Autoencoder vs. Regression Neural Networks for Detecting Manipulated Wine Ratings

Michaela Baumann
Business Intelligence / Analytics Competence Center
NÜRNBERGER Versicherung
Nürnberg, Germany
email: michaela.baumann@nuernberger.de
ORCID: 0000-0001-5066-9624

Michael Heinrich Baumann
Department of Mathematics
University of Bayreuth
Bayreuth, Germany
email: michael.baumann@uni-bayreuth.de
ORCID: 0000-0003-2840-7286

***Abstract*—In this study, we analyze the ability of different (neural network based) detection methods to identify manipulated wine ratings for two "vinho verde" datasets. We find that autoencoders outperform regressions in terms of true/false positive rates. All in all, neural network based autoencoders seem to detect best, while classical linear models show the smallest performance variability. Most interestingly, linear model based autoencoders perform well within a reasonable computation time. Furthermore, hyperparameter tuning via sequential accumulative selection is established.**

***Keywords*—anomaly detection; manipulation identification; wine preferences; artificial neural networks; autoencoder.**

I. Introduction

In a world of increasingly differentiated products and customers who frequently change their buying behavior, it is difficult to assess whether the price-performance ratio is appropriate before making a purchase. An important and much-used assistance in such buying decisions are ratings. In this study, we are going to approach the question of whether and how manipulated ratings can be detected using wine quality ratings as an example. When ratings come from an official or non-official authority (such as Gambero Rosso's *Vini d'Italia* [1], Robert Parker's *The Wine Advocate* [2], *Gault&Millau* [3], or *Guide Michelin* [4], when dealing with wines, hotels, restaurants, or related topics), it is possible to verify with little effort whether ratings given by a merchant or producer are genuine by simply looking up the relevant work. However, since by far not all wines are represented and rated in one of the works published by an authority, there are countless other ratings. These other ratings, which are not given by an authority, are difficult to verify for authenticity, and it might even be possible that they are not objective, but rather paid for by someone. In the following, we are going to show possibilities for detecting such manipulated or faked ratings.

A very basic idea for how to identify manipulated ratings would be via (linear) regressions. That means, when we have other, exactly measurable features, such as alcohol content, pH value, or density, we can learn how to predict the rating using these independent variables on correctly rated data objects. Ratings that differ (strongly) from the predicted ones on unseen data might be suspicious. The described methodology is commonly used in many contexts, such as in economics and finance, and often leads to useful results (cf. [5]). However, a linear regression does not lead to good results in our case, i.e., when trying to detect manipulated wine ratings. Thus, the research question is how manipulated wine ratings may be detected in a better way. Since artificial neural networks are currently en vogue, one can of course use a regression by means of a neural network (cf. [6]–[9]). Note that a linear regression is the same as an exactly trained, fully connected neural network without any hidden layer with linear activation functions (i.e., id resp. pass-through), when adding a dummy column (filled with 1s) in the data for the intercept and using Mean Squared Error (MSE) as loss. Regressions based on shallow or deep neural networks are likely to outperform a linear regression.

Especially when dealing with outlier detection, so-called autoencoders (resp. reconstruction networks or auto-associative neural networks) are a common means [10]–[13]. Autoencoders consist of two parts (i.e., two regressions), an encoder and a decoder. The encoder compresses the input data to a lower dimensional representation usually referred to as the code; the decoder takes the code as input and aims to reconstruct the original input.

Given a well trained autoencoder, when the input and the output differ (strongly), the data might be manipulated (or in other contexts: an outlier, an anomaly, fraudulent, or suspicious). Note that there are much more application areas of autoencoders, such as dimensionality reduction, data compression, or denoising. Although it is in principal assumed that the quality depends on the other features, the autoencoder does not use this information, that is, the quality and all other features are considered as coequal input (and output) variables. Since the autoencoder does not use all the information that is actually available, it would be very interesting if it nevertheless achieved better results.

In the work at hand, we investigate how *Regression Neural Networks* (RNN) and *Neural Network based Autoencoders* (NNA) can be used to identify manipulated data. Additionally, as benchmark models we use a *linear regression* (Linear Model; LM; see [5]) and an autoencoder that implements two linear regressions (*Benchmark Autoencoder;* BA; see Section IV-D). Clearly, there are several other data analytics methods that might be applied, e.g., support vector machines [14], however, an investigation is postponed to future work,

Copyright (c) IARIA, 2022. ISBN: 978-1-61208-972-0

as it would go well beyond the scope of this paper, especially since the number of those techniques keeps growing (see [15][16]).

