# Initial Measurement of Data Quality

Mitchell J. Kinney, Ph.D.*[a]
[a]The MITRE Corporation, 7515 Colshire Drive, McLean, VA USA 22012

## ABSTRACT

The quality of a dataset is extremely hard to gauge in the age of big data because of the overwhelming amount of data needed to train deep learning models. To estimate the applicability of our data to a deep learning solution, we often must fit our model. This is a time and resource intensive process. The method we present here is a quick triage if the data may not be worth the time or if it may deserve more thorough vetting before fitting a solution. We propose a data quality score that is closely associated with the amount of separability within the data. Our target application is for a large amount of unstructured data such as images and text. We use pre-trained models to do feature generation and use an approximate nearest neighbor solution for speed in understanding the local neighborhoods of data points. In simulation and through examples on well-known toy datasets our method performs as expected and is able to identify when there may be problems when training a classifier.

**Keywords:** Data, Artificial Intelligence, Assurance, Model Agnostic

## 1. INTRODUCTION

In Artificial Intelligence (AI) and Machine Learning (ML), the effectiveness of classification in supervised learning is largely determined by the quality of the data used to construct predictive models. A common phrase about model training is “Garbage in, garbage out,” referring to data quality. When faced with an incredibly large amount of high dimensional data, such as images or text, it is hard to know the quality of the dataset before fitting a model. Determining dataset quality is often a long and expensive process. We consider dataset quality to be proportional to the measure of class separability and this paper presents an efficient way to determine this measure of class separability. Figure 1 illustrates how classification performance corresponds to data overlap between classes. In the figure, the example on the left is visually more separated than the data on the right which leads to a better accuracy and F1 score if the dashed line is our decision boundary. Figure 1 also shows the data quality score proposed by this paper. Since class separability is inherently model agnostic, a data quality score should be attainable before any model selection is performed. The data quality score will be high for higher class separation.

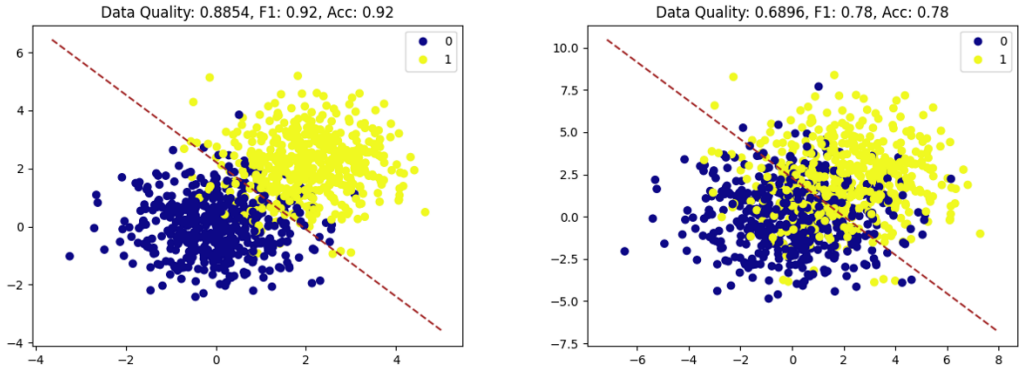



Figure 1. Example of how class overlap affects model performance. Data quality score (proposed by this paper) and common performance metrics are shown in each image along with the decision boundary from a trained logistic regression classifier. Source: MITRE

The method we use to measure class separability takes advantage of the plethora of pretrained models that exist to encode unstructured data into features. Our focus is on labeled data that is being used for classification. We use an approximate nearest neighbor search with weighted rankings for each point. To calculate a score of class separability we average the weighted rankings over each class. Our intuition is that when close points share the same class this should result in a high data quality score.

*mkinney@mitre.org
Additional research is done on the identification of points that may be outliers or exist along a potential decision boundary between classes. Part of the process of measuring class separability is collecting scores from each point which identifies points that may fall far away from a class cluster or fall in between class clusters. The three labels we associate with points are either cluster for points that are within a class cluster surrounded by points of the same class, outlier for points that are surrounded by points that are not of the same class, and edge for points that are close to both points in the same class and different classes. We use leave-one-out on each point to measure the change in ranked scoring for each class. This helps to determine not only what points were close to the left-out point, but also what points had the left out point in their neighborhoods. This two-way inspection of each point gives a better sense of the point's location within the data cloud. Additional post processing ensures point pairs (such as cluster and outlier within the same class) are not in each other's neighborhoods.

The contributions of this report are a score for calculating class separability for unstructured data such as image and text in a timely manner, a model agnostic approach for labeling points as outliers or edge points, and results and examples of how the contributions can be used with simulation and well-known toy datasets.

## 2. PREVIOUS RESEARCH

The topic of classification complexity and data separability has been studied intensely. A seminal paper [6] defined three different categories for measuring a classification dataset for separability: (i) measures of overlap of individual feature values; (ii) measures of the separability of classes; and (iii) geometry, topology and density of manifolds measures. Our contribution falls into the second category. Other measures that exist in this category come from a survey [7] and include neighborhood measures. One example is Fraction of Borderline Points, [6] that builds a Minimum Spanning Tree from the data and counts the number of edges shared by points with different classes. Building the graph edges accounts for the distance between the points and comparing classes accounts for the neighborhood approach. Critical points are those whose distances are small between points that are of differing classes. The T1 distance or Fraction of Hyperspheres Covering Data builds hyperspheres around each data point and expands them until they encompass a point of another class. Then any hypersphere completely contained in another hypersphere is eliminated. The count of remaining hyperspheres is a measure of classification complexity.

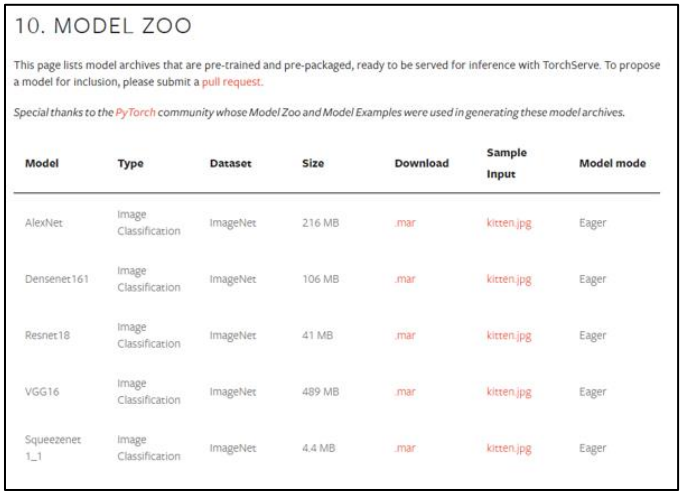
Another more recent measure is in [5] which computes the ratio between the intra distance of classes and between distances of classes. These intra class distances are calculated by finding the pairwise distance of all points in a class and the between class distance is the distance between points from different classes. The ratio of these two distances is compared using the Kolmogorov-Smirnov distance for each class. We differentiate our method from these techniques by focusing on speed and unstructured data. Since a lot of time can be spent on these methods for very large datasets, our method uses an approximate solution rather than a complete one. Also, all these metrics require structured data which we create in our method. Therefore, the methods mentioned here could be of use after converting the unstructured data into features.

