OUTLIER DETECTION BY MODEL COMPLEXITY

A NEW DEEP LEARNING METHOD

A Thesis

by

SUNG JE BANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Duncan Henry M. Walker
Committee Members,	Jun Kameoka
	Theodora Chaspari
Head of Department,	Scott Schaefer

August 2020

Major Subject: Computer Engineering

ABSTRACT

In this research, we developed a new method of outlier detection and removal from point-based data sets utilizing deep learning. To do this, we focused on creating an outlier detection method that would tie the outlier detection procedure and a model-building process together. Using the different behaviors of outliers and inliers, we used model complexity as an indicator for outliers in data sets. In this context, "complexity" of a model means the weight of non-zero edges in the model. This include features of a model such as number of layers and number of nodes per layer.

Our proposed method of using model complexity to detect outliers consists of several steps. First, a model of low complexity (low number of layers or low number of nodes per layer) should be made and trained on a data set, and its predicted values for each instance of the data set must be recorded. Second, we need to build multiple neural network models of differing number of layers or number of nodes per layer and find a group of models of specific number of layers with the best average performance values on a given data set. Performance in this context includes general classification accuracy or mean squared error values of models. Third, within the group, we pick the model with the highest number of nodes per layer and use its predictions for each instance of the data set and compare them with the predicted values of the low-complexity model from the first step. The instances with different prediction values by both models should then be labeled as outliers and thus removed.

Two factors must be noted about this method. First, the lower the correlation that attributes have to the output values in a data set, the fewer outliers the method will detect.

Second, the larger and more complex a data set becomes (such as having many attributes), the fewer outliers the method will find. These factors must be noted when using this method.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professors Duncan M. (Hank) Walker Theodora Chaspari of the Department of Computer Science and Engineering, and Professor Jun Kameoka of the Department of Electrical and Computer Engineering.

The analysis of the prediction values of neural network models was partly assisted by Professor Theodora Chaspari.

All other work conducted in the thesis was completed by the student independently.

Funding

Graduate study was supported by the Texas Aggie Graduate Grant provided by Texas A&M University.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

1. INTRODUCTION

1.1 The Outlier Problem

1.1.1 Data Sets and Neural Network

In any professional field, people have to make decisions. When determining if a CPU chip is good enough to be sold, a person has to determine if it performs at a high enough level. When determining if an individual in a picture is a man or a woman, a person has to take note of key features in the individual's face and body. To make these decisions efficiently, people turned to collecting and analyzing data from their respective fields. This practice became bigger in scale and more sophisticated in methodologies as computers became more advanced to handle big data, and now experts rely nearly entirely on digital data for data analytics [1]. Such methods include data condensation, sampling, divide and conquer, and more [2]. The scale of digital data storage and usage grew so large that the market for big data, in total, reached $16 billion in 2014 [3]. People extensively used data in other fields as well, such as disease research and prevention [4], businesses [5], and smart cities [6]. Data usage, storage, and research continues to grow today, which makes the concept of data more and more important.

As data grew, people developed new methods of analyzing data sets more efficiently. During the 1960's, Bayesian Inference, a statistical method of predicting outputs from instances of data based on existing data, was developed for machine learning [7]. After facing a decline in popularity in the 1970's, machine learning began to resurface and gain even more popularity as backpropagation, a method for finding the gradient, was rediscovered in the 1980's [8][9]. Eventually in the 1990's, machine learning became more focused on building models by training on digital data sets [10]. During this time period, methods such as the Support Vector Machine (SVM), machine learning models used for analyzing classification and regression problems in

data [11], became popular. In the 2000's, unsupervised machine learning became widely available and popular [12]. In 2009, people found that neural networks, a robust but expensive machine learning method that normally cost too many computer resources to be practical, could be run to create models efficiently by training on data sets using Nvidia's graphical processing units (GPU) [13]. GPUs make it possible to do neural network training on average computers [14]. A powerful and robust machine learning method, neural networks can be used to analyze and predict output values in many topics and fields, such as automatic speech recognition [15] and image recognition [16]. Neural network models are currently one of the most powerful and relevant machine learning methods for studying and utilizing data.

1.1.2 Outliers, the Parasite of Data Sets

One big issue that negatively impacts machine learning as a whole exists in data sets: outliers. According to Frank E. Grubbs, a statistician, an outlier is an instance in a data set that "appears to deviate markedly from other members of the sample" [17]. Outliers can be caused by two issues. First, such values can be the result of groups of data within a data set being too dispersed, causing such values to be undetected until further data analysis or a larger sample size. Second, outliers can be caused by an error during the construction of their respective data set, such as a physical disturbance or measurement error [18]. Outliers in the second case are meaningless data, and they do not represent the nature of their respective data set in any way. It is critical these values are removed so that they do not muddle the actual behaviors of the real data, termed inliers.

For this research, we focus on a method of detecting and removing outliers from data sets. Outliers continue to be a significant issue for all forms of data analytics, and people continue to create sophisticated methods of detecting and removing outliers. Despite this, outliers

continue to be a significant problem, so we created a new method of outlier detection. However, because of time constraints, we limited the scope of our research to point-based numerical data sets.

1.1.3 Background

People started developing methods of detecting outliers long before neural networks existed. During the mid-1800's, Benjamin Peirce created Peirce's Criterion, a classical outlier removing method derived from the Gaussian distribution, and it allows for at least two outliers to be labeled as outliers within a data set [19]. In this method, we first have to calculate the mean and standard deviation of the complete data set. We then have to collect the R-value that matches with the number of measurements in Peirce's table. We then find the maximum deviation of the value ($|x_i - x_m|_{max}$). For any values that appear to be outliers, we find the $|x_i - x_m|$ value. If $|x_i - x_n|$ is greater than $|x_i - x_m|_{max}$, then the value must be labeled as an outlier [20]. Shortly after this, William Chauvenet created the Chauvenet's Criterion, another classical outlier removing method [21]. When using this method on a data set, we first find the sample mean and sample standard deviation. Then, for all data points that might be outliers (X_i), we calculate ($|X_i$ - sample mean| / sample standard deviation) and store each value. Next, we take that calculated value, find its complementary error function (erfc), and multiply it by the number of instances in the data set. If the resulting value is less than 0.5, then the value is an outlier [20]. In 1950, Frank E. Grubbs created the Grubb's Test, which was a different method that can only be used on univariate data sets [22].

In the modern era, experts took inspirations from the classical methods and created outlier detection methods with neural networks in mind. One type of method created with neural networks in mind is the Deep Anomaly Detection (DAD) technique. As data sets grow larger in

size and more complex in types (images or large number of attributes), traditional outlier detection became ineffective. To address this issue, people created the DAD technique. This technique allows models to learn “hierarchical discriminative features” from given data sets [23]. One sophisticated DAD method is Outlier Exposure. In this method, learning neural network models are exposed to outliers outside of their training data set to more effectively detect outliers in the input for the models [24]. The DAD technique is efficient and is being used in industrial practice, such as credit card fraud detection [25], cyber-intrusion detection [26], and the medical domain [27].

1.1.4 Guide to this Paper

In this research, we focused on developing an outlier detection method that would intuitively tie the outlier detection procedure and a model-building process together. We did this to find a new, more efficient method of detecting, labeling, and removing outliers from point-based data sets.

The second chapter focuses on why we chose this particularly method for detecting outliers. We explain the logic behind our choice and demonstrate what kind of procedures our method follows. To note, we use point-based data sets due to the limited time constraints of our research..

We show our experimentation and findings in the third chapter. We explain in more detail what steps we took, what data sets we tested this on, what results we gained, and what said results showed. We also discussed the potential errors and hidden negative factors that may have muddled our results.

In the fourth chapter, we conclude our research and summarize the research’s proposed problem and new solution. We then discuss how this research can be improved in future work.

2. PURPOSE OF RESEARCH

2.1 A New Method of Outlier Detection

2.1.1 A New Approach to an Old Problem

The purpose of this research is to find an effective method of detecting and labeling outliers within any point-based data set. To do this, this research focuses on developing an outlier detection method that intuitively ties the outlier detection procedure and a model-building process together. By implementing this feature, the process of outlier detection can be further simplified while maintaining efficiency.

2.1.2 The Narrowed Scope of Data Sets

As mentioned earlier, the research focuses on detecting outlier values in point-based data sets analyzed in a batch process. In this context, a point-based data set is a pre-made data set that contains all necessary data values. As opposed to a sequential data set, a point-based data set is not reliant on time, which means that it will not be collecting any more data values as time passes.

