Autoencoder vs. Regression Neural Networks for Detecting Manipulated Wine Ratings

Michaela Baumann
BI/Analytics Competence Center
NÜRNBERGER Versicherung
Nürnberg, Germany
`michaela.baumann@nuernberger.de`

Michael Heinrich Baumann
Department of Mathematics
University of Bayreuth
Bayreuth, Germany
`michael.baumann@uni-bayreuth.de`

The 17th International Multi-Conference on Computing in the Global Information Technology, ICCGI 2022, Venice, Italy

Presenter Resume

Michaela Baumann studied mathematics, statistics resp. economics at LMU Munich (Bachelor's degree) and at University of Bayreuth (Master's degree) and received a doctoral degree (Dr. rer. nat.) in computer science at University of Bayreuth, Germany.

Currently, she is working as a Data Scientist/AI Specialist at a German insurance company. She is interested in the application of AI and machine learning in the insurance industry, especially fraud detection, and is also a member of several economics-related research groups.

The opinions expressed here are her own and not necessarily those of her employer.

Research Question

- We analyze the ability of different detection methods to identify manipulated wine ratings
- We consider regression models and autoencoders
- The detection ability is measured through true/false positive rates
- The hyperparameters for the neural network based models are tuned via sequential accumulative selection

Overview

- Introduction
- Related Work
- The Data
- Methodology
- Results
- Future Work

Introduction

- There are prestigious rating authorities concerning wines, hotels, or restaurants, such as Gambero Rosso's *Vini d'Italia*, Robert Parker's *The Wine Advocate*, *Gault&Millau*, or *Guide Michelin*
  - $\rightarrow$ Publicly approved
  - $\rightarrow$ Check for genuineness is possible
- By far, not all wines are represented and rated by the above or related authorities; countless other ratings exist
  - $\rightarrow$ Verification of authenticity is difficult
  - $\rightarrow$ Objectivity can be dubious (ratings may even be paid for)

Related Work

Analytics of Wine Quality

- *P. Cortez et al., Modeling wine preferences by data mining from physicochemical properties (2009), Using data mining for wine quality assessment (2009)*
  Wine preferences are predicted with several data mining approaches using measurable wine features
- *Y. Gupta, Selection of important features and predicting wine quality using machine learning techniques (2018)*
  The most relevant features for machine learning models predicting wine quality are selected
- *À. Nebot et al. Modeling wine preferences from physicochemical properties using fuzzy techniques (2015)*
  Fuzzy inductive reasoning is used for classifying wine preferences
- *S. Kumar et al. A deep neural network approach to predict the wine taste preferences (2020)*
  Deep neural networks are applied for classifying wine ratings

Related Work

Anomaly Detection and Fraud Identification

- *V. Chandola et al., Anomaly Detection: A Survey (2009)*
  "Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior."
- *V. Hodge and J. Austin, A Survey of Outlier Detection Methodologies (2004)*
  Detection approaches usually fall into one of these categories
  - → Unsupervised methods, e.g., clustering
  - → Supervised methods, e.g., classification
  - → Semi-supervised methods, e.g., autoencoders
- *M. H. Bhuyan et al., Network Anomaly Detection: Methods, Systems and Tools (2014)*
  Apart from tabular data, there are also methods working on highly connected data represented through graphs

The Data

- Our approach is applicable to various working areas; we use wine data as an example
  - → Good data availability
  - → Data innocuousness
  - → Metric, clearly defined and exactly measurable explaining variables
  - → Unique target feature
- "Wine Quality Datasets" containing vinho verde wines from Universidade do Minho (by P. Cortez et al.)
  - → 1,599 entries (red wines) and 4,898 entries (white wines), in total 6,497 entries
  - → 13 columns, among them "quality," the target feature, and a categorical feature for the wine type (red/white)
  - → rating ("quality"): from 0 ("very bad") to 10 ("excellent")