Indeed, we find that neural network based autoencoders are suitable for detecting manipulated wine ratings. Since one of the main application areas of autoencoders is anomaly detection, this result is, at a first glance, not unexpected. However, because regressions from measurable data to wine quality are used in the literature to estimate or predict wine quality, it might be unexpected that such regressions do not perform well concerning manipulation detection. Most interestingly, we find that a linear model based autoencoder, which we use as a benchmark, can be a good tradeoff between detection ability and computation time.

The remainder of this paper is organized as follows: Section II reviews both the literature on wine data analysis and those on anomaly detection in general while Section III specifies the data we are using. Sections IV and V describe the method we use and Section VI presents the results. Finally, Sections VII and VIII conclude and describe possibilities for ongoing work.

II. Literature Review

The closely related literature roughly splits into two groups, namely data analytics of wine quality and general outlier/anomaly detection resp. fraud identification. The analytics of wine quality mostly covers the prediction of wine ratings based on measurable features. Cortez et al. [17][18] compare several data mining regression methods for predicting wine preferences based on easily available data during the certification of wines. In this context, they originally published the two datasets that are also used in the work at hand. They use the vinho verde white wine dataset [18] and both the vinho verde red wine and white wine datasets [17]. Besides these papers, also the importance of the selection of the most relevant features before predicting the wine quality with machine learning regression methods is investigated for the vinho verde datasets [19]. The vinho verde white wine dataset is used for classifying wine preferences via fuzzy inductive reasoning [20]. Deep neural networks are applied for classifying wine ratings (in detail: multiple classification) on the vinho verde datasets [21].

Several regression models for assessing wine quality are developed with data from southern France consisting of altogether 137 variables (vineyard variables and enological variables) to assist the winemakers in their business [22]. Also, the effect of weather and climate changes as well as the effect of expert ratings on the prices of Bordeaux wines are analyzed [23][24]. With this, the efficiency of the Bordeaux wine market is assessed. Tree models are used for predicting the relative quality of German Rhinegau Riesling considering terrain characteristics obtained through cartographic studies [25]. A frameworks is developed that automatically finds an appropriate set of classifiers and hyperparameters via evolutionary optimization for predicting wine quality for arbitrary wine datasets [26].

Having in mind the literature reviewed above, which predicts wine ratings or conducts data analyses of wine quality, the work at hand contributes by connecting wine rating predictions and anomaly detection. The topic of outlier/anomaly detection and fraud identification is addressed in a lot of related work in various contexts (see, for example, the surveys and summaries [27]–[31]) and we can only touch on this broad topic here. Generally, according to Chandola et al., "*Anomaly detection* refers to the problem of finding patterns in data that do not conform to expected behavior" [27]. Usually, anomalies have to be identified throughout the analysis of data so that they can be treated separately and do not distort the results of the analysis of "normal" data. However, in the case of fraud and also in our case of manipulation detection they are of special interest. Fraudulent and manipulated data objects inhibit abnormal patterns but they try to appear as normal. The detection of anomalies, especially of intentional, malicious anomalies, such as fraud or manipulation, is very challenging and there are many approaches that try to accomplish this task. The approaches basically fall in one of the following three categories [28]:

- Unsupervised methods (e.g., clustering); labels are not needed here and new patterns (normal ones and outliers) may be processed correctly.
- Supervised methods (e.g., classification); this needs prelabeled data, however, anomalies are usually very rare and the labeled datasets are, thus, highly unbalanced; new patterns are unlikely to be processed correctly.
- Semi-supervised methods (e.g., autoencoders); normal behavior is known, i.e., (a part of) the training data is labeled as normal, and new, unlabeled data objects are compared to the normal case.

In addition to methods that require tabular data (a priori tabular data, but also image, audio, or video data transferred to tabular data) there are methods that operate on graph based data [32], which are especially useful when identifying anomalies in highly connected data. The approach of the work at hand falls into the third category, i.e., semi-supervised methods, and works on tabular data.

III. Data

The approach described in this work is applicable to various working areas (see Section VIII). We demonstrate it using wine data as an example because of the following reasons.

A rather simple advantage is the good data availability and (if no wine names or winemaker names are used) the innocuousness of the data. Further, the explaining variables (except for wine or winemaker names) are metric, clearly defined, and exactly measurable (e.g., alcohol content, acid, pH value, red/white). The used datasets further have a unique target feature and not a list of ratings (see also Section VIII).

