Technische Universität Wien

184.702 Machine Learning

Exercise 1: Classification

Mentors:
Univ.Mag.rer.soc.oec. Dipl.-Ing. Rudolf Mayer,
Priv.-Doz. Dr. Nysret Musliu,
Ao.univ.Prof. Dr. Andreas Rauber.

Students:
Betim Maloku 11728298,
Gent Rexha11832486,
Ilir Osmanaj 11770999

Vienna, November 2018

Table of contents

Characteristics of data sets, classifiers, performance measures and pre-processing 2
Data sets 2
Bank dataset 2
Image Segmentation dataset 3
Breast Cancer dataset 4
KDD-98 dataset 4
Classifiers 5
Decision Tree 5
Random Forest 5
k-Nearest Neighbors 6
Naive Bayes 6
Explanation of choice for data sets and classifiers 6
Performance measures 6
Accuracy 7
Precision 7
Recall 7
F1 score 8
ROC_AUC 8
Pre-processing 8
Handling of missing values 8
Scaling 8
Handling of categorical data 9
Feature selection 9
Principal Component Analysis (PCA) 9
Sampling 9
Experiments and results 9
KDD-98 dataset 9
Breast Cancer dataset 11
Bank dataset 12
Image segmentation dataset 12
Experiment with different parameter settings 13
Record (approximate) runtimes of the classifiers 15
Conclusion 17
Bibliography 18

Characteristics of data sets, classifiers, performance measures and pre-processing

Data sets

We have chosen four datasets with these different characteristics:

a. Bank dataset, the data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (or not) subscribed.
b. Image Segmentation dataset, the instances were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel. Each instance is a 3x3 region, wherein its aim is to classify each region in one of the classes: (i) brickface, (ii) sky, (iii) foliage, (iv) cement, (v) window, (vi) path, and (vii) grass.
c. Breast Cancer dataset (taken from Kaggle), the features are computed from a digitized image of a fine needle aspirate of a breast mass. They describe characteristics of the cell nuclei present in the image, based on which its aim is to classify if somebody has breast cancer (benign or malign).
d. KDD-98 dataset (taken from Kaggle), presents a subsample of the KDD Cup 1998 Dataset, which presents the results of one of Paralyzed Veterans of America's (PVA) recent fundraising appeals. This mailing was sent to a total of 3.5 million PVA donors who were on the PVA database as of June 1997. Everyone included in this mailing had made at least one prior donation to PVA. The aim of the kaggle assignment was to find who donated.

Bank dataset

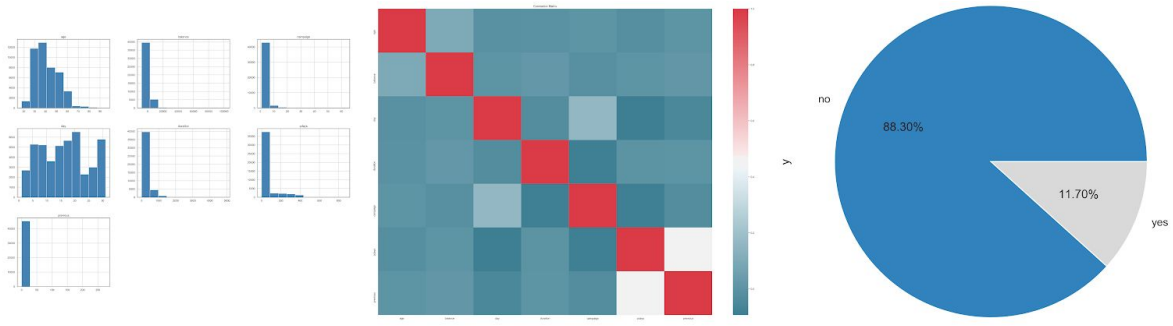

Figure 1: Visual Exploration of the Bank dataset: (a) Feature Histograms showing distribution of data, (b) Correlation matrix of all features, and (c) target feature percentage.

It contains these characteristics:

1. It has a large number of samples (49762)
2. It has a small number of features (16)
3. It contains few classes (2)
4. It doesn't contain missing values
5. The target feature has a big imbalance (88% no, 11% yes)
6. It needs a step of preprocessing. More accurately, it needs One-hot encoding. Scaling, feature selection and PCA are applied just for results comparison

Image Segmentation dataset

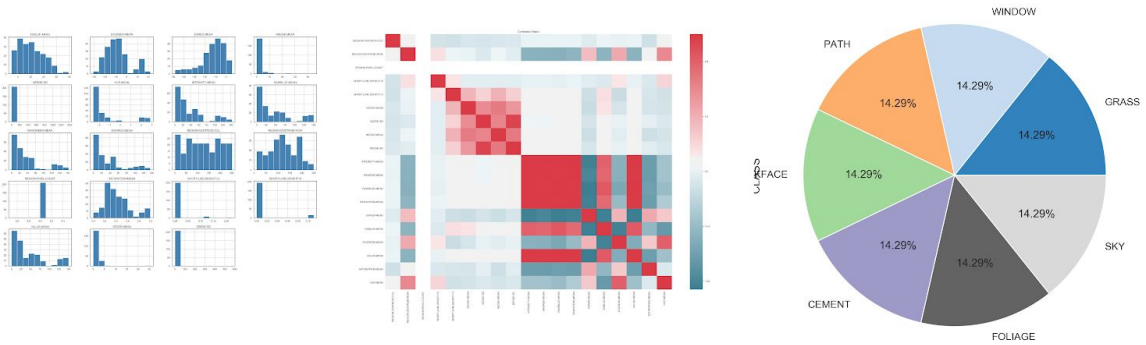

Figure 2: Visual Exploration of the Image Segmentation dataset: (a) Feature Histograms showing distribution of data, (b) Correlation matrix of all features, and (c) target feature percentage.