The second area that this report covers is labeling cluster, edge and outlier points. We could not find work that attempts to categorize points as possibly being on the decision boundary before model fitting. However, there is a lot of work on outlier detection. Methods that are comparable to our contribution are specifically supervised anomaly detection. One method is by [4] which combines outlier detection algorithms with supervised modeling to optimize outlier detection hyperparameters. This method is similar to ours because it does not rely on known class distributions to calculate the outliers. Our method focuses on neighborhoods of points and the differing classes instead of exploiting class distributions.

## 3. DATA QUALITY

In this section, we introduce our approach to creating a model-agnostic measurement of data quality, focusing on class separability. The steps to calculate our measurement are: (i) generate features for each data point using pre trained deep learning models; (ii) reduce the dimension; (iii) input into a database and perform approximate nearest neighbor search for each point; and (iv) aggregate weighted neighborhood rankings for each data point.

The first step involves using a model zoo to generate features from unstructured data such as images or text as shown in Figure 2. Many pre-trained models exist that have been trained and validated on generalized tasks and well-known benchmarks. These datasets are representative of a multitude of tasks such as classification, entailment, object detection, etc. and can provide a good baseline for transfer learning to a new dataset. We leverage this existing resource by removing the model's head, where the objective typically resides, and extract features from the last layer. To create a high-dimensional object for each data point we concatenate the features together. We recommend using model pairings that have been trained on different datasets to avoid redundant information. All models we use are from pytorch [10] or huggingface [14].



10. MODEL ZOO

This page lists model archives that are pre-trained and pre-packaged, ready to be served for inference with TorchServe. To propose a model for inclusion, please submit a pull request.

*Special thanks to the PyTorch community whose Model Zoo and Model Examples were used in generating these model archives.*

| Model | Type | Dataset | Size | Download | Sample Input | Model mode |
|---|---|---|---|---|---|---|
| AlexNet | Image Classification | ImageNet | 216 MB | .mar | kitten.jpg | Eager |
| Densenet161 | Image Classification | ImageNet | 106 MB | .mar | kitten.jpg | Eager |
| Resnet18 | Image Classification | ImageNet | 41 MB | .mar | kitten.jpg | Eager |
| VGG16 | Image Classification | ImageNet | 489 MB | .mar | kitten.jpg | Eager |
| Squeezenet 1_1 | Image Classification | ImageNet | 4.4 MB | .mar | kitten.jpg | Eager |

Figure 2. Pytorch's [10] offering of pre trained models available to download and use.

The next step is to use dimension reduction techniques to search through the data in an efficient manner. The approximate nearest neighbor solution that we use is time dependent on the number of features within each data point, so to ensure a quick method we must reduce the feature size. Two methods that we explored are random projections and principal component analysis [1]. A visualization of how random projections are computed is shown in Figure 3. Random projections have been shown to provide dimension reduction while still maintaining the relative distance between the points for a chosen threshold if certain criteria are met [3]. One of these criteria is the Johnson-Lindenstrauss lemma [8] that is a calculation of the dimension size the features can be reduced to and still maintain the threshold relative distance given the number of samples in the dataset. The Johnson-Lindenstrauss lemma [8] is a worst case scenario. In Python package sklearn [11] the dimension reduction is done by generating a matrix $N^{p \times d}$ of iid Gaussian random variables with mean 0 and variance $\frac{1}{d}$. Here $p$ is the original dimension size and $d$ is the reduced dimension size. Then to get the reduced features, sklearn matrix multiplies the original data matrix with the generated matrix. Principal component analysis was also explored, but reducing the dimension by compressing the information in the features was not as applicable in our setting as maintaining relative distances between points. Therefore, all results and examples shown were done using random projections to reduce the dimension.

The approximate nearest neighbor solution that we used is ANNoy [2] which is a library that utilizes random projections to build trees that separate features by layers of hyperplanes. The distance formula to measure distance between points and hyperplanes used is Euclidean. Tuning parameters that exist for the library are how many hyperplanes deep to go in the trees, and the number of nodes to search for similar items. Both have a tradeoff between performance and speed. For all the tests we used a consistent set of parameters because performance did not seem to be impacted when using different parameters.

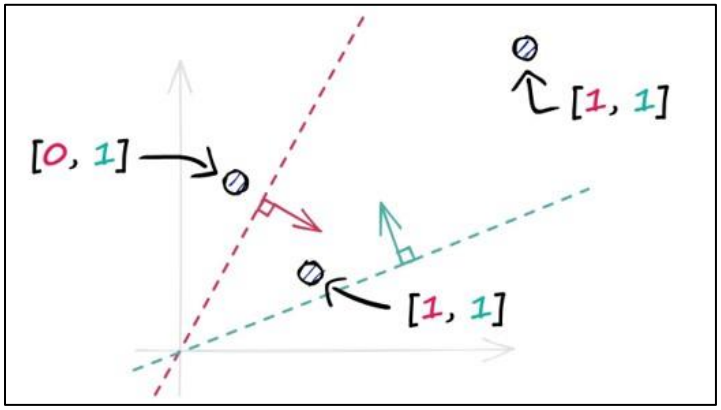



Figure 3. Random projections aid in reducing the dimension of the data. The dimension of the points will be reduced to the number of hyperplanes (dashed lines) chosen and the reduced vectors will be dependent on the relative location of the points to each hyperplane. Source: https://www.pinecone.io/learn/locality-sensitive-hashing-random-projection/

After neighborhoods have been built for each point, we rank points by distance and sum the ranks within class. For a neighborhood of approximate nearest neighbors of size $K$ we sort the neighboring points from closest to farthest. We use inverse ranks as the score so the closest point is scored as $K$, and the furthest point in the neighborhood is scored as 1. An example is shown in Figure 4. For each point in the dataset we use its neighborhood to sum up the inverse rank scores over the classes. If the current point has class $c_0$ then the cumulative score for the same class is $\sum_k^K k\ 1(c_k = c_0)$. The equation for all other classes replaces $c_0$ with the appropriate class label. After calculating these scores for each point we sum the matching class pairs to produce a matrix. The matrix cell values are associated with the current point class and neighborhood point class respectively. Intuitively the matrix shows the clustering of classes along the diagonal and overlap between classes on the off diagonals. Note this is not a symmetric matrix since the existence of point *A* in the neighborhood of point *B* does not guarantee the point *B* will be a nearest neighbor of the point *A*. If these two points are from different classes the matrix will not be symmetric. The matrix is then normalized across rows, so the rows add up to one to highlight the proportionality of clustering and overlap within a specific class. We do not average on the number of labels in each class because we are not comparing the measurements between classes.

