Exercise 1 - Classification

Machine Learning

Johannes Breitenecker(01346716), Emil Daub(12143524), Aileen Dick(11706782)

April 26, 2022

1 Introduction

The goal of this exercise is to try different classification algorithms on different data sets. Among the classification models, various experiments with different settings, parameters, etc. are performed.

This report aims to present the modelling process for each data set. In total, two have four different data sources. Two have been assigned via Kaggle, one is taken from the previous Exercise and the last one is a new one selected by us.

In the following sections, the pre-processing, modeling and evaluation of each data set will be explained. Followed by a summary paragraph at the end.

1.1 Data partitioning

Each data set is split into three partitions to find an appropriate model: 75 percent is used for training, 17.5 percent for evaluation/testing, and the remaining 7.5 percent are used for calculating a cross validation score to validate each model.The split is done stratified by "class". This means it tries to keep the balance of the classes when splitting the data.

1.2 Chosen classifiers

To make the outcomes of this exercise comparable, we will train a decision tree, random forest, and the k-nearest neighbor approach on each data set. Moreover, to guarantee reproducibility the random seed has been fixed for decision trees and random forests.

1.3 Validation

For each model 5-fold cross-validation is performed in order to check stability of a model. This is done only using the holdout data.

1.4 Chosen metrics

For evaluating and comparing results, it was decided to look at the confusion matrix, the accuracy, precision, and recall on each model. Throughout this report, we will judge and compare results using these mentioned metrics.

2 Voting data

This data set is the smaller one provided by Kaggle, Congressional Voting data. The official training data set contains of 218 rows and 18 columns. The ultimate goal is to predict the "class", which is either "republican" or "democrat".

The first column of the data set is simply the ID of a person, called "ID". The remaining 16 features describe the interests of the people: 'handicapped-infants', 'water-project-cost-sharing', 'adoption-of-the-budget-resolution', 'physician-fee-freeze', 'el-salvador-aid', 'religious-groups-in-schools', 'anti-satellite-test-ban', 'aid-to-nicaraguan-contras', 'mx-missile', 'immigration', 'synfuels-crporation-cutback', 'education-spending', 'superfund-right-to-sue', 'crime', 'duty-free-exports', 'export-administration-act-south-africa'.

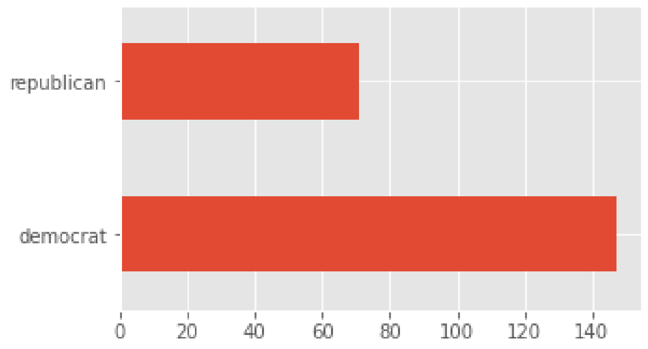


Figure 1: Distribution of classes

All of them only contain either ”y” for yes, or ”n” for no support in that topic. Among all predictors, there are also some missing values, included as category ”unknown”.

2.1 Preprocessing

2.1.1 Missing values

In total there are 177 missing values, but they are not obviously missing, but rather encoded as category ”unknown”. From these 16 features available, we looked for those rows where more than half of the variables are missing. It turns out that this only affects one row. In that instance, 15 out of 16 features are ”unknown”. Therefore, we have decided to drop that row completely since it makes no sense to impute all but one predictor.