It contains these characteristics:

1. It has a medium number of samples (2310)
2. It has a small number of features (19)
3. It contains many classes (7)
4. It doesn't contain missing values
5. The target feature is balanced across all classes
6. It doesn't need preprocessing. Scaling and feature selection are applied just for results comparison

Breast Cancer dataset

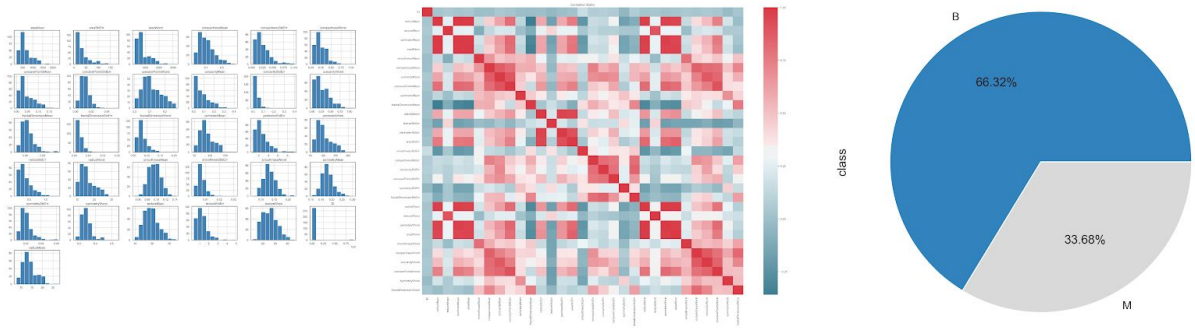

Figure 3: Visual Exploration of the Breast Cancer dataset: (a) Feature Histograms showing distribution of data, (b) Correlation matrix of all features, and (c) target feature percentage.

It contains these characteristics:

1. It has a small number of samples (569)
2. It has a medium number of features (31)
3. It contains few classes (2)
4. It doesn't contain missing values
5. The target feature has a small imbalance (66% B, 33% M)
6. It doesn't need preprocessing. Scaling and feature selection are applied just for results comparison

KDD-98 dataset

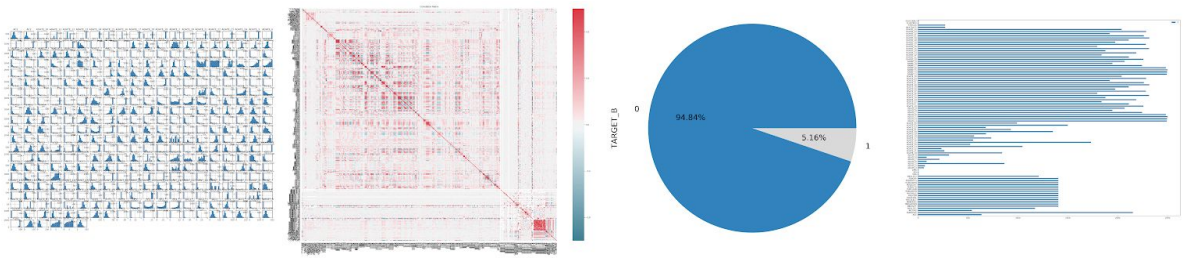

Figure 4: Visual Exploration of the KDD dataset: (a) Feature Histograms showing distribution of data, (b) Correlation matrix of all features, (c) target feature percentage, and (d) features with missing values.

It contains these characteristics:

1. It has a large number of features (480)
2. It has a medium number of samples (5000)
3. It contains many classes (2)
4. It contains missing values
5. The target feature has a big imbalance (95% 0, 5% 1)

6. It needs preprocessing. More accurately, it needs the transformation of some nominal variables into dummy variables, in order to work just with purely numerical data. And it needs to handle missing values. Scaling (normalizing), feature selection (CFS) and PCA are applied just for results comparison

Classifiers

We have chosen four classification algorithms (Classifiers):

a. Decision Tree (DT) - Non-parametric , tree model, with metrics: Gini Impurity, Information Gain and Variance Reduction [1]
b. Random Forest (RF) - Non-parametric, Ensemble of decision trees[2]
c. k-Nearest Neighbors (kNN) - Non-parametric and it is a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel[3]
d. Naive Bayes (NB)- GaussianNB - Parametric and probabilistic classifier[4]

Decision Tree

It contains these characteristics:

Advantages:

