Assignment 1, Group 5

Nikolas Ioannou(2671383), Paris Sfikouris(2671387), and Casper van Ek(2556917)

Introduction

This document was made for the course Data Mining Techniques 2020-2021, given at the Vrije universiteit Amsterdam. It consist of 3 parts. In the first part we made an exploration on a dataset that collected from the students of the course and next we create and run two classifiers on South African Heart Disease dataset. In the second part we mainly focus on preprocessing data and used classifiers to model the prediction of the survivors on the Titanic dataset. In the third part we concern ourselves with text mining and error measures.

1 Explore a small dataset

1.A Exploration

First contact: Before we begin with the cleaning process we first check the CSV and retrieved some basic information. From a first look We can see that the dataset has 313 instances and 16 attributes. Additionally We can see that we expect the type of attributes to be string, boolean or numeric.

Cleaning process: We used Python, pandas and numpy libraries to clean the dataset. First we try to find a pattern for each attribute to make values as manageable as possible. For the first column **'What programme are you in?'**, we collect all programs with the same meaning and write them as one. Moreover, we mark as NaN values with no actual meaning. Columns with boolean values we let them as it was. For the columns **'When is your birthday (date)?' and 'Time you went to be Yesterday'**, function todatetime() applied convert not date type to Nan. Furthermore, columns **'What is your stress level (0-100)?', 'Number of neighbors sitting around you?', 'You can get 100 euros...would deserve then?'**, we use similar decisions. About stress level, we convert all the values that are above 100 to 100, below 0 to 0 and everything else that is not numeric to NaN. For the neighbors we decide to set a range between 0-10 and so, everything that is below 0 we convert it 0 and everything over 10 convert it to 10. Finally, for the 3rd column, again, we convert everything that is over 100 converted to 100, below 0 to 0 and not numeric to NaN.

Information retrieved: After finished the cleaning we check the attributes with NaN values and try answer to question **"In which attributes We can found more NaN values and Why ?"**. As we can see from the figure 1 below highest NaN values is column **'Time you went to bed Yesterday' (132 NaN)** and the reason for that is because at the cleaning process no correct date types (we found a lot) converted to NaN.

Moving to the second attribute with the most NaN values, **'When is your birthday' (73 NaN)**,it is not random that this attribute has the most NaN

values, because is asking a sensitive question. Students did not fill in correct the question and showed caring about privacy. In the other hand We can't noticed that privacy is not the only reason for NaN but also uncaring from users to fill in correct the questionnaire.

The third most highest attribute with NaN is the **You can get 100 euros...would deserve then?' (53 NaN)**. We believe that the reason of that is the irrelevance of the question but also the complication that has in it. The answer to the user wasn't came out naturally and that might affect the results. Again the uncaring of user to fill in the questionnaire was one of the reasons of many NaN.

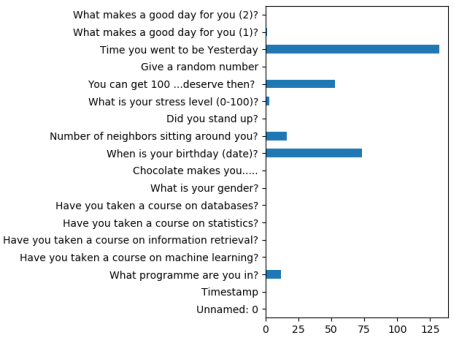

Fig. 1. Number of NaN values in

As second question we have **Which Gender is care more about privacy?**. Results of figure 2 below, showed that gender is not a measurement type for privacy.

1.B Basic classification/regression

We have select to work at data regarding coronary heart disease (CHD) in South Africa [8]. The reason We choose new dataset is because it will be more interesting to make predictions on a real case that targets a real problem.

Algorithm used: Our research question is, "He/She has CHD?", we use binary classification because we interested in classifying data into one of two binary groups. Chosen algorithms, Support Vector Machines and Random Forests.

Parameters of algorithms: For both algorithms we used a dataset of 462 records and 10 attributes. We make a cleaning because algorithms works only with numeric values. Then we train our model with 80 percent of our data and with the other 20 percent we test our models. For SVM we choose as parameters "kernel='linear'", "C=1", randomState = 42 and for Random Forest maxDepth = 2, randomState=42.

