Predicting purchasing intent of website visitors with deep feature learning

Jelmer van Hoek
University of Twente
PO Box 217, 7500 AE Enschede
the Netherlands
j.m.vanhoek@student.utwente.nl

ABSTRACT

In E-commerce, there is a growing interest in understanding website visitors. Being able to accurately predict when a visitor has an intention to make a purchase or not can be very valuable for e-commerce businesses. Several studies have aimed to make these predictions based on machine learning algorithms. Different datasets were used to test these methods, often consisting of data about a particular website or behavior of visitors. However, most of these methods only use raw data to predict the buying behavior of visitors. It is not yet clear if that is the most effective way to make these predictions. The objective of this paper is to test this issue by using deep feature learning to predict the purchasing intent of website visitors, based on website data, clickstream data and session data. Results of four different autoencoders in combination with a SVM/WSVM classifier is compared to a SVM/WSVM that only uses raw data. Of all the autoencoders, the deep autoencoder shows the best results with an average accuracy of 0,61 and an average TPR of 0,71. However, traditional methods used on this dataset like a WSVM with raw data and decision trees significantly outperform the autoencoders. Nevertheless, the use of autoencoders in the prediction of buying behavior can potentially become a lot more effective when improvements are made to the configuration of the autoencoders.

Keywords

E-commerce, machine learning, deep feature learning, buying behavior, purchasing intent, SVM, autoencoders.

1. INTRODUCTION

E-commerce is growing faster than ever. Especially with the recent global pandemic, more and more businesses are transitioning into a more online-focused way of doing business, because e-commerce is becoming the primary place for customers to buy their favorite products.

At the same time, there is also a shift happening in the way that businesses advertise their products or services. The traditional way of advertising, such as via newspapers, television, radio and billboards has undergone and is still undergoing a rapid transition into online advertising. Today, it is almost impossible to surf the web or use social media without seeing numerous ads, often tailored to your interests and your previous activities online [1].

Compared to traditional physical stores, the amount of competition is much higher in the online world. Store owners no longer need to have a physical location to sell their products, so the investment costs for online stores are lower, resulting in low barriers to entry. Also, for customers, it is much easier to find the products or services they need. With the help of search engines and comparison sites, potential customers can easily narrow down their search and find products tailored to more specific needs, giving smaller niche stores the opportunity to reach their audience more easily.

As a consequence of this transition, a business needs to understand these customers better. Businesses are eager to know what the intention of a visit on their webshop is. For example, is a visitor just browsing or intending to buy a product?

This is relevant information for businesses because, with this information, they will be able to target their advertising more efficiently, offer targeted discounts, or just learn about what aspects of their webshop are generating most of the revenue [2].

On online platforms, it is possible to collect much more information about visitors compared to visitors in physical stores. Especially clickstream data and session data (clicking behavior, visited pages, product views, conversions, etc.) gives highly detailed information about all website visitors. This allows businesses to understand all visitors and not only the visitors that end up making a purchase [3]. However, this information is very detailed and it is often hard to derive meaningful conclusions out of this data without a thorough analysis. Machine learning methods can provide a solution to this problem.

Several studies have aimed to achieve this goal by analyzing visitor data, clickstream data and website data to determine behavioral patterns that can predict the event of making a purchase on an e-commerce website.

In existing works, described in further detail in section 2, a lot of attention has been paid to traditional machine learning classification methods(Random forest, Hidden Markov Model, Decision trees) but there is not yet a comprehensive quantitative analysis of the potential of unsupervised deep feature learning in combination with a support vector machine for the prediction of online buying behavior based on clickstream data.

This paper aims to contribute to the existing work by providing a comprehensive analysis and comparison between the performance of a traditional machine learning classifier, in

33th Twente Student Conference on IT, July 3rd, 2020, Enschede, The Netherlands.

particular the Support Vector Machine (SVM), that is provided with raw clickstream data and the performance of a SVM in combination with Autoencoders.

The objective of this research is to predict the purchasing intention of website visitors with the use of unsupervised deep feature learning, specifically Autoencoders, to learn better representations of buying behaviors. By testing the performance of autoencoders with this dataset, it will become clearer what the impact is of deep feature learning on the prediction performance of buying behavior.

The rest of the paper is organized as follows. In section 2, the research questions are presented. In section 3, several related works are given to give a clear representation of the state of the art. In section 4, necessary background information is given to help with understanding the domain of this research. In section 5 the research methodology is described. In Section 6, the results of the research are presented and following are the conclusions in section 7.

2. RESEARCH QUESTION

How effective is the use of deep feature learning algorithms for prediction of buying behavior based on clickstream data, session data and website data?

2.1 Sub-questions

1. What is the performance of deep feature learning algorithms on this dataset?
2. How does this performance compare to machine learning methods that only use raw data?

3. RELATED WORK

[4] used a Hidden Markov model in combination with browsing data to predict purchasing intent. However, their data consisted of all the data that someone downloaded on a website, including pictures and sounds, which makes the dataset completely different than the one used in this research. Their method had a low precision (0,51) and not a very outstanding Recall (0,73).