1. Simple to understand and interpret
2. Able to handle both numerical and categorical data
3. Requires little data preparation
4. Uses a white box model
5. Possible to validate a model using statistical tests
6. Performs well with large datasets
7. Mirrors human decision making more closely than other approaches
8. Robust against co-linearity
9. In built feature selection
10. Decision trees can approximate any Boolean function eq. XOR

Disadvantages:

11. Trees can be very non-robust
12. The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts
13. Decision-tree learners can create over-complex trees that do not generalize well from the training data
14. For data including categorical variables with different numbers of levels, information gain in decision trees is biased in favor of attributes with more levels

Random Forest

It contains these characteristics:

Advantages:

1. There is no need for feature normalization
2. Individual decision trees can be trained in parallel

3. Random forests are widely used
4. They reduce overfitting

Disadvantages

5. They're not easily interpretable
6. They're not a state-of-the-art algorithm

k-Nearest Neighbors

It contains these characteristics:

Advantages:

1. Robust to noisy training data
2. Effective if the training data is large

Disadvantages:

3. Need to determine value of parameter K (number of nearest neighbors)
4. Distance based learning is not clear which type of distance to use and which attribute to use to produce the best results.
5. Computation cost is quite high because we need to compute distance of each query instance to all training samples.

Naive Bayes

It contains these characteristics:

Advantages:

1. It's relatively simple to understand and build
2. It's easily trained, even with a small dataset
3. It's fast!
4. It's not sensitive to irrelevant features

Disadvantages:

5. It assumes every feature is independent, which isn't always the case
6. If you have no occurrences of a class label and a certain attribute value together then the frequency-based probability estimate will be zero.

Explanation of choice for data sets and classifiers

We have chosen RF and DT because they are very effective with large datasets, NB due to its speed in training and works well even with small datasets, and kNN because it is robust to noisy training data.

Performance measures

We've taken into account these measurements:

a. Accuracy

b. Precision
c. Recall
d. F1 score
e. Roc_Auc

Accuracy

1. The accuracy of a system is the closeness degree of quantity measurements to that quantity's true value.
2. It is the ratio of correct results (both TPs and TNs) among the total number of cases (TP+TN+FP+FN), i.e. Accuracy = (TP+TN) / (TP+TN+FP+FN).
3. When we have a large (majority) class compared to others, then accuracy suffers from Accuracy Paradox. I.e. If we take the case of breast cancer, and there are 95% of people benign and 5% of them malign. We can configure classifier to classify arbitrary all the people as benign, and we would have an high accuracy, but as we can see it's a terrible measure in these case (with zero predictive power).

Precision

1. Known as positive predictive value, it is the ratio of relevant samples over the total amount of the retrieved samples.
2. It is the ratio of true positive samples among predicted positive samples, i.e. Precision = TP / (TP+FP).
3. If we have a perfect classifier, then Precision=Recall=1.0 (which is not possible to achieve in real classification).
4. We can increase the precision in account of recall, i.e. if we increase the precision the recall will decrease (also making the classifier useless).
5. Precision is more important in the cases we really want to be precise and not assume anything to be positive unless we are very confident, for instance let's take the breast cancer case. In this case, we really want to be precise and not to misinform patients that they have malign breast cancer.

Recall

1. Known as sensitivity, it is the ratio of relevant samples that have been retrieved among of relevant instances.
2. It is the ratio of true positive samples among all positive samples, i.e. Recall = TP / (TP+FN).
3. Also, We can increase the recall in account of precision, i.e. if we increase the recall the precision will decrease (also the classifier would be made useless).
4. Recall is more important in the cases we don't want to miss positive samples. For example, when the doctor has to assess if we have benign or malign breast cancer, then recall is more important, because we don't want to miss any of cases with malign breast

cancer, or if we take the bank dataset, we don't want to happen not to call a potential client (with high probability that he would deposit his money in the bank).

F1 score

1. It is a evenly weighted harmonic mean of recall and precision, i.e. F1_score = 2 / (1/Recall + 1/Precision).
2. It's a better measure when FPs and FNs have different costs.
3. It is resilient to very unbalanced classes (One very huge class, the other very small). For instance if we take the example from Accuracy subsection (with breast cancer), it will give us high accuracy, but zero F-score.

ROC_AUC

1. ROC (Receiver operating characteristic curve) - plots recall against inverse recall, or equivalently true positive rate (TPR = TP / (TP+FN)) against false positive rate (FPR = FP / (FP+TN)).
2. ROC_AUC (Area Under the ROC Curve) - measures the two-dimensional area under the ROC curve (like integral for surface calculation).
3. It measures discrimination, i.e. the ability of the classifier to correctly classify those with and without the breast cancer, or said with other words it indicates how well the probabilities from the positive classes (with breast cancer) are separated from the negative classes (without breast cancer).

Pre-processing

Handling of missing values

To handle the missing value we have used two approaches:

1. For numerical feature, the missing value is filled by the mean of that feature, i.e. we have filled the missing value with the sum of all values of the corresponding feature divided by the number of values of that feature (Missing value = sum(n-feature) / n-feature).
2. For nominal feature, the missing value is filled by the mode of that feature, i.e. by the most recursive nominal value of that feature.