Methodology

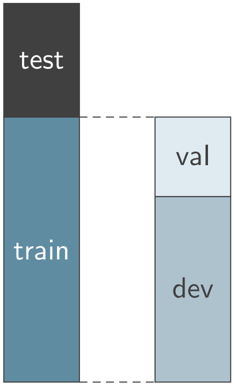
General Approach

- We train several network based models on correctly labeled data
- We make predictions on unseen data, where a certain part is manipulated (feature “quality” is not correct)
- We compare the predicted quality with the provided quality
- Hypothesis: Data objects where the predicted value differs strongly from the provided one are more likely to be manipulated
- To prevent overfitting we apply bootstrapping

Methodology

Boostrapping and Data Splitting

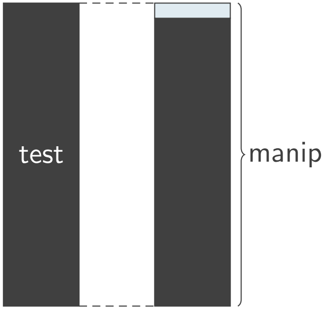
- Bootstrapping is, in our case, a Monte-Carlo-like approach of repeatedly and independently data splitting and model training (100 runs)
- Data splitting:
  - $\rightarrow$ Train-test-split: 70:30
  - $\rightarrow$ Dev-val-split (of the training data, needed for hyperparameter optimization): 70:30
- To allow for reproducibility, we initially draw a vector of seeds, one seed for each bootstrap run



Methodology

Data Manipulation

- We manipulate the 5% worst ranked test data
  - $\rightarrow$ test $\rightarrow$ manip
  - $\rightarrow$ Manipulation: Averaging between the original rating and the highest possible (10)
- The manipulated data objects are flagged for the later evaluation



Methodology

Data Normalization

- The independent features are normalized by min-max-scaling
  - $\rightarrow$ For the regression models, the target “quality” is not normalized (dependent variable)
  - $\rightarrow$ For the autoencoder models, there is no target and “quality” is normalized like all the other features
  - $\rightarrow$ To obtain comparable results, the performance of the regression models is normalized afterwards
- The train set serves as reference for normalization

Methodology

Models

- We consider two simple models as benchmark
  - → Linear regression (LM)
  - → Flat autoencoder (neural network with one hidden layer and linear activation) (BA)
- With the simple models we set benchmarks for the general performance measured for all four models on the test data

- We also consider two (deep) neural network models
  - → Regression neural network (RNN)
  - → Deep autoencoder (mimicking two nested regressions) (NNA)
- With the deep models, we measure the detection performance on the manipulated test data

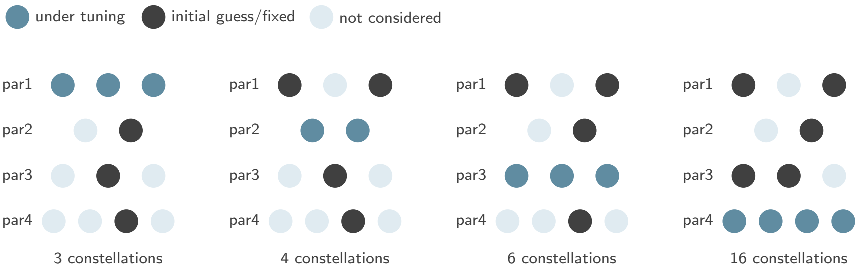
Methodology

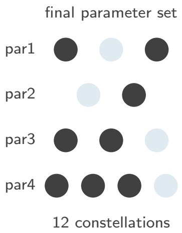
Hyperparameter Tuning

- For every bootstrap run, the deep models are trained with hyperparameter optimization over two respective grids
- The grids are obtained through sequential accumulative selection
  - → We start with an initial, quite large set of possible hyperparameters and initial guesses for plausible parameter values
  - → We fix all parameters to the initial guesses except the already processed ones and the one under tuning
  - → We perform 50 model training runs on the dev set over all constellations and consider those parameters further on, that were picked at least once after evaluation on val
  - → We iterate over all hyperparameters until a final (smaller) set for the actual hyperparameter optimization is found