[5] aimed to use more than one data source and next to clickstream and session data also used touch-interactive behavior to analyze behavioral patterns with the use of Deep Intent Prediction Network (DIPN). The goal of this study was to predict future buying behavior instead of predicting buying behavior in the current session.

Also, [6] aimed to predict buying behavior in real-time with the use of long short-term memory(LSTM) recurrent neural network (RNN) to process clickstream data, session data and website data. He succeeded in predicting the probability that the user will leave the site within a given time period with greater accuracy compared to more common machine learning methods.

[7] used Deep Belief Networks(DBN) and Auto-encoders to predict when a user has a high probability of buying a product on an e-commerce website, based on page views of products, basket-views, buy events, ad clicks and ad views.

[8] Used Naïve Bayes and Decision trees to predict the user's interest in E-commerce websites. It is not clear what kind of data they specifically using data from the browser cache of users.

In table 1, an overview of all the related work and their corresponding methods is presented.

Table 1. Overview of mentioned papers with methods used

Paper	Research	Methods
Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks	Real-time prediction of probability that the user will leave the website without making a purchase.	LSTM-RNN, MLP, SVM, Decision trees
Prediction of the Intention of Purchase of the User Surfing on the Web Using Hidden Markov Model	Real-time prediction of the visitors buying intention	Hidden Markov Model
Buying or browsing?: Predicting Real-Time Purchasing Intent using Attention-based Deep Network with Multiple Behavior	Real-time prediction of a user's subsequent purchasing behavior within a given time	DIPN with multi-task learning
Prediction of User's Purchase Intention Based on Machine Learning	Prediction of user's interest in websites	Naïve Bayes, Decision trees
Predicting online user behavior using deep learning algorithms	Prediction of high probability that a user will make a purchase on an ecommerce website	Deep Belief Network, Auto-encoders

4. BACKGROUND

4.1 Dataset

The dataset used in this paper comes from the UCI machine learning repository website [9]. It consists of 12,330 user sessions with each session having a total of 17 attributes. Each session belongs to a unique user in one year, to make sure there is no bias towards a particular campaign, period, user, or special day. A description of the attributes in the dataset is given in table 2. The ranking of the attributes is taken from the results of the filter-based feature selection done by [6]. This ranking will later be used to test the performance of the SVC on raw data.

Table 2. Description of the dataset.

Attribute	Description	Type	Ranking[6]
Administrative	Number of administrative pages visited the user (e.g. profile page or account management)	Numerical	6
Administrative Duration	Amount of time spent by the user on administrative pages	Numerical	9
Informational	Number of informational pages visited by the user (e.g. contact page, "about us" page)	Numerical	8
Informational duration	Amount of time spent by the user on informative pages	Numerical	13
Product related	Number of product-related pages visited by the user (e.g. browsing products or individual product pages)	Numerical	3
Product-related duration	Amount of time spent by the user on product-related pages	Numerical	4
Bounce rate	Google Analytics metric. Refers to the percentage of visitors that enter the website but leave without any interaction with the website. The value in this dataset is the average bounce rate of all pages visited by the user	Numerical	5
Exit rate	Google Analytics metric. Refers to the percentage of page views of a specific web page that was followed by the visitor leaving the website. The value in this dataset is the average exit rate of all pages visited by the user	Numerical	2
Page value	Google Analytics metric. This refers to the average value for a web page that a user visited before making a purchase. The value in this dataset is the average page value of all pages visited by the user.	Numerical	1
Special day	Indicates closeness to a special day (e.g. Valentine's Day or Christmas). This value increases based on how close a purchase is to a special day	Numerical	10
Operating System	The Operating system the visitor uses	Categorical	14
Browser	The Browser software the visitor uses	Categorical	17
Region	The Region from which the visitor visits the website	Categorical	16
Traffic Type	The source from which the user enters the website (e.g. Google Ads, Facebook, Instagram, organic)	Categorical	12
Visitor Type	Indicates whether the visitor is a new visitor or returning visitor	Categorical	7
Weekend	Indicates whether the visit is made in the weekend	Categorical	15
Month	Month that the visit was made	Categorical	11
Revenue	Class variable: True or False based on whether the visitor has made a purchase on the website or not	Class	

4.2 Support Vector Machine

The main goal of a SVM is to classify data into two or more different classes. A SVM is a supervised machine learning algorithm that is commonly used for these kinds of classification problems, where the algorithm tries to separate the data into two categories [10]. Also, the SVM has applications in cybersecurity, image processing, biometrics and applied statistics [11]. The basic idea behind the algorithm is that it looks for a hyperplane, which serves as a decision boundary, that most accurately separates the classes in a dataset, in this case, the buyers and the non-buyers. the SVM uses a kernel function to find these hyperplanes in higher dimensional spaces, where the data is more easily separable.