Scaling

We have performed scaling using:

1. Normalizing (.Normalize) - rescales the vector for each sample to have unit norm
2. Min-Max Scaling (.MinMaxScaler) - scales all numeric variables in the range [0,1]
3. Standardization (.scale) - transforms variables to have zero mean and unit variance.

Handling of categorical data

To handle nominal and ordinal data we have used the following approach for each case:

1. For nominal data, we have used One-hot encoding, i.e. the transformation of some nominal variables into dummy variables, in order to work just with purely numerical data.
2. For ordinal data, we have done an integer-transformation (for example, primary:1, secondary:2, tertiary:3)

Feature selection

In order to perform feature selection, we have used three methods:

1. SelectKBest (with a chi2 function) - Select top k-features according to the highest scores. Where Chi-squared stats of non-negative features for classification.
2. Decision Trees - the features with zero importance aren't exploit to split any nodes, so we can remove these zero-importance-feature.
3. Drop highly correlated feature (threshold 0.8)
4. The Correlation Feature Selection (CFS) - evaluates subsets of features based on the hypothesis: "Good feature subset contain features highly correlated with the output.

Principal Component Analysis (PCA)

PCA transforms a number of (possibly) correlated features into a smaller number of uncorrelated features called principal components.

Sampling

In order to handle imbalanced dataset, we have performed two kinds of sampling:

1. Under sampling - which reduced the number of samples of the major dataset to the number of minor dataset (Major: 500 samples; Minor: 100 samples ⇒ Major & Minor = 100 samples)
2. Over sampling - which increases the number of samples of the minor dataset to the number of major dataset (Major: 100 samples; Minor: 20 samples ⇒ Major & Minor = 100 samples)

Experiments and results

KDD-98 dataset

In Figure 5 we have plotted four charts, which indicate the performance measure of the four classifier by applying preprocessing in the KDD-98 dataset.

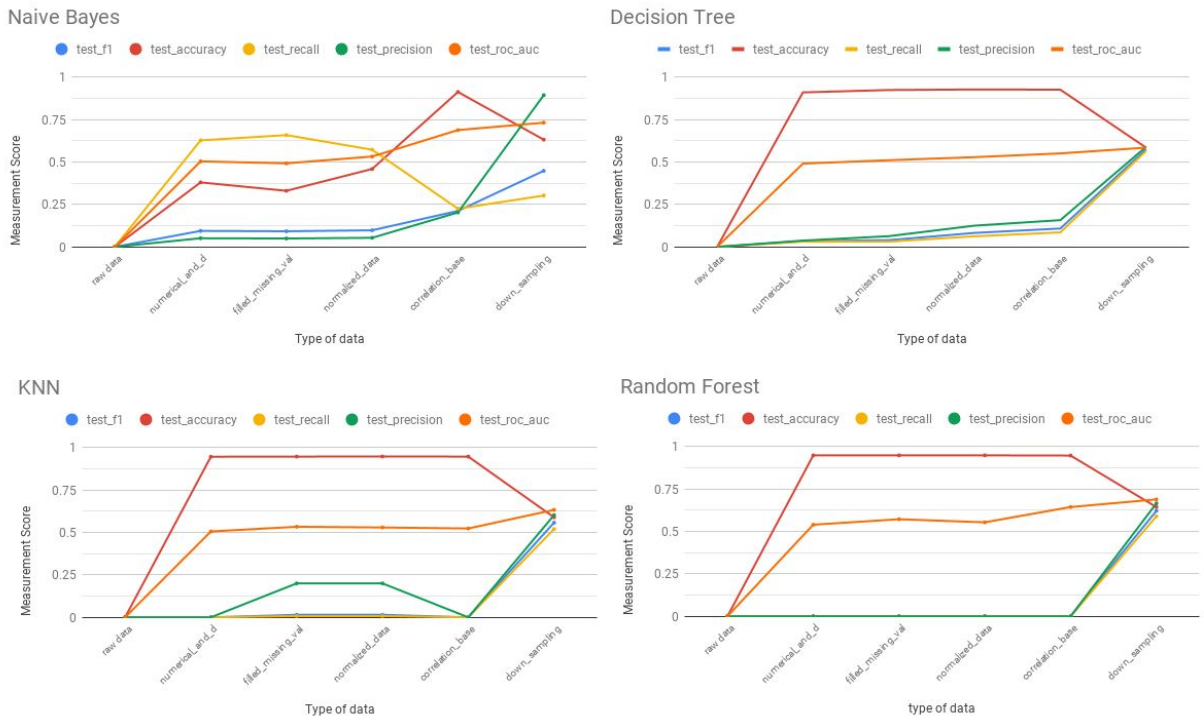

Figure 5: Performance measurements with default parameters by adding different steps of preprocessing in: a.Naive Bayes, b. Decision Tree, c. kNN and, d. Random Forest in the KDD-98 dataset

As seen in Figure 5, all measurements score are 0 for raw data, because the classifiers from the sklearn library don't work with datasets that contain missing values and nominal data. This data is quite noisy, high dimensional, with a lot of missing values and just with 5% of positive cases. Feature selection and preprocessing will be vital for a good model.

Furthermore, accuracy will always be very high for all classifiers, considering the big imbalance between the amount of target variable outputs, therefore comparing the classifiers on other measurement scores is more desirable.

