Data Mining 2019

Project Report on Assignment 1

Basic Version

Luisa Ebner, Frederic Chamot, Philipp Lintl

April 19, 2019

1 Task 1

1.1 Data Exploration

The present data stems from a self-report questionnaire filled in by 276 students of the VU/UVA during a lecture of the course Data Mining at the VU. The questionnaire comprised 17 open question and multiple choice items about the current programme, prior courses taken, birthday, gender, bedtime, stress level and several other personal attributes. The items selected for further discussion in this section are shown in the table in 1.1.2. The data was collected anonymously and redistributed for educational purposes to the students of that class.

1.1.1 Data Cleanup & Preprocessing

All data manipulation was done using Python 2.7 with use of pandas v0.24.2 and scikit-learn v0.20.3. After loading the .csv into python the several decimal separators used in the whole set were converted to ".". All nominal variables in the dataset, namely those regarding the prior education of the students, were recoded into 0/1 format and handled as categorical data. Missing values were imputed using the mode of the respective attribute. In order to extract nominal data from the open question regarding the current study programme we categorized the answers using keywords. Each string was searched for programme-specific keywords, e.g. "Artificial", "artificial", "Intelligence", "intelligence", "AI", "ai" for the programme Artificial Intelligence. Using this technique we extracted the four major programme groups Artificial Intelligence, Computer Science, Computational Science and Business Analytics, accounting for 219 of the 276 observations, and created a new nominal feature consisting of this subset.

To convert the unformatted data into numeric format for the features bedtime, stress level and birthday we used regular expression patterns. This allowed us to extract time data from the string formats *hh:mm* and *h:mm*, which covered 156 entries, as well as 138 entries from the birthday item that were either in format *dd/mm/yyyy* or *dd-mm-yyyy*. From the birthday data we then derived another numerical feature depicting the age of the respondents. Finally we restricted the recorded answers regarding the stress level of the students to the permitted values 0-100 (values higher than 100 were recoded as 100) and categorized the remaining 268 data points into a novel categorical feature classifying the respondents in either "low", "medium" or high" stress group, corresponding to stress levels between 0-33, 34-66 and 67-100, respectively.

1.1.2 Descriptives

Programme	[CS(93),AI(73), BA(28), CLS(24)]
Previous ML	[yes(171), no(102)]
Previous Stats	[yes(242), no(24)]
Previous IR	[yes(157), no(114)]
Previous DB	[yes(142), no(130)]
Age	24.42(1.93)
Stress level	38.40(33.40)
Stress Cat.	[low(144, med(66), high(66)]

We did not find indications for any strong correlation in the data, but some features showed signs of weak biserial correlation with age (prior ML: $r = -.24$, prior stat: $r = -.24$) and stress (prior IR: $r = -.17$). Age and stress did not correlate ($r < .1$). Figure 1 shows that our suspicion that bedtime might interact in some way with stress level is not reflected in the data.

1.2 Classification

Two models that make use of the present data come to mind intuitively. Firstly, the association between stress levels and sleep duration has been reported more than once [1, 2], but unfortunately our data does not show any signs of reflecting this relationship. Secondly, and maybe even more intuitive, the correlation between study programme and the previously followed courses.