4.3 Autoencoders

In this paper, autoencoders have been used for deep feature learning. The goal of an autoencoder is to learn features about data or images. It can be used to denoise datasets and re-build a clearer representation of the data, or it can be used as a feature-extractor, to improve the results of a classifier. The latter option is the purpose of the autoencoder in this paper. An autoencoder is a neural network that attempts to compress its input data into a lower dimension. Then, it tries to re-create the input based on the representation of the lower dimension data. It is trained to minimize the difference between the re-creation and the original input. Therefore, the autoencoder has to learn the most essential features about the dataset, leaving out the unnecessary or redundant ones, that show little to no relation to the class variable [12]. The most common real-world applications of autoencoders today, are data/image denoising and dimensionality reduction for data visualization [13].

In this paper, different implementations of autoencoders are used to compare the performance on this particular dataset: Deep autoencoders, Sparse autoencoders, 2Dconvolutional autoencoders and 1Dconvolutional autoencoders.

4.4 Tools

All programming was done in *Python 3.8*. the *Jupyter notebook* environment helps with keeping the flow of the code structured and makes it easy to share the progress with supervisors. It is also useful for re-running specific code blocks when that is necessary. packages that were used are *scikit-learn, matplotlib, Keras, pandas* and *numpy.*

Scikit-learn was used for the implementation of the SVM and the hyper-parameter tuning. *matplotlib* was used for making scatter plots to give a visual representation of the dataset during the programming process. *Keras* was used for everything related to the implementation of autoencoders. *Pandas and numpy* were used for the pre-processing of the dataset. Finally, the results of the classifier were exported to *Microsoft Excel*, for the creation of the graphs.

5. METHOD

5.1 Data pre-processing

First, the dataset is loaded into 2 pandas data frames, separating the class variable "Revenue" from the attribute variables. To be able to experiment with different amounts of features, a dictionary is created to create 10 additional data frames, where for each additional data frame, an attribute(column) is dropped, starting from 10 attributes. So, after these operations, there are 12 different data frames: One for the class variable, one for all the 17 attributes and the additional 10 data frames with a decreasing number of attributes, ranging from 10 to 1. The selection of the attributes that were dropped is based on the ranking in Table 2.

After the data frames are in place, the data is normalized using a MinMaxScaler from *Scikit-learn*. The is done so that attributes with different ranges become comparable, by mapping all the features to a range of (0,1) (include reference that this is important) The formula that it uses is as follows:

$x\text{i} = (x\text{i} - \min)/(\max-\min)$

Where xi refers to a single data point in the dataset and min/max the minimum and maximum value for the attribute that this data point belongs to.

5.2 Training data and testing data

After normalizing, the data is split into a training set and a test set, using the *train_test_split* function. The reason for this is simple: Training data is needed to train the autoencoders and fit the classifier and testing data is necessary to evaluate and compare the performance of the algorithms. the test size parameter is set to 0.2 Also, a fixed random seed is used for the *random_state* parameter: 32342. This is to make sure that every algorithm is trained on the same training data and tested on the same testing data.

5.3 SVC implementation

For the classification of non-buyers and buyers, the *SVC* implementation from *scikit-learn* was used The SVC was chosen because it implements the optional *class_weight* parameter, which allows the modification of the regularization parameter C based on the class variable. Given that the dataset is unbalanced since 84,5% of all sessions have a negative class variable, setting the *class_weight* parameter to "balanced" could turn out to give better results compared to the default implementation. Therefore, both the default SVC and the SVC with balanced class weight are tested to compare results. In the rest of the paper, the latter version is referred to as the WSVC.

GridSearchCV was used to find the best performing parameters for each time the SVC makes a prediction based on its input data. It optimizes the parameters based on the C variable and the kernel choice: either RBF or SIGMOID.

5.4 Autoencoders implementation

For all the autoencoders, the implementations are based on the default implementation of *Keras* [13]. The following parameters were used: Mean Squared Error (MSE) as loss function and the ADADELTA was used as an optimizer. The network is trained with the training data with all 17 attributes. A batch size of 3000 is used to minimize execution time and the autoencoders run on 50 epochs to achieve stable train/test loss values. The network consists of 5 hidden layers, where the last layer, also known as the "bottleneck" layer is used for training the classifier. The data in this layer is re-normalized with the MinMaxScaler, otherwise, some values in this layer are higher than 1, which is not optimal for the SVC classifier.

The complete implementation used for the autoencoders as well as the SVC/WSVC can be found on my GitHub page [14].

5.5 Metrics

For the evaluation of the results, the following variables are important:

- True Positives (TP): the amount of correctly predicted buyers.
- True Negatives (TN): the amount of correctly predicted non-buyers.
- False Positives (FP): the amount of incorrectly predicted buyers.
- False Negatives (FN): the amount of incorrectly predicted non-buyers.