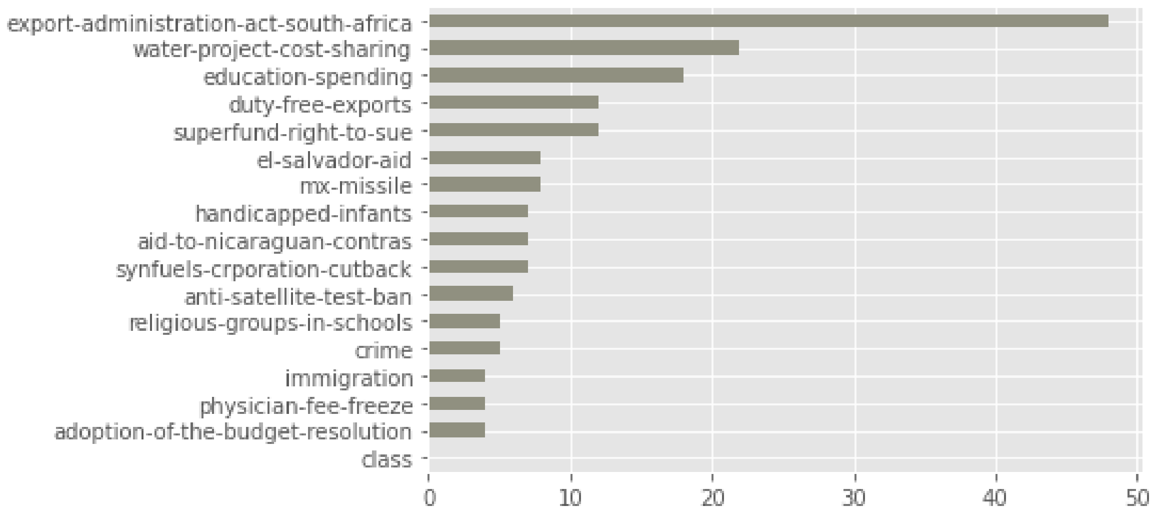


Figure 2: Missing values by column

Besides that, the unknowns are kept as separate variable for modeling, because it seems too risky to estimate a binary variable since it can result in the exact opposite. In addition, it could be that the unknowns also form a pattern. Another reason for keeping the unknown is, that if all the rows , where there is any unknown would be dropped, it would remove 97 rows which is almost half of the data set in total.

2.1.2 Encoding

Since all columns contain strings, a label-encoder is used to transform the strings into numeric data. In the response variable ”class”, 1 stands for ”republican”, and 0 for ”democrat. In all other variables, the values assigned are 0 for ”n”, 1 for ”unknown”, and 2 for ”y”.

2.1.3 Partitioning

As a last step in pre-processing the data is divided into a training,test and a holdout set. So the training part contains 53 republicans and 109 democrats, and the test set 12 and 26.

2.2 Models

2.2.1 Decision tree

The first classifier trained on this data is the Decision tree. With the default settings - gini criterion, best splitter strategy at each node, one minimum sample leaf and two minimum samples to split a node - already great results could be observed. There was only one case misclassified, hitting an accuracy of 97.34 percent and a precision of 100.

		predicted	
		0	1
actual	0	26	0
	1	1	11

Table 1: Confusion matrix of the tree.

By changing the splitting criterion to ”entropy” and set the minimum samples required at a split to 5, a perfectly classified tree is the outcome.

However, the cross-validated scores show, that all decision tree experiments perform the same. CV results of all intermediate experiments are (1,0.75,1,1,1).

2.2.2 Random forest

A more complex version of the decision tree is the random forest. With the default settings - gini criterion and a hundred trees - similar results like with single trees can be observed. There is one misclassified case leading to an accuracy of 97.37 percent and a recall of 100 percent. The cross-validated results are a little worse than with the decision tree.

After experimenting with settings of criterion, number of trees, and minimum samples, the outcome never changed. There was always this 1 misclassified case. Only when changing the seed, a perfect classifier could be achieved. The random effect plays obviously a more important part. Regarding performance and speed, the random forest using 50 trees, the gini criterion and minimum 5 samples to split led to the best model here.If setting the random seed to 1, there will even be no incorrectly classified data points.

2.2.3 K-nearest neighbor algorithm

As the third model we chose to use KNN. Since there are only categorical inputs, the hamming distance should be used to compute the nearest neighbors. Using the 5 nearest neighbors, there is again one case misclassified among the test data. Testing different values it looks like 5 is the optimal value to consider. From there onwards, there occur a little more misclassified data points.

		predicted	
		0	1
actual	0	25	1
	1	0	12

Table 2: Confusion matrix of 5NN.