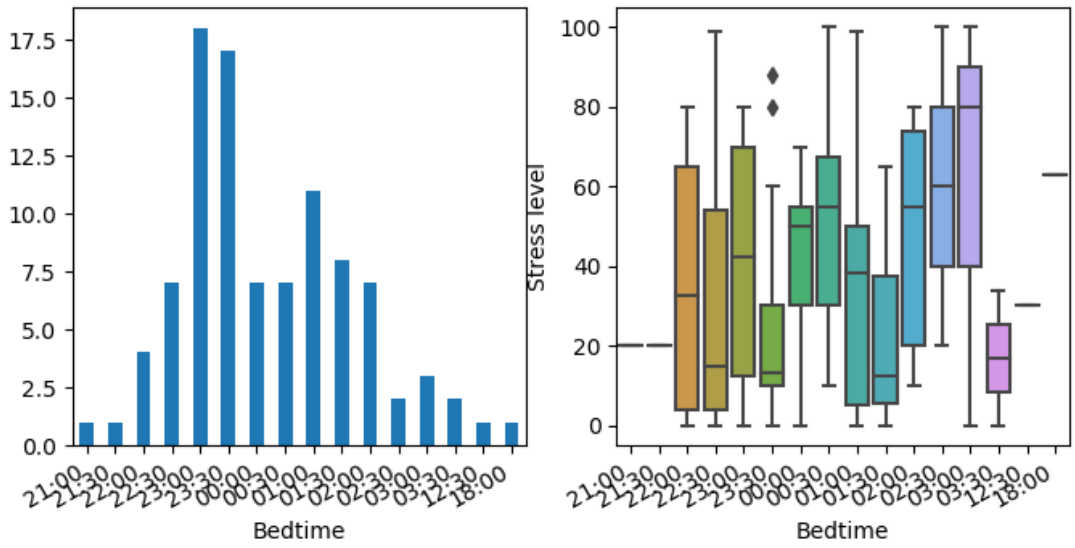


Figure 1: Distribution of bedtime datapoints (left). Distribution of stress level conditional on bedtime (right).

We built a model including the four nominal features describing the prior experience in statistics, database, information retrieval and machine learning that aims to predict the current study programme. In order to achieve a reasonable ratio of predictive features and possible target categories we restricted the dataset to AI and CS students and reduce the problem to a binary classification problem.

1.2.1 Classifiers

In order to compare the performance on the classification problem we constructed both a baseline classifier applying the naive bayes theorem and a decision tree classifier instance. From the four predicting features eight dummy features were derived which were used for both classifiers. In order to investigate the generalizability of our models we apply 3-fold cross validation on, i.e. split our data randomly into three subsets, and perform three iterations of training our model on two of the three subsets and testing it on the remaining subset to evaluate performance on unseen data. To measure the performance we calculate the accuracy (proportion correctly classified) and compare that to the no information rate which we derive from the confusion matrix for the specific subset of data the models are tested on.

1.2.2 Results

Running the naive bayes classifier on the three different folds yielded an average prediction accuracy of .55 (std = .011) at a no information rate of .58, which implies that it performs worse than a classifier that exclusively predicts the more frequent target category (in this case CS). The average accuracy of the decision tree classifier (see figure 3) with a restriction on maximum depth of two showed to only be marginally better at .60 ($std = 0.015$). Increasing the maximum depth did not change the magnitude of the prediction accuracy significantly. The poor predictive power of the tree is also notable in the gini impurity values that - for a majority of nodes - lie close to .5, thus indicating that values indeed are not well separated by the decisions in the tree.

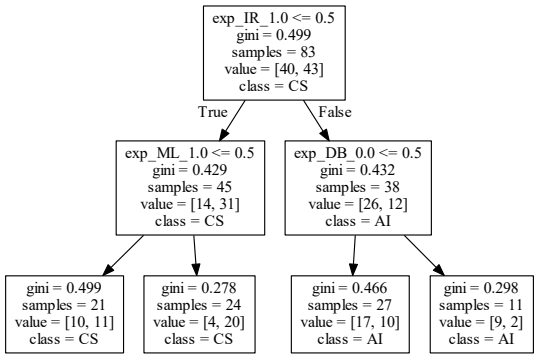


Figure 2: Decision tree with max depth=2 for illustration purposes.Dummies coding(1-yes/0-no). Predictive performance was constant over all tree depths.

1.2.3 Discussion

Several reasons in the structure of data and methodological choices may account for poor classifier performance. A primary concern regards the presumption of the naive bayes approach to data classification, which assumes that features are independent, an assumption which in the present data is rather dubious. The weak performance of the decision tree however signals that the data itself may also partly be responsible for the overall bad classification accuracy. The previously followed courses might very well overlap to a great extent specifically for Computer Science and Artificial Intelligence students, providing low decisive power to any classification algorithm.

Task 2

2.A Preparation

We start off with a description of the dataset and its variables. In Table 1, each variable is shown with its classes and the number of instances in brackets for categorials and min, mean and max for numericals.