We use the "Wine Quality Datasets" [17] from the Universidade do Minho [33], more specifically the datasets "White Wine Quality—Simple and clean practice dataset for regression or classification modelling" [34] and "Red Wine Quality—Simple and clean practice dataset for regression or

classification modelling" [35] downloaded from *kaggle,* which are licensed under "Database Contents License (DbCL) v1.0," *Database: Open Database, Contents: Database Contents* [36]. Both datasets contain anonymized *vinho verde* wines and have the same twelve columns, i.e., features, namely: "fixed acidity," "volatile acidity," "citric acid," "residual sugar," "chlorides," "free sulfur dioxide," "total sulfur dioxide," "density," "pH," "sulfates," "alcohol," and "quality." There, *quality* is the wine rating, which is supposed to depend on the other, explaining features, called independent. All values except for the ratings are in some meaningful physical unit, while the ratings range from 0 (very bad) to 10 (excellent) in integer steps [17]. The red wine dataset consists of 1,599 entries while the white wine data has 4,898 rows, leading to a combined data set with 6,497 rows and 13 columns. For the distinction of red and white wines we added a binary encoded categorical column. Please note that we do include neither descriptive statistics like plots or correlations, nor explorative analyses, such as clusterings, nor distribution estimations etc. in this work. There is already a lot of such work done for the vinho verde datasets. Such statistics and many more analyses can be found in the work of Cortez et al. [17][18], in other papers [19]–[21], and further tutorials or notebooks [37]–[41].

IV. Methodology

As outlined in Section I, the aim of this work is to identify manipulated ratings. For this, we train several network based models on the provided, correct data. We then make predictions on unseen data objects where we manipulate a certain part of these objects. As manipulation, we increase the original rating of very low rated wines as this seems to be a "reasonable" manipulation in the context of wine ratings (when someone wants to increase sales numbers). By comparing the provided, potentially manipulated data and the predicted data we aim at identifying the manipulated data objects. Objects for which the predicted values strongly differ from the provided data are more likely to be manipulated. We assess the models' detection performance, i.e., their ability to identify manipulations through calculating the true and false positive rates when marking the most deviating data objects as suspicious. To prevent overfitting and account for other random effects we apply bootstrapping. That is, we repeat the process of randomly splitting the data and training the models. Finally, we take among others the median over the particular results.

In the following, we explain our methodology in detail. The implementation is done in `R` using the `Keras` library, which is an API to `TensorFlow`, for the neural networks.

A. Bootstrapping and Data Splitting

The bootstrapping is in our case a Monte-Carlo-like approach of repeatedly and independently splitting the complete dataset $\underline{all}$ (6,497 rows, 13 columns) with a ratio of 70:30 into training data (4,547 rows, 13 columns) and test data (1950 rows, 13 columns) 100 times: $\underline{all} = \underline{train} \dot\cup \underline{test}$. To make this process reproducible, we set an initial seed and randomly draw 100 seeds $(seed_1, \ldots, seed_{100})$. Before every splitting we explicitly set the seed to the respective run's seed. The training data is further split with a ratio of 70:30 into development data (3,182 rows, 13 columns) and validation data (1,365 rows, 13 columns): $\underline{train} = \underline{dev} \dot\cup \underline{val}$.

B. Data Manipulation

Exemplarily, we manipulate the 5% worst ranked test data by averaging the original rating and the highest possible rating (10) and rounding up. That is, we split the test data $\underline{test} = \underline{low} \dot\cup \underline{high}$ with a ratio of 5:95 (with a random tie breaking), manipulate $\underline{low} \mapsto \underline{low_{manip}}$ and get the manipulated test data $\underline{manip} := \underline{low_{manip}} \dot\cup \underline{high}$. We also add a flag column to the manipulated test data for marking the manipulated entries for evaluation purposes.

C. Data Normalization

The independent features are all normalized by min-max-scaling where $\underline{train}$ serves as reference. That means, also the test datasets are normalized with the minimum and maximum values of $\underline{train}$. For LM and RNN, the target variable "quality" is not normalized. For BA and NNA, "quality" is an input variable like the others and, hence, normalized. To obtain comparable results, the performance of the regression models is normalized afterwards (using $\underline{train}$).

