Predicting FDA Risk Classification of Medical Devices Using Machine Learning Techniques and Public Data

Fahim Tajwar
Undergraduate Student, Computer Science
Stanford University
California 94305, USA
tajwar93@stanford.edu

Zach Harned
J.D. Candidate, Stanford Law School 2020 &
M.S. Stanford Symbolic Systems Program 2020
Stanford University
California 94305, USA
zharned@stanford.edu

Abstract

The Food and Drug Administration (FDA) plays a critical role in promoting the safety and efficacy of a number of products ultimately used by or on the public, including medical devices. The FDA classifies these devices in order of increasing risk (viz. Class I, II, III), each with an attendant clearance process, becoming more onerous and expensive with the associated increase in risk. As an innovator, it can be tremendously valuable to know what risk class the developed device is likely to be classified as, hence we have developed and experimented with different machine learning models that takes standard input features of a medical device that must be reported to FDA about the device — and hence are easily accessible — and output the most likely classification of the device. Our paper is application based, we describe developing models (e.g. weighted softmax classifier, support vector machine, a fully connected neural network) to do binary classification of a device's risk level into the classes Risky (Risk Class III) or Not Risky (Risk Class I and II), and show that this can be done with reasonable accuracy. We then explore the possibility to do the classification of a device's risk level into our 3 original risk classes used by FDA. We explore the difficulties (i.e. lack of features and high bias problem) for this 3 class classification, and also describe our methods and experiments (i.e. using SVM with Gaussian Kernel, extracting word features from device description) to overcome these limitations. We finally show that predicting the FDA risk classification of Medical devices can be done with reasonable accuracy.

1. Introduction

The United States Department of Health and Human Services (HHS) is a cabinet-level department with the mission to "enhance and protect the health and well-being of all Americans [4]. HHS is home to the US Food and Drug Administration (FDA). It is estimated that the products regulated by the FDA account for roughly 20 cents of each dollar spent by consumers in the US [6].

Given the rising importance of digital health, this paper focuses on the FDA's Center for Devices and Radiological Health (CDRH), which is tasked with "regulating firms who manufacture, repackage, relabel, and/or import medical devices sold in the United States." [3] A primary mandate of CDRH is to ensure the safety and effectiveness of medical devices marketed in the US [2]. The chief legal authority granting FDA this responsibility is found in the Federal Food and Drug Cosmetic Act (FD&C Act) passed by Congress in 1938.[1] One of the ways that CDRH regulated medical devices is by categorizing them into one of three classes: Class I, Class II, and Class III, in terms of increasing level of risk. Each of these classes is associated with a different clearing process, with each process becoming increasingly difficult (and expensive) the higher the risk/class of the device.

Therefore it is very helpful for entrepreneurs, innovators, and device manufacturers to be able to accurately estimate what classification their device is likely to receive, and hence whether the approval process will be cost effective, influencing whether production is viable. So we use machine learning methods, specifically, weighted softmax classification, support vector machines and fully connected neural networks to first take the publicly available features (e.g. whether it is implanted or not, in what medical setting is it used, etc.) and then classify it as Risky (equivalent to FDA risk class III) or Not Risky (equivalent to FDA risk class I or II). Later, we address the issue of predicting the risk class directly into the 3 classes, develop a support vector machine with Gaussian kernel, and extract additional

[1] This is by no means the only source. See the FDA's history of medical device regulations and oversight webpage for additional details.

features from the dataset, to implement models for this purpose.

2. Related Work

The FDA risk classification is, at least in the final step, decision of humans. But it would be helpful for us nonetheless to explore the factors influencing this decision, and see if we can predict it ourselves. We did not find any direct work done before on our particular task. However, we found it interesting to see how machine learning is used on social science data, and as described by N.C. Chen et al [1], we saw some good results obtained from transforming qualitative data into numerical values. This paper does not go into detail of any particular social science issue, but is a