Features: To create our models and analyzed them we make use of pandas in combination with sklearn. Pandas help us to read the dataset and clean it. For

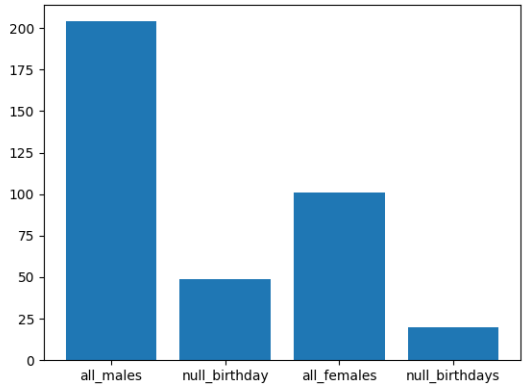

Fig. 2. Which Gender cares more about privacy?

the machine learning part sklearn offered ready functions, like trainTestSplit, svm.SVC, RandomForestClassifier for training the model and crossvalscore, confisionMatrix for evaluating the Algorithm.

	predict no	predict yes
actual no	55	6
actual yes	27	5

	predict no	predict yes
actual no	153	8
actual yes	18	14

Table 1. Confusion Matrix of Random Forest (left), SVM (right)

	precision	recall	f1-score	support
0	0.67	0.9	0.77	61
1	0.45	0.16	0.23	32
accuracy			0.65	93
macro avg	0.56	0.53	0.5	93
weighted avg	0.6	0.65	0.58	93

Table 2. Random Forest, classification report.

	precision	recall	f1-score	support
0	0.75	0.87	0.80	61
1	0.64	0.44	0.52	32
accuracy			0.72	93
macro avg	0.69	0.65	0.66	93
weighted avg	0.71	0.72	0.71	93

Table 3. SVM, classification report.

Accuracy score of Random Forest: 0.6451612903225806
Accuracy score SVM: 0.7204301075268817
(SVM) 0.72 score mean with a standard deviation of 0.08
(Random Forest) 0.67 score mean with a standard deviation of 0.11

Analyze/compare results, As we can see from the above results, True negatives of Random Forest are 55 in contrast with SVM 153. False negatives again SVM is better with 18 against 27. True positives and False positives for SVM are 14 and 8 respectively and for Random Forest 5, 6 respectively. From **classification report and Accuracy score,** We can see that SVM outperformed Random Forest algorithms and that is logical because SVM is intrinsically suited for two-class problems, while Random Forest for multiclasss problems. From the other hand, using **cross-validation**, Random Forest perform slightly better than SVM.

2 Titanic competition

2.A Preparation

	Attributes	Types	Distribution
1	name	boolean	0/1
2	survived	alphanumeric	-
3	pclass	discrete	[1,3]
4	sex	boolean	F/M
5	age	doubles	[0.42,80]
6	# of siblings	integers	[0,8]
7	# of parents/children	integers	[0,6]
8	ticket number	alphanumeric	-
9	fare	doubles	[0,512.3232]
10	cabin	alphanumeric	[A,B,C] classes
11	embarked	characters	[S,C,Q] ports

Table 4. Features statistics

Exploring the data we can observe some correlations between certain attributes and surviving the Titanic. Firstly we see that survivors were mostly females. Moreover we can see that people with age of less than 40 years old had higher chances of surviving. People embarked at Southampton and with one or no siblings also had a higher chance but that may be because these two attributes were the dominant values in the specific attributes.

For preparing the data to be more usable in the learning algorithm we need to first discard attributes that do not add any value in determining the outcome or may cause noise in the system . For that we will remove the attributes of name and ticket number as they do not play any role regarding surviving the Titanic and we will discard the cabin attribute as most of its values are missing. Additionally some of the values for age attributes are also missing thus we replaced them with the mean value. For the attributes of sex and embarked port we will transform to integers. We assigned females to zero value and males to one value and for the embarked port Southampton, Cherbourg and Queenstown with 0, 1 and 2 respectively.

2.B Classification and evaluation

For building our models to predict the outcomes we first need to setup and prepare the data. Firstly we transformed and removed some attributes in the dataset as described above and then based on the dataset we downloaded from Kaggle we created a training and a test set for modeling our classifiers.

Classifiers used were Decision Tree, Logistic Regression and KNeighbors. Decision tree because of its efficiency and ability to process large amount of data that require classifying categorical data based on their attributes. Logistic Regression because is one of the basic and popular algorithm to solve a classification problem and KNeighbors as a third classifier for a better comparison with the others.