Name	Statistics
Survival	[Yes(549) , No(342), `NA`(418)]
Pclass	[1st(323), 2nd(277), 3rd(709)]
Sex	[male(843), female(466)]
Age	min(0.17), mean(29.88), max(80), `NA`(263)
Sibsp	[0(891), 1(319), 2(42), 3(20), >4(37)]
Parch	[0(1002), 1(170), 2(113), 3(8), >4(16)]
Fare	min(0), mean(33.2), max(512), `NA`(1)
Embarked	[C(270), Q(123), S(914), `NA`(2)]
Title	[Master(61), Miss(264), Mr(757), Mrs(198), rare(29)]

Table 1: Variables with some statistics. Notation: C = Cherbourg, Q = Queenstown, S = Southhampton, Pclass = class of ticket, Sibsp = number of siblings or spouse on board, Parch = number parents or children on board, Fare = ticket prize, Embarked = city

Cat.	Sex	Pclass	Embarked	Title
1	ma(0.18)	1st(0.63)	C(0.55)	Master(0.57)
2	fe(0.74)	2nd(0.47)	Q(0.39)	Miss(0.70)
3		3rd(0.24)	S(0.33)	Mr(0.15)
4				Mrs(0.79)
5				rare(0.34)

Table 2: Survival rates per categorial variable. ma = male, fe = female

Table 4 reveals that sex, the class of the ticket and the embarking station could have an influence on the survival. However, the survival rates within the variables vary strongly.
We do not apply proper feature selection as in applying significance tests for a model with the respective vs. without. However, we delete variables, that very likely do not contribute greatly. For instance *Cabin* has 186 categories with very few instances. Also *PassengerId, Name, Ticket* and *Surname* are disregarded for similar reasons.
In terms of transforming given variables in order to increase the quality of the information, also called Feature engineering, we limited our efforts to the *Title* variable. We obtained the Title by clipping the name variable and chose it due to the survival rates as seen in Table 4. There are noticeable differences among the titles. Most clearly, Miss and Mrs survived to a much higher rate than Mr.. We could also investigate the size of families on the survival rate. However, there are a lot of unique surnames (875). This does not indicate a big influence of families or their sizes. Eventually, we experimented with a few transformations of the given variables. Details are found in the respective `Group34-Task2.Rmd` file.
There is a striking amount of missing values. In fact, 263 age values and 1014 cabin values out of a total of 1309 total observations appear to be missing. This is problematic, as the power of classification algorithms depends strongly on the amount and quality of data. We could disregard observations with missing values, which drastically decreases the amount of data and therefore will yield a less powerful classifier. Another approach to deal with this is called Imputation. However, this technique also poses problems, as we do not want to insert incorrect data or emphasize even more on the characteristic values. There exist quite a few possible techniques, such as imputating median, mean or predicted (e.g. linear regression or Random Forest) values for numerical values or the mode (most appearing class) for categorical variables. In general, it depends on the kind of variable and the amount of missing data. We chose mean / mode imputation for missing age and fare values.

2.B Classification and evaluation

We decided to use the setup enabled by the `mlr` package in `R`. It allows for a very intuitive handling of the classification task, as it integrates many classifiers, tuning, Imputation and performance evaluation. As the test set from Kaggle does not include the true Survived values (otherwise one could easily cheat), we split the train set by a ratio of 80/20. Thus, 179 test and 712 train observations stay for our setup. Based on these, we apply several classifiers with increasing complexity to illustrate performance differences. Performance is assessed by applying the trained models on the test set we created. We evaluated using accuracy and precision, as accuracy alone is known as a non sufficient

measure. It only considers the correct labels. However, if a dataset contains much more 0s than 1s, a high accuracy could also be achieved by always setting the output to zero and not learn anything. To counteract this, precision measures the percentage of correct labels for survived = 1. We start with a simple logistic regression model, followed by a simple decision tree, a random Forest model and finished with a *Xgboost* classifier. Both Random Forest and *XGBoost* incorporate randomness to counteract overfitting, which is why we train/test over several runs and report the average performance. For the Kaggle submission, we train on the full training set, that was downloadable.
As presumed, logistic regression does not classify well, which happens, when the decision boundaries are not linear. Decision trees in general perform well for classification tasks with a lot of categorial variables. Therefore, it is not surprising that both the cart and the random forest perform better.
After performing all four models with default parameter values, we observe that the Random Forest yields highest precision. One way to improve performance, is to apply parameter tuning. Therefore, we tune the number of trees created for the Random Forest, as well as the number of variables sampled in each split. We observe almost no difference to the basic version, which indicates that the default model is close to the tuned version. For the *XGBoost* classifier, we encounter an even worse performance after tuning. This can happen, if the tuning happens on a cross validation setup but the performance evaluation is based on one particular train/test split. Therefore we could improve our performance be integrating the tuning in the training, as for instance in nested cross validation, where we furtherly split our train split into train and validation. For the sake of simplicity, the intuition is that performance depends strongly on the training and test observations. So, it is better to assess performance on a cross validation setup. Due to time limits we could not implement it properly.