D. Models

We consider four different kinds of models: LM, RNN, BA, and NNA. The two simple models LM and BA are solely for benchmarking the general performance of the two corresponding (deep) neural network models on the unmanipulated test data $\underline{test}$. We measure the manipulation detection performance for the two (deep) neural network models only. LM uses `R`'s `lm` function. BA is a fully connected, three layer network with input layer (size 14), code layer (size 4), and output layer (size 14). The input is the 13 dimensional data plus a constant column of 1s (intercept) in order to mimic two nested linear regressions. Thus, linear activation functions and MSE are used.

E. Hyperparameter Tuning

In every step during the bootstrapping, the deep models are trained with hyperparameter optimization over a grid. How these grids are obtained is outlined in Section V. The hyperparameter grid for RNN is:

- Activation function (hidden layers): `linear`, `softplus`, `ReLU`
- Activation function (output layer): `linear`
- Number of hidden layers: 1, 3, 5, 7
- Dropout rate: 0%, 5%, 10%
- Number of neurons in each hidden layer: 32, 64, 128
- Number of neurons the input layer: 12
- Number of neurons the output layer: 1
- Batch size: 32, 64
- Learning rate: 5%, 10%
- Patience for early stopping: 15

- Patience for learning rate reduction: 7
- Loss function: MSE
- Evaluation measure: Mean Absolute Error (MAE)
- Optimizer: `Adam`
- Number of epochs: 75
- Batch normalization: between every layer

The grid for NNA is defined as follows:

- Activation function (hidden layers, except the code): `softplus`, `ReLU`
- Activation function (code layer and output layer): `linear`
- Number of hidden layers (except code layer): 4, 6
- Dropout rate: 0%
- Number of neurons in each hidden layer (except the code): 64, 128
- Number of neurons the input layer as well as in the output layer: 13
- Number of neurons the code layer: 4
- Batch size: 32, 64
- Learning rate: 5%, 10%
- Patience for early stopping: 15
- Patience for learning rate reduction: 7
- Loss function: MSE
- Evaluation measure: MAE
- Optimizer: `Adam`
- Number of epochs: 75
- Batch normalization: between every layer

F. The Algorithm

The bootstrapping, model training, and evaluation algorithm is depicted in the algorithm in Figure 1. All individual steps are described above. The algorithm is parallelized.

1: **begin**
2: **for** i=1 **to** n **do**
3: **begin**
4: Prepare datasets with $seed_i$ (split, manipulate, normalize);
5: Train the two benchmark models on $\underline{train}$;
6: Optimize RNN's and NNA's hyperparameters (train on $\underline{dev}$, validate on $\underline{val}$ using MAE as performance measure) and retrain the best model in each case on $\underline{train}$;
7: Measure all four models' performance on $\underline{test}$;
8: Measure RNN's and NNA's detection performance on $\underline{manip}$;
9: **end**
10: **end**

Figure 1. Procedure for model training and evaluation. Input: the original dataset; a seed vector $(seed_1, \ldots, seed_{100})$; two hyperparameter grids. Output: list of performance data.

As one can see from the algorithm, both approaches (RNN, NNA) are semi-supervised. We use labeled data to train the networks, but only data that is labeled as "correct," i.e., that is not manipulated. Although in the analysis, "correct" and "incorrect," i.e., manipulated, data are used, no incorrect data are used for training—that is, one does not need a data set where "incorrect" data are already identified as incorrect. We use the information about which data entries are really "incorrect" only for the statistical analysis of the results for this paper.

G. Detection Performance

The detection performance is measured as follows: For RNN, we calculate the squared difference of the predicted quality and the given quality (which is possibly manipulated) for each data object (Squared Error; SE). For NNA, we compute for all data objects the sum over all features of the squared differences between the predicted feature and the respective given (possibly manipulated) feature (Sum of Squared Errors; SSE). We sort the data in descending order according to these deviation values (once for RNN and once for NNA): $\underline{manip} \mapsto (\underline{manip_{reg}}, \underline{manip_{auto}})$. Then, we determine the true/false positive rates when marking the first $q_i\%$ of the data objects in the sorted sets $\underline{manip_{reg}}$ and $\underline{manip_{auto}}$ as suspicious for $q_i = i$, $i = 1, 2, \ldots, 99$. The True Positive Rate tpr is defined as $tpr = TP/(TP + FN) = 1 - fnr$ and the False Positive Rate fpr is $fpr = FP/(TN + FP) = 1 - tnr$, where TP is the number of True Positives, i.e., of manipulated objects that are marked suspicious, TN is the number of True Negatives, i.e., of unmanipulated objects that are not marked, and FP and FN are the respective False Positives/Negatives and fnr and tnr the respective Rates. If one would assign the "suspicious marks" randomly with equal probabilities to $q\%$ ($q \in [0, 100]$) of the data, the expected true/false positive rates would equal q, i.e., $\mathbb{E}[tpr] = \mathbb{E}[fpr] = q$, independent of the share of real positives/negatives. The values for $q = 0$ and $q = 100$ are meaningless since in the former case no object would be marked as suspicious and in the latter case all objects would be marked as suspicious. To summarize the results of all runs, we calculate all quartiles of tpr and fpr for every q_i, i.e., minimum, first quartile, median, third quartile, maximum. Before presenting the results of our analysis in Section VI, we describe how the set of possible hyperparameters is found.