Methodology

Hyperparameter Tuning – Illustrative Example





Here, we need to consider 41 hyperparameter constellations in total (29 for the sequential accumulative selection and 12 for the final hyperparameter optimization) instead of 72 when trying all possible combinations from the initial set

Methodology

Detection Performance

- RNN: We calculate the squared difference of the predicted quality and the given (possibly manipulated) quality for each data object
- NNA: We compute the sum over all features of the squared differences between the predicted and the given (possibly manipulated) features for each data object
- Separately for RNN and NNA
  - $\rightarrow$ We sort the data in descending order according to the deviations
  - $\rightarrow$ For $q_i, i = 1, \ldots, 99$, we mark the first q_i% of the data objects as suspicious
  - $\rightarrow$ We compute the true positive rate ($tpr = TP/(TP + FN)$) and false positive rate ($fpr = FP/(FP + TN)$) for all q_i
  - $\rightarrow$ To summarize the results of all bootstrap runs, we calculate the quartiles of tpr and fpr for every q_i

Methodology

Detection Performance – Illustrative Example

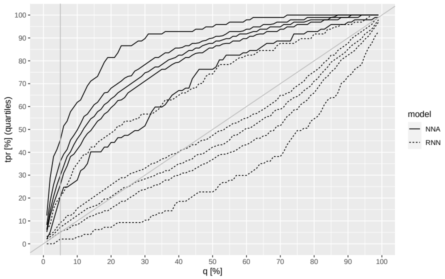
wine	diff.	manip.
wine1	0.99	1
wine23	0.95	0
wine456	0.92	1
wine57	0.90	0
wine3454	0.89	1
wine345	0.86	0
wine99	0.81	0
⋮	⋮	⋮

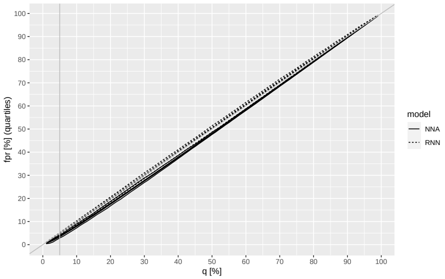
wine	diff.	manip.
wine1	0.99	1
wine23	0.95	0
wine456	0.92	1
wine57	0.90	0
wine3454	0.89	1
wine345	0.86	0
wine99	0.81	0
⋮	⋮	⋮

- Assume we have 10 manipulated wines
- q_1 (first three rows): $tpr = 2/10 = 20\%$
- q_2 (first six rows): $tpr = 3/10 = 30\%$
- ...

Results

Detection Performance

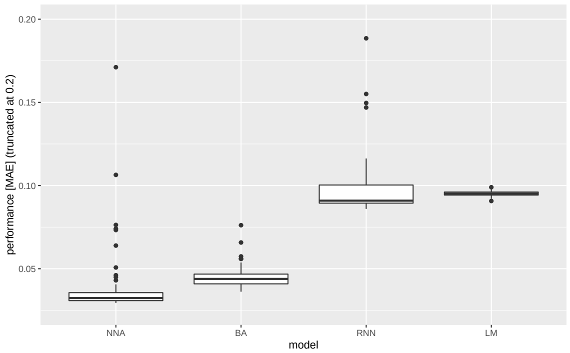




- NNA outperforms RNN in most of the cases
- Often, RNN is worse than randomly guessing
- Runtime of RNN was ca. 2h14'06" and thus much larger than those of NNA (ca. 15'40.8")

Results

General Prediction Performance (truncated)



- NNA is best (in median)
- Autoencoders are better than regressions. Even the simple BA performs well (best compromise?)
- RNN is better (in median) than LM, but has a large interquartile distance
- LM is most stable model

Future Work

- Test the approach on other manipulation strategies
- Identify faked ratings when there are multiple ratings per product
- Apply other models, e.g., SVMs, and compare the results
- Consider other hyperparameter optimization approaches
- Explain the results with XAI methods

Thank you for your attention!

If you have any comments or questions,
don't hesitate to contact me