For all steps of preprocessing, the performance measurements are almost the same (they don't indicate any improvement by performing extra steps, except for the down sampling method when the accuracy falls down, and F1 and Precision are rising).
Random Forest ROC_AUC score is outperforming other classifier's ROC_AUCs. Others performance measures (except ROC_AUC and Accuracy) are zero almost zero for all classifiers (except for NB) due to high imbalance dataset.

Breast Cancer dataset

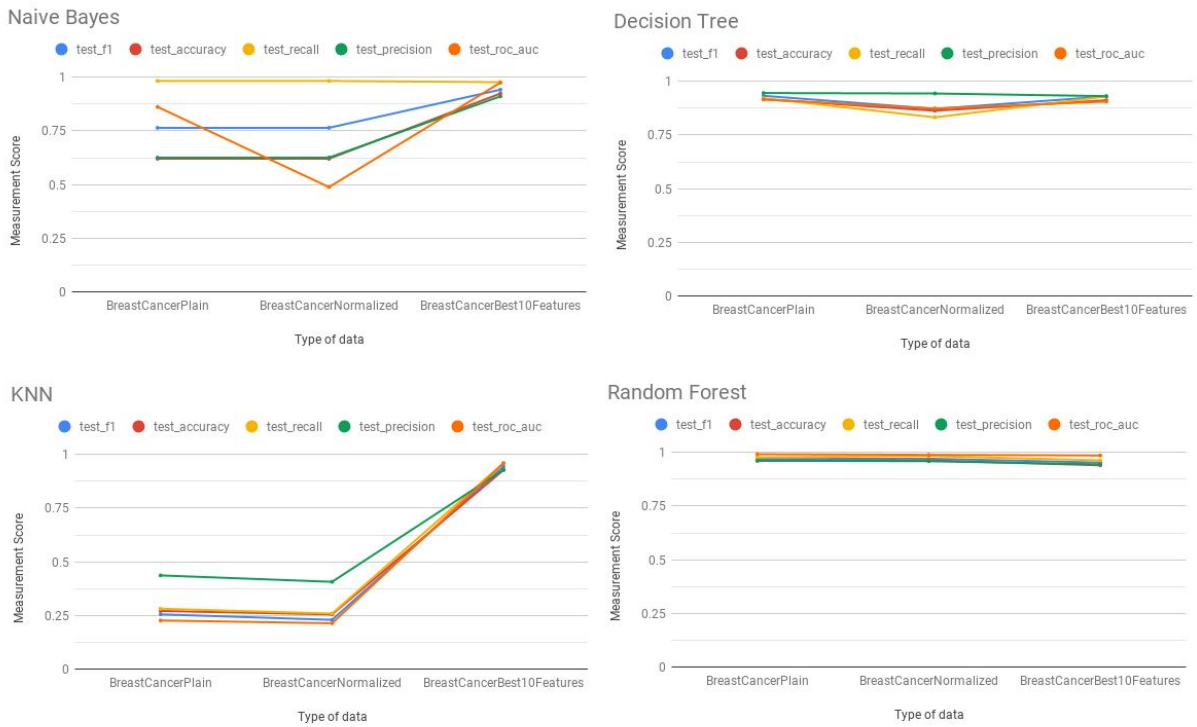

Figure 6: Performance measurements with default parameters by adding different steps of preprocessing in: a.Naive Bayes, b. Decision Tree, c. kNN and, d. Random Forest in the Breast Cancer dataset

As we can see from Figure 6, DT and RF outperform other classifiers, wherein their metrics stand unchangeable even after performing normalization and feature selection. NB's ROC_AUC falls down after normalization but raises after feature selection. The kNN metrics don't change with normalization but raise just after applying feature selection.