V. Hyperparameters

Since basically the set of possible hyperparameters is infinite, it is quite natural that this set has to be shrunk. In doing so, we start with an initial set for possible hyperparameters and with an initial guess for a plausible setting (underlined). This is done based on comparisons to similar problem as well as extensive trail-and-error pre-tests.

The initial hyperparameter grid for RNN is:

- Activation function: `linear`, `softplus`, `ReLU`, `tanh`, `sigmoid`
- Number of hidden layers: 0, 1, 3, 5, 7
- Dropout rate: 0%, 5%, 10%
- Number of neurons in each hidden layer: 32, 64, 128
- Batch size: 32, 64
- Learning rate: 5%, 10%

The initial grid for NNA is:

- Activation function: `linear`, `softplus`, `ReLU`, `tanh`, `sigmoid`
- Number of hidden layers (excluding the code layer): 0, 2, 4, 6
- Dropout rate: 0, 0.05, 0.1
- Number of neurons in each hidden layers (except the code layer): 32, 64, 128
- Batch size: 32, 64
- Learning rate: 0.05, 0.1

All other parameters are fixed to the values of Section IV-E. Note that we intentionally did not include varying numbers of neurons for the code layer (in the autoencoder case). This is because higher numbers of neurons in the code lead to a higher performance, but a lower compression. Since both values are important for outlier detection, based on comparisons to similar examples, we chose four as a promising tradeoff.

Using the heuristic strategy of *sequential accumulative selection,* this set is further shrunk so that the hyperparameter optimization in the algorithm in Figure 1 (in Section IV) performs within a reasonable runtime. Next, we explain the sequential accumulative selection: We started with performing 50 runs with the hyperparameters fixed to the underlined, plausible values except for the activation function, which was allowed to be any of the given possibilities. All activation functions that were taken at least once in the hyperparameter optimization in the 50 runs were declared to be also plausible, all others were deleted. In the same fashion, next, the number of hidden layers was analyzed, i.e., the hyperparameter optimizer had to optimize over the set of the plausible activation functions (due to step one there is possibly more than one plausible activation function) and the number of hidden layers. All values for the hidden layers that were chosen at least once were declared to be also plausible, all others deleted. The plausible activation functions remain the same. This procedure is repeated in the following order with number of neurons, dropout rate, batch size, learning rate. The results of the sequential accumulative selection, i.e., of the diminution of the possible hyperparameters can be found in Section IV-E. For clear, the procedure of sequential accumulative selection is done separately for RNN and NNA.

VI. Results

Here, we do set neither an explicit threshold for the share of data objects to be marked as suspicious nor an explicit threshold for the SE resp. SSE beyond which the data objects have to be marked as suspicious since the aim of this work is not to find a classifier for manipulated wine data quality, but the comparison of the two neural network based models. How a threshold can be found is, e.g., outlined in [42]. To illustrate the detection performance of RNN and NNA, we calculate tpr and fpr for all Monte-Carlo-like runs and for all $q_i = 1, \ldots, 99$. For all q_i, we calculate the five quartiles of tpr and fpr for RNN and NNA and plot these values against q. The results are depicted in Figures 2 (tpr) and 3 (fpr). The respective quartiles of NNA are drawn solid and of RNN dashed.

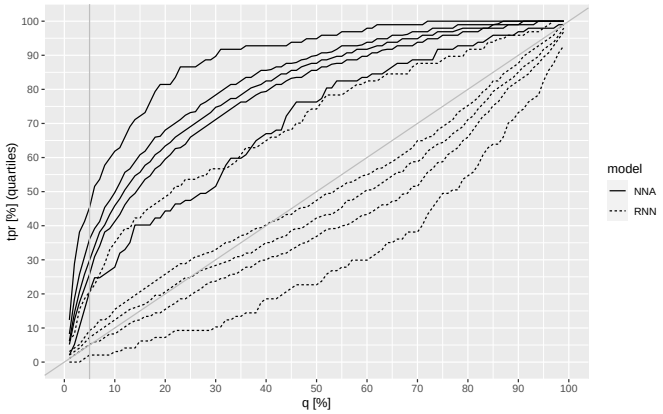

Figure 2. The five quartiles of tpr for varying q and RNN (dashed), NNA (solid), resp. Additionally, the diagonal and the 5% line are depicted. NNA outperforms RNN, which is often worse than random draws.