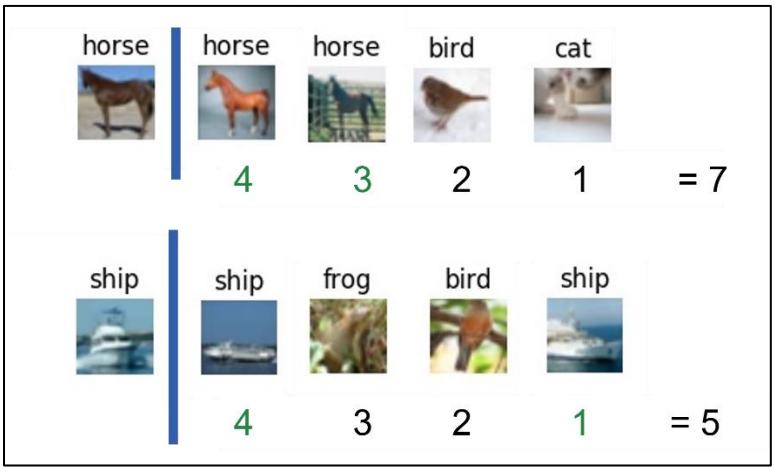



Figure 4. Example of scoring using inverse rankings for the same class. Image to the left is the data point being evaluated. The other images in the row are the neighborhood sorted by closeness. Source: MITRE

We expect that the larger the scores along the diagonals, the more clustered the individual classes. Likewise, the larger the scores on the off diagonals, the more overlapped the two classes. Since AI/ML solutions will typically do better the more separated data is, lower scores along the diagonal would indicate a worse performing classification model. To get a one number summary we can average along the diagonal to get an average same class data quality

score. This was done in Figure 1 where we showed how poorer data quality scores resulted in lower classification scores. In our results section we show through simulation how the data quality score changes with scenarios that are known to affect data separation. Also, we show scores for well-known toy datasets.

## 4. LABELING POINTS

Part of the process of calculating dataset spread is to calculate individual separability scores for each point. We take advantage of this to label points as possible problems when fitting a model. The distinctive labels we choose are whether points are outliers, along the edge of class clusters, or within a class cluster. If the neighborhood of a point is made up of points with class labels different than the main point we consider the main point an outlier, whereas if the neighborhood is made up of points of the same class, then it is within a class cluster, and if there is a mix of classes within the neighborhood then it is an edge point. We differentiate between outlier in the canonical sense of being far away from a distribution. Here a point is only an outlier if we expect the point to influence the decision boundary of the not-yet-fit classifier. An example of an outlier as defined in this paper is shown in Figure 5. If a point is not close to its class cluster but still the nearest points are all the same class, then this would not be labeled as an outlier. Looking at only the neighborhood of the point itself leaves out the information about the neighborhoods that the point may be included in. To account for this additional information, we use leave-one-out and compare the scores between the point being included and not. This is instead of a method like treating the problem like a classification task and using KNN classification. Depending on the change in the same class score and different class scores, we can assign the appropriate label.

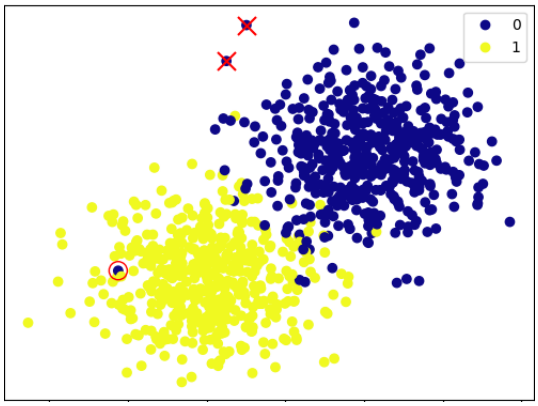



Figure 5. Circled is a point that is considered an outlier. The X'd points are ones that our method does not consider outliers. Source: MITRE

The process of doing leave-one-out on a nearest neighbor search would be extremely long so instead we reuse the scores found in the previous section when we calculated the data quality score matrix. To do this we keep track of the neighborhoods of each point and artificially reduce the number of nearest neighbors by one. This will initially change the data quality score slightly. Then for each point we remove the point from all the neighborhoods we know it's a part of and we remove the scores for when the point was the central point. When a neighborhood loses the point all other points in the neighborhood are moved "closer" and the artificially left out point becomes part of the neighborhood again. Then the overall data quality score is recalculated adjusting only for the changes instead of going over the entire data set again. One difference in this step from the previous section is we do average over the number of points in each class. Since we are comparing between runs with varying number of observations it is important to average so we can control this variation. An example of how the scores change with leave-one-out is shown in Figure 6. In Figure 6 the points are color coded to show which class they belong to. The darker blue column are the points left out of the neighborhood in the initial calculation. As point 4 is eliminated in each these points become a part of the neighborhood if point 4 was previously in that neighborhood. The Same Class Score is then updated accordingly based on the class of the new point. Keep in mind that when re calculating the new overall data quality score, the Same Class Score for point 4 would also be eliminated. The top example in Figure 6 shows point 4 as an edge point, middle example a cluster point, and bottom example an outlier point.

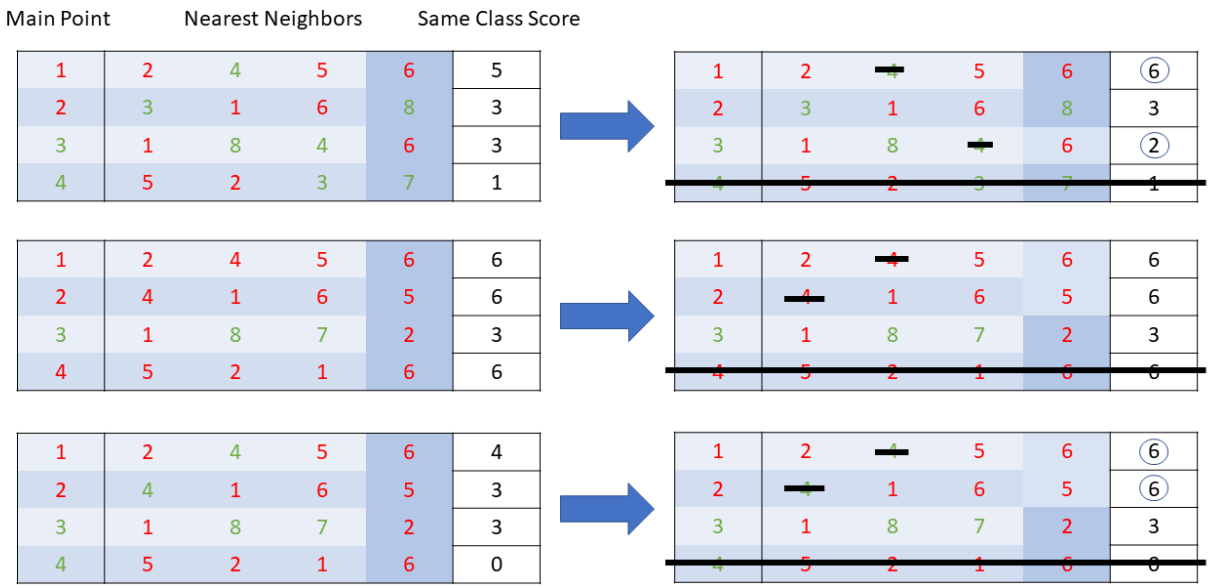



Figure 6. Each horizontal example shows the calculation for determining point 4 is an edge (top), a cluster (middle) or an outlier (bottom). Each number is color coded to correspond to a class. The circled numbers represent a change in score when point 4 is removed. The dark blue column indicates points that are not used in the initial score but are included if point 4 was a part of the neighborhood when removed. For example, we see that in the top example a score for a point in a different class went up and the same class went down indicating an edge point. Source: MITRE