2.2.4 Conclusion

All in all, the models perform well on this data. They even perform equally good. The latest score in the kaggle competition is computed from **the decision tree**. It was chosen,because it is the fastest one with a run time of 0.006 seconds compared to 0.008 in KNN, and 17 from the best random forest. In addition, the cross validation results performed best here.

	accuracy	precision	recall
DT	1	1	1
RF	1	1	1
KNN	0.9736	0.9231	1

Table 3: Final metrics

	part1	part2	part3	part4	part5
DT	1	0.75	1	1	1
RF	1	0.75	0.6667	1	1
KNN	1	0.75	0.6667	1	1

Table 4: Cross-validation scores

Another reason for choosing the decision tree, is because of the run time. KNN and the decision tree have a comparable run time of around 0.007 seconds. Whereas the random forest took 0.31 with one minimum sample leaf for splitting, and 0.15 with 5 minimum sample leaves. So the performance can be optimized but will never reach the one of the decision tree, because it computes several trees itself.

3 Census Income data

This data set is the one already chosen for Exercise0. It is available at Census income. It contains of 32561 instances and 15 column, with a mixture of categorical and numeric data. The goal is to predict if someone earns more or less than 50K a month.

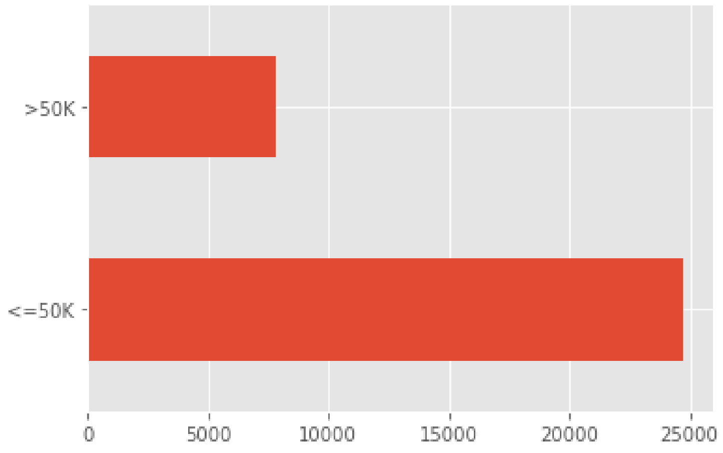


Figure 3: Distribution of income classes

The predictors are: 'age', 'workclass', 'fnlwgt', 'education', 'education_num','marital_status', 'occupation', 'relationship', 'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country'. "fnlwgt" describes some final weight originally defined by the data collectors. Moreover, "workclass","marital_status","occupation","relationship","race","sex","native_country" are categorical variables with various different levels.

name	values
workclass	Private, Self-emp-not-inc,Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked, missing values as '?'
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct,Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv,Protective-serv, Armed-Forces, missing values as '?'
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
sex	Female, Male
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece,South,China, Cuba, Iran, Honduras, Philippines, Italy,Poland, Jamaica,Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic,Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala,Nicaragua, Scotland, Thailand,,Yugoslavia, El-Salvador, Trinada and Tobago, Peru, Hong,Holand-Netherlands, missing values as '?'

The variable "age" ranges from 17 to 90 year-old's. Furthermore, "fnlwgt" is distributed between 122850 and 1484705. There are also the working hours of a person per week, ranging from 1 to 99. The columns "capital_gain" and "capital_loss" have a rather interesting distribution, because in most of the cases it is around, but when it contains values, they are pretty high in comparison. The further usage of them, will be discussed in the next section.