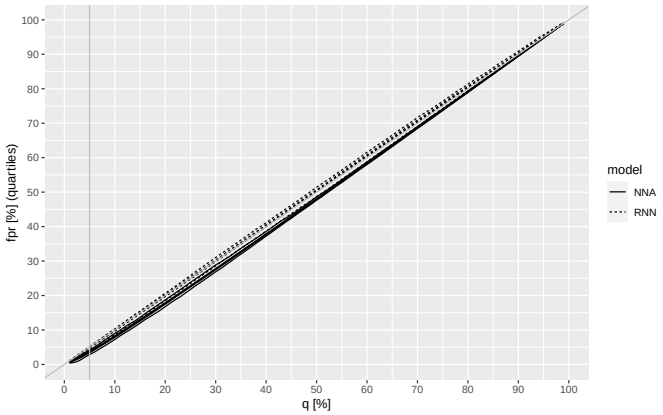

Figure 3. The five quartiles of fpr for varying q and RNN (dashed), NNA (solid), resp. Additionally, the diagonal and the 5% line are depicted. NNA outperforms RNN, which is often worse than random draws.

As we can easily observe, the autoencoder outperforms the regression network in most of the cases (despite the fact that the autoencoder does not use the information about the assumed dependency) concerning tpr and fpr. A further notable point is that often RNN is worse than randomly guessing (cf. the diagonals in the figures). Additionally, the average runtime of RNN was with ca. 2h14'06" much larger than those of NNA (ca. 15'40.8"). The performance of all four models on the unmanipulated test data is depicted in Figure 4. We see that NNA is best (in median), while autoencoders are better than regressions (in median). RNN is (in median) better than LM, however, its interquartile distance is the largest among all models, while those of LM is the smallest.

VII. Conclusion

We analyzed the ability of different (neural network based) methods for detecting manipulated wine ratings. It turns out that in our study autoencoders outperform regressions. It is very interesting that and worth further investigation why regressions are often worse than randomly guessing. Interest-

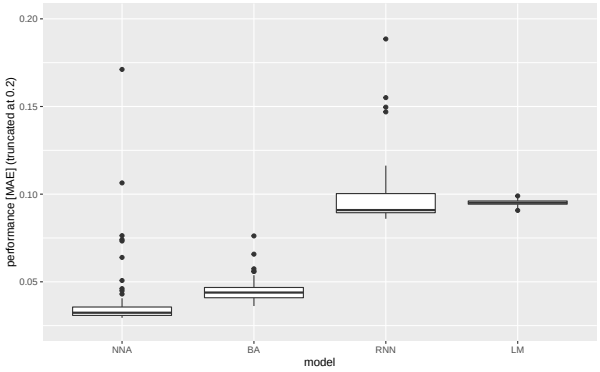

Figure 4. Boxplot of the performance (MAE) of NNA, BA, RNN, LM. There are some outliers that are not depicted. In median, NNA is best, whilst the interquartile distance is the smallest for LM.

ingly, even the very basic linear model based autoencoder (BA) outperforms regressions.

In detail, we analyzed four models for detecting manipulated wine ratings with one resp. two specific datasets, the vinho verde data. We find that a neural network based autoencoder performs best while a linear regression shows the smallest variability in the results. The linear model based autoencoder exhibits a good tradeoff between computation time and detection performance. We note that despite there is a lot of literature concerning regressions from measurable data to wine quality, such regressions do not work well in our case for semi-supervised manipulation identification. Presumably, this is because regressions are basically used for supervised and not for semi-supervised methods. Further, we established the procedure of sequential accumulative selection for finding appropriate hyperparameters.

VIII. Future Work

In this paper, we assumed that it is reasonable that manipulations are applied to low rated wines to make them appear better to increase sales numbers. It would be interesting to test our approach also on other manipulation strategies, including, e.g., intentional and unjustified down ratings. Future work could also deal with the detection of faked ratings when there are multiple ratings per product as it is typical for many online stores or rating portals. Are there ways to detect the faked/manipulated ratings (whether better or worse) when there are many ratings for the same product? In this context, many stores and portals offer the possibility to write a review in addition to the plain rating. The processing of such information (via Natural Language Processing; NLP) is likely to be useful here.