To determine which label to assign each point, thresholds on the amount of change in score of same class and different classes are applied. When we remove points labeled as a class cluster, we expect the same class score to change minimally and the different class scores to remain the same. If a point is surrounded by points of the same class then the point should not be in the neighborhood of any point of a different class and should be replaced by points of the same class when removed from neighborhoods of same class points. For points labeled as an outlier we expect the same class score and different class scores to increase. A main point that has a neighborhood with points of a different class, will negatively affect the class data quality score. If that main point is in the neighborhoods of different class points, then the different class quality scores will also be negatively affected. Therefore, removing the outlier will increase all scores. The edge points are a mixed bag. Empirically we've observed that edge points will increase in one score and not the other. For instance, if an increase in the same class score is observed and no increase in different class scores then we label it an edge. This can be imagined as the point having a neighborhood of different class points but not being in the neighborhood of those different class points. And if a small decrease is observed in the same class score and a large increase in different class scores is observed then we also label it an edge point. This scenario can be imagined as a point having a lot of the same class points in its neighborhood and being in the neighborhood of a lot of different class points. Figure 7 provides a visualization of the neighborhood of a cluster point, edge point and outlier point.

The thresholds that are chosen to consider a score has "changed" is a percentile of the distribution of all changed scores and independent for same class or different class. After calculating all the scores when performing leave-one-out we observe the distributions are highly skewed. Most points fall close to minimal change for both same and different classes which is to be expected if we have a candidate dataset for an AI/ML classification solution. Empirically we use the 99$^{th}$ percentile for each score change as the cutoff. If a point falls above the 99$^{th}$ percentile of both score changes then it is an outlier, if it falls above only one cutoff then it is an edge, otherwise it is a class cluster point. In Figure 4-4 an example of the spread of score changes and the corresponding labeled points is shown. Note that the red lines indicate the 99$^{th}$ percentile of the score changes, so there is a lot of data bunched around the origin.

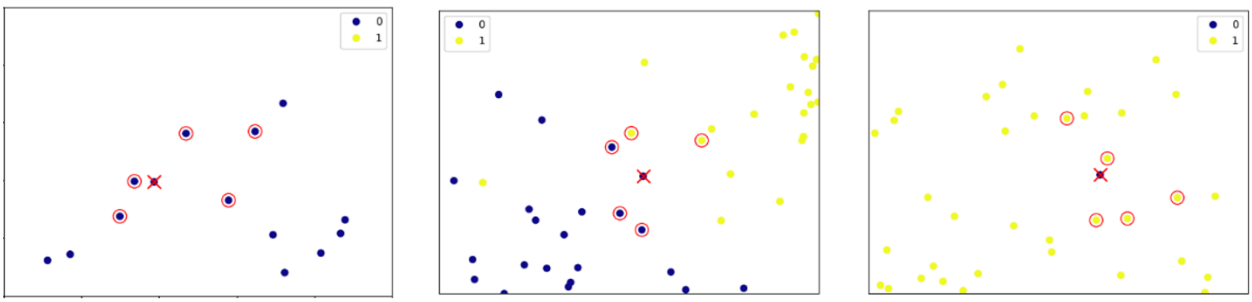



Figure 7. In each plot the point with an X is the central point and the circles are the neighborhood points. The left is the neighborhood of a cluster point. The middle is a neighborhood of an edge point. The right is the neighborhood of an outlier point. Source: MITRE

The thresholds that are chosen to consider a score has "changed" is a percentile of the distribution of all changed scores and independent for same class or different class. After calculating all the scores when performing leave-one-out we observe the distributions are highly skewed. Most points fall close to minimal change for both same and different classes which is to be expected if we have a candidate dataset for an AI/ML classification solution. Empirically we use the 99$^{th}$ percentile for each score change as the cutoff. If a point falls above the 99$^{th}$ percentile of both score changes then it is an outlier, if it falls above only one cutoff then it is an edge, otherwise it is a class cluster point. In Figure 8 an example of the spread of score changes and the corresponding labeled points is shown. Note that the red lines indicate the 99$^{th}$ percentile of the score changes, so there is a lot of data bunched around the origin.

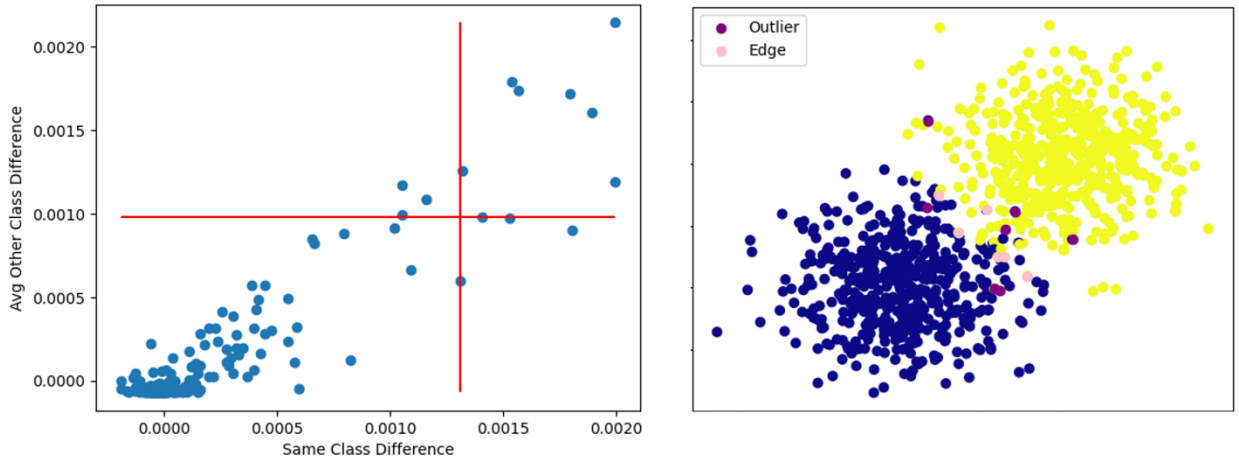



Figure 8. On the left shows a scatter plot of the score differences for same and other class. The red lines are the 99th percentile of each. Each point in the top right quadrant is labeled as either an edge or outlier. On the right is the resulting plot showing which points got labeled as edges and outliers. Source: MITRE

We also do post processing on the points to ensure close points do not have labels that contradict. In the neighborhood of a class cluster point there should not be a point of the same class with the label of outlier. If an edge point is surrounded by too many cluster points of a different class then we change this to an outlier. We determine "too many" cluster points by a percentage of the weighted rankings in the neighborhood of the edge point. This percentage is tunable, but empirically we use if 80% of the weighted rankings belong to a different class then we convert the edge to an outlier. Further work can be done to identify the correct threshold percentage rather than settle for an empirical observation.