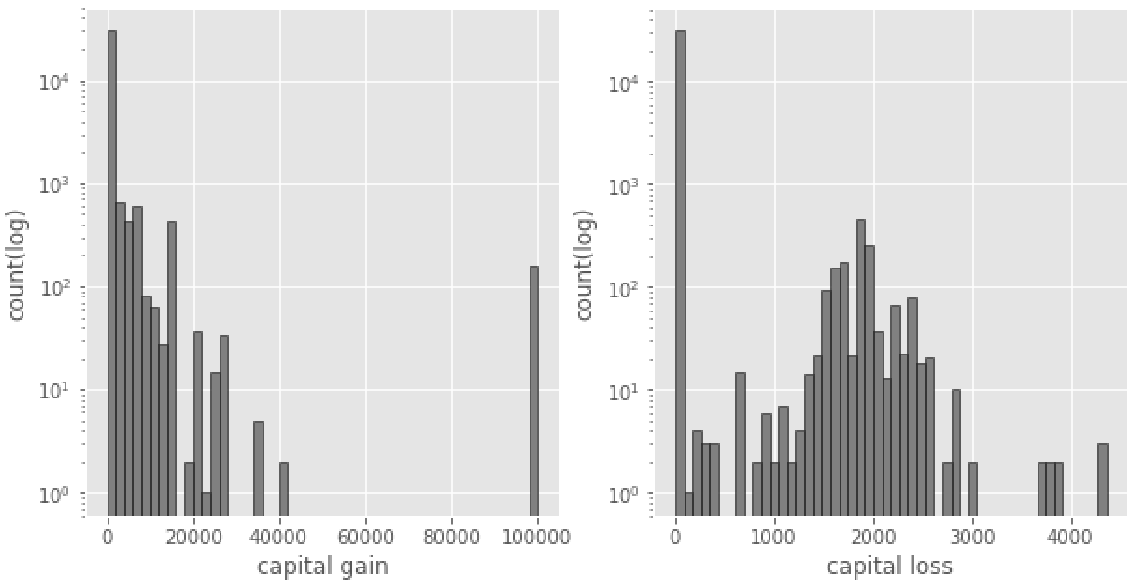


Figure 4: (Logarithmic) Distribution of capital_gain(left) and capital_loss(right)

3.1 Preprocessing

3.1.1 Missing values

The missing values are not obviously NaN here. In this data they are marked as question marks. This question mark appears in total 1836 times in column workclass, 1843 times in occupation, and 583 times in native_country. So the first step would be to rename these question marks to "unknown"-simply for more convenient usage.

3.1.2 Encoding

As a second step, the categorical data should be encoded into numbers. Among experiments, both, label-encoding and dummy-encoding has been considered. That means once training all models on dummy data, and once on labeled data. The final method chosen will be communicated after modeling.

3.1.3 Outliers

Furthermore, there are clearly outliers in capital gains which are located quite far from the rest, as it was visible in the plot above. There are 159 values that are exactly 99999. The assumption is, that these are probably missing values, or it is just an odd coincidence that they all have the same value. Either way they need to be handled.

3.1.4 Partitioning

Now the data gets partitioned in three parts as described at the beginning.

3.1.5 Imputation

After dividing the data, the "99999" in capital gains will be imputed using the 3 nearest neighbors from the data set. For imputing training data, only training data is used; for imputing test data only test data; and so on.

K nearest neighbor imputation has been chosen, because capital gain values are in most of the cases very small. Choosing the median would make no sense, since it is 0. The KNN imputation will give more reliable results than mean and median imputation here.

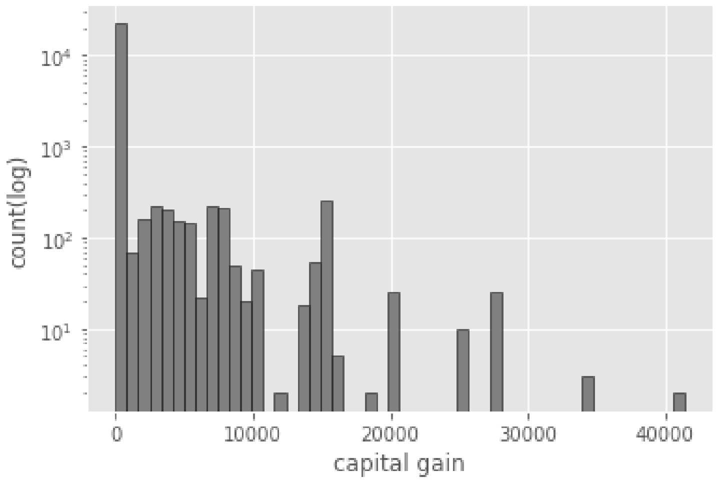


Figure 5: (Logarithmic) Distribution of capital_gain in the training data after imputation

3.2 Modeling

3.2.1 Decision tree

As a first model, a decision tree with default settings was trained using the dummy encoded data, which contain 99 predictors. Using thiss method, around 1070 cases were miss-predicted. This is probably caused by the imbalanced data set. It results in a accuracy of 0.81, a precision of 0.60, and a recall of 0.62. The cross-validated scores using the holdout data is stable around 0.80.