KNeighbors Classifier

	Predicted 0	Predicted 1
Actual 0	149	37
Actual 1	47	64

Score 0.71717

Decision Tree Classifier

	Predicted 0	Predicted 1
Actual 0	161	25
Actual 1	32	79

Score 0.80808

Logistic Regression

	Predicted 0	Predicted 1
Actual 0	156	30
Actual 1	31	80

Score 0.79461

Table 5. Classifiers statistics

As we can see Decision Tree had the most accurate score with Logistic regression falling behind by 0.1. KNeighbors did not perform as well as the other classifiers.

Score on leadership board: 0.50239

The score is quite lower than expected as in the training set the decision tree classifier scored 0.80808.

3 Research and theory

3.A State of the art solutions

Often times, which techniques should be used depends highly on the context. In this section we will give an example of this, describing a competition that was held and what techniques were used by the winner of said competition. Contrasting those techniques with those chosen by the other contestants, so that we may gain insight into why the competition ended the way it did.

The competition: The competition we have chosen to research is the "Partly Sunny with a Chance of Hashtags" competition, hosted on the website Kaggle[1]. This competition, held in 2013, had as a goal to create an algorithm that can make judgements on the weather based upon tweets. Judging a tweet's sentiment towards the weather, when the weather occurred in relation to the tweet and what kind of weather occurred.

Thus the competition concerned itself with a multi output problem. As input the algorithm is given the exact text of the tweet. Also included are the location & state where the tweet was made, if this data is available. The state and location being in plain text, with no set format and inconsistent naming. As output it has to generate 3 sets of variables, each variable being a value in the range [0,1].

Each set of variables is about one of the qualities which are to be discerned about the weather. The variable value of 1 indicating total confidence in a certain judgement, 0 indicating no confidence. The set of variables about the sentiment and chronological order each sum up to 1. In total there are 24 variables.

To evaluate the performance of a solution the competition used Root Mean Squared Error (RMSE). The consequence of using this evaluation method is that solutions which are consistently good trump those that are occasionally great and usually average. This is because, using his method, larger errors have a disproportionately large effect on performance. [7]

The winner: The winner of this competition was Aliaksei Severyn, a PhD student from the University of Trento. In the discussion board of the competition he explains his approach, and elaborates on the reasoning behind his decisions.

At the start of his explanation, he mentions he uses stacking. However, when he elaborates on this it seems he is actually using cascading. Which means, as he himself states, that he uses multiple models where the output of one model is the input for another.

His solution first tokenized the tweets. Using the tf–idf of the 1 to 7 ngrams. Which means that the text was split up into tokens consisting of 1 up to 7 words, with each token being a value of how often the token occurs in the tweet minus how often it occurs in all tweets. To save the algorithm some time in learning which tokens are associated with which sentiment, a sentiment dictionary was used. This proved very helpful in predicting the first set of variables.

The second step was to use the previously extracted values to make an initial prediction of the 24 variables. The total number of extracted values was exceedingly large, totaling to around 1.9 mil. To deal with this ridge regression was used, as it is especially good at dealing with a large number of attributes[4].

The third step was to tackle one of the weaknesses of ridge regression. This weakness is that it treats its outputs as independent. Of course this is not the case, for instance if the weather is sunny then it is more likely to be perceived as positive. To solve this, the output of the first model is given to a second model. This second model used Random forest, the reasoning given being that it deals well with a small number of features.

Finally, it is checked whether the output of the second model has the properties the output should have. Namely that the first and second sets of features sum up to 1 and that all values fall in the range [0,1]. The first problem being solved with L1 normalization, the second by clipping the results if needed.

The runner ups: The winner was not the only one to use ridge regression. But his approach stood above the others by making use of secondary techniques to counteract the weaknesses of his main strategy.

Even those that did not make use of ridge regression made choices that caused the same issue to manifest itself. The issue that the outputs are treated as independent variables. This is because they noticed that training three models, each for one of the sets of features, increased performance. However, they then did not implement techniques that would make the results of these 3 models dependent on each other.

3.B MSE vs MAE

Just as RMSE described before, Mean Squared Error (MSE) and Mean Absolute Error (MAE) are ways to measure the error between two paired sets of data. In the context of regression this often is the set of predicted values versus the actual values.