Model	prec.	acc.	Kaggle
logreg	0.628	0.770	0.708
cart	0.736	0.98	0.779
rforest*	0.746	0.822	0.784
xgboost*	0.733	0.826	0.765
rforest_tuned*	**0.747**	**0.821**	**0.789**
xgboost_tuned*	0.638	0.782	0.722

Table 3: Model perfomances on our evaluation setup and Kaggle score. prec. = precision, acc. = accuracy, logreg = logistic regression, cart = decision tree, rforest = Random Forest, *: average over 5 train/test runs, as randomness involved, `KAGGLE TEAM: DM-VU-2019-Group34(2)`

Further suggestions

More advanced imputation methods, such as using random forests to imputate missing age and fare values or applying nested resampling to tune hyperparameters not only on one train/test split should furthely increase accuracy and precision. Also, further feature engineering transformations, as well as using other classifiers should increase performance. All in all, considering the best teams do not seem to get higher accuracies than 85% with a lot more efforts, out best result is acceptable.

Task 3

3.A. State of the art solutions

Striking state-of-the-art solutions in the field of machine learning and data mining are developed annually in the course of the Knowledge Discovery and Data Mining (KDD) Challenge cup.[3, 4] The KDD Cup is a particularly popular data mining competition, organized annually since 1997. Open to everyone, this cup challenges machine learning and statistics communities with real-life, predictive modelling tasks on data sets from various research domains or the industry.[4] Typically, the participants are handed a raw data set together with a corresponding modelling task, that which requires a good understanding of the domain, task-specific, efficient evaluations and sophisticated modelling strategies.[5, 6] In 2007, the KDD cup was held in substantive cooperation with the Netflix Prize competition. The advertised data mining challenge was to predict different aspects of movie rating behavior among Netflix users.[3] As a database, the 2007 KDD Cup was based on the *Netflix Prize* training data set, published by the American media-services provider Netflix. This remarkably large data set comprises more than 100 million ratings from over 480 000 randomly chosen, anonymous customers on almost 18 000 movie titles. On Netflix, a movie can be rated with stars on a scale from 1 to 5. The data was collected over a time period of more than 7 years, from October 1998 until December 2005.[3, 6] By making these data available for the KDD Cup, Netflix pursued the self-interest to gain creative solutions, that would improve the accuracy of their recommendation system.[4] On that note, the Cup developed two separate tasks: The first task was called *"Who Rated What in 2006"*. Its subject was to predict the probability, that a certain user rated a certain movie in the year 2006. Therefore, 100 000 nonrated (user id, movie id) pairs drawn from the Netflix Prize training data set were made available.[3, 6] The second task, entitled as *"How many Ratings in 2006"*, entailed predicting the number of additional ratings, that the users from the Netflix Prize training data set gave to a subset of movies in the training set. For this analysis, the participants were supplied with 8863

movie ids, sampled from the training data set.[3, 6] The competition initiated an abundance of well-performing data mining techniques in order to understand, explore, structure and analyze the Netflix data set. However, the first prize for the *"How many Ratings in 2006"* task was awarded to Saharon Rosset, Claudia Perlich and Yan Liu, all working in the Data Analytics group at IBM research. [7] All presented modelling approaches were ranked according to the root mean squared error (RMSE) between their model predictions and the correct answers (the actual numbers of ratings received in 2006). The qualifying answer sets were published after the competition.[6] In the following, the winner team's modelling solution shall be presented and discussed in more detail. Explanations and remarks are based on the winner's report *Making Most of Your Data: KDD Cup 2007 "How Many Ratings"*[7].
Overall, the predictive modelling task was viewed as a regression problem with the number of ratings during 2006 as the model's target variable. An initial aspect of the modelling approach was the selection of features, indicative of of a movie's popularity and thus the unknown amount of ratings in 2006. Among these are the movie's age, it's availability duration on Netflix and other characteristics implying the movie's general popularity over time. The temporal dynamic of the given data set made it possible to view the "How many ratings" - task as a supervised learning problem with the target variable given for 1998 - 2005 and unknown for 2006. Predictions for 2006 could be achieved by an adequate time series model taking full account of the historical rating behavior since 1998 (all lagged ratings). Eventually, the number of ratings in 2006 can be estimated as a function of past ratings and movie specific features over time.[7] As the target variable expresses the number n of events (ratings of movies m) occurring in a fixed interval of time t (the year 2006), the same was considered to have a marginal Poisson distribution.