Of course, other application areas apart from wine can be investigated with our approach, for example, ratings for products in online stores, restaurants, hotels. The detection of fraud in telecommunication, insurance, etc. [43] is also closely related. It could be of interest to identify the similarities and differences between these applications and how they should be addressed. When analyzing wine ratings, in addition to extending our approach to other, larger datasets with more features, such as countries, producing regions, price segments, etc., it is also worthwhile to apply other models, e.g., SVMs [14], and compare the results to the neural network based models. Further, an extensive comparison with other methodologies concerning the topic of manipulation detection for wine ratings could be done in future work.

The procedure of *sequential accumulative selection* (as explained in Section V) can be further analyzed. One might investigate whether and how the order of the features is important. Comparisons to other hyperparameter selection methods are also possible (cf. [26]). Last but not least, it should be noted that the topic of explainable AI and responsible AI is rapidly growing in importance [44]. For example, one can ask how to explain which data sets are marked as suspicious. As few as possible false positives are to be marked, whereas all manipulated ones are to be recognized if possible. So how can the decisions of the recognition algorithms be (understandably) explained?

Acknowledgment

Michaela Baumann is with NÜRNBERGER Versicherung, Germany. The opinions expressed here are her own and not necessarily those of her employer.

The authors thank Lars Grüne and Bernhard Herz, both with University of Bayreuth, Germany, as well as Claire Pfeil and Heinrich Fritzlar, both with NÜRNBERGER Versicherung, Germany.

References

[1] G. Rosso, Italian Wines 2021 *(English Edition),* Gambero Rosso, 2021
[2] R. Parker, The Wine Advocate, https://www.robertparker.com/articles/the-wine-advocate, accessed: 2022-02-02
[3] Gault&Millau, https://www.gaultmillau.com/, accessed: 2022-02-02
[4] Guide Michelin, https://guide.michelin.com/en, accessed: 2022-02-02
[5] D. Freedman, R. Pisani, and R. Purves, Statistics, 4th ed., W. W. Norton & Company, Inc., New York, London, 2007, Chapters 10-12
[6] E. Gelenbe, Z. H. Mao, and Y. D. Li, "Function approximation with spiked random networks," in IEEE Transactions on Neural Networks, vol. 10, no. 1, 1999, pp. 3-9
[7] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," in Neural Computataion, vol. 1, no. 4, 1989, pp. 502-510
[8] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review," in International Journal of Automation and Computing, vol. 14, no. 5, 2017, 503-519
[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," in nature, vol. 521, no. 7553, 2015, pp. 436-444
[10] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier Detection Using Replicator Neural Networks," Data Warehousing and Knowledge Discovery, 2002, pp. 170-180
[11] M. Sakurada and T. Yairi, "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction," Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, 2014, pp. 4-11
[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," in science, vol. 313, no. 5786, 2006, pp. 504-507
[13] J. D. Kelleher, Deep learning, MIT press, 2019
[14] I. Steinwart and A. Christmann, Support Vector Machines, Springer, 2008
[15] D. L. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality," in AMS math challenges lecture, 2000