Formulae: Given set $X = \{x_1, x_2, ..., x_n\}$ and set $Y = \{y_1, y_2, ..., y_n\}$, we can create the set of differences between all x and y. This is set $D = \{d_1, d_2, ..., d_n\}$, where $d_i = |x_i - y_i|$. The calculation of the MSE and MAE of the paired sets X and Y is as follows:

$$\text{MSE} = \frac{\sum_{i=1}^{n} d_i^2}{n} \qquad \text{MAE} = \frac{\sum_{i=1}^{n} d_i}{n}$$

As can be seen in the above formula, MSE exhibits quadratic growth and MAE shows linear growth in relation to the difference between X and Y. MSE is thus similar to RMSE, in that large differences have a disproportionately large effect on the final value.[5] This causes the largest differences to be the main denominator of the final result. This can be both a good and bad thing, depending on context.

A situation where this can be advantageous is if one cannot afford any large errors, cases where the consequences of a significant mistake are unacceptable. An example would be deciding the correct dosage of medicine for a patient, where a significantly wrong dosage could have deadly consequences.

However, there are situations where using MAE is preferable. If the consequences of a few outliers are negligible, and one prefers an on average better performance, then MAE is a viable choice. Especially if there are a few extreme

outliers, then those can be disruptive to the performance of a model if one uses MSE. There have been cases where only %0.5 of the instances determined the final error measure when using MSE [3].

Example: There can be cases where MSE and MAE return the same result. This happens when for all d in D we have $d = d^2$. A trivial example where this is the case is if we have $d = 0$ for all d in D, as $0 = 0^2$.

The previous example given is about the case where $X = Y$. This means that all paired values of X and Y are the same. In the context of regresssion, if X was the set of predicted values and Y the set of actual values, then all predictions were perfect and there is no error.

Experiment: Earlier it was mentioned that extreme outliers have a greater effect when using MSE than when using MAE. We seek to show this by adding one outlier to an existing dataset. To easily visualize the effects we choose to employ simple linear regression.

The dataset we use is a record of heights and weights, the goal of the regression being to judge someone's height based upon their weight[2]. The dataset consists of 10 000 entries, we have grabbed a sample of 200 entries and added one extreme outlier.

We employ linear regression twice, first using MAE as our loss function and then MSE. Stochastic gradient ascent was used, the model training for 300 epochs. After the last epoch the MSE and MAE were measured of both resulting models. The results can be found in fig. 3 and Table 6.

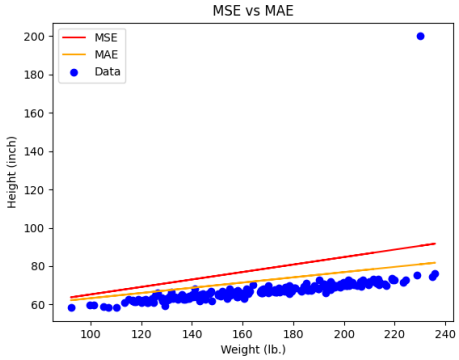

Fig. 3. MSE and MAE linear regression

	MSE loss	MAE loss
MSE score	102.7	11.7
MAE score	195.7	4.5

Table 6. MSE and MAE final score

As can be seen in fig. 3, the model using MSE is 'pulled' closer to the outlier that the model using MAE. When looking at their final MSE and MAE score, we can see that both models performed better at the score they used as their loss function. This is of course to be expected, as that is what they are optimizing. The observation that the model using MSE has moved closer to the outlier

supports the claim that using MSE means outliers have a greater effect on the final results.

3.C Dataset analysis

The earlier competition we discussed in section 3.A dealt mostly with text data as input. In this section we will deal with a similar dataset of text messages, a dataset of SPAM and HAM messages. SPAM being messages which are unwanted, HAM being all other messages. We will first discuss general techniques which can be applied to text, then discuss ways text data can be transformed, and finally describe a model we made with the SPAM/HAM dataset.

Text mining techniques: Text has quite a varied number of techniques which can be applied to it[6]. At the highest level we can still apply both supervised and unsupervised learning.

Most supervised learning tasks on text are classification tasks. Text specific classification problems include sentiment analysis, topic analysis and intent detection. Sentiment analysis was one of the tasks of the competition we described earlier. Topic analysis and intent detection are similar to sentiment analysis, but seek to classify the topic and intent of a text respectively. Each of these classification problems can also be treated as a regression problem, the output not being a certain label but a numerical confidence rating for each possible label.

Unsupervised learning can also be applied to text. There are a number of technique which are specifically focused on text, an example of this would be summarization. Summarization is the task of shortening and summarizing huge documents of text such that they still retain their key essence or theme. The technique can be further divided into extraction-based and abstraction-based summarizaion. The former being focused on highlight the most important parts of the original text, the latter to paraphrase the text using sentences not necessarily present in the original.