The following metrics are going to be used to measure and compare the results of the classifier:

- Accuracy: rate of correctly predicted data points. Calculated by: **(TP+TN)/(TP+TN+FP+FN)**
- True-Positive rate: rate of correctly predicted buyers. Calculated by **TP/(TP+FN)**
- True-Negative rate: rate of correctly predicted non-buyers. Calculated by: **TN/(TN+FP)**
- False-Positive rate: rate of incorrectly predicted buyers. Calculated by **FP/(FP+TN)**
- False-Negative rate: rate of incorrectly predicted non-buyers. Calculated by **FN/(FN+TP)**

6. RESULTS

Every machine learning method used in this paper: SVC with raw data, WSCV with raw data, Deep autoencoder with SVC, Sparse autoencoder with SVC and convolutional autoencoder(2D and 1D) were run with two different implementations of SVC: default and weighted. The weighted SVC(WSVC) is different from the default version because its class weight parameter is set to 'balanced'.

Every training and testing run is repeated I times. Where I refers to the length of the set of feature amounts. {10,9,8,7,6,5,4,3,2,1} that the SVC is given as input.

Below you will find the graphs created in *Microsoft Excel*.

FPR and FNR are the direct opposites of TNR and TPR, respectively. Therefore, only the Accuracy, TPR and TNR are shown in the graphs and the table to maintain readability.

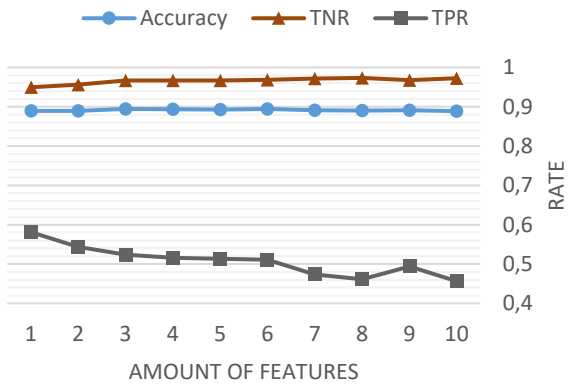


Figure 1. Performance of SVC with raw data. Amount of features (1,10)

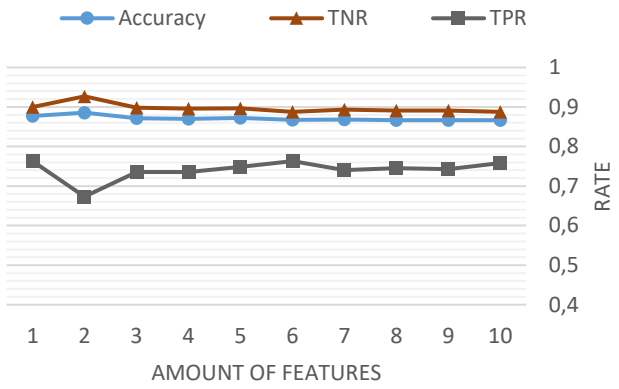


Figure 2. Performance of WSVC with raw data. Amount of features (1,10)

6.1 Single SVC/WSCV

In figures 1 and 2, It is clear that the WSVC shows superior results over the default SVC. Especially when you look at the difference in TPR. with an average TPR of 0,74 over 0,5 respectively, the default SVC underperforms on this level. The default SVC treats the negative class samples(non-buyers) with the same weight as the positive class samples(buyers), which makes the classifier less able to separate the buyers from the non-buyers. The WSVC puts more weight on the positive class samples, thus compensating for the imbalanced dataset. However, the increase in TP is at the cost of a small decrease in trade-off happening here with the accuracy of the classifier. A higher TPR is obtained at the cost of a lower TNR. At first glance, the overall result looks less impressive compared to the results of the WSVC with raw data. However, from a business perspective, When the goal of a business is to maximize their sales purely to increase their market share, then naturally, you must be willing to decrease your desired profit margins to achieve higher sales volumes, especially because in e-commerce, the conversion rates of visitors are generally relatively low [15]. A higher TPR at the cost of accuracy is then a more desirable outcome. Also, whether it is important to minimize the amount of False Positives cases is dependent on the cost per lead of an e-commerce business. Furthermore, the results show a decrease in TPR of ~0,25 when the number of features gets below 3.

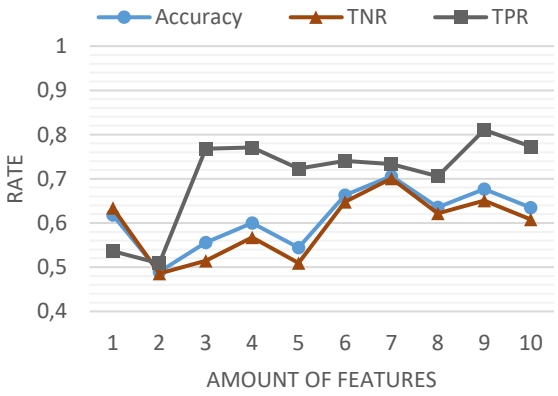


Figure 3. Performance of deep AE with WSCV. Amount of features (1,10)

accuracy of ~2%.