Copyright (c) IARIA, 2022. ISBN: 978-1-61208-972-0

[16] J. W. Tukey, “The future of data analysis,” in The Annals of Mathematical Statistics, vol. 33, no. 1, Institute of Mathematical Statistics, 1962, pp. 1-67
[17] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, “Modeling wine preferences by data mining from physicochemical properties,” in Decision Support Systems, vol. 47, no. 4, 2009, pp. 547-553
[18] P. Cortez et al., “Using data mining for wine quality assessment,” in International Conference on Discovery Science, Springer, Berlin, Heidelberg, 2009, pp. 66-79
[19] Y. Gupta, “Selection of important features and predicting wine quality using machine learning techniques,” in Procedia Computer Science, vol. 125, 2018, pp. 305-312
[20] À. Nebot, F. Mugica, and A. Escobet, “Modeling wine preferences from physicochemical properties using fuzzy techniques,” in SIMULTECH, 2015, pp. 501-507
[21] S. Kumar, Y. Kraeva, R. Kraleva, and M. Zymbler, “A deep neural network approach to predict the wine taste preferences,” in Intelligent Computing in Engineering, Springer, Singapore, 2020, pp. 1165-1173
[22] P. Abbal, J. M. Sablayrolles, E. Matzner-Lober, and A. Carbonneau, “A model for predicting wine quality in a rhône valley vineyard,” in Agronomy Journal, vol. 111, no. 2, 2019, 545-554
[23] O. Ashenfelter, “Predicting the quality and prices of Bordeaux wine,” in The Economic Journal, vol. 118, no. 529, 2008, F174-F184
[24] O. Ashenfelter, “Predicting the quality and prices of Bordeaux wine,” in Journal of Wine Economics, vol. 5, no. 1, 2010, 40-52
[25] R. Schwarz, “Predicting wine quality from terrain characteristics with regression trees,” in Cybergeo: European Journal of Geography, 1997
[26] T. H. Y. Chiu, C. Wu, and C. H. Chen, A generalized wine quality prediction framework by evolutionary algorithms,” in International Journal of Interactive Multimedia & Artificial Intelligence, vol. 6, no. 7, 2021, pp. 60-70
[27] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection: A Survey”, ACM Comput. Surv., vol. 41, no. 3, 2009, article no. 15, pp. 1-15
[28] V. Hodge and J. Austin, “A Survey of Outlier Detection Methodologies.” Artificial Intelligence Review, vol. 22, 2004, pp. 85-126
[29] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Network Anomaly Detection: Methods, Systems and Tools,” in IEEE Communications Surveys & Tutorials, vol. 16, no. 1, 2014, pp. 303-336
[30] A. Patcha and J.-M. Park, “An overview of anomaly detection techniques: Existing solutions and latest technological trends,” in Computer Networks, vol. 51, no. 12, 2007, pp. 3448-3470
[31] R. Chalapathy and S. Chawla, “Deep Learning for Anomaly Detection: A Survey,” preprint on arXiv, https://arxiv.org/abs/1901.03407, 2019
[32] C. C. Noble and D. J. Cook, “Graph-Based Anomaly Detection,” Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 631-636
[33] Wine Quality Datasets, Universidade do Minho, http://www3.dsi.uminho.pt/pcortez/wine/, accessed: 2022-02-02
[34] kaggle (Piyush Agnihotri), White Wine Quality—Simple and clean practice dataset for regression or classification modelling, https://www.kaggle.com/piyushagni5/white-wine-quality, accessed: 2022-01-19
[35] kaggle (UCI Machine Learning), Red Wine Quality—Simple and clean practice dataset for regression or classification modelling, https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009, accessed: 2022-01-17
[36] Open Data Commons—Legal tools for Open Data, Database Contents License (DbCL) v1.0, https://opendatacommons.org/licenses/dbcl/1-0/, accessed: 2022-01-19
[37] T. Shin, “Predicting Wine Quality with Several Classification Techniques” towards data science, 2020, https://towardsdatascience.com/predicting-wine-quality-with-several-classification-techniques-179038ea6434, accessed: 2022-02-02
[38] D. Nguyen, “Red Wine Quality Prediction Using Regression Modeling and Machine Learning,” Towards Data Science, 2020, https://towardsdatascience.com/red-wine-quality-prediction-using-regression-modeling-and-machine-learning-7a3e2c3e1f46, accessed: 2022-02-02
[39] F. Rodríguez Mir, “Red Wine Quality,” Data UAB, 2019, https://datauab.github.io/red_wine_quality/, accessed: 2022-02-02
[40] *unknown* “Wine Quality Prediction,” cppsecrets.com, 2021, https://cppsecrets.com/users/10126100104105114971061121141111061019964103109971051084699111109/WINE-QUALITY-PREDICTION.php, accessed: 2022-02-02
[41] D. Alekseeva, “Red and White Wine Quality,” RPubs, https://rpubs.com/Daria/57835, accessed: 2022-02-02
[42] N. Japkowicz, C. Myers, and M. Gluck, “A Novelty Detection Approach to Classification,” IJCAI, vol. 1, 1995, pp. 518-523
[43] M. Baumann, “Improving a rule-based fraud detection system with classification based on association rule mining,” INFORMATIK 2021, 2021, pp. 1121-1134
[44] M. Baumann, “Data science challenge 2021: explainable machine learning,” https://github.com/DeutscheAktuarvereinigung/Data-Science-Challenge2021_Explainable-Machine-Learning, accessed: 2022-02-04

Copyright (c) IARIA, 2022. ISBN: 978-1-61208-972-0