Also, the type of data sets that used in this research will be strictly quantitative rather than qualitative. We avoided implementing mixed methods research due to its time-consuming nature. Also, many data sets containing both quantitative and qualitative data may have been poorly collected; using such data sets in this research could lead to inaccurate results [28]. Also, we chose to use quantitative data for our research because detecting association rules in quantitative data have been effective to a certain degree [29]. Because of this, there is a higher chance to find quantitative data sets with decent association between attributes and results in comparison to qualitative data sets.

Because of this limited scope, this work does not make any claims on whether the proposed detection method will work on time-based sequential data sets. It does, however, show that the method will function on point-based data sets.

2.2 Model Complexity as Metric

2.2.1 The Proposed Method of Outlier Detection

The goal of this research is to develop a sophisticated process of using model complexity as an indicator for outliers in any point-based data set. This will allow for any outlier point in a data set to be labeled as an outlier, which would then be easily removed by a trained model. We developed this method due to the differing characteristics of outliers and regular inliers in any data set.

To explain generally, outliers and inliers behave differently in all data sets. Inlier values usually stay grouped in a single or multiple clusters in their data set, maintaining one or more specific patterns neural network models can detect. Outlier values, however, differ considerably from the rest of the inlier values, causing them to exist outside of the general clusters of the inliers. Each outlier can be viewed as being drawn from a different population than the inliers, with each outlier potentially from its own unique population. Notably, in nearly all proper data sets, inlier values far outnumber outlier values [30]. Despite this, a considerable number of outliers usually exist in all data sets, and the nature of their unexpected behavior in data sets disrupts the learning process of neural network models.

2.2.2 Procedure of Method

To elaborate on our proposed detection process, our method will use a neural network model's complexity as a metric of measuring and detecting outliers. The complexity of a model,

in this context, means the weight of non-zero edges in the model. This includes features such as number of nodes per layer or number of layers within a model. Normally, a neural network model with a set complexity that trains well on a data set with no outliers will struggle to train on the same data set with outliers included. The model will have more difficulty learning the characteristics of inliers due to outlier values disrupting the expected patterns in a data set. In this case however, if we use a similar model but with more complexity added, then that model will be able to detect the unexpected behaviors of outliers, detect them as outliers, and allow for a model to learn from the same data set while avoiding the labeled outliers. In this context, adding more complexity to a neural network model can mean adding more layers or adding more nodes per layer. After this, the model of the same or similar complexity can be used to learn from the same data set and detect the different behaviors of outliers the original model did not catch. This kind of behavior can be used to compare the two models' predicted output values for each instance in the data set. A different prediction value for the same instance would mean that it is an outlier, since theoretically, the original low-complexity model would not have detected the outlier while the higher-complexity model would have detected it; this would most likely cause a difference in prediction for the outlier instance.

Table 1, the Example Data Set, shows a demonstration. We have a data set with 5 instances with 1 input attribute and 1 output attribute that has 2 possible output classes: A or B. One of the instances is an outlier while the rest are inliers.

Attribute	Output
1	A
1	A
2	B
2	B
5 (outlier)	A

Table 1: Example Data Set

For this scenario, a single-layer neural network model with 32 nodes is built. However, after training on the data set, the model improperly learns the data set due to the outlier and predicts the last instance in the data set as “B” while predicting the rest correctly.

A model with two layers and 32 nodes per layer, however, has a better performance. It has a 100% accuracy on the training value and can predict all instances correctly. This means that it predicted the last instance’s class correctly as “A”.

We would then compare the prediction values of the two models. The comparison is demonstration in Table 2, which shows the Two Models’ Prediction Values for Each Instance from Example Data Set.

32 Model	32-32 Model
A	A
A	A
B	B
B	B
B	A

Table 2: Two Models’ Prediction Values for Each Instance from Example Data Set

The last instance is predicted differently by the two models. This means that the last instance is an outlier. We can then label the last instance as an outlier, which would allow for easy removal for future training. This is the general summarization of what we plan to show during the experimentation in the next chapter.

Notably, the more outliers there are and the farther they deviate from inliers in a data set, the more likely they will disrupt the learning process of a neural network model. This means that

a model will have more difficulty detecting inlier behaviors and differentiate them from outlier behaviors. Considering this, some data sets have outlier values that are more severe in deviation from the inliers than other data sets. In this case, the more severe the outliers become, the higher the complexity can be raised in the model for training. We must, however, also keep note of how many inliers the model correctly classifies. In other words, we must also check if the model is classifying most, if not all, inliers as inliers rather than outliers. We must beware of cases where models incorrectly classify outliers as inliers and a significant number of inliers as outliers. By keeping these factors in mind, we may be able to find a method of using model complexity as a means of detecting outliers while correctly classifying inliers as inliers in any data set.

2.2.3 Avoiding the Pitfall

Importantly, one cannot use a neural network model of very high complexity on a data set by default. To optimally detect outliers in a data set, we need to start with a low complexity, train the model on the data set, and record its general accuracy on the test data set. After this, we need to raise the model's complexity slightly, train the model on the data set, and compare its general accuracy on the test data set to the previous model's results. This process of raising the complexity continues until the general accuracy of the model peaks and begins to drop with higher complexity. This rise and fall of performance (accuracy of a model) happens because of underfitting. When a model gains too high of a complexity, it begins to over-analyze the behaviors of both inliers and outliers [31]. This causes a model to classify more behaviors than normal within a data set, which results in the model mislabeling data points. This means that the model becomes inefficient at detecting outliers when it has more complexity than needed for a data set. This factor forces model complexity to remain as a metric for outlier severity in data sets, so model complexity cannot be set high on default to detect outliers.

2.2.4 The Scope of Complexity

Considering the information explained in the chapter, we may be able to detect outliers in any point-based data by altering model complexity. We will do this by altering the non-zero edges within the model being used for each data set we will use for experimentation. This includes changing the number of nodes per layer or changing the number of layers of a neural network model.

3. EXPERIMENTATION

3.1 General Procedures

3.1.1 Generalized Approach Explained

As mentioned earlier, we performed research on finding a new outlier detection method that utilizes model complexity. To do this, we completed two tasks. First, we took three publicly available labeled data sets, manually replaced some of their inlier values with outlier values, and tested models of differing complexities. Second, we selected the models with the overall good performances in the first task and observed how they reacted to the outlier values within the data sets.

For building a neural network model, we used a Keras Sequential model. Keras is a TensorFlow-based API that allows programmers to easily build a neural network model with features of their choice, ranging from applying different number of layers to creating a model for specific types of inputs such as images or text [32]. For data analysis through code execution, we used Google Colab, a cloud service provided Google that allows an individual to run Python code on a browser [33]. This service provides a versatile virtual environment for data scientists and programmers. These two allowed for speedy and intuitive experimentation in our research.

The first task focused on observing which model complexity allowed for the best performance when training on data sets with differing severity of the deviations of outliers. We performed the following sequence of steps:

1. We took a sample labeled data set and trained multiple models with different types of complexities. For example, after creating a test data set from the sample data set, we trained a model on the sample data set with one layer of 64 nodes, a model with a layer of 128 nodes, a model with two layers of 64 nodes, and so on.

2. We tested each model on the test data set and measured their performance, measured in either overall accuracy or Mean Squared Error (MSE). In this context, accuracy is a fraction of the number of the model's correct predictions of data points' classes over the total number of data points in the data set. We recorded the performance values for each model and repeated the process five or ten times. Once we completed this, we recorded the average performance value for each model on the sample data set.
3. We took the sample data set and found the one or two attributes that affected the output values the most. We did this instead of picking random attributes so that the outliers would be able to muddle the data set most effectively. Then, we calculated the average value and the standard deviation (sigma) for each attribute.
4. Using this, we took about 10% of the data set's values and replaced them with the summed value of the average value and the sigma value, manually creating outliers within the data set. To maintain some uniformity, we did our best to evenly distribute the outliers among each class in the data set, although this proved to be difficult to perform consistently since some data sets had significantly more values in one class than the others. No changes were made to the testing data set.
5. Once we completed the data set with 1-sigma outlier values, we repeated the process of taking the same models of different complexity, recording their performance values on the unchanged testing data set ten times, and recording their average performance values.
6. After completing the same process on the 1-sigma data set, we created a 2-sigma version of the original data set by replacing 10% of the data set's values (the same ones that were changed to create the 1-sigma version of the data set) with summed values of the average value and 2-sigma value (2 multiplied by the attribute's standard deviation value).

7. We then repeated the process of finding the same models' performance on the 2-sigma data set and recording their average performance values on the unchanged testing data set when trained on the 2-sigma version of the data set.
8. Finally, we created a 3-sigma data set and repeatedly trained on it and tested on the same testing data set. The process was completed once all the averaged performance values were recorded.

The following two example charts illustrate the task described above. Table 3 is the Example Chart for Different Models' Accuracy Performance on Untouched Example Data Set Every Iteration & Average, and Table 4 is the Example Chart for Different Models' Accuracy Performance on Sigma-1 Example Data Set Every Iteration & Average. In a real experiment, there would also be tables for 2-sigma and 3-sigma data sets.