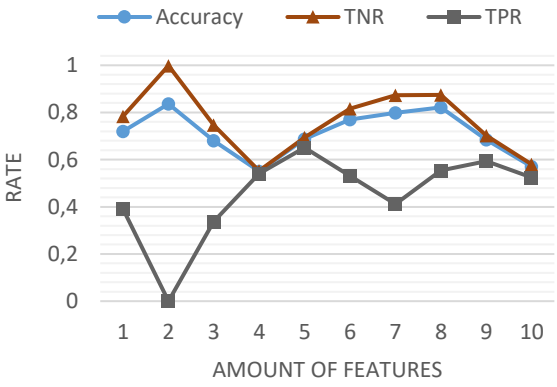


Figure 4. Performance of sparse AE with WSCV. Amount of features (1,10)

6.2 Autoencoders

For the autoencoders, the results of the autoencoders in combination with the default SVC have been left out in this section. The reason for this is because the WSVC was clearly outperforming the default SVC, which is also made clear with the graphs of a single SVM versus a single WSVM. However, the results of the autoencoders with the default SVC can be found in the Appendix.

The deep autoencoder with WSCV is showing an average TPR of 0,70, which is slightly lower than the WSCV with raw data. However, the maximum TPR is 0,81, which is the highest value compared to the maxima of the other methods. There is also a

The Sparse autoencoder with WSCV has the least stable results compared to the ot her autoencoders. The TPR ranges from a maximum value of 0,65 to a minimum of 0 and the Accuracy ranges from roughly 0,55 to 0,83.

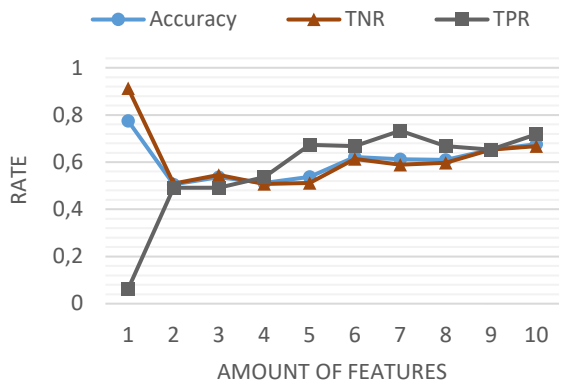


Figure 5. Performance of 2DConvolutional AE with WSCV. Amount of features (1,10)

The 2D convolutional autoencoder with WSCV is showing roughly stable results until the amount features in the hidden layer is set to 1. This is because the MaxPooling2D layer, which is responsible for the dimensionality reduction needs at least 2 values to effectively compress the data into a smaller dimension, making the last layer with only one feature not suitable for this type of autoencoder. Even though the results are more stable compared to the sparse autoencoder, with an average TPR of 0,45 and average accuracy of 0,57 this method gets the least desirable results compared to the other autoencoders

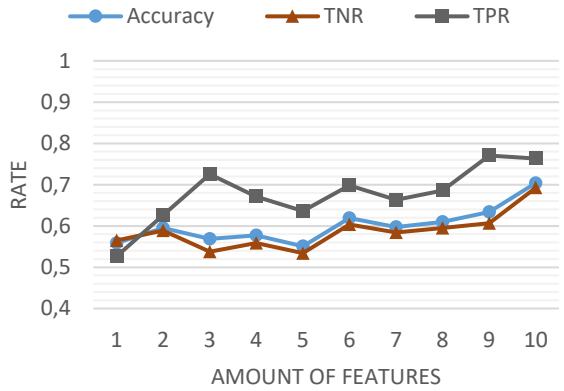


Figure 6. Performance of 1DConvolutional AE with WSCV. Amount of features (1,10)

Finally, the 1D convolutional autoencoder has the most stable results of all autoencoders. With an average TPR of 0,67 and average accuracy of 0,6, the results are very comparable to the results of the deep autoencoders.

Overall, it can be observed that all implementations of autoencoders find it more difficult to recognize buyers when the set of features become smaller. This is most likely caused by two reasons: The autoencoders are trained on an imbalanced dataset. unlike the WSVC, the autoencoders themselves treat every sample with the same weight. With only 15,5% positive samples, it is too hard for the autoencoders to identify meaningful features. Also, the dimension of the datapoints become too small for the autoencoder to effectively separate the buyers from the non-buyers. Then, it either fails to classify buyers consistently, or it does classify buyers, but at the cost of accuracy.

Also, likely, this dataset is not optimal for an autoencoder. Looking at other studies that used autoencoders for different types of classification, it seems that autoencoders perform best when given a higher-dimensional dataset, around 100-300 different features [16].

in Table 3 a summary of all the results is given with the average and maximum values of Accuracy, TPR and TNR.

Table 3. summary of maximum and average results of different machine learning methods.