Bank dataset

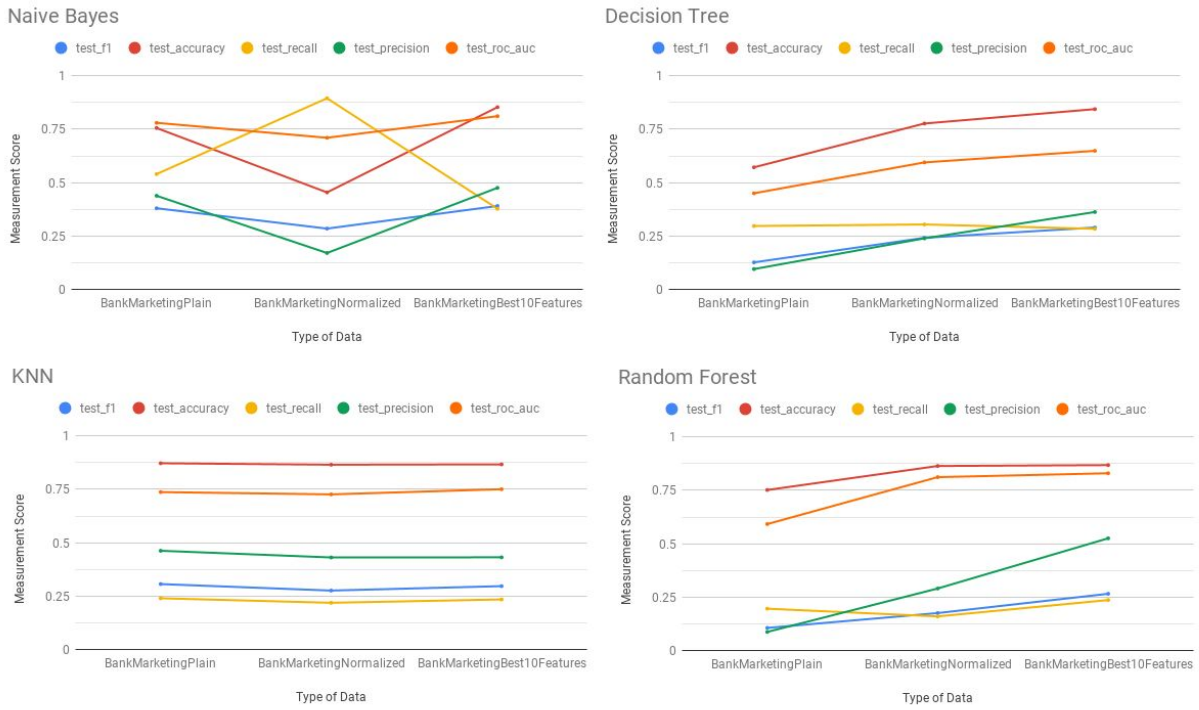

Figure 7: Performance measurements with default parameters by adding different steps of preprocessing in: a.Naive Bayes, b. Decision Tree, c. kNN and, d. Random Forest in the Bank dataset

As shown in the Figure 7, doing preprocessing (in this case normalization) or doing a parameter selection (in this case select 10 best with chi squared function) yields much better results as plain data.

Observing the results from different classifiers, it is clear that naive bayes is quite stable with the ROC_AUC metric, whereas decision tree increases the accuracy while using less features.
Strangely, kNN is quite indifferent to normalization and feature selection.
For the random forests model, each step - normalization and feature selection - increase the metrics for ~5% in total. Furthermore, random forests also yields the best accuracy results.

Image segmentation dataset

Image segmentation is quite balanced dataset and contains only numerical values, which resulted in better out of the box performance. The only difference with the other datasets it's the number of classes, but in general the same procedure was followed for this dataset, too. One interesting fact was all of the classifiers performed worse after having the data normalized.

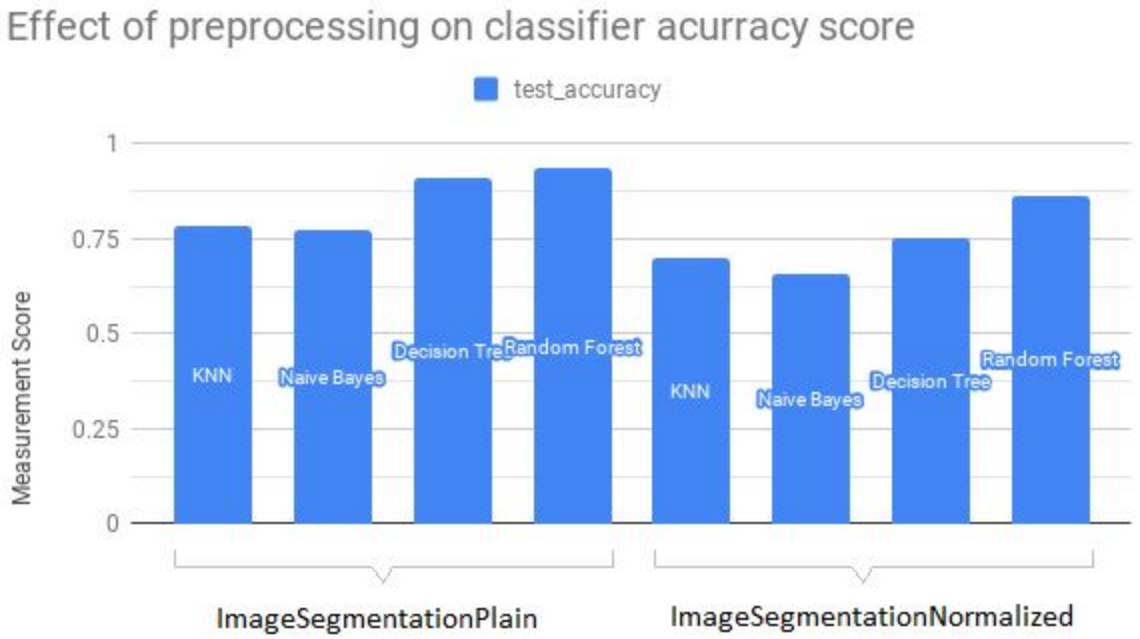

Figure 8: Performance measurements with default parameters by normalizing the data in: a.Naive Bayes, b. Decision Tree, c. kNN and, d. Random Forest in the Image Segmentation dataset