By changing parameters, the scores did not improve to much. Experimenting with criterion, minimum samples for splitting, and minimum number for a leaf, the accuracy could be pushed to 82.70 using the same seed. In addition, experimenting with the seed didn't lead to improvements in results.The best tree was obtained by having entropy as splitting criterion, using 5 minimum samples for a leaf.

		predicted	
		0	1
actual	0	3898	428
	1	543	829

Table 5: Confusion matrix of the best decision tree.

3.2.2 Random forest

As a next step, a random forest using gini as criterion and 100 trees was computed. This resulted in an accuracy of 85.03, with cross-validated scores as well pushed up to 85. Recall is the same as with the tree, but the precision increased to 73.

This forest could be a little improved by using now entropy, 50 trees, and a 5 samples required for a leaf.The accuracy is 85.66; the precision increased to 78; but the recall dropped to 56.34.

		predicted	
		0	1
actual	0	4108	218
	1	599	773

Table 6: Confusion matrix of random forest.

However, there were no major changes when experimenting with the parameters or seeds here.

3.2.3 KNN

As previously, the KNN algorithm is the next one to model. Since we have numeric data here, the data needs to be scaled. The "standard"- scaler is chosen, which scales the data to having mean 0 and variance 1. Using the 5 nearest neighbors and the euclidean distance, similar results to the random forest can be observed with an accuracy of 84.19. The drawback is that it takes four times longer to calculate than the random forest.

3.2.4 Naive Bayes

Since the models still leave some space for improvements, a simple naive Bayes has been trained. It led to an accuracy similar to decision tree with 0.79, but overall the recall got much worse. It dropped to 0.31. Interesting about this approach is when looking at the overall accuracy and the cross-validated scores on the holdout data it performs similar to decision tree, and thus is not that far from random forest results as well.

3.2.5 Conclusion

All in all, the random forest performed best here. Considering run time, the NB is the fastest with 0.04. Followed closely by KNN. Then decision trees with around 1 to 2 seconds. The best random forest took even 5 times longer than the decision tree. So in case of speed, the decision tree or naive Bayes should be considered.

In addition, the whole modeling approach was tested using label encoder as well for the categorical columns. Apparently, with this data it doesn't matter which encoding to use, because all results of the models are almost the same as for dummy encoded variables. However, in terms of run time it improves, because when everything is dummy encoded we have a much higher number of dimensions.Now the run time is comparable to the random forest.

	accuracy	precision	recall
DT	0.8127	0.6097	0.6173
RF	0.86170	0.7818	0.5904
KNN	0.8264	0.6613	0.5721
NB	0.79484	0.6579	0.3083

Table 7: Final metrics

	part1	part2	part3	part4	part5
DT	0.7853	0.7832	0.8078	0.7971	0.7992
RF	0.8589	0.8609	0.8609	0.8361	0.832
KNN	0.816	0.818	0.7832	0.8013	0.79713
NB	0.7894	0.7853	0.8037	0.8074	0.7971

Table 8: Cross-validation scores

Results in this tables are reported from the models using label encoding, since it performs almost the same but is less expensive in memory and run time.

4 Eucalyptus data

The Eucalyptus data set was newly searched and has not been introduced in the first assignment. This data set was found on openML Eucalyptus data. The objective was to determine which seedlots in a species are best for soil conservation in seasonally dry hill country.Determination is found by measurement of height, diameter by height, survival, and other contributing factors.

4.1 Data

The data set consists of 736 instances, has 20 features and is divided into 5 classes. As mentioned above, with the classes the tree is generally assessed the classes are: none,low,average,good,best. The data set has 448 missing values in total 95 instances are affected.

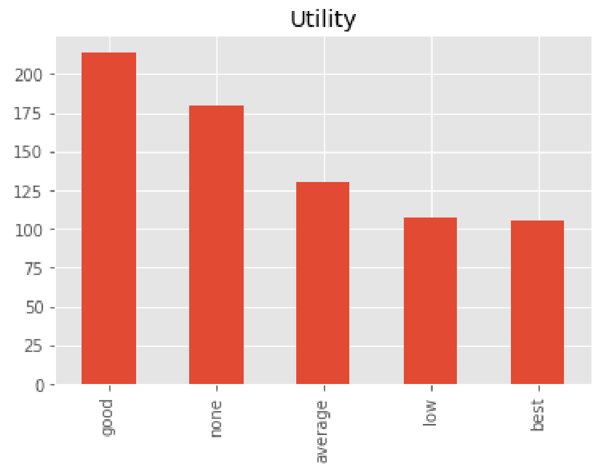


Figure 6: eucalyptus class distribution

The class distribution is very uneven. there are half as many low / best entries as good. this could become a disadvantage.