Example Chart 1 - Models' Accuracy Value on Unchanged Imaginary Data Set						
	64	128	64-64	128-128	64-64-64	128-128-128
Trial 1	.9	.95	.92	.94	.92	.93
Trial 2	.9	.95	.92	.94	.92	.93
Trial 3	.9	.95	.92	.94	.92	.93
Trial 4	.9	.95	.92	.94	.92	.93
Trial 5	.9	.95	.92	.94	.92	.93
Average	.9	.95	.92	.94	.92	.93

Table 3: Example Chart for Different Models' Accuracy Performance on Untouched Example Data Set Every Iteration & Average

Example Chart 2 - Models' Accuracy Value on Sigma-1 Imaginary Data Set						
	64	128	64-64	128-128	64-64-64	128-128-128
Trial 1	.88	.91	.92	.95	.93	.94
Trial 2	.88	.91	.92	.95	.93	.94
Trial 3	.88	.91	.92	.95	.93	.94
Trial 4	.88	.91	.92	.95	.93	.94
Trial 5	.88	.91	.92	.95	.93	.94
Average	.88	.91	.92	.95	.93	.94

Table 4: Example Chart for Different Models' Accuracy Performance on Sigma-1 Example Data Set Every Iteration & Average

The charts would be used to observe which model performed the best on the original data set, and what kind of complexity change, such as change in number of layers or change in number of nodes per layer, would allow a model to maintain its performance when having to train on the same data set but with manually inserted outlier values. Keep in mind that the hypothesis is that as the outliers' sigma values increase, the complexity has to rise accordingly. We also had to know which kind of complexity affected the performance more positively: prioritizing on raising the number of nodes per layer while statically keeping the layers, or prioritizing on raising the number of layers while statically maintaining the number of nodes per layer. Once all this procedure is completed, the first task would be completed.

In the second task, we selected the models with good overall test performances from the first task and observed what predictions they would make for injected outliers. To do this, we first observe which model performed the best on the original data set. In Example Chart 1 from the first task, a model with one layer and 128 nodes performed the best, and in Example Chart 2 a model with two layers and 128 nodes per layer and a model with three layers and 128 nodes per layer performed very well compared to other models. In this case, we observe that increasing the total number of nodes while maintaining the same amount of nodes per layer is the most effective way of increasing model complexity to maintain high performance against outliers. We selected the 128 model as the base model, since it performed the best on the original data set, and the 128-128 model and the 128-128-128 model for comparison to see what predictions they would make for injected outliers. Then, we take these models, train them on the 1-sigma data set, and record their prediction values for each data point in the data set. We then compare the base model's prediction values for data points with injected outlier values against the other models' prediction values for the same data points. The logic here is that trained models of lower

complexity will have more difficulty properly predicting the correct output values for outliers compared to higher complexity models. When this type of difference in prediction is detected for the outlier data, the data point could be labeled as an outlier within the data set, which would then allow for easy removal. During this phase, we focus on finding which of the models (excluding the base model) detects the most outliers. After this completed, the same process is repeated for 2-sigma and 3-sigma data sets. By comparing the best outlier-detecting models for each sigma data sets, we would also be able to observe the pattern between rising sigma outliers in data sets and rising complexity in a neural network model.

3.1.2 Data Sets Used

We evaluated our outlier detection method on three different data sets: The Iris Data Set, the Haberman's Survival Data Set, and the Red Wine Quality Data set. We also go forward with the assumption that these data sets contain negligible outliers.

The Iris Data Set is a classic data set in the field of Data Science and is one of the three data sets used in this experiment. A biologist and statistician, R.A. Fisher, created the data set in 1936 [34]. The data set contains three types of iris flowers, and its purpose is to show that each iris type can be determined from their four aspects: sepal length, sepal width, petal length, and petal width. In the data set, the attributes are named "SepalLengthCm", "SepalWidthCm", "PetalLengthCm", and "PetalWidthCm", respectively. The data set contains a total of 150 instances and three iris types, or output classes: Iris-setosa, Iris-versicolor, and Iris-virginica. Each class, or output class, has 50 instances each [35].

From the Iris Data Set, we created a testing data set. We took 8 random instances from each class. This means that the entire testing data set contained 24 instances taken from the original Iris Data Set.

As described in the *Generalized Approached Explained* section, we created several different versions of the data set where we manually replaced some data values with our own outlier values. First, we chose several attributes to change. Figure 1, the Scatter Matrix for the Iris Data Set, shows the analysis of every attribute and how the output values relate to them. Through this data analysis, we found that the PetalLengthCm and PetalWidthCm have the clearest division of data points by their output classes, so we chose those two attributes. Next, we chose to replace 6 data points' PetalLengthCm and PetalWidthCm values for each class, which means we replaced 18 data points' values in total. Because the data set was nicely divided so that each flower class has 50 data points, we faced no difficulty evenly distributing the outlier values. Thus, for the first version of the data set, we found the sum of the average value of PetalLengthCm and its 1 sigma value and a sum of the average value of PetalWidthCm and its 1 sigma value. Thus, for the first version of the data set, we replaced a total of 18 data points' PetalLengthCm values with the sum of the attribute's average and 1 sigma value and the PetalWidthCm values with the sum of that attribute's average and 1 sigma value. We repeated the process for the second and third versions of the data set, but we instead used 2 sigma and 3 sigma, respectively.

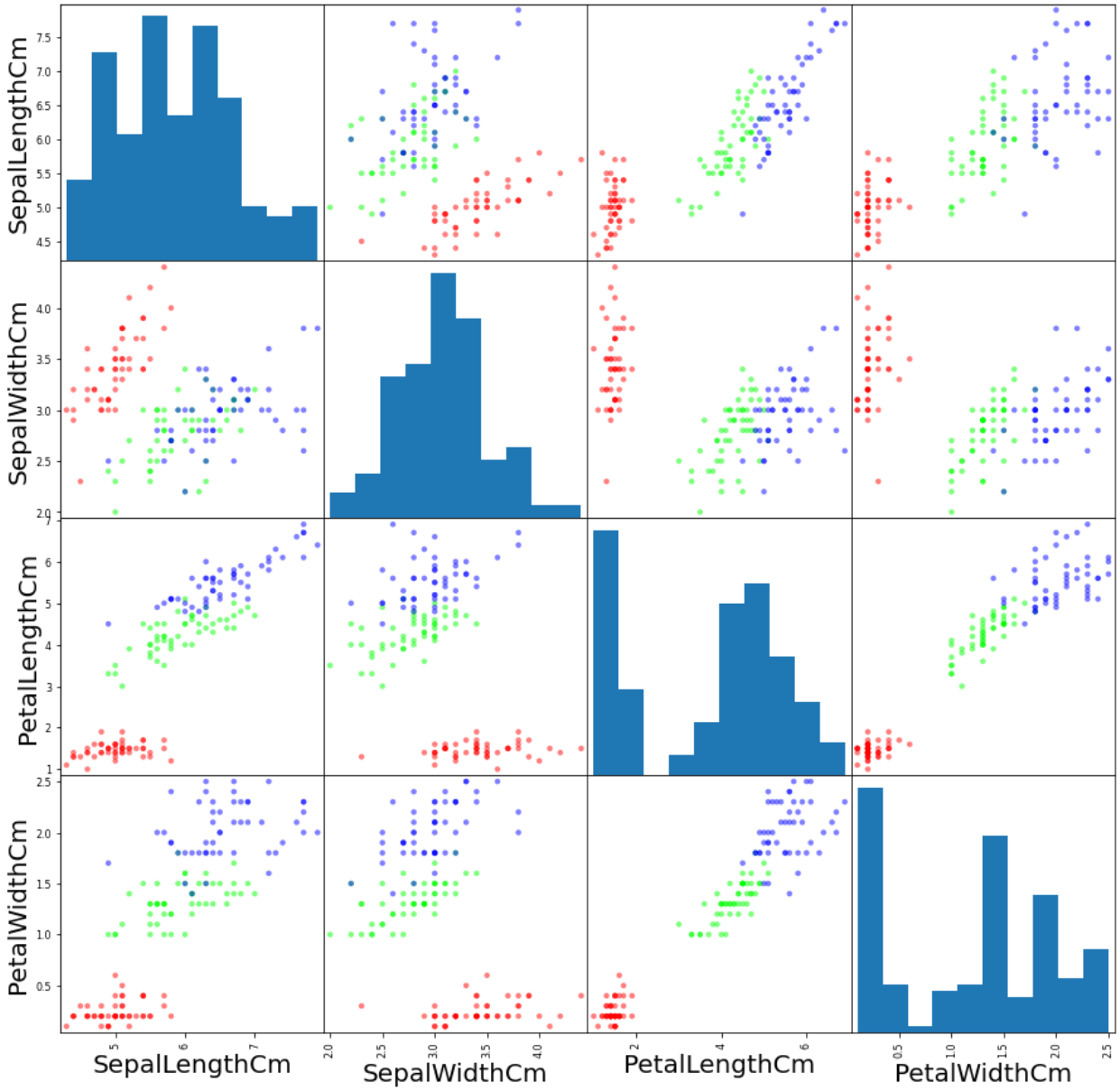

Figure 1: Scatter Matrix for the Iris Data Set

The Haberman's Survival Data Set is a result of research, conducted from 1958 to 1970, on breast cancer surgery patients at the University of Chicago's Billings Hospital. The data set describes the conditions of the patients and shows which of them survived the operation or died within 5 years after the surgery. For each patient, the data set describes the person's age when they underwent the surgery, the year they received the surgery, the number of positive axillary