Method	Max / avg ACC	Max/ avg TNR	Max/ avg TPR
Single SVM	0,89/0,89	0,97/0,97	0,58/0,50
Single WSVM	0,89/0,87	0,93/0,9	0,76/0,74
Deep AE with WSVM	0,71/0,61	0,7/0,59	0,81/0,71
Sparse AE with WSVM	0,84/0,71	0,998/0,76	0,65/0,45
2DConvolutional AE with WSVM	0,77/0,57	0,91/0,6	0,67/0,45
1DConvolutional AE with WSVM	0,70/0,6	0,69/0,59	0,77/0,68

6.3 Autoencoders versus traditional methods

Taking the average of all the implementations of autoencoders tested in this paper into account, an average accuracy of 0,62, average TNR of 0,64 and average TPR of 0,57 is achieved. Compared to the traditional single WSVM, the WSVM beats the autoencoders with an average accuracy of 0,87, TNR of 0,9 and TPR of 0,74.

Compared to other traditional methods, The Markov model achieved an accuracy of 0,56 with a TPR of 0,73 and a TNR of 0,32, performing better on TPR but a bit less on accuracy [8]. However, the Markov Model was used with a different dataset, so this comparison should be considered as an indication. Decision trees used on the same dataset achieved an accuracy of 0,82, with a TPR of 0,79 and a TNR of 0,85 [6]. This result is close to the results of the WSVM. Therefore, the decision trees outperform the autoencoders as well on almost every level.

7. CONCLUSION AND FUTURE WORK

This research shows an analysis of the application of autoencoders in the prediction of online buying behavior. A comparison is made between the classification performance of the SVM/WSVM in combination with four different types of autoencoders and a more traditional method: a single SVM/WSVM using only raw data. Also, the performance of the autoencoders is compared to the performance of other machine learning methods used in related research papers.

The performance of the autoencoders on this dataset show to be highly dependent on the dimensionality of the data was chosen for a low amount of features result in less desirable results. In general, it can be concluded that the deep feature learning algorithms used in this paper show potential in their ability to recognize buyers in an unbalanced dataset. However, the

accuracy of the predictions made with the use of these algorithms is not ideal. Comparing the different types of autoencoders tested in this paper, the deep autoencoder is the best choice for this dataset if you are willing to sacrifice accuracy for a higher TPR. The sparse autoencoder is the best choice if you are only interested in the most accurate classification.

Generally, on this dataset, the traditional methods that only use raw data are showing the best and most stable results compared to autoencoders. the WSVM with raw data has the highest average score in all areas and is, therefore, the most balanced option to use for this dataset.

However, that does not mean that it can be concluded that autoencoders, in general, perform worse on this dataset compared to methods that only use raw data. This paper shows the performance of autoencoders with default parameters. However, there are a lot of parameters that can still be tweaked and tested, which has not yet been done in this research.

Also, there is one type of autoencoder that has not been tested in this research: the denoising autoencoder. These autoencoders corrupt the data on purpose, to create "noise" in the dataset. Then, by having to learn to "denoise" the dataset afterward, it has to learn features about the dataset in a different way. It would be interesting to see what the results are when this implementation is used with the WSVM.

Also, all the tests have only run one time. Training and testing all different implementations is a time-consuming task and there was a limited time for this research. So, it is worth it to investigate the results when the autoencoders and SVM run more times with small variations in the training and testing sets.

8. ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Dr. Le Viet Duc for the guidance with the research and the programming of the autoencoders. His experience and expertise in the field of machine learning have been of great value for this research.

9. REFERENCES

[1] J. D. Ratliff and D. L. Rubinfeld, "ONLINE ADVERTISING: DEFINING RELEVANT MARKETS," *J. Compet. Law Econ.*, vol. 6, no. 3, pp. 653–686, 2010, doi: 10.1093/joclec/nhq011.

[2] A. V Hausman and J. S. Siekpe, "The effect of web interface features on consumer online purchase intentions," *J. Bus. Res.*, vol. 62, no. 1, pp. 5–13, 2009, doi: 10.1016/j.jbusres.2008.01.018.

[3] D. Van den Poel and W. Buckinx, "Predicting online-purchasing behaviour," *Eur. J. Oper. Res.*, vol. 166, no. 2, pp. 557–575, 2005, doi: 10.1016/j.ejor.2004.04.022.

[4] F. Wu, I. H. Chiu, and J. R. Lin, "Prediction of the intention of purchase of the user surfing on the web using hidden Markov model," in *2005 International Conference on Services Systems and Services Management, Proceedings of ICSSSM'05*, 2005, vol. 1, pp. 387–390, doi: 10.1109/ICSSSM.2005.1499501.

[5] L. Guo, L. Hua, R. Jia, B. Zhao, X. Wang, and B. Cui, "Buying or browsing?: Predicting Real-time Purchasing Intent using Attention-based Deep Network with Multiple Behavior," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1984–1992, doi: 10.1145/3292500.3330670.

[6] C. O. Sakar, S. O. Polat, M. Katircioglu, and Y. Kastro, "Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks," *Neural Comput. Appl.*, vol. 31, no. 10, pp. 6893–6908, 2019, doi: 10.1007/s00521-018-3523-0.