Text transformations: A lot of the transformations used by the contestants of the competition can also be applied to the SPAM/HAM dataset. One of the first things the contestants do is that they tokenize the messages into n-grams. An n-gram is a sequence of words of length n that occurs in a text. For instance, "has a" is a 2-gram of the sentence "My father has a boat". Splitting up the text into n-gram tokens makes it easier for an algorithm to process the text and find patterns.

After the text is tokenized it is likely to contain both redundant and useless tokens. Both are a source of noise, making it harder to reason about the text and find meaningful patterns. Redundant tokens are those tokens that might use different spelling, but mean roughly the same. An example of two redundant tokens would be "walk" and "walking", if these two tokens are treated as separate then a ML algorithm has to spend the time learning the meaning of both.

A way to reduce the amount of redundant tokens is to use lemmatization, which groups together the inflected forms of words. Other causes of potentially redundant tokens are capitalization and the use of abbreviations. The process of removing redundant tokens is called normalizing the tokens.

After tokenization and normalization, we still have the problem of useless tokens. Useless tokens are those tokens that tell us nearly nothing about the text, and are thus a waste to process. One common source of useless tokens are stopwords, words such as "I" and "the". Removing these words means there is less clutter when trying to find the relevant parts of a text.

There is still much more that can be done with a text after these steps. Examples are the correction of spelling errors to reduce redundancy, the calculation of tf-idf as described in section 3.A and the usage of dictionaries such as sentiment dictionaries to prevent a model from having to learn these associations.

SPAM detection model: We will apply a subset of these transformations on the SPAM/HAM dataset. First we tokenized the text messages into 1-grams, removing special icons. After the messages were tokenized, we applied lemmatization on the words and removed stop words.

To then create a model that is able to give a prediction whether a message is SPAM or HAM we split the dataset in half. The first half functioning as our training set, the second as our test set. We then create a dictionary which for each word in the messages of the training set stores a value in the range [1,-1]. The value being the number of times the word occurs in HAM messages minus the number of times the word occurs in SPAM messages, divided by the total number of times the word occurs in the messages. The value 1 indicating the word only occurs in HAM messages, -1 indicating it only occurs in SPAM messages.

To predict whether a message is SPAM or HAM, the model first performs the same pre-processing on it as was performed on the training set. Then for every word in the message it checks whether the word is in the dictionary it created. If a word is present in the dictionary, then the model squares the value that is associated with the word and stores this value. During squaring it preserves the sign of the value. If the sum of these values is positive then the word is judged as HAM, else it is judged as SPAM. The reason why the values are squared is to make words that appear disproportionately often in HAM or SPAM messages have a greater effect on the outcome.

Of the words that occur at least a 100 times in the dataset. The top 5 words that occur most often in HAM messages are: "gt", "lt", "say", "come" and "ok". The top 5 words that occur most often in SPAM messages are: "txt", "â", "free", "call" and "4". With this model 99.5% of the HAM messages in the test set were correctly labeled, and 82.3% of the SPAM messages in the test set were correctly labeled.

One major weakness of the current model is that it treats the words as independent. We process messages as 1-grams, individual words, and don't use techniques which can discover patterns between the n-grams we extract. We also don't look at sentence structure, where a word is placed in a sentence. An example where discovering such context is important is for words such as "hot". Take the sentence "this is a hot day" and "hot TA's in your area". The first being HAM, the second SPAM. Whether the word "hot" indicates SPAM is

highly dependent on context, something which our current model is not able to discover.

References

1. Partly sunny with a chance of hashtags, https://www.kaggle.com/c/crowdflower-weather-twitter/
2. Ali, M.: weight-height.csv (May 2018), https://www.kaggle.com/mustafaali96/weight-height?select=weight-height.csv
3. Chatfield, C.: What is the 'best'method of forecasting? Journal of Applied Statistics **15**(1), 19–38 (1988)
4. Kennedy, P.: A guide to econometrics. John Wiley & Sons (2008)
5. Makridakis, S., Hibon, M.: Evaluating accuracy (or error) measures (1995)
6. Sarkar, D.: Text analytics with python (2016)
7. Willmott, C.J., Matsuura, K.: On the use of dimensioned measures of error to evaluate the performance of spatial interpolators. International Journal of Geographical Information Science **20**(1), 89–102 (2006)