nodes the patient had, and if the patient survived the operation for five years or longer or died within five years [35]. For these descriptions, the attributes are labeled “age”, “op_year”, “axil_nodes”, and “surv_status”, respectively. The data set contains a total of 306 instances and two output classes for “surv_status” [36]. The class is either 1 or 2. A class value of 1 signifies that the patient survived for five years or more after the surgery, and 2 means that the patient died within five years after the surgery. 225 instances had the surv_status of 1, and 81 instances had the surv_status of 2 [35]. Unfortunately, the data set is not balanced, so outlier distribution will have to be uneven.

We created a testing data set for the Haberman Data Set also. We took 10 random instances from each class. This means that the testing data set contained 20 instances taken from the Haberman Data Set.

As we did for the Iris Data Set, we created several versions of the Haberman Data Set where we replaced some data values with our own outlier values. Figure 2, the Scatter Matrix for Haberman’s Survival Data Set, shows the analysis of every attribute in the Haberman’s Survival Data Set and how its output values relate to them. We first analyzed the attributes of the data sets to identify the attributes to use for outliers. Through data analysis, we found that no pair of attributes have a significant impact on a patient’s survivability. We examined the individual impact of each attribute and found that “axil_nodes” has the most effect on the output value of “surv_status”. The effect is not strong, but stronger than other attributes, singly or in pairs. Figure 3, which is a Comparison of the Values of “axil_nodes” and “surv_status”, shows the relationship between the survival status of patients and the number of axillary nodes they carried. After choosing “axil_nodes” as the attribute for outlier replacements, we found the sum of the attribute’s average value and 1 sigma value. Upon completing this, we replaced the “axil_node”

values of 20 instances that have the surv_status of 1 and 10 instances that have the surv_status of 2 with this sum. Finally, as we did for the Iris Data Set, we repeated the process with 2 sigma and 3 sigma values, respectively.

Figure 2: Scatter Matrix for Haberman's Survival Data Set

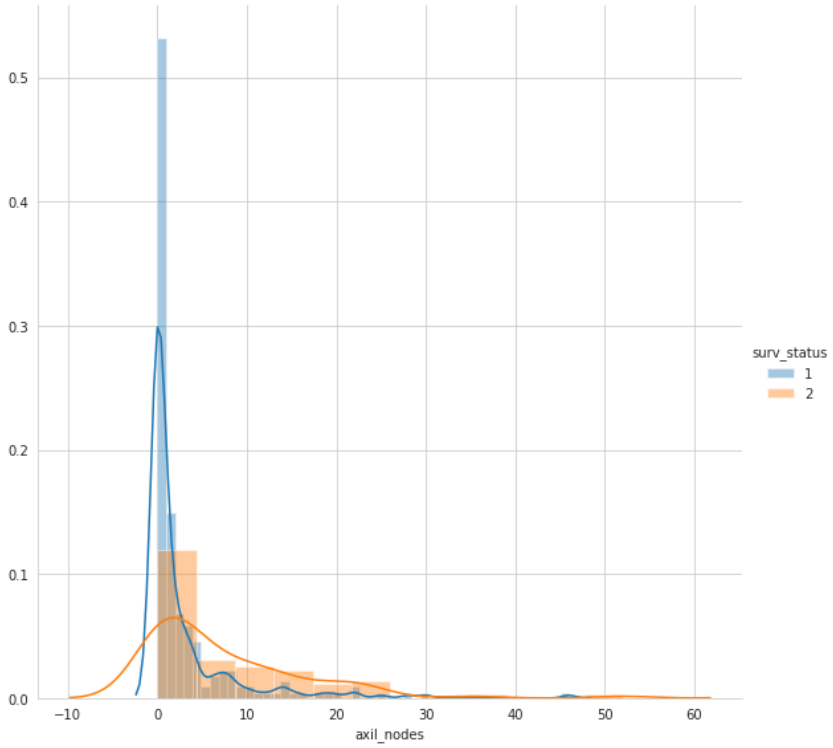

Figure 3: Comparison of the Values of "axil_nodes" and "surv_status"

The final data set is the Red Wine Quality Data Set, which is derived from the Wine Quality Data Set, made by P. Cortez and his group, which measures the overall quality of thousands of red wine and white wine samples [37]. This red wine data set contains different samples of the Portuguese "Vinho Verde" wine, measures each instance's different qualities based on physicochemical tests, and finally rates the overall quality of the wine based on sensory data. The different qualities include fixed "acidity", "volatile acidity", "citric acid", "residual sugar", "chlorides", "free sulfur dioxide", "total sulfur dioxide", "density", "pH", "sulphates", and "alcohol". The output variable is "quality", which is a numerical score between 0 and 10. The data set contains 1,599 instances and the "quality" values of 3, 4, 5, 6, 7, and 8. There are 10 instances with the quality score of 3, 53 instances with the score of 4, 681 instances with the score of 5, 638 instances with the score of 6, 199 instances with the score of 7, and 18 instances

with the score of 8 [38]. Since the data set is unbalanced, outliers will have to be distributed unevenly.

We created a testing data set using 320 random instances from the Red Wine Quality Data Set. Like the previous two data sets, we created several versions of the Red Wine Quality Data Set, where we replaced some data values with our own outlier values. We first analyzed the attributes of the data set using the entire data set's correlation plot shown in Figure 4, which is the Correlation Chart for the Red Wine Quality Data Set. Through it, we found that "quality" has the most correlation with "volatile acidity" (correlation score of -0.378372), "sulphates" (correlation score of 0.242596), and "alcohol" (correlation score of 0.472676). For each of the three attributes, we found the sum of the attribute's average value and 1 sigma value, and then replaced the attribute value with the sum values. We replaced values of 2 instances with the score of 3, 4 instances with a score of 4, 51 instances with a score of 5, 47 instances with a score of 6, 14 instances with a score of 7, and 2 instances with a score of 8. We did the same for the 2-sigma and 3-sigma versions with 2- sigma values and 3-sigma values, respectively.

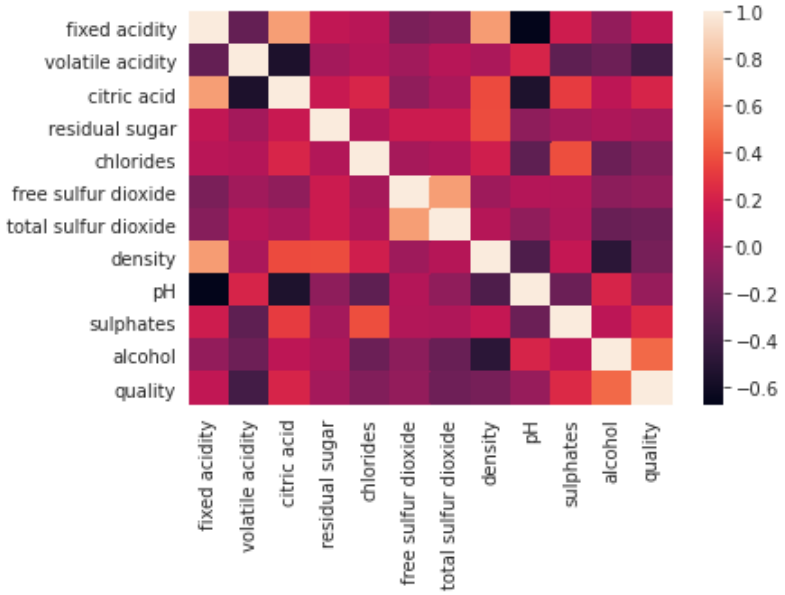

Figure 4: Correlation Chart for the Red Wine Quality Data Set

3.2 Experimentation

3.2.1 What to Look For

With the general procedures and the data sets (along with their three outlier versions) prepared, we moved on to the experimentation. While we experimented, we looked for what kind of complexity showed more positive effect on models' performances as outliers' sigma values rose: increasing the number of nodes per layer while maintaining the number of layers or increasing the number of layers while maintaining the number of nodes per layer. We also checked to see what kind of relationship existed between models' performance and the models' ability to detect many outliers.