$$m_i \sim Poisson(\lambda_i t)$$

Consequently, the regression problem specified as a Poisson regression problem and the dependency of rating counts $n_1, ..., n_m$ per movie on the chosen feature set x was expressed by a generalized linear model (GLM) with natural log link:

$$log(\lambda_i) = \sum_j \beta_j x_{i,j}$$

In order to define the missing scaling parameter λ_i, representing a movie's constant rating rate, the "life cycle" of the movie m_i was studied and finally, $_i$ was estimated in a separate modelling exercise on the basis of lagged data sets.[7]
At that point in the modelling process, the winning team came up with the idea to exploit the information given in the *"Who Rated What"* test set of task 1. This alternative task of the 2007 KDD Cup is a classification task, where the model output shall represent the probability for (movie x, user y) pairs, indicating that some user y rated the movie x. The sampling probability of a particular (user, movie) pair is calculated according to the product of the marginal rating distributions of the user and the movie in 2006. The latter marginal is now proportional to the absolute number of ratings a movie received. This proportionality can be utilized to model task 2. In concrete terms, this lead to a 2-stage modelling approach. In the first stage, the appearance of a certain movie in the *"Who Rated What"* test set was used as the target variable to train a model in predicting the number of ratings, a movie received in 2006. In the second step, this trained model was applied to the *"How Many Ratings"* test set and evaluated though cross validation and training-test splits. This strategy enabled the IBM team to both take account of all recent rating information and model the dynamics of the 2006 ratings. According to the team members, the inclusion of the *"Who Rated What"* test set in the model training phase was the crucial component towards their model's prediction success.[7]

MSE verse MAE

MSE

The mean squared error is a standard statistical metric to describe average model performance errors. It is a measure of average model inaccurracy on continuous variables and calculated based on the difference between forecasts or predictions and eventual outcomes.[8] In practice, the MSE is used to examine, which model formulation among a set of possible models yields the most accurate estimates of the target variable. [8]. It is the default in many software packages and thus (most) commonly used in practice.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2$$

Put in words, the MSE is the average of the squared differences between the model's predictions $\hat{y}_j$ and the actual observations $y_j, j = 1, ..., n$.

MAE

The mean absolute error is an alternative metric to depict average model error, which averages the absolute error magnitude in a set of model predictions. It is the average over the test sample of the absolute differences between a model forecast and the corresponding, actual observation. [9]

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

As can be seen from the formula, MAE does not consider the direction of errors. Also, MAE is a linear score which means that all individual errors are weighed

equally, irrespective of their magnitude.[10]
Summing up, both the mean squared error and the mean absolute error can be employed in model evaluation studies to assess model performance. [11] Both measures express average model accuracy on the basis prediction errors. They range from 0 to ∞, where 0 signifies a perfect model and lower values generally denote better performance then larger ones. [9]