[7] A. Vieira, "Predicting online user behaviour using deep learning algorithms." 2016, [Online]. Available: https://www.researchgate.net/publication/284219029_Predicting_online_user_behaviour_using_deep_learning_algorithms.

[8] L. Bing and S. Yuliang, "Prediction of User's Purchase Intention Based on Machine Learning," in *Proceedings - 2016 3rd International Conference on Soft Computing and Machine Intelligence, ISCMI 2016*, 2017, pp. 99–103, doi: 10.1109/ISCMI.2016.21.

[9] O. Sakar and Y. Kastro, "Online Shoppers Purchasing Intention Dataset Data Set," *UCI machine learning Repository*, 2018. https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset (accessed Jun. 21, 2020).

[10] "Support Vector Machines," *Scikit-learn*, 2019. https://scikit-learn.org/stable/modules/svm.html#svm (accessed Jun. 21, 2020).

[11] G. Guodong and Y. Ma, *Support Vector Machines Applications*. Springer International Publishing, 2014.

[12] J. Wang, H. He, and D. V Prokhorov, "A Folded Neural Network Autoencoder for Dimensionality Reduction," *Procedia Comput. Sci.*, vol. 13, pp. 120–127, 2012, doi: 10.1016/j.procs.2012.09.120.

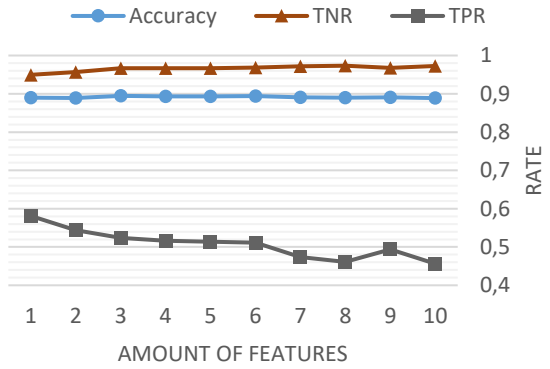
[13] F. Chollet, "Building autoencoders in Keras," *Keras*, 2016. https://blog.keras.io/building-autoencoders-in-keras.html (accessed Jun. 21, 2020).

[14] J. van Hoek, "Research Project," *GitHub*, 2020. https://github.com/Jelmervh/Researchproject/ (accessed Jun. 21, 2020).

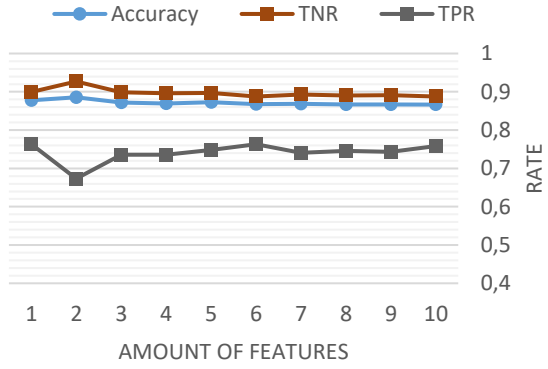
[15] A. W. Ding, S. Li, and P. Chatterjee, "Learning User Real-Time Intent for Optimal Dynamic Web Page Transformation," *Inf. Syst. Res.*, vol. 26, no. 2, pp. 339–359, 2015, doi: 10.1287/isre.2015.0568.

[16] Y. Ju, J. Guo, and S. Liu, "A Deep Learning Method Combined Sparse Autoencoder with SVM," in *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2015, pp. 257–260, doi: 10.1109/CyberC.2015.39.

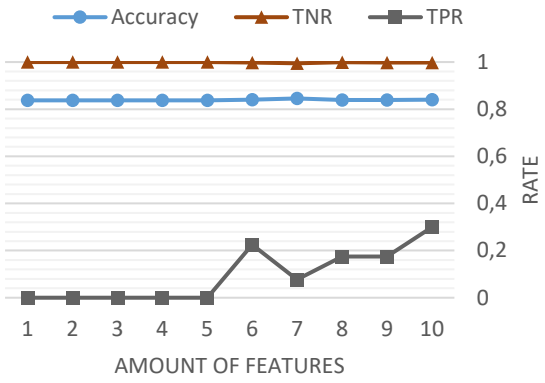
APPENDIX



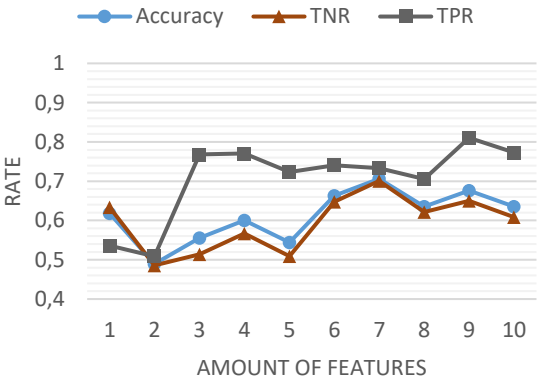
Performance of single SVC. Amount of features (1,10)