Overall, this method helps with identifying potentially problematic points that may decrease the data quality or influence the decision boundary location between class distributions. In our results section we show examples from simulation and well-known datasets highlighting the data points our method picks out.

## 5. RESULTS

The results shown from both methods are from datasets generated through simulation and well-known toy datasets. All simulation data was derived from sklearn [12] and the toy datasets were downloaded through pytorch [10]. We will first show how the data quality score is affected by adjusting parameters known to affect data spread in simulation to demonstrate that the data quality score reacts as expected. Then we will show the data quality scores for well known toy datasets CIFAR10 (image) and 20 newsgroup (text). For the method to label points, we show what data points were labeled as edge and outliers within the simulated and well-known toy datasets. Finally, we can demonstrate how the data quality score correlates with performance on simulated data and if removing outlier points improves the classification score.

### Data Quality Score

The simulations were run for different parameter combinations to observe how changes would affect the data quality score. Overall, the changes in score that were seen align with what was expected. Since a big part of the method is timing, we display timing results in Table 1. These results display how long the method takes to run *after* generating the features. The cells in the table left blank indicate that there was not enough memory to run the simulation. We can see that for all values in the table, they are significantly lower than what would be required to train a deep learning classifier which is the goal. The next tables show the data quality score averaged over the same class score or the diagonal of the computed matrix. Table 2 shows the data quality scores for a changing number of features and samples. Important parameters that were held constant throughout the simulation is the number of classes were 2, and there were no dirty labels. From the table we can see that as the samples increase data quality increases and as features increase data quality decreases. As the number of samples increase and distribution stays the same we most likely see more data being added to cluster centers rather than edges or outliers causing the data to be more compact. As the number of features increase and we use our dimension reduction techniques the reduced data may be stripped of more and more information because even low variable dimensions may have useful information for separability. This is an unsupervised dimension reduction so the risk is that information that would help with separability is being lost. This is not seen as a large issue because our goal is only to evaluate data quality and not classification.

Table 1. The number of seconds needed to run our method with simulated data.

| | | Number of Samples | | | |
|---|---|---|---|---|---|
| | | **1 x $10^3$** | **1 x $10^4$** | **1 x $10^5$** | **1 x $10^6$** |
| Number of Features | **1 x $10^2$** | 0 | 3 | 42 | 578 |
| | **1 x $10^3$** | 0 | 5 | 63 | - |
| | **1 x $10^4$** | 1 | 20 | - | - |

Table 2. Data quality score changes over number of samples and features.

| | | Number of Samples | | | |
|---|---|---|---|---|---|
| | | **1 x $10^3$** | **1 x $10^4$** | **1 x $10^5$** | **1 x $10^6$** |
| Number of Features | **1 x $10^2$** | 0.891 | 0.9283 | 0.96 | 0.976 |
| | **1 x $10^3$** | 0.6972 | 0.7168 | 0.7271 | - |
| | **1 x $10^4$** | 0.6461 | 0.6541 | - | - |

Table 3 shows how the data quality score changes over the parameters of dirty labels, amount of separation and number of classes in the data. The number of samples are held constant at of 1 x $10^5$ and the number of features as well at of 1 x $10^3$. We can see expected results of data quality score degrading when we increase the number of

classes and keep everything else constant. More data in a confined area will most likely lead to less data separability. Also we see the score degradation when we increase the proportion of dirty labels being introduced into the data which is expected. Data separation is a measure of how far apart the centers are when the data is being generated and as the separability increases so does the data quality which is expected. Overall, the score changes align with what we would expect.

Table 3. Data quality score changes over number of classes, data separation, and dirty labels.

Number of classes = 2

| Data Separation | Proportion Dirty Labels | | | |
|---|---|---|---|---|
| | **0** | **0.01** | **0.05** | **0.10** |
| **0.1** | 0.6548 | 0.652 | 0.6405 | 0.6273 |
| **1.0** | 0.6738 | 0.6703 | 0.6556 | 0.6392 |
| **2.0** | 0.732 | 0.7262 | 0.7023 | 0.6754 |

Number of classes = 4

| Data Separation | Proportion Dirty Labels | | | |
|---|---|---|---|---|
| | **0** | **0.01** | **0.05** | **0.10** |
| **0.1** | 0.4296 | 0.4279 | 0.4216 | 0.4138 |
| **1.0** | 0.4464 | 0.4443 | 0.4361 | 0.4263 |
| **2.0** | 0.4494 | 0.4961 | 0.4822 | 0.4467 |

The final simulation compares how data quality score is affected by changes in number of nearest neighbor and number of dimensions we reduce the data down to. We hold all other parameters constant. In Table 4 the data quality score increases by dimension and decreases with an increase of number of nearest neighbors being used. Keeping the dimensions high is expected to help with the data quality score, but why adjusting the nearest neighbors higher negatively affects the score is not as clear. There were only two classes and each class has much more data than nearest neighbors. A plausible explanation would be that the score becomes more normalized to the overall data as the nearest neighbor count become larger. This exploration would be interesting to follow up on in a future study.

Table 4. Data quality score changes over number of nearest neighbors and feature size.

| Reduced Dimension Size | Number of Nearest Neighbors | | | |
|---|---|---|---|---|
| | **3** | **11** | **101** | **1001** |
| **10** | 0.7096 | 0.59 | 0.5235 | 0.5118 |
| **100** | 0.7711 | 0.6738 | 0.6055 | 0.5689 |
| **1000** | 0.9058 | 0.8556 | 0.8106 | 0.7489 |

We also provide visuals of the data quality score for different amounts of overlap, classes and data distributions. In the following simulated data we only simulate in two dimensions so no dimension reduction techniques are used. In Figure 9 we show the overlap of two different simulated datasets and the average data quality score for the same class. Table 5 shows the overlap scores between the classes for each of the two datasets. In the 2 dimension case this aligns with our expectations that more overlap of the dataset would result in a lower data quality score. For the second simulated dataset in Figure 9 we see the overlap scores are not even between the two classes. Based on the table, class 0 is not as separated from class 1 as class 1 is separated from class 0. This can happen when points do not appear in the neighborhoods of points in their own neighborhood.