Discussion

Both MSE and MAE are statistical metrics. They condense a set of error values into a single number. Thus, both provide only one respective projection of the model errors, whereby either metric emphasizes on different aspects of the error characteristics.[11] Accordingly, there are situations in which the data, the problem at hand or the modelling objective make one metric more suitable than the other and vice versa.
Generally, the MSE is said to be more appropriate in representing model performance when the error distribution is expected to be Gaussian and the sample size is large enough. [11] In fact, this is a reasonable assumption for many real-world model settings. If so, the MSE can better illustrate the model's error distribution. The MAE on the other hand is particularly suited to describe uniformly distributed errors. [11]
Since the errors are squared before being averaged, the MSE gives comparatively high weight to large errors.[11, 10] One may claim, that MSE is sensitive to outliers. This can be disadvantageous insofar as only one bad prediction can skew the metric towards overestimating the model's faultiness. This is particularly problematic, when data is noisy. On the other hand, if almoste all predictions are wrong to a minor degree (≤ 1), the same may likewise be underestimated. Due to the squaring of errors, MSE increases the the variance of the frequency distribution of error magnitudes. [8, 10] Consequently, MSE is considerably larger in data settings with large error outliers, and rather similar to MAE only for small variances in the error distribution. However, it depends on the data set and the individual modelling objective, whether a penalization of great error magnitudes is desirable or not. In any scenario, where being off by 20 more than twice as bad as being off by 10 - that is to say, being further off the mark is an increasingly undesirable condition - MSE is beneficial. To conclude, MSE is advantageous whenever one is concerned about large errors whose consequences are much bigger than equivalent smaller ones. The MSE is e.g. widely quoted in climatic and environmental literature. [8] If on the other side being off by 20 is exactly twice as bad as being off by 10, MAE is more appropriate. On that account, the MAE is widely used in finance, where a 20$ error is usually exactly twice as worse as a 10$ error. As MAE is more robust towards outliers, its use is beneficial whenever data is inherently noisy and outliers are unexpected values, rather than particularly problematic conditions.
When it comes to interpretation, considering a model's absolute prediction errors is more intuitive and easier to justify. In his article on a comparison of the RMSE and the MAE, Willmott advocates the MAE as the "more natural measure of average error as $|e_i|$ is the actual size of each error and, in turn, the sum of all $|e_i|$s is a meaningful measure of total error". [8, 10]
On another note, the MAE may be discredited for its use of absolute values, which are highly undesirable for most mathematical calculations. [11, 10] Indeed, MAE is not differentiable at its zero point. In machine learning, the loss function L is often defined in terms of MSE. The same is continuously differentiable and a sufficient statistic for the Gaussian distribution. In the case of a simple linear model $y = ax + b$, L can easily be differentiated according to a and b to gain a model update. MAE to the contrary, requires more complicated tools such as linear programming to compute make use of eg. gradient descent approaches in model development.
Weighing up the aforementioned arguments for and against both metrics, one may conclude none of the two to be superior for every model and in every situation. [11] Instead, it depends on the model's objective and the underlying data set, whether MSE or MAE is more appropriate. Consequently, it is important to understand the context of the modelling problem before choosing a metric and in case of doubt, a combination of both metrics can be most qualified to assess model performance.[11]

Considerations on the size relation between MAE and MSE

Whenever both MSE and MAE are calculated, the relation $MAE \leq MSE$ holds. Put in words, MSE is always larger or euqal to MAE. This is obvious, as error values enter the MSE in a squared and the MAE only in an absolute fashion. [11, 8] Finally, MSE equals MAE only under very specific restrictions on the data:

$$MSE = MAE \tag{1}$$

$$\leftrightarrow \quad \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \tag{2}$$

$$\leftrightarrow \quad \frac{1}{n}\sum_{j=1}^{n}(e_j)^2 = \frac{1}{n}\sum_{j=1}^{n}|e_j| \tag{3}$$

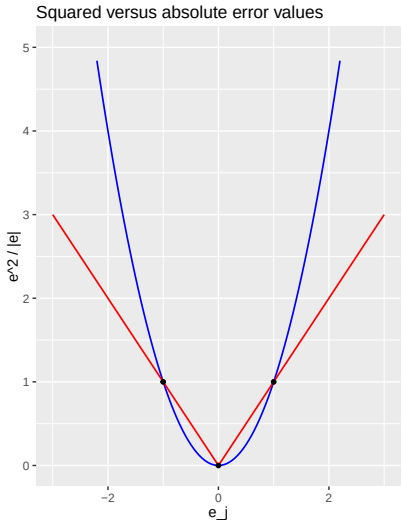
$$\leftrightarrow \quad \sum_{j=1}^{n}(e_j)^2 = \sum_{j=1}^{n}|e_j| \tag{4}$$

$$\leftrightarrow \quad (e_1^2 - |e_1|) + ... + (e_n^2 - |e_n|) = 0 \tag{5}$$

$$\tag{6}$$

Case Analysis:

$$1. \quad \forall j, j \in \{1, ..., n\} : (e_j)^2 = |e_j| \tag{7}$$



$$(5) = 0 \leftrightarrow \forall e_j : e_j \in \{-1, 0, 1\} \tag{8}$$

As can be seen in the graph above, absolute error values equal their squared equivalents only for the values $-1, 0$ and 1. This equivalence propagates to any sum of n (same) error values. Finally, for case 1 the MSE and the MAE are equal if and only if all differences between the model predictions and the actual observations have values in {-1, 0, 1}. For all values in]-1, 1[, the squared value is smaller than the absolute and for all difference/error values in $]-\infty, -1[\cup]1, \infty[$, e^2 is larger than $|e|$.