3.2.2 Iris Data Set

We obtained the Iris Data Set and created three other versions of the same data set. The first version has 18 of its instances replaced with the sum of each attribute's average value and its 1 sigma value. The second version has 18 of its instances replaced with the sum of each attribute's average value and its 2 sigma value. The third version has 18 of its instances replaced with the sum of each attribute's average value and its 3 sigma value.

For the original version of the data set, we first trained multiple models varying in complexity and tested them on the testing data set 10 times. We found that the model with one hidden layer of 128 nodes, along with a *sigmoid* activation function, performed the best on average with a 95.668% accuracy. In this context, accuracy means how accurately a model can determine every instance's iris type.

We then tested different model complexities on the outlier data sets. Along with the 128-model, we trained multiple models with varying complexities on the 1-sigma, 2-sigma, and 3-sigma data sets. This includes a model with one layer and 256 nodes, two layers with 64 nodes

per layer (64-64), two layers with 128 nodes per layer (128-128), and so on. Table 5, chart for Different Models' Accuracy Average Values on Sigma-X Data Sets of the Iris Data Set, shows the average performance value of each model on the x-sigma data sets (average of 10 trials).

Different Model Complexities and Their Performance Averages on Outliers (Iris)								
	128	256	128-128	256-256	128-128-128	256-256-256	128-128-128-128	256-256-256-256
1-sigma	.92667	.93	.89001	.89667	.95334	.94334	.97002	.94669
2-sigma	.92334	.93336	.93668	.96003	.95668	.9667	.95668	.9667
3-sigma	.94334	.93002	.96002	.96336	.96336	.96336	.96336	.95002

Table 5: Different Models' Accuracy Average Values on Sigma-X Data Sets of the Iris Data Set

For 1-sigma, the models generally performed well with 3 or 4 layers. Having 4 layers is better, but this is only by a small margin. Notably, the general performance of models are similar according to the number of their layers, so there is a relationship between outlier detection and model complexity in layers. Raising the number of nodes while maintaining a similar number of layers does improve accuracy, but this is secondary compared to raising the number of layers while maintaining a similar number of nodes.

For 2-sigma, the models generally performed well with 3 or 4 layers. Once again, 4 layers is slightly better, and more layers improves accuracy more than more nodes.

For 3-sigma, models generally perform very well when having 2, 3, or 4 layers. They do still, however, perform better than when the model has one layer.

Next, we must check how accurately models are able to detect outliers. We determine if a model detected an outlier if they predict a different value from the original 128-model on data points with replaced outlier values. Table 6, a chart for Different Models' Average Percentage of Outliers Detected on Sigma-X Data Sets of the Iris Data set, shows the fraction of the outliers the models were able to detect for each sigma data set.

Different Model Complexities' Outlier Detection Measurement (Iris)							
	256	128-128	256-256	128-128-128	256-256-256	128-128-128-128	256-256-256-256
1-sigma	.15556	.14444	.20556	.17222	.20556	.12778	.16667
2-sigma	.11667	0.18333	.35556	.50556	.48333	.57222	.5
3-sigma	.12778	.15	.37778	.52778	.67778	.75556	.76111

Table 6: Different Models' Average Percentage of Outliers Detected on Sigma-X Data Sets of the Iris Data Set

The outlier detection experimented yielded interesting results. For 1-sigma, the 256-256-256 model performed the best while the 128-128-128-128 model performed the worst. Generally, 3-layer models detected the most outliers among the injected outliers while the 128-128 model performed decently. For 2-sigma, the 128-128-128-128 model performed the best while the 256 model performed the worst. For 3-sigma, the 256-256-256-256 model performed the best while the 256 model performed the worst.

Finally, we collected the percentage of the number of inliers the models detected differently from the original 128 model. Table 7, a chart for Different Models' Average Percentage of Inliers Classified as Outliers on Sigma-X Data Sets of the Iris Data Set, shows how much of the inliers the models were not able to detect consistently for each sigma data set.

Different Model Complexities' Wrong Inlier Detection Measurement (Iris)							
	256	128-128	256-256	128-128-128	256-256-256	128-128-128-128	256-256-256-256
1-sigma	0.03939	0.06061	0.05606	0.07121	0.05606	0.05152	0.06515
2-sigma	0.06364	0.04848	0.05076	0.05606	0.05227	0.05227	0.08333
3-sigma	0.06667	0.05682	0.05530	0.07652	0.07348	0.07348	0.06591

Table 7: Different Models' Average Percentage of Inliers Classified as Outliers on Sigma-X Data Sets of the Iris Data Set

From these results, having more nodes per layer led to models classifying more inliers correctly for models with two or three layers, but having less nodes per layer led to models classifying more inliers correctly for models wtih four layers. For 1 sigma, the 256 model

classified the most inliers correctly, and the models began to classify more inliers incorrectly as outliers as they rose in number of layers and nodes per layer. For 2 sigma, the 128-128 model classified the most inliers correctly. The number of inliers being classified incorrectly as outliers grew as the models rose in number of layers and number of nodes per layer. For 3 sigma, the 256-256 model detected the most amount of inliers correctly. The number of inliers being classified incorrectly started relatively high for the 256 model, but dropped as it gained another layer. But afterward, the models began to incorrectly classify more inliers as outliers as it gained more layers or rose in number of nodes per layer.

Generally, we found interesting results from this data set. For the 1-sigma data set, the 256-256-256 model and the 256-256 model both performed the best in outlier detection, but they performed average in correctly classifying inliers as inliers. The 256 model retained the most amount of inliers, but it did a below-average job in detecting outliers within the 1-sigma data set. For 2 sigma, the 128-128-128-128 model performed the best in outlier detection, and it did an above average job in classifying inliers correctly. It must be noted that 256-256-256-256, despite detecting a decent percentage of injected outliers, did poorly in classifyign inliers correctly. The 128-128 model performed the best in inlier detection. For 3 sigma, the 256-256-256-256 model performed the best in outlier detection and performed average in inlier detection. The 256-256 model classified the most inliers correctly in this data set.

3.2.3 Haberman's Survival Data Set

We created three outlier versions of the Haberman's Survival Data Set. The first version has 30 of its instances replaced with the sum of each attribute's average value and its 1 sigma value. The second version has 18 of its instances replaced with the sum of each attribute's

average value and its 2 sigma value. The third version has 18 of its instances replaced with the sum of each attribute's average value and its 3 sigma value.

For the original version of the data set, we trained multiple models varying in complexities and tested them on the testing data set 10 times. We found that the model with one hidden layer of 64 nodes, along with a *sigmoid* activation function, performed the best on average by scoring 77.287% accuracy. In this context, "accuracy", similar to the Iris Data Set, means how accurately a model is able to determine every patient's survival status.

We tested different model complexities on the three outlier data sets. As we did for the Iris Data Set, we trained different models with varying complexities on the 1-sigma, 2-sigma, and 3-sigma data sets. Table 8, a chart for Different Models' Accuracy Average Values on Sigma-X Data Sets of the Haberman's Survival Data Set, shows the average performance value of each model on the data sets (average of 10 trials).

Different Model Complexities and Their Performance Averages on Outliers (Haberman)								
	64	32-32	64-64	128-128	64-64-64	128-128-128	64-64-64-64	128-128-128-128
1-sigma	.77124	.7722	.77285	.76928	.76471	.76568	.7696	.76632
2-sigma	.77222	.77123	.77482	.77156	.76993	.76012	.77285	.76861
3-sigma	.77613	.78267	.78136	.77483	.77286	.76862	.77219	.77286

Table 8: Different Models' Accuracy Average Values on Sigma-X Data Sets of the Haberman's Survival Data Set

For 1-sigma, the models generally performed about the same with 1 or 2 layers. More layers caused a slight decrease in model accuracy.

For 2-sigma, the models performed slightly better when having 2 layers, and then worse with more layers. This does show that it is helpful to add more hidden layers to models the more severe the outliers become in deviation.

For 3-sigma, the models performed slightly better when having 2 layers, and then slightly worse for more layers. This, again, serves as small evidence that it is helpful to add more hidden layers to models the more severe the outliers become in deviation.

Next, we need to see observe how accurately models detect outliers. As with the Iris Data Set, we determine that a model detected an outlier if it predicts a different from the original 64-model on data points with replaced outlier values. The following chart shows how much of the outliers the models were able to detect for each sigma data sets.