Experiment with different parameter settings

Param alpha	Param fit_prior	Accuracy	Recall	Precision	F1-score	ROC AUC
0	TRUE	0.6231155779	1	0.6231155779	0.7677993332	0.5221720849
0	FALSE	0.4045226131	0.04434416983	1	0.08344754073	0.5221720849
0.3	TRUE	0.6231155779	1	0.6231155779	0.7677993332	0.5221720849
0.3	FALSE	0.4045226131	0.04434416983	1	0.08344754073	0.5221720849
0.6	TRUE	0.6231155779	1	0.6231155779	0.7677993332	0.5221720849
0.6	FALSE	0.4045226131	0.04434416983	1	0.08344754073	0.5221720849
0.9	TRUE	0.6231155779	1	0.6231155779	0.7677993332	0.5221720849
0.9	FALSE	0.4045226131	0.04434416983	1	0.08344754073	0.5221720849
1	TRUE	0.6231155779	1	0.6231155779	0.7677993332	0.5308139336
1	FALSE	0.4045226131	0.04434416983	1	0.08344754073	0.5308139336

Table 1: Naive Bayes classifier applied in breast cancer dataset

In Table 1 is shown breast cancer dataset classified with Naive Bayes with different parameter settings. The best parameters (for alpha=0 and fit_prior= True) are found by applying grid-search. We can notice that when fit_prior is true the recall is 1 and precision is smaller, and when fit_prior is false the precision is 1 and recall is less. Parameter alpha hasn't an evidently impact in performance measures.

Param Criterion	param max_depth	param n_estimators	Accuracy	Recall	Precision	F1-score	ROC AUC
entropy	15	1185	0.6511627907	0.5968992248	0.6755368814	0.6325687703	0.6675680548
entropy	50	642	0.6472868217	0.5968992248	0.665381141	0.6288169482	0.6695511087
entropy	38	2000	0.6240310078	0.5813953488	0.6395663957	0.6080729728	0.6645033351
gini	4	2000	0.6317829457	0.5736434109	0.6606283007	0.607987383	0.6800072111
entropy	50	914	0.6240310078	0.5813953488	0.6379855465	0.607795861	0.6625202812
entropy	38	1728	0.6279069767	0.5736434109	0.6495670996	0.6075969513	0.663601947
entropy	38	100	0.5968992248	0.5658914729	0.6043247654	0.5841003981	0.654407788
gini	22	100	0.6046511628	0.5503875969	0.6198376787	0.5819635171	0.6460248783
gini	27	100	0.6085271318	0.5426356589	0.6270833333	0.5802409639	0.6499909861

Table 2: Random Forest classifier applied on KDD dataset

In Table 2 is presented KDD dataset classified by Random Forest with different parameter settings. The best parameters (for criterion=entropy, max_depth= 15 and estimators=1185) are found by applying grid-search. It is noticeable that all three parameters should be tunable in specific value in order to achieve the best performance, otherwise the performance measure will decrease.

param_n_neighbors	param_p	param_weights	Accuracy	Recall	Precision	F1-score	ROC_AUC
3	1	distance	0.8969849246	0.9485919795	0.9008940003	0.9225326858	0.9271981442
6	1	uniform	0.8894472362	0.9288570302	0.9035750006	0.9152106361	0.9311384339
7	1	uniform	0.8894472362	0.9446507045	0.891922432	0.9165393375	0.9314033487
15	1	distance	0.8743718593	0.9486205537	0.8705528732	0.9067201013	0.9365062296
5	2	distance	0.8693467337	0.9407380037	0.8708643963	0.9029843915	0.9263772887
2	2	uniform	0.8391959799	0.8182727362	0.9244035068	0.8644807122	0.8975942423

Table 3: kNN classifier in Breast cancer dataset

In Table 3 is presented breast cancer dataset classified by kNN with different parameter settings. The best parameters (for neighbors=3, p= 1 and weights=distance) are achieved by applying grid-search. It is clearly seen from the table that when parameters neighbor change, the performance of classification starts to fall down. Two other parameters hasn't a huge impact in the performance classification.

Record (approximate) runtimes of the classifiers

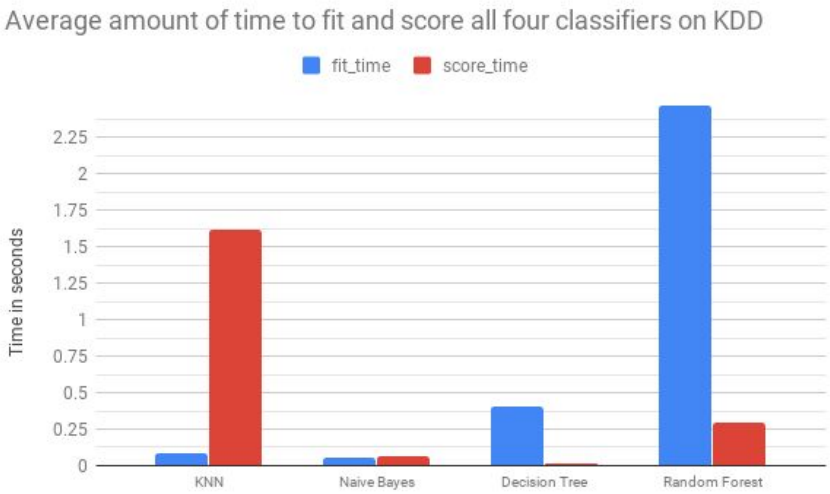

Figure 9: Runtimes (fit and score time) for four classifications on KDD dataset

From Figure 9 we can notice that the Naive Bayes is the fastest classifier. kNN has better runtime than DT and RF in fitting (training) time (as it is lazy learner) and has worse runtime in scoring time.