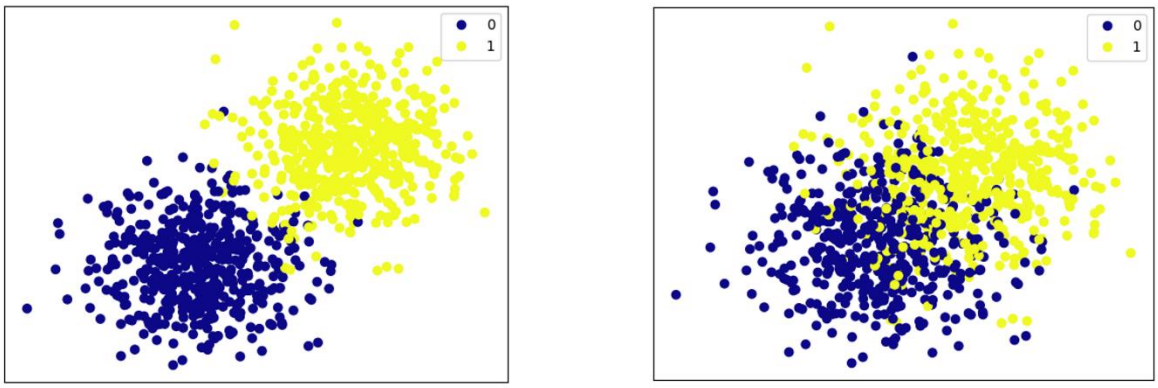


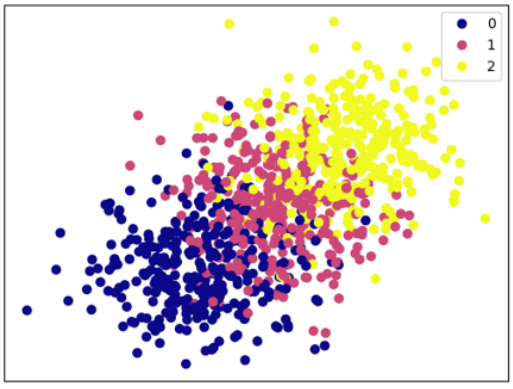

Figure 9. The left simulated data has a data quality score of 0.965 and the right simulated data has a data quality score of 0.763. Source: MITRE

Table 5. Overlap scores for each of the two simulated datasets in Figure 9.

| | 0 | 1 |
|---|---|---|
| **0** | 0.966 | 0.034 |
| **1** | 0.036 | 0.964 |

| | 0 | 1 |
|---|---|---|
| **0** | 0.757 | 0.243 |
| **1** | 0.232 | 0.768 |

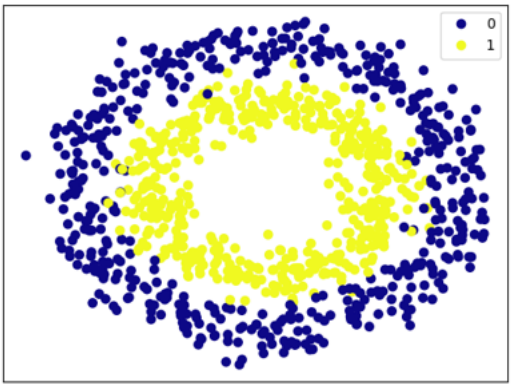
To show separability for more than two classes Figure 10 shows a simulated dataset of three classes with the corresponding overlap scores. We can see that class 1 has a much lower data quality score than class 0 or 2 because it is being overlapped from both directions. Also to note is that the overlap scores between class 0 and 2 are low compared to the other overlap scores.




| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | 0.766 | 0.215 | 0.019 |
| **1** | 0.198 | 0.569 | 0.233 |
| **2** | 0.021 | 0.231 | 0.748 |

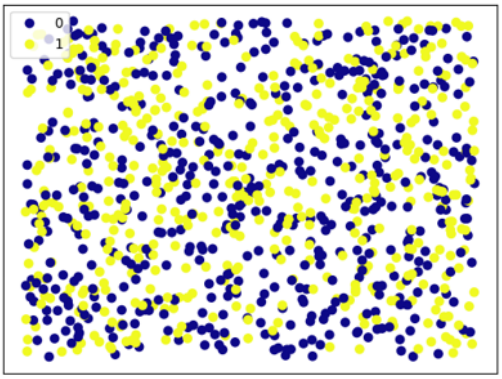
Figure 10. The simulated data and corresponding overlap scores for a dataset with three classes. Source: MITRE

In Figure *11* and Figure 12 we show the data quality scores for different simulated data shaped in unique ways. We show these to demonstrate that our method is flexible to different types of decision boundaries. We do not expect our method to align with any specific classification method. It is important to remember that scores are normalized across rows. This implies that there is not an overall threshold for poor performance. It matters more how the scores along the diagonal compare to the scores off the diagonal.




| | 0 | 1 |
|---|---|---|
| **0** | 0.948 | 0.052 |
| **1** | 0.045 | 0.955 |

Figure 11. Simulated Circles dataset which shows good separation and good data quality scores. Source: MITRE




| | 0 | 1 |
|---|---|---|
| **0** | 0.522 | 0.478 |
| **1** | 0.531 | 0.469 |

Figure 12. Simulated completely random data shows poor data quality scores. Source: MITRE

We now measure the data quality score for well-known datasets CIFAR10 and 20 Newsgroups. These datasets are of images and texts respectfully and are known to be good test sets for various classification methods. Table 6 shows the data quality scores for the CIFAR10 dataset. This dataset contains 10,000 training images and we used EfficientNet Small and ResNet34 for feature generation. Overall, this took 18 seconds. We see expected results that among the animals there is some higher overlap scores as well as with the non-biological categories there is also some high overlap score. Otherwise, the diagonal scores are high indicating a classification solution should be possible for this data. In
Table 7 the 20 Newsgroups data quality scores are shown. We used DistilBert for feature generation. We see these scores are not as high on the diagonal indicating some groups will be hard to discriminate. These include combinations of groups that are not surprising. The overlap scores within the computer topics are all quite high. Also the overlap scores for the religious topics are high.

**Data Labeling**

We also provide examples through simulation and well-known toy datasets of points that have been labeled as cluster, edge and outlier. Examples of simulated 2d data align with data seen in the previous section. In Figure 13 we show labeled data from a 3-class simulated dataset. We see a lot of examples of points that might be considered as along a potential decision boundary between classes. We also see outliers that potentially moved too far into a neighboring data cloud to be close to the decision boundary.

Table 6. CIFAR10 data quality scores.