Next, we need to see observe how accurately models detect outliers. As with the Iris Data Set, we determine that a model detected an outlier if it predicts a different value from the original 64-model on data points with replaced outlier values. Table 9, a chart for Different Models' Average Percentage of Outliers Detected on Sigma-X Data Sets of Haberman's Survival Data Set, shows how much of the outliers the models were able to detect for each sigma data set.

Different Model Complexities' Outlier Detection Measurement (Haberman)						
	64-64	128-128	64-64-64	128-128-128	64-64-64-64	128-128-128-128
1-sigma	.03	.09333	.04	.05333	.04667	.07667
2-sigma	0.03333	.07	.05333	.09667	.05333	.06667
3-sigma	0.08333	.11333	.07333	.10333	.05	.11333

Table 9: Different Models' Average Percentage of Outliers Detected on Sigma-X Data Sets of Haberman's Survival Data Set

The outlier detection experimented showed led to some interesting findings. For 1-sigma, the 128-128 model performed the best while the 64-64 model performed the worst. Overall, though, 4-layer models detected the most outliers among the injected outliers, and models with 128 nodes per layer performed decently. For 2-sigma, the 128-128-128 model performed the best while the 64-64 model performed the worst. For 3-sigma, the 128-128 model and the 128-128-128-128 model performed the best while the 64 model performed the worst.

Finally, we collected the percentage of the number of inliers the models detected differently from the original 64-64 model. Table 10, a chart for Different Models' Average Percentage of Inliers Classified as Outliers on Sigma-X Data Sets of Haberman's Survival Data Set, shows how much of the inliers the models were not able to detect consistently for each sigma data set.

Different Model Complexities' Wrong Inlier Detection Measurement (Haberman)						
	64-64	128-128	64-64-64	128-128-128	64-64-64-64	128-128-128-128
1-sigma	0.04161	0.08077	0.03846	0.055594	0.040210	0.069580
2-sigma	0.03846	0.05140	0.05	0.057692	0.047902	0.049650
3-sigma	0.05682	0.05530	0.076515	0.073484	0.073485	0.065909

Table 10: Different Models' Average Percentage of Inliers Classified as Outliers on Sigma-X Data Sets of Haberman's Survival Data Set

From these results, having more nodes per layer led to models classifying more inliers correctly for models with two or three layers but having less nodes per layer led to models classifying more inliers correctly for models with four layers. For 1 sigma, the 256 model classified the most inliers correctly, and the models began to classify more inliers incorrectly as outliers as they rose in number of layers and nodes per layer. For 2 sigma, the 128-128 model classified the most inliers correctly. The number of inliers being classified incorrectly as outliers grew as the models rose in number of layers and number of nodes per layer. For 3 sigma, the 256-256 model detected the most amount of inliers correctly. The number of inliers being classified incorrectly started relatively high for the 256 model, but dropped as it gained another layer. But afterward, the models began to incorrectly classify more inliers as outliers as it gained more layers or rose in number of nodes per layer.

Generally, we found interesting results from this data set. For the 1-sigma data set, the 256-256-256 model and the 256-256 model both performed the best in outlier detection, but they performed average in correctly classifying inliers as inliers. The 256 model retained the most

amount of inliers, but it did a below-average job in detecting outliers within the 1-sigma data set. For 2 sigma, the 128-128-128-128 model performed the best in outlier detection, and it did an above average job in classifying inliers correctly. It must be noted that 256-256-256-256, despite detecting a decent percentage of injected outliers, did poorly in classifying inliers correctly. The 128-128 model performed the best in inlier detection. For 3 sigma, the 256-256-256-256 model performed the best in outlier detection and performed average in inlier detection. The 256-256 model classified the most inliers correctly in this data set.

3.2.4 Red Wine Quality Data Set

We obtained the Red Wine Quality Data Set and created three other versions of the same data. The first version has 320 of its instances replaced with the sum of each attribute's average value and its 1 sigma value. The second version has 18 of its instances replaced with the sum of each attribute's average value and its 2 sigma value. The third version has 18 of its instances replaced with the sum of each attribute's average value and its 3 sigma value.

Unlike the Iris Data Set and the Haberman's Survival Data Set, however, this data set is a regression data set. This means that a neural network model would not be classifying the output value based on an instance's attributes. It would calculate a numerical score for each instance, which would fit this data set since it focuses on scoring each instance from 0 to 10. This also means that the performances of the models that would learn from this would be shown through their Mean Squared Error (MSE) values.

For the original version of the data set, we trained multiple models varying in complexities and tested them on the testing data set. We found that the model with two hidden layers and 64 nodes per layer, along with *relu* activation functions and a dropout value of 0.25 for every hidden layer, performed the best on average by scoring a MSE value of 0.3788. Since

all score values are in integer values ranging from 0 to 10, we consider this MSE value good for the Red Wine Quality Data Set.

Next, we must check how accurately models are able to detect outliers. We determine if a model detected an outlier if they predict a different value from the original 64-64 model on data points with replaced outlier values. Table 11, a chart for Different Models' MSE Average Values on Sigma-X Data Sets of the Red Wine Quality Data Set, shows the performances (in terms of MSE) of models on each data set.

Different Model Complexities' Outlier Detection Measurement (Red Wine Quality)						
	64-64	128-128	64-64-64	128-128-128	64-64-64-64	128-128-128-128
1-sigma	.3823	.4	.4179	.4174	.4063	.4151
2-sigma	.3869	.3655	.4164	.4066	.41	.4135
3-sigma	.4007	.3752	.3890	.4167	.41	.4188

Table 11: Different Models' MSE Average Values on Sigma-X Data Sets of the Red Wine Quality Data Set

For 1-sigma, the models generally performed the same when having multiple layers. However, there is a slight worse performance when the number of layers increases to 3 layers or more. This trend continued to both 2-sigma and 3-sigma.

Next, we need to see observe how accurately models detect outliers. Similarly with the previous two data sets, we determine that a model detected an outlier if it predicts a different from the original 64-64 model on data points with replaced outlier values. Table 12, a chart for Different Models' Average Percentage of Outliers Detected on Sigma-X Data Sets of the Red Wine Quality Data Set, shows how much of the outliers the models were able to detect for each sigma data set.

Different Model Complexities' Outlier Detection Measurement (Red Wine Quality)					
	128-128	64-64-64	128-128-128	64-64-64-64	128-128-128-128
1-sigma	.0509375	.0484375	.058125	.0503125	.040625
2-sigma	.0525	.033125	.056875	.0309375	.0384375
3-sigma	.0475	.0309375	.046875	.0384375	.0321875

Table 12: Different Models' Average Percentage of Outliers Detected on Sigma-X Data Sets of the Red Wine Quality Data Set

The outlier detection experimented yielded interesting results. For 1-sigma, the 128-128-128 model performed the best while the 128-128-128-128 model performed the worst. Generally, 3-layer models detected the most outliers among the injected outliers while the 128-128 model performed decently. For 2-sigma, the 128-128-128 model performed the best while the 64-64-64-64 model performed the worst. It must be noted, however, that the 64-64-64 model and the 64-64-64-64 model performed poorly. For 3-sigma, the 128-128-128 model performed the best while the 64-64-64 model performed the worst.

Finally, we collected the percentage of the number of inliers the models detected differently from the original 64-64 model. Table 13, a chart for Different Models' Average Percentage of Inliers Classified as Outliers on Sigma-X Data Sets of the Red Wine Quality Data Set, shows how much of the inliers the models were not able to detect consistently for each sigma data set.

Different Model Complexities' Incorrect Inlier Detection Measurement (Red Wine Quality)					
	128-128	64-64-64	128-128-128	64-64-64-64	128-128-128-128
1-sigma	0.150733	0.157981	0.194219	0.150820	0.167731
2-sigma	0.152804	0.132787	0.155651	0.121225	0.156428
3-sigma	0.172476	0.112770	0.169974	0.140811	0.171096

Table 13: Different Models' Average Percentage of Inliers Classified as Outliers on Sigma-X Data Sets of the Red Wine Quality Data Set

Interestingly, in each layer, models with less nodes generally detected more inliers correctly. It also seems that for every sigma, a model that detected the most outliers also

incorrectly identified the many inliers as outliers. It must be also be noted that the 64-64-64 and 64-64-64-64 models identified inliers correctly the most. In 1-sigma, the 64-64-64-64 model performed the best in inlier detection. For 2-sigma, the 64-64-64-64 model detected the most inliers, but the 64-64-64 model had a similarly good performance in inlier detection also. In 3-sigma, the 64-64-64 model clearly performed the best in inlier detection.