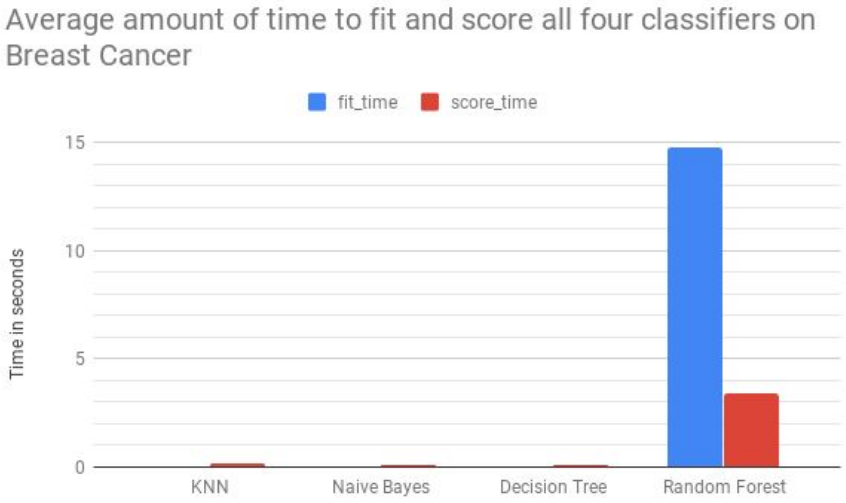

Figure 10: Runtimes (fit and score time) for four classifications on Breast Cancer dataset

In Figure 10 is shown that RF has the worst runtime compared to other classifiers.

Figure 11: Runtimes (fit and score time) for four classifications on Bank dataset

From Figure 11, we notice that NB has the fastest fit and score time. In the second place for fitting time is kNN, while for scoring time is DT, while RF has the worst time in all competition.

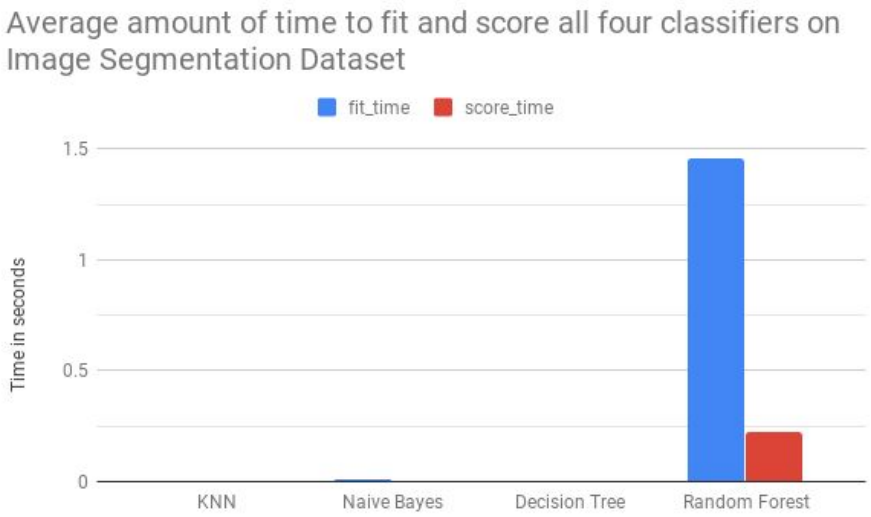

Figure 12: Runtimes (fit and score time) for four classifications on Image Segmentation dataset

Conclusion

In this report we've taken four classifier (DT, RF, NB and kNN) and classified them on four different datasets (Bank, Breast Cancer, KDD-98, Image Segmentation). To measure the performance measure we have used accuracy, precision, recall, F1_score and ROC_AUC.

Some of the datasets have needed to be preprocessed due to nominal data and missing values. In order to get best performance measure we have applied other preprocessing steps (such as scaling, oversampling, undersampling, feature selection, filling missing values, dropping missing values) in datasets and tried to find optimal hyper parameters through grid search.

We spent most of our time coding and exploring all the options the sklearn and all the other machine learning libraries in python offer. The options and possibilities on what to do are limitless. We often found ourselves trying out new stuff and finding out that it didn't help our measurement scores as much as we though.

Then, we've tried changing the parameter settings manually in order to compare the performance of the classifiers with different settings and to see which parameters have more impact in classification performance.

We've limited the report to just some of the charts, figures and tables we've created due to the constraint in the report's number of pages.

Bibliography

[1] “sklearn.tree.DecisionTreeClassifier — scikit-learn 0.20.1 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier. [Accessed: 27-Nov-2018].

[2] “3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.1 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 27-Nov-2018].

[3] “sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.20.1 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. [Accessed: 27-Nov-2018].

[4] “sklearn.naive_bayes.GaussianNB — scikit-learn 0.20.1 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB. [Accessed: 27-Nov-2018].