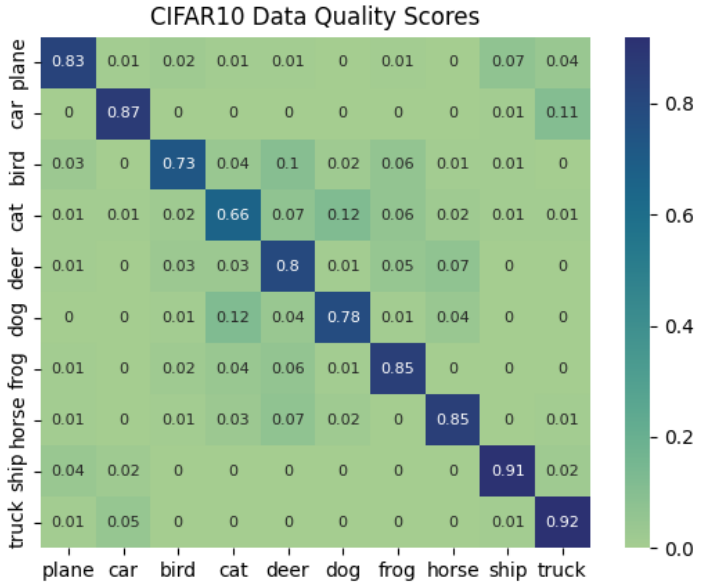
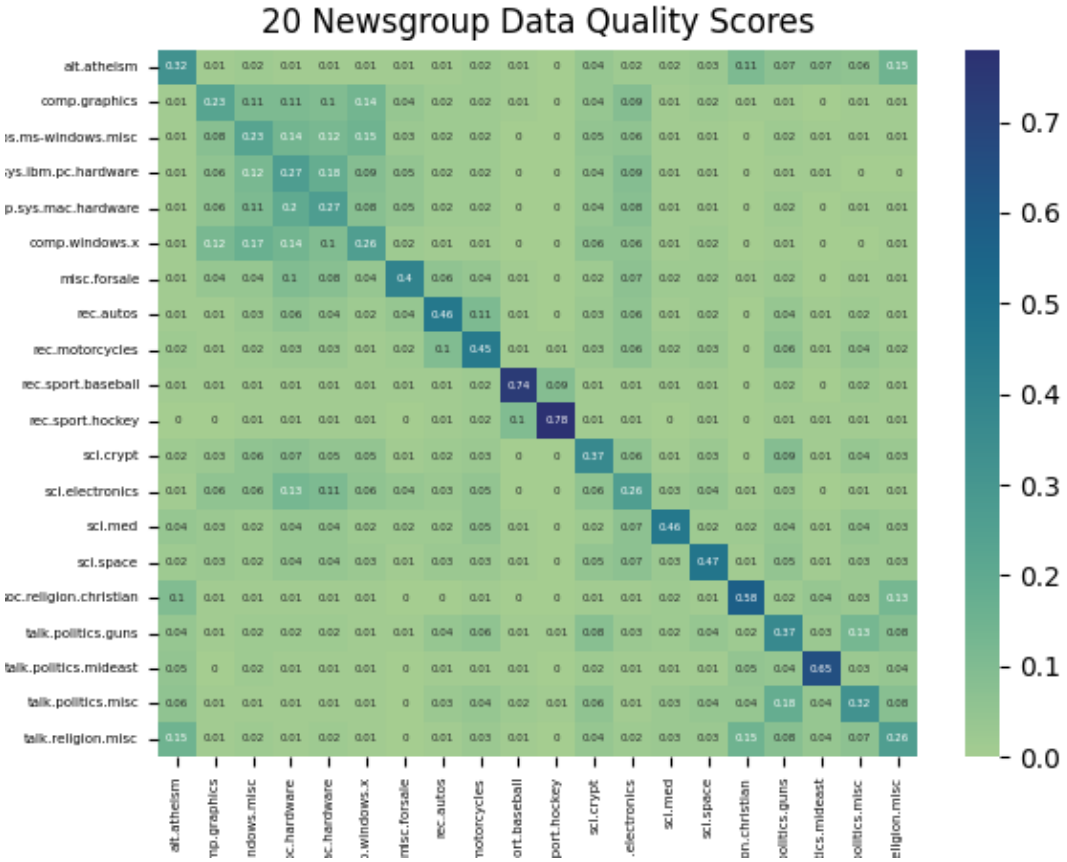



Table 7. 20 Newsgroup data quality scores.



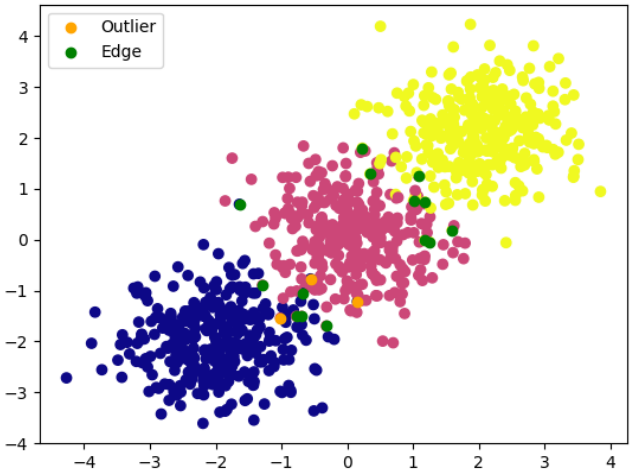



Figure 13. Labeled outliers and edges from a 3-class simulated dataset. Source: MITRE

In Figure 14 the simulated circles data shown in Figure 11 is shown. The points labeled seem appropriate to consider as an edge or outlier. They may warrant additional review to ensure they were labeled correctly.

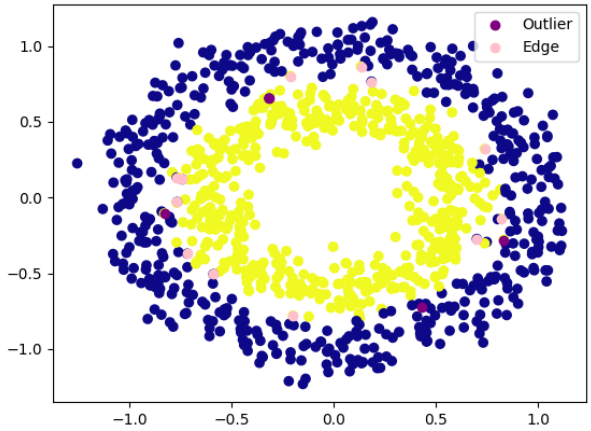



Figure 14. Labeled outliers and edges from the simulated circles dataset. Source: MITRE

Examples from CIFAR10 and the 20 Newgroups datasets are also shown. In Figure 15 we can see examples labeled as outliers by our method. They are both from the class of Horse and show one image that is only a partial view of a horse and another example that is of a human riding a horse. In Figure 16 we show an outlier and an edge from the 20 Newsgroup dataset. On the left is an outlier letter from the Hockey mailing list and on the right is an edge letter from the Atheism mailing list. In the hockey letter there is no mention of hockey and only the city where a hockey team is located so it is understandable this may be considered an outlier and not relevant to hockey. The Atheism letter talks about Jesus so may be along the line with other letters from the Christian mailing list. With our method it is not clear what edge the samples may be on.

Figure 15. Examples of outlier images for the class Horse from CIFAR10.