$$2. \quad \forall j, j \in \{1, ..., n\} : (e_j)^2 > |e_j| : \tag{9}$$

In this case, there is no MSE-MAE equality. All summands in formula (5) are greater than zero. Their sum is it therefore, too.

$$3. \quad \forall j, j \in \{1, ..., n\} : (e_j)^2 < |e_j| :$$

Also here, there is never MSE-MAE equality, since conversely all summands - and thus also the whole sum in (5) - are negative.

$$4. \quad else :$$

In all other cases, however, it is not possible to make a clear statement about the number of zeros. If there are both negative and positive summands in (5), there are infinitely many ways in which partial sums can balance out to zero in absolute terms. Nevertheless, the equality of MSE and MAE remains a very unlikely case here.

Next up, the MSE and MAE shall be calculated for different regression models built on the basis of a suited dataset available from the web.

In this regard, we chose the *Boston Housing* data set, a default dataset available on R, which is one of the most popular data sets to introduce and present regression problems in a machine learning context. Indeed, the Boston Housing data have been used extensively throughout the literature to benchmark algorithms.
With 506 observations and 14 variables, Boston Housing is a rather small data set. This enables for clarity in structure in model building processes.
In terms of content, The Boston Housing dataset concerns housing values in different suburbs of Boston, collected by the U.S Census Service in 1970. Accordingly, the target variable 'medv' reveals the median values of various houses Boston. Besides, it contains 13 feature variables that could potentially be related to the housing prices. The features provide information on socio-economic ('crim', 'age') and environmental conditions ('indus', 'nox'), educational facilities ('ptratio', 'lstat') and other factors relevant to rate the value of houses. 12 features are numerical, the one remaining is categorical.
Finally, the modelling objective is to predict the value of prices of the house using (a powerful subset of) the given features. For this purpose, we built different linear regression models in order to comparatively evaluate them by means of MSE and MAE. Firstly, we employed the step-up method to add significant features to an intercept model corresponding according to R^2. Thereafter, we applied the step down method, removing features from the full model. Both times the same linear model with 11 features resulted. Finally, we created a linear model penalized according to the lasso method. The latter contains 10 of the previously 11 feature variables.

	step up/down model	**lasso model**
MSE	1867.907	2115.477
MAE	309.240	322.897

As expected, MSE and MAE are very similar to the two (similar) models and the MSE values are significantly larger than the MAE.

3.C

The dataset can be modelled as a binary classification task on the document level with a target variable. Document level means, that each sms is one observation with a target being spam or ham. In order to represent each document in a feature space, we need to tokenize the text documents. Therefore, sentences are split into tokens, whereas each word and sign of punctuation is a token. Then, the so called Document-Term-Matrix (DTM) can be created. In it, each row stands for a

document (here sms) and each column is a token (all words and signs of punctuation). In the easiest case, the values in the matrix represent the frequency of each token in each document. Thus, each document is represented in feature space. The size of the vocabulary (unique tokens in the text corpus) can be large and thus the vector space is highly sparse, meaning that mostly 0s appear in the DTM. This noisy representation weakens classification algorithms. Thus, countermeasures such as deleting stop words (highly frequent but non-informative words, such as 'the' or 'he') and signs of punctuation. Also, manipulating the words themselves, as done in stemming or lemmatization can improve modelling results. Also, integrating n-grams into the DTM usually help to represent the data in a more informative way. N-grams mean that tokens can be combined to be a new token and then the appearance of the combination of the two tokens is measured. In terms of the Vector space, merely using frequencies usually does not suffice. Thus, transformations such as tf-idf (term-frequency-inverse-document-frequency) furtherly help to weight more informative words higher. Tf-idf accomplishes this by measuring the frequency of a token in a document over the entire frequency in all of the documents. More recently, word embeddings, such as word2vec or glove vectors help to represent text data in a much more dense and informative way. They are trained by using simple neural networks to decrease the dimensionality of the corpus or for word2vec by predicting the neighboring words of a large corpus such as Wikipedia. These pre-trained vectors can then be used, to represent the documents of the corpus. In the same time, they incorporate semantic meaning, as their place in the vector space is related to its meaning.