We retrieved interesting results from this data set. For all sigma tests, the 128-128-128 model performed the best in outlier detection but also performed consistently bad in inlier detection. Generally, models with 128 nodes per layer detected more outliers when their number of layers rose from two to three, but dropped as it went from three to four layers. The same type of story cannot be told for inlier detection. For 1-sigma, models with 128 nodes per layer incorrectly classified inliers as outliers when the number of layers rose from two to three, but dropped as it went from three to four layers. For 2-sigma, the number of incorrectly classification of inliers as outliers rose as the number of layers rose from two to three, but increased significantly less when rising from three to four layers. For 3-sigma, the models with 128 nodes per layer generally performed similarly badly, incorrectly detecting around 17% of the inliers as outliers. Meanwhile, the models with 64 nodes per layer detected less outliers than their 128-node counter parts in each layer. Regardless, models with 64 nodes per layer detected more outliers as the number of layers rose from three to four for sigma 1. For sigma 2, the models detected less outliers as the number of layers rose from three to four. For sigma 3, the models detected more outliers as the layers changed from three to four. Inlier detection for the models with 64 nodes per layer were interesting. For sigma 1, more inliers were classified correctly as the number of layers changed from three to four. For sigma 2, less inliers were classified

correctly as the number of layers changed from three to four. For sigma 3, less inliers were classified correctly as the number of layers rose from three to four.

3.3 Results

3.3.1 The Pattern between Models' Performance and Outlier Detection Efficiency

From the information we gathered from this experiment, we found that there is a pattern between model complexity and outlier detection that can be used to detect and label some outlier values. Unfortunately, the level of complexity of the models does not clearly show the sigma value of the outliers in data sets. However, model complexities do point to what type of model will do very well in detecting outliers in data sets.

There is a common behavior that shows nearly all the time in these data sets. When a group of models of the same number of layers has the best performance for a data set compared to models of different number of layers, then a model of that number of layers with the highest number of nodes per layer in the group will do the best job at detecting outliers. On top of this, this model correctly classifies most inliers as such, but becomes worse at doing this as the data set it trains on becomes more complex.

For example, in the 1 Sigma version of the Red Wine Quality Data Set, the 64-64 model and the 128-128 model perform the best on average MSE compared to average performance of 64-64-64 model and 128-128-128 model and the average performance of the 64-64-64-64 model and 128-128-128-128 model. On top of this, a model with two layers along with 128 nodes per layer performs the best in detecting outlier compared to other models. It also classifies a good number of inliers as such compared to other models, but all models do not do well with this task in general by classifying over 15% of the inliers as outliers at minimum. In the 2 Sigma version,

the 64-64 model and the 128-128 model performs the best on average compared to the average performances of other groups of models of different layers. Although the 128-128 model does not perform the best in outlier detection in the chart, it does perform very well in comparison to other models in the figure. It detects 5.25% of the manually-replaced outliers while the only model that performs better than this (128-128-128) detects 5.69% of the outliers. This is close compared to other models that only detected about 3.5% of the outliers. The models in this data set, however, do poorly in inlier classification, classifying over 10% of the inliers as outliers at minimum. In the 3 Sigma version, the 64-64 model and the 128-128 model perform the best on average compared to other groups of models of different numbers of layers. On top of this, a model with two layers along with 128 nodes per layer performs the best in detecting outliers compared to other models. However, the said models and the other models tested in this data set do poorly in inlier classification here again. As shown in this data set, the model with the same number of layers as the group of models that had the lowest MSE average value and with the same number of nodes per layer as the group's highest number of nodes per layers performed very well generally in detected outliers, but they also all generally do poorly in inlier classification.

In another example, in the 1-sigma version of the Haberman's Survival Data Set, the 64-64 model and the 128-128 model performed the best in accuracy compared to the average accuracies of other groups of models of different number of layers. On top of this, a model with two layers along with 128 nodes performed the best in outlier detection compared to other models. The 128-128 model does badly in inlier classification, although its performance is nowhere near as bad as the models in the Red Wine Quality Data Set. In the 2 sigma version, the 64-64 model and the 128-128 model performed the best in accuracy once again. Although the

128-128 model did not perform the best in detecting outliers by detecting 7% of the injected outliers, it did perform the second best and scored lower than only the 128-128-128 model which detected 9.67% of the outliers. The 128-128 model did the worst in inlier classification, but the model's percentage of inliers incorrectly classified as outliers was considerably low. In the 3 sigma version, the 64-64 model and the 128-128 model performed the best in accuracy. And this time for outlier detection, the 128-128 model performed the best among the models. The inlier classification performance of the 128-128 model was also good. For this data set also, the model with the same number of layers as the group of models that had the highest accuracy average value and with the same number of nodes per layer as the group's highest number of nodes per layers performed very well generally in detected outliers.

Lastly, in the 1-sigma version of the Iris Data Set, the 128-128-128-128 model and the 256-256-256-256 model performed the best in accuracy compared to the average accuarcies of other groups of models of different number of layers. Notably, the 256-256-256-256 model performed the fourth best among the models, detecting 16.67% of injected outliers while both the 256-256 and 256-256-256 detected the most (20.56%). This was the only case where the model with the same number of layers and the highest number of nodes per layer as the group of models performed at a mediocre level. The 256-256-256-256 model also did below average in inlier classification compared to other models in the data set, but percentage of inliers wrongly classified as outliers was generally low. For the 2-sigma version, the group of the 128-128-128 model and the 256-256-256 model and the group of the 128-128-128-128 model and the 256-256-256-256 model performed the best with the same average accuracy value. However, the 256-256-256-256 model detected more outliers than the 256-256-256 model. We can hypothesize from this that in the case of a tie between two models, the model with the higher

number of layers will perform better in outlier detection. The 256-256-256-256 model detected 50% of the outliers, losing to the 128-128-128-128 model (which detected 57.22%) and the 128-128-128 model (which detected 50.56%). It must be noted, however, that the 256-256-256-256 model performed very well, detecting a very similar number of outliers as the 128-128-128 model, which detected the second most number of outliers. The 256-256-256-256 model had an average inlier-classification performance also, but the percentage of inliers incorrectly classified as outliers was low once again. For the 3-sigma version, the group of 3 layer models, the 128-128-128 model and the 256-256-256 model, had the highest accuracy on average. The 256-256-256 model's outlier detection performance was the third highest; the performance itself was high also. The model's inlier classification was below average, but models in the data set generally classified most inliers as such correctly. From the experimentation on this data set also, the model with the same number of layers as the group of models that had the highest accuracy average value and with the same number of nodes per layer as the group's highest number of nodes per layers generally performed very well in detected outliers.

3.3.2 Verdict and Error Possibilities

From this experiment, we found that the model with the same number of layers as the group of models that had the highest average of performance value (accuracy or MSE) and with the same number of nodes per layer as the group's highest number of nodes per layers, will perform well in detecting outliers. Although, rarely, the model will not be the optimal model for detecting outliers, this method will generally find a model complexity that will allow for a model to detect outliers at high efficiency. Also, said model will normally classify most inliers as inliers correctly, but will grow more error prone as the data set it trains on becomes more complex by having more instances or having more attributes.

This method will allow for outlier detection and removal. First, a model of low complexity should be made and trained on a data set. It should also save the predicted values for each instances. Next, the method presented here can be used; different models of different levels of complexities should be tested on the data set, and the average highest performance values (whether it be accuracy on classification or MSE value) of a group of a same number of layers should be found. Then, within that group, a model with the highest number of nodes per layer should predict each instance output. That list of prediction should be checked with the list of prediction of the original low-complexity model made and trained earlier. All instances that have different prediction values should be removed, which will leave with a data set with considerably less outliers.

Although we have finished this experiment, there are points of interest that must be mentioned. Firstly, some of the data sets we used, namely the Haberman's Survival and the Red Wine Quality Data Sets, may have issues that are difficult to pinpoint. Throughout the experiment, the data sets have shown hints that it may have already contained outliers in the first place. Another possible issue is that the data sets may not have had attributes with strong correspondence with the output values. The strongest evidence of the two possible factors is that the data set with no injected outliers was not able to produce a neural network model with an accuracy value higher than 90% or an MSE value lower than 0.1. Because of at least one of the listed two issues, the results yielded by the two data sets may be slightly misleading. Secondly, this method works efficiently when the sigma of outliers grow. When the sigma values of the outliers are high, the method has a tremendous efficiency in finding the model with the right complexity that can detect outliers. When the sigma values of the outliers are low, however, the method struggles to find the right complexity that would allow for the model to detect a high

number of outliers. Finding a solution to these issues may lead to a more conclusive result.

4. CONCLUSION

4.1 Final Results and Thoughts

4.1.1 Main Results

In the end, in a group of models with the same number of layers that had the highest average of performance value, the model with the highest number of nodes per layer detected outliers at a high level at the cost of misclassifying an average number of inliers most of the time. In most cases, this type of model generally detected outliers consistently well, which would allow many outlier values to be labeled as outliers in their respective data sets. In nearly all cases, such models misclassified an average number of inliers as outliers compared to other models of varying complexities.