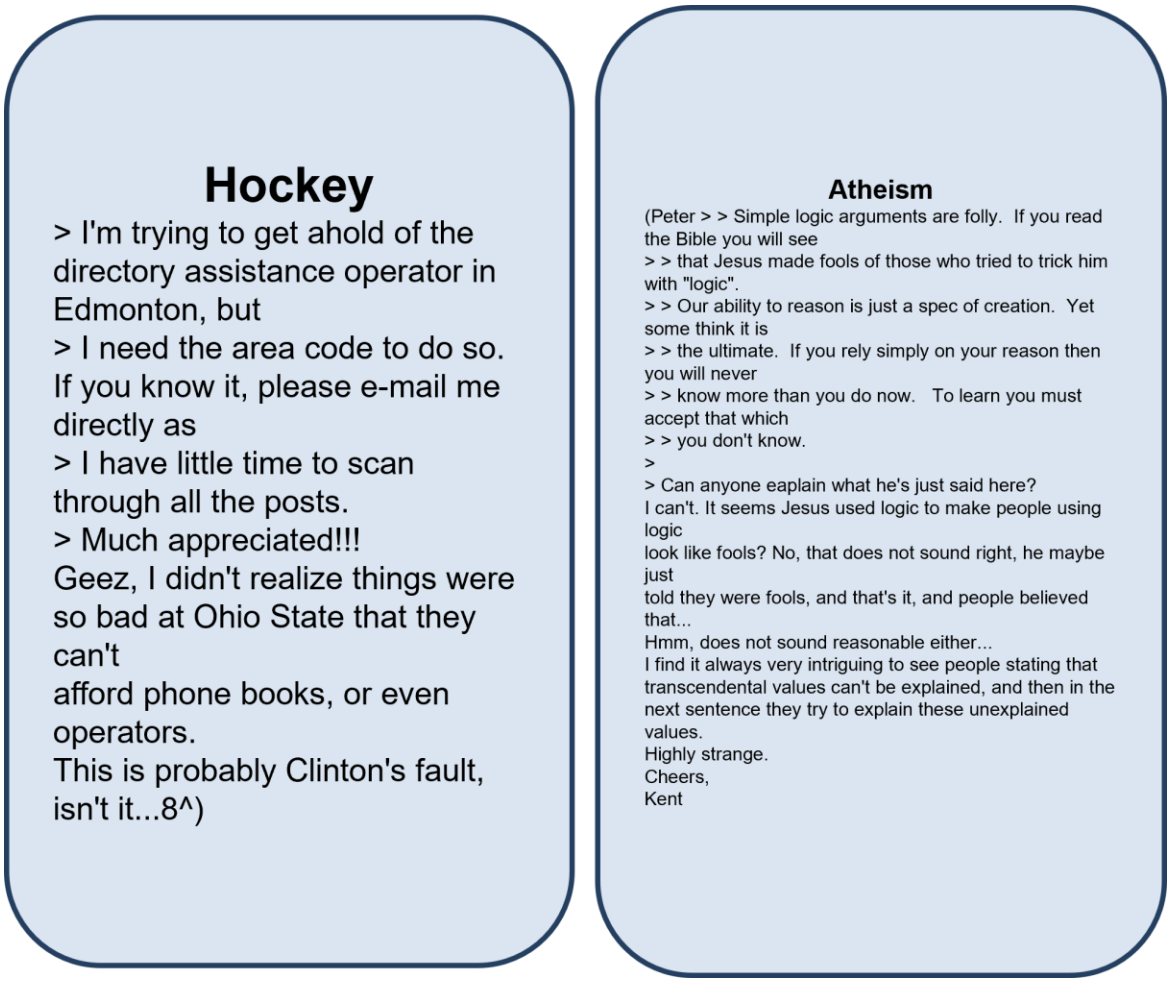



Figure 16. Example letters from the 20 Newsgroup datasets labeled as outlier and edge respectively. The left letter is from the Hockey class, the right is from the Atheism class.

## 6. CONCLUSION

To train a good classification model with deep learning is a resource and time intensive effort. These methods help an analyst understand when that investment may not be worth it. Datasets these days are made up of a large number of samples that are impossible to individually comb through. And even when visually inspected, an analyst may not fully understand what may or may not be separable to a model. Many times, analysts see the quality of their dataset after fitting a model since it is the first opportunity to understand the performance. Before fitting, the methods in this

paper can alert an analyst when data quality may be lacking. However, we do not recommend these methods to be used for a prediction on how good of a performance an analyst can expect.

The data quality score uses an approximate nearest neighbor approach to determine the separability of the data for each class by converting unstructured data into features using pre-trained models. These pre-trained models give a good representation of features out of the box. A part of calculating the data quality score is evaluating individual points for how separated they are compared to other points in the class. We take advantage of this, and use leave-one-out to calculate the discrepancy of data quality scores. The amount of change in score dictates whether we believe the point should be labeled as a part of the cluster center, a point that may affect the decision boundary or an outlier. Points labeled as edge and outlier can be individually evaluated for label change or removal. With the necessity for a large amount of data to train a meaningfully useful model, it is difficult to know how well attuned the data is to an AI/ML classification solution. These methods can provide an early warning that there may be something amiss.

## 7. FUTURE WORK

A component of this work that would be instructional would be to add in an explanation for how the data points can be fixed, or what is wrong with the individual data points that make them be labeled as an outlier or edge point. This involves using counterfactuals [11] to understand what happens when data is changed, or the minimal amount of change that is needed to be classified differently. Being able to move a point in feature space and see the difference in the image space is a difficult problem and still being explored [9].

Another area of future work would be to compare the method if a complete nearest neighbor search was done, rather than an approximate one. This would increase the amount of time substantially but would be beneficial to know how much is lost by speeding up the implementation. Tangentially related is to vary the way scores are calculated based on the rankings. We decided to use a linear ranking system, but other work could compare ranking systems. Finally, another direction to explore is how to distinguish the edge points when there are more than two classes being compared. It is not clear looking only at the scores which classes the points labeled as edges fall near, so a better understanding of the close classes would be a useful addition.

## ACKNOWLEDGEMENTS

The author would like to thank Jesse Galef, James McCeney, Marcus Tyler and Alex Odeh for their thoughtful comments when peer reviewing this work.

## REFERENCES

[1] Arya, Sunil, et al. "An optimal algorithm for approximate nearest neighbor searching fixed dimensions." Journal of the ACM (JACM) 45.6 (1998): 891-923.
[2] Bernhardsson, Erik. “Annoy.” https://github.com/spotify/annoy. 2013.
[3] Bingham, Ella, and Heikki Mannila. "Random projection in dimensionality reduction: applications to image and text data." Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. 2001.
[4] Fernández, Ángela, Juan Bella, and José R. Dorronsoro. "Supervised outlier detection for classification and regression." Neurocomputing 486 (2022): 77-92.
[5] Guan, Shuyue, and Murray Loew. "A novel intrinsic measure of data separability." Applied Intelligence (2022): 1-17.
[6] Ho, Tin Kam and M. Basu, "Complexity measures of supervised classification problems," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 3, pp. 289-300, March 2002, doi: 10.1109/34.990132.
[7] Lorena, Ana C., et al. "How complex is your classification problem? a survey on measuring classification complexity." ACM Computing Surveys (CSUR) 52.5 (2019): 1-34.
[8] Johnson, William B. "Extensions of Lipschitz mappings into a Hilbert space." Contemp. Math. 26 (1984): 189-206.
[9] Kinney, Mitchell. “Exploring Latent Space for Classification Explainability.” *MITRE Technical Report* (2022).
[10] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library [Conference paper]. Advances in Neural Information Processing Systems 32, 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[11] Pearl, Judea. "The seven tools of causal inference, with reflections on machine learning." *Communications of the ACM* 62.3 (2019): 54-60.
[12] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830.
[13] Tipping, Michael E., and Christopher M. Bishop. "Mixtures of probabilistic principal component analyzers." Neural computation 11.2 (1999): 443-482.
[14] Wolf, Thomas, et al. "Huggingface's transformers: State-of-the-art natural language processing." arXiv preprint arXiv:1910.03771 (2019).

## NOTICE

**This software (or technical data) was produced for the U. S. Government under contract 17-D-0211, and is subject to the Rights in Data-General Clause 52.227-14, Alt. IV (DEC 2007)**