features	description	type
Abbrev	site abbreviation	enumerated
Rep	site rep (is saved as a number)	integer
Locality	location where the tree stands	enumerated
Map_Ref	also describes the site	enumerated
Latitude	Latitude oft the Tree location	enumerated
Altitude	Altitude oft the Tree location	enumerated
Rainfall	rainfall at the site in pa mm	integer
Frosts	Lowest measured temperature in Celsius	integer
Year	Year of planting the tree	integer
Sp	species code of the Tree	enumerated
PMCno	seedlot number	integer
DBH	best diameter base height (cm)	real
Ht	height in meter of the Tree	real
Surv	survival	integer
Vig	vigour	integer
Ins_res	insect resistance	real
Stem_Fm	stem form	real
Crown_Fm	crown form	real
Brnch_Fm	branch form	real
Utility	general evaluation oft the Tree	Class: enumerated

Table 9: Eucalyptus data.

4.2 Preprocessing

OpenML provides an .arff file containing the data and the data description. For the sake of simplicity, the data from the .arff file was transferred to a .csv file. This allows an easier way of working in the following steps.

The following properties have been deleted as they do not directly describe the tree. ”Abbrev”, ”Locality”, ”Map_Ref”, ”Latitude”, ”Altitude”, ”Rep”. Abbrev, Map_Ref , Latitude, Altitude and Locality contain the information about the location of the trees. The weather conditions are reflected in other parameters and therefore the processing of the location is invalid. The Rep feature has also been deleted as the Rep has no influence on the rating of the tree. The Rap only has an influence before the data is recorded.

In addition, rows with more than 5 missing values have been deleted, otherwise the classification is almost impossible. All other missing values were replaced with -1.

All enumerated type features were converted to integer numbers using a LabelEncoder. These included ”Latitude”, ”Sp” and the ”Utility” classifier.

The final shape of the data set (667, 17) is achieved in this way.

4.3 Models

4.3.1 Decision tree

First, the decision tree was trained with the data set. With the default settings, the decision tree achieved an accuracy of 62.06%. Through the visualization it is possible to determine the size of the tree, in total the tree has 16 layers.

If the maximum depth of the tree is set to 9, the accuracy improves to 66.37%.
Better values could not be achieved with the tree.
Using a smaller tree, also less time is needed.

Settings	default	[max_depth=9]
execution time [in ms]	49	47

Table 10: Execution time of Decision tree.

4.3.2 Random forest

Next, the Random Forest is tested, with the default settings. It has an accuracy of 70.68% and is already better than the simple tree.

With the setting criterion=”entropy” the results are improved in general as well.
By Using the settings [n_estimators=80, criterion=”entropy”, max_depth=8] a accuracy of 75.86% can be achieved.

Settings	default	criterion=”entropy”	Best
execution time [in ms]	918	998	782

Table 11: execution time of Random forest.

As expected, the run time of the best settings is below the default, because with [n_estimators=80, max_depth=8] the values are below default ([n_estimators=100] is the default value).

4.3.3 KNN

In order to use KNN, the data must be scaled. The ”standard”-scaler is also used for this purpose.

With the KNN default algorithm an accuracy of 48.27% is achieved and is therefore the worst default algorithm.

Using the settings [n_neighbors=3, weights=”distance”, p=1], the accuracy improve by almost 10% to 58.68% and thus significantly better.
By further testing it was found that [n_neighbors=10, weights=”distance”, p=1] can push the result even to 61.20%.
Even deleting all missing values did not show any significant improvement.

Settings	default	[n_neighbors=3,+2]	[n_neighbors=10,+2]
execution time [in ms]	30	26	33

Table 12: execution time of KNN.

KNN generally performs training very quickly, but as also expected here, more time is needed when the number of neighbors is increased.

4.4 Conclusion

Over all, the results delivered by the models are not satisfactory. Since the class value is composed of the underlying data, it is regrettable that none of the models used did not perform nearly as well, but as expected, the Tree models (Tree and Random Forest) performed best.