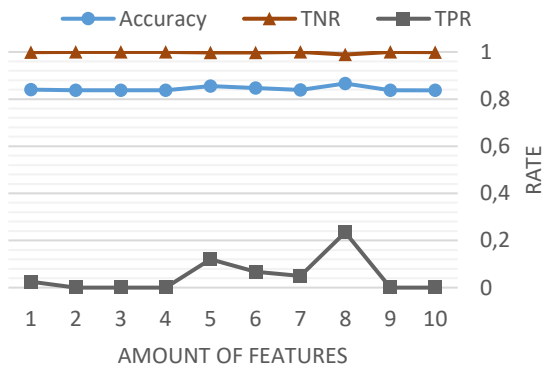
Performance of single WSVC. Amount of features (1,10)



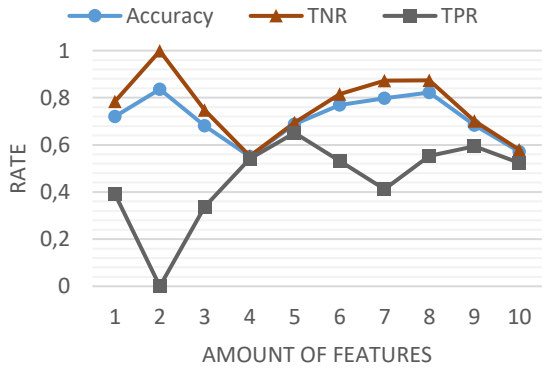
Performance of deep AE with SVC. Amount of features (1,10)



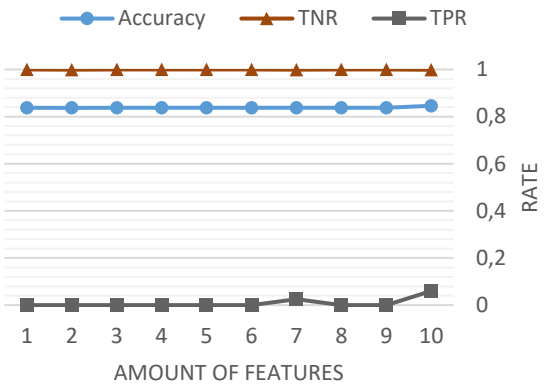
Performance of deep AE with WSVC. Amount of features (1,10)



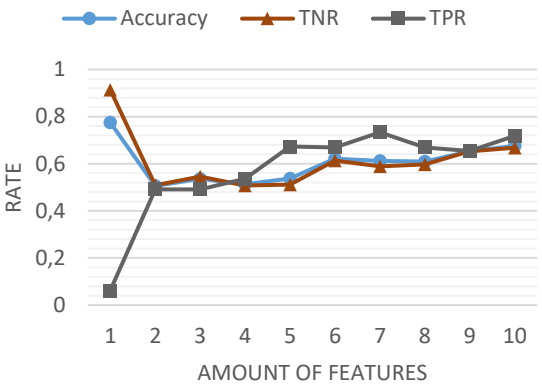
Performance of sparse AE with SVC. Amount of features (1,10)



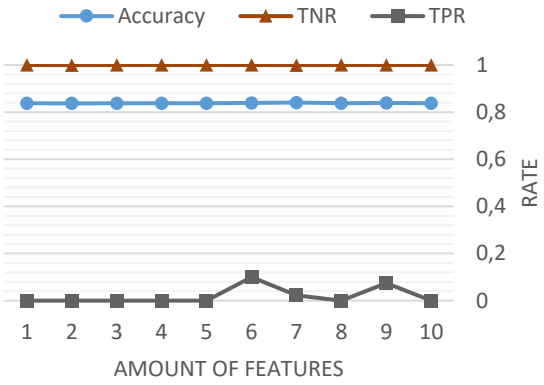
Performance of sparse AE with SVC. Amount of features (1,10)



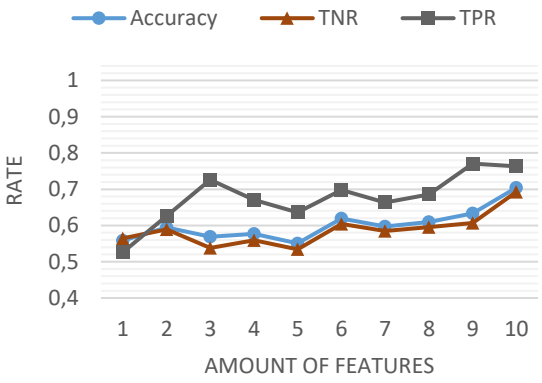
Performance of 2DConvolutional AE with SVC. Amount of features (1,10)



Performance of 2DConvolutional AE with WSVC. Amount of features (1,10)



Performance of 1DConvolutional AE with SVC. Amount of features (1,10)



Performance of 1DConvolutional AE with WSVC. Amount of features (1,10)