Again, analysis is done in `R`. After loading the data, we preprocess with with deleting special characters, signs of punctuation, numbers, set every letter to lower, delete stopwords from a stopword list and create a stemmed document term matrix additional to the normal one. Figure 1.2.3 illustrates the most frequent words within the ham and spam documents respectively. The ham documents do seem to be of more positively sentimented words, such as *love, care* or *happy*. Spam documents on the other hand most prominently contain rather negatively sentimentdd words in the context of giving money or a prize contest, such as *cash, urgent, win, collect* or *chance*.

After preprocessing the documents, we apply a simple classification algorithm called Naive Bayes, which as a rather non complex procedure yields acceptable results for text data. This is mainly because the otherwise strong assumption of feature independence is not necessarily violated for text. Other, simple and well performing text classification algorithms are Support Vector Machines.

As above, we split by a train / test ratio of 80/20 and evaluate accuracy and precision on the test data.

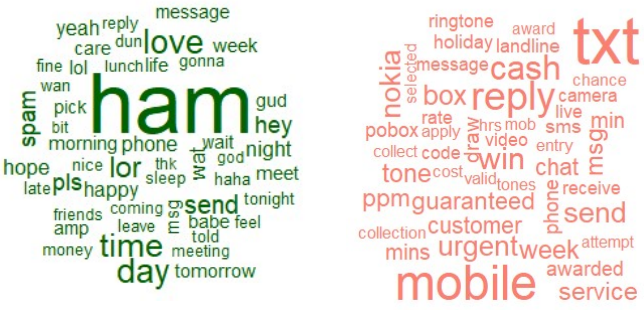


Figure 3: Wordclouds of most frequent words in ham and spam.

Model	Accuracy	Precision
Naive Bayes	0.975	0.981
Naive Bayes stem	0.974	0.983

Table 4: Performance of Naive Bayes model on sms test data.

The data seem to be very well separable, as the word distributions do not have many overlaps, especially after removing stopwords and applying all preprocessing steps. Other textclassification tasks require much more sophisticated steps as described before. For those, the classification results can be improved by using tf-idf weighting, adding bi- and tri-grams, as well as proper crossvalidation procedure, such as nested crossvalidation, in order to increase generalizability. The biggest increase in accuracy would lie in the application of another classifier, which in the best case is applied on word embeddings. This dataset, however already yields acceptable results for simpler models, as both classes have very distinct words, which do not appear often in the other class. If we had more and overlapping classes, more sophisticated text classification algorithms, which consider the order of words, will yield a better accuracy.

References

[1] Eui-Joong Kim and Joel E Dimsdale. The effect of psychosocial stress on sleep: a review of polysomnographic evidence. *Behavioral sleep medicine*, 5(4):256–278, 2007.

[2] Grant Benham. Sleep: an important factor in stress-health models. *Stress and Health*, 26(3):204–214, 2010.

[3] James Bennett, Charles Elkan, Bing Liu, Padhraic Smyth, and Domonkos Tikk. Kdd cup and workshop 2007. *ACM SIGKDD Explorations Newsletter*, 9(2):51–52, 2007.

[4] Marian Garcia Martinez. *Open innovation in the food and beverage industry*. Elsevier, 2013.

[5] Cathy O'Neil and Rachel Schutt. *Doing data science: Straight talk from the frontline.* " O'Reilly Media, Inc.", 2013.

[6] Bing Liu. Kdd cup and workshop 2007, August 2007.

[7] Saharon Rosset, Claudia Perlich, and Yan Liu. Making the most of your data: Kdd cup 2007" how many ratings" winner's report. *SIGKDD Explorations*, 9(2):66–69, 2007.

[8] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[9] Georgios Drakos. How to select the right evaluation metric for machine learning models: Part 1 regression metrics, August 2018.

[10] unknown. Mae and rmse - which metric is better? mean absolute error versus root mean squared error, March 2016.

[11] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.