As in this case, the setting of the model parameters is very important, because it can gain a lot of percentages.
The execution time also changes due to the changes in model parameters. In this case, it did not, but it has a greater influence on larger data sets.

	accuracy	precision	recall
DT	0.6637	0.6706	0.6642
RF	0.7586	0.7804	0.7292
KNN	0.6120	0.6205	0.5672

Table 13: Final metrics

	part1	part2	part3	part4	part5
DT	0.3636	0.4	0.3	0.4	0.3
RF	0.4545	0.6	0.5	0.5	0.6
KNN	0.5454	0.5	0.4	0.5	0.3

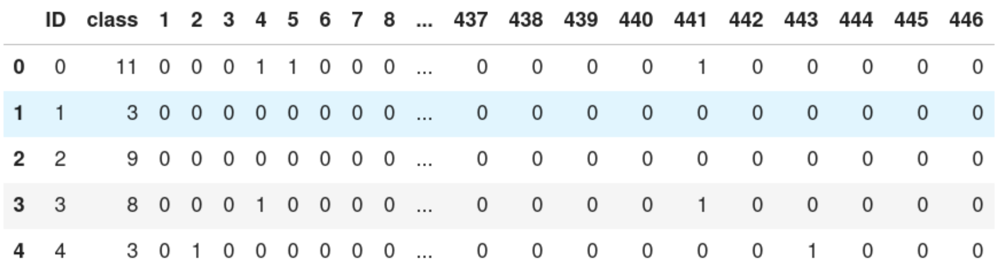
Table 14: Cross-validation scores

5 Location data

The Location data set were already chosen as part of the kaggle exercise. It can be downlaoded from Kaggle- Location This data set contains 446 binary features which represents a person visiting a certain location. There are 30 different geosocial types. The classification task is to predict a geosocial type for users based on their location data.

5.1 Data

In the data set are 4000 instances and 446 features with 30 classes. Per instance it is possible that zero or multiple features are selected, which is indicated with an 1.



	ID	class	1	2	3	4	5	6	7	8	...	437	438	439	440	441	442	443	444	445	446
0	0	11	0	0	0	1	1	0	0	0	...	0	0	0	0	1	0	0	0	0	0
1	1	3	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	2	9	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	3	8	0	0	0	1	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
4	4	3	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0

Figure 7: Location data overview

Following bar plot shows the count of each class that is to predict.

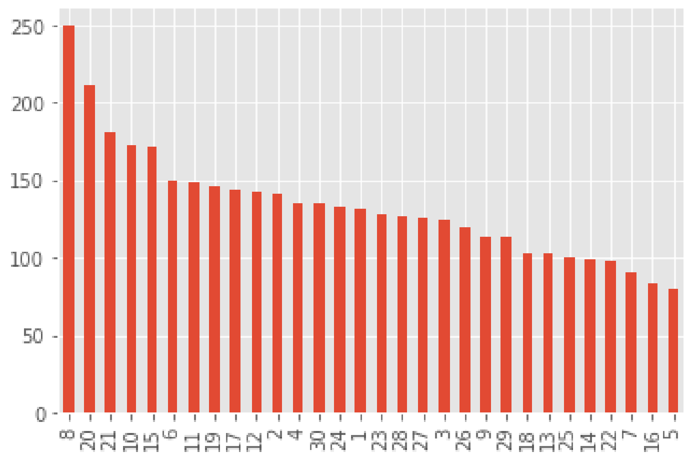


Figure 8: Distribution of location classes

Some classes are more present than others, but all in all it is not highly imbalanced.

5.2 Preprocessing

Since there were no missing values, or other limitation there was not much pre-processing necessary. The data just needs to be split into training, test and holdout data set as usual.

5.3 Models

5.3.1 KNN

The K-nearest neighbor algorithm got selected as our first model. Because there is only binary data, the hamming distance is used as metric parameter. When considering the 5 nearest neighbours, we get an accuracy of 0.375. Experimenting with amount of nearest neighbors, selecting the 30 nearest neighbours got the best accuracy of 0.458, a precision of 0.52, and a recall of 0.43. The execution time is relatively high, since there are many columns to consider.