4.1.2 Other Factors in the Results

It must be noted, however, that the findings pointed at other interesting behaviors. Neural network models generally detected less outliers the less correspondence attributes within the data set had with the output values. This is shown in the comparisons between the Iris Data Set and the Haberman's Survival Data Set. The Iris Data Set had two highly correlating attributes, PetalLengthCm and PetalWidthCm. The Haberman's Survival Data Set, however, had one attribute, axil_nodes, that showed correspondence with the output value. The attribute, however, had lower correspondence with the output than the two attributes for Iris Data Set had with their respective output attribute. Another factor to note is that neural network models generally detected fewer outliers the more complex the data set. By becoming more complex, we mean that data set has more correlating attributes or more instances. For example, the models for the Red Wine Quality Data Set detected fewer outliers in general compared to the models for the Iris Data Set. The Red Wine Quality Data Set had three highly correlating attributes while the Iris

Data Set only had two. On top of this, the Red Wine Quality Data Set contained many more instances than the Iris Data Set.

4.1.3 Possible Errors

There are possible factors that may have muddled our findings from this experimentation. First, it is possible one or more of our data sets contained attributes that did not have high enough correlation to the output. Namely, the Haberman's Survival Data Set did not have attributes with high correlation to the survival status of patients. Contextually, it may also be possible that the data set was built without enough attributes in mind. For instances, the data set did not describe how each patients were treated, what type of health issues each patient had before the surgery, and more. Second, it is possible one or more of our data sets contained outliers to begin with. Finding outier-free data sets proved to be very difficult, and only the Iris Data Set showed a very high chance of it having very few outliers. If one wishes to use this work as reference for future work, these factors must be considered for improved findings.

REFERENCES

[1] Lyman P, Varian H. How much information 2003? Tech. Rep, 2004. [Online]. Available: http://www2.sims.berkeley. edu/research/projects/how-much-info-2003/printable_report.pdf.

[2] Xu R, Wunsch D. Clustering. Hoboken: Wiley-IEEE Press; 2009.

[3] Press G. $16.1 billion big data market: 2014 predictions from IDC and IIA, Forbes, Tech. Rep. 2013. [Online]. Available: http://www.forbes.com/sites/ gilpress/2013/12/12/16-1-billion-big-data-market-2014-predictions-from-idc-and-iia/.

[4] Mayer-Schonberger V, Cukier K. Big data: a revolution that will transform how we live, work, and think. Boston: Houghton Mifflin Harcourt; 2013.

[5] Chen H, Chiang RHL, Storey VC. Business intelligence and analytics: from big data to big impact. MIS Quart. 2012;36(4):1165–88.

[6] Kitchin R. The real-time city? big data and smart urbanism. Geo J. 2014;79(1):1–14.

[7] Solomonoff, Ray J. "A formal theory of inductive inference. Part II." Information and control 7.2 (1964): 224–254.

[8] Goodfellow, Bengio & Courville (2016, p. 221), "Efficient applications of the chain rule based on dynamic programming began to appear in the 1960s and 1970s, mostly for control applications (Kelley, 1960; Bryson and Denham, 1961; Dreyfus, 1962; Bryson and Ho, 1969; Dreyfus, 1973) but also for sensitivity analysis (Linnainmaa, 1976). ... The idea was finally developed in practice after being independently rediscovered in different ways (LeCun, 1985; Parker, 1985; Rumelhart *et al.*, 1986a). The book *Parallel Distributed Processing* presented the results of some of the first successful experiments with back-propagation in a chapter (Rumelhart *et al.*, 1986b) that contributed greatly to

the popularization of back-propagation and initiated a very active period of research in multilayer neural networks."

[9] Rumelhart; Hinton; Williams (1986). "Learning representations by back-propagating errors" (PDF). *Nature*. **323** (6088): 533–536. Bibcode:1986Natur.323..533R. doi:10.1038/323533a0.

[10] Marr, Bernard. "A Short History of Machine Learning – Every Manager Should Read". *Forbes*. Retrieved 28 Sep 2016.

[11] Siegelmann, Hava; Sontag, Eduardo (1995). "Computational Power of Neural Networks". *Journal of Computer and System Sciences*. **50** (1): 132–150. doi:10.1006/jcss.1995.1013.

[12] Bennett, James; Lanning, Stan (2007). "The netflix prize"(PDF). *Proceedings of KDD Cup and Workshop 2007*.

[13] "Nvidia CEO bets big on deep learning and VR". Venture Beat. April 5, 2016.

[14] "From not working to neural networking". *The Economist*.

[15] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," Audio, Speech, and Language Processing, IEEE Transactions on, vol. 20, no. 1, pp. 30–42, 2012.

[16] Cireşan, Dan; Meier, Ueli; Masci, Jonathan; Schmidhuber, Jürgen (August 2012). "Multi-column deep neural network for traffic sign classification". *Neural Networks*. Selected Papers from IJCNN 2011. **32**: 333–338. CiteSeerX 10.1.1.226.8219. doi:10.1016/j.neunet.2012.02.023. PMID 22386783.

[17] Grubbs, F. E. (February 1969). "Procedures for detecting outlying observations in samples". *Technometrics*. **11** (1): 1–21. doi:10.1080/00401706.1969.10490657. An

outlying observation, or “outlier,” is one that appears to deviate markedly from other members of the sample in which it occurs.

[18] Grubbs 1969, p. 1 stating “An outlying observation may be merely an extreme manifestation of the random variability inherent in the data. ... On the other hand, an outlying observation may be the result of gross deviation from prescribed experimental procedure or an error in calculating or recording the numerical value.”

[19] Peirce, Benjamin (May 1877 – May 1878). “On Peirce’s criterion”. *Proceedings of the American Academy of Arts and Sciences*. **13**: 348–351. doi:10.2307/25138498. JSTOR 25138498.

[20] Rochim, Adian. (2016). Chauvenet’s and Peirce’s Criterion (literature review).

[21] Lily Lin and Paul D Sherman. Cleaning Data the Chauvenet Way. SESUG 2007: The Proceedings of the SouthEast SAS Users Group, (c):1–11, 2007

[22] Quoted from the *Engineering and Statistics Handbook*, paragraph 1.3.5.17, http://www.itl.nist.gov/div898/handbook/eda/section3/eda35h.htm

[23] Chalapathy, Raghavendra & Chawla, Sanjay. (2019). Deep Learning for Anomaly Detection: A Survey. https://arxiv.org/abs/1901.03407 **[cs.LG]**

[24] Hendrycks, Dan & Mazeika, Mantas & Dietterich, Thomas. (2018). Deep Anomaly Detection with Outlier Exposure. https://arxiv.org/abs/1812.04606 **[cs.LG]**

[25] Aderemi O Adewumi and Andronicus A Akinyelu. A survey of machine-learning and nature-inspired based credit card fraud detection techniques. International Journal of System Assurance Engineering and Management, 8(2):937–953, 2017.

[26] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. A survey of deep learning-based network anomaly detection. Cluster Computing, pages 1–13, 2017.

[27] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sanchez. A survey on ´ deep learning in medical image analysis. Medical image analysis, 42:60–88, 2017.

[28] Almalki, Sami. (2016). Integrating Quantitative and Qualitative Data in Mixed Methods Research—Challenges and Benefits. Journal of Education and Learning. 5. 288. 10.5539/jel.v5n3p288.

[29] Gosain, Anjana & Bhugra, Maneela. (2013). A comprehensive survey of association rules on quantitative data in data mining. 2013 IEEE Conference on Information and Communication Technologies, ICT 2013. 1003-1008. 10.1109/CICT.2013.6558244.

[30] Singh, Karanjit & Upadhyaya, Shuchita. (2012). Outlier Detection: Applications And Techniques. International Journal of Computer Science Issues. 9.

[31] A. Ghasemian, H. Hosseinmardi and A. Clauset, “Evaluating Overfit and Underfit in Models of Network Community Structure,” in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2019.2911585.

[32] Chollet, F., & others. (2015). Keras. https://keras.io.

[33] Colaboratory – Google. (2018). Retrieved from https://research.google.com/colaboratory/faq.html

[34] R. A. Fisher (1936). “The use of multiple measurements in taxonomic problems”. Annals of Eugenics. 7 (2): 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x. hdl:2440/15227.

[35] Dua, D., & Graff, C.. (2017). UCI Machine Learning Repository.

[36] de Sousa, G. R. (2017). Haberman's Survival Data Set. Retrieved from https://www.kaggle.com/gilsousa/habermans-survival-data-set

[37] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physiochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

[38] Red Wine Quality. (2017, November 27). Retrieved from https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009