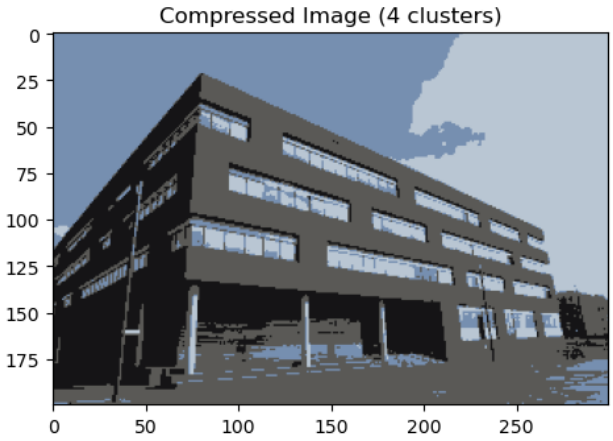
Figure B.3: UHG image

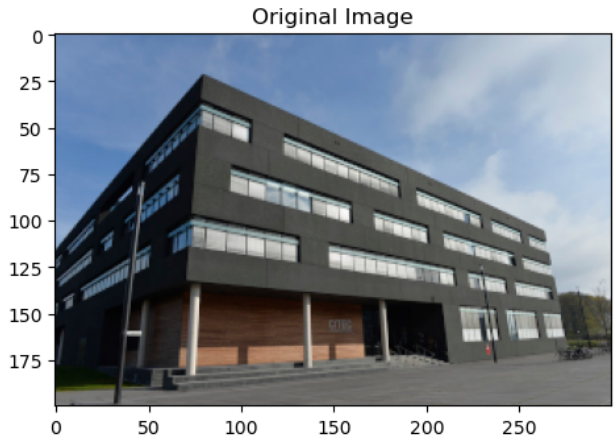
C Bonus task results

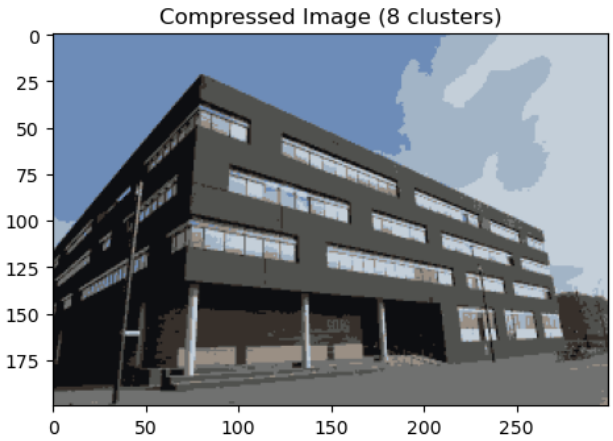




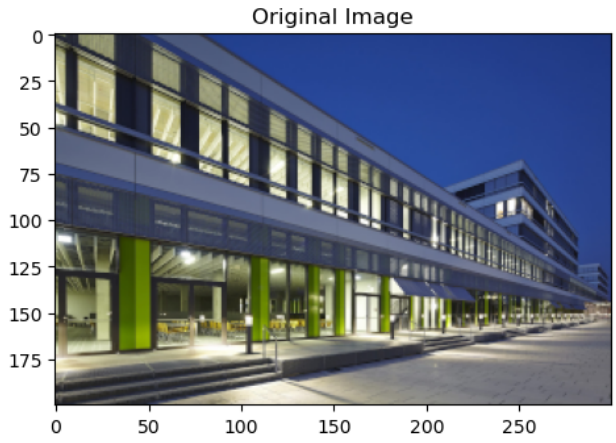


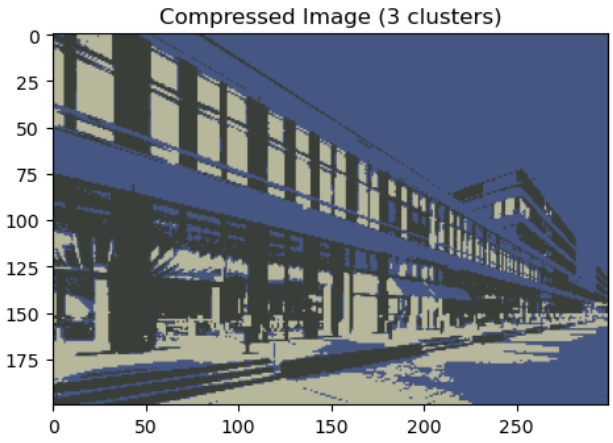


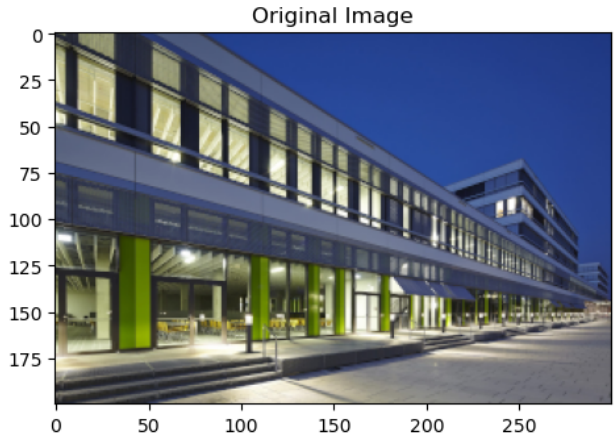


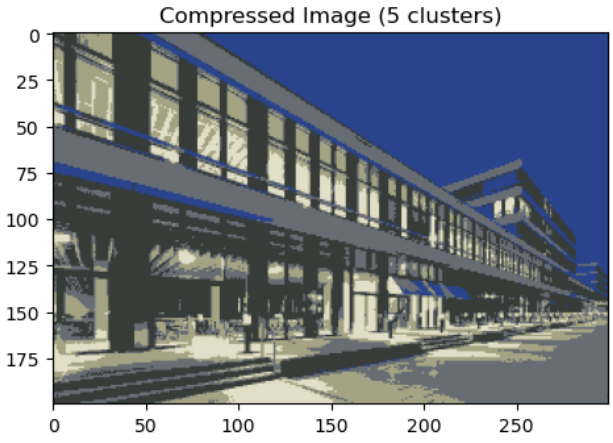


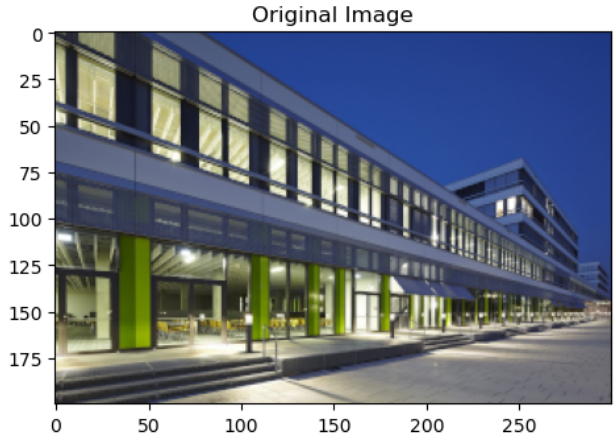
C Bonus task results

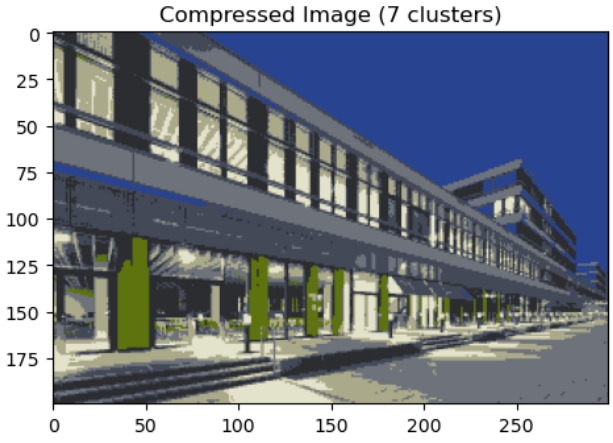












C Bonus task results