Settings	[n_neighbors=5]	[n_neighbors=10]	[n_neighbors=30]
execution time [in ms]	320	374	379

Table 15: execution time of KNN.

5.3.2 Decision Tree

For the next model the decision tree was chosen. This single did not work well on our data. The default settings achieved an accuracy of 0.2686 and a precision of 0.2642.

5.3.3 Random forest

Again, several random forest with different settings have been tested. A random forest has been train with the gini criterion, 50 trees, a minimum samples split of 5 and a min samples leaf of 3. For that we got an accuracy of 0.567 and a precision of 0.5816. Since all binary features are highly imbalanced, we have tried to balance the class weight using the option class_weight="balanced". This led to an accuracy of 0.544, a precision of 0.529. We also tried the entropy criterion and a minimum samples split of 10, which got a accuracy of 0.541 and 0.576.

Settings	[criterion=gini, min_samples_leaf=3]	[class_weight=balanced]	[criterion=entropy]
execution time [in ms]	219	209	316

Table 16: execution time of Random Forest.

5.4 Conclusion

In summary, the random forest model gave the best results. It was also used for the kaggle competition. The decision tree model is not suitable for this data set as it achieved very low values in accuracy and precision.

	accuracy	precision	recall
DT	0.2686	0.2642	0.2538
RF	0.5443	0.529	0.5356
KNN	0.4586	0.5245	0.4320

Table 17: Final metrics

	part1	part2	part3	part4	part5
DT	0.2833	0.2167	0.2667	0.2833	0.3333
RF	0.4167	0.4	0.3667	0.3167	0.4167
KNN	0.2833	0.2167	0.2667	0.2833	0.3333

Table 18: Cross-validation scores

Considering the high amount of classes and the sparse binary input data, it has been difficult to find a suitable model.

6 Summary of results

In this exercise, three different classification algorithms have been performed on four different data sets. In particular, decision trees, random forests, and k-nearest neighbor method were used in an attempt to classify given response variables as good as possible.

For all methods, there were different experiments conducted. Both tree methods were trained using different splitting criterion, maximum depths, minimum samples needed for splitting, et cetera. The KNN method required adjustments in the metrics depending on the data set. Furthermore, several values for possible k have been tested.

The final best results for each data set using each model is documented in the table below.

Model	Setting	Data	accuracy	precision	recall
DT	criterion="entropy",min_samples_split=5	Voting	1	1	1
DT	criterion="entropy",min_samples_split=5	Income	0.8127	0.6097	0.6173
DT	max_depth=7	Eucalyptus	0.6551	0.6601	0.6628
DT	criterion="gini"	Location	0.2686	0.2642	0.2538
RF	n_estimators=50,criterion="gini" ,min_samples_split=5	Voting	1	1	1
RF	n_estimators=50,criterion="entropy" ,min_samples_split=5	Income	0.8617	0.7818	0.5904
RF	n_estimators=80, criterion="entropy", max_depth=8	Eucalyptus	0.75	0.7775	0.7278
RF	n_estimators=100,criterion="gini" ,min_samples_split=5,class_weight="balanced"	Location	0.5443	0.529	0.5356
KNN	k=5, distance="hamming"	Voting	0.9736	0.9231	1
KNN	k=5, distance="euclidean"	Income	0.8264	0.6613	0.5721
KNN	n_neighbors=10, weights="distance", p=1	Eucalyptus	0.6120	0.6205	0.5672
KNN	k=30,distance="hamming"	Location	0.4586	0.5245	0.4320

Table 19: Final best results

According to our metrics, accuracy,precision, and recall, the voting data could be modeled best with the selected models. Followed by the income model. The results for Eucalyptus and Location data are not as satisfying as expected. In these cases, either more input data, or cleaner input data, or just different model approaches are necessary to fit better models.

Nevertheless, it is to say that almost all categories to classify are rather imbalanced. Therefore it makes sense, that the model sometimes cannot perform as good because there is just not enough data available.

In addition, when building a model one has to take care of how to set which parameters because there are variations expected depending on the choice. Also the chosen seed, has impact on the outcome, as we saw among our experiments.

In the end, when deciding on an algorithm to take, it may also be a trade of between performance, and computation time

The voting and location data sets are the one from the kaggle competitions. For voting, the best random forest has been the last one submitted, and for the location set the as well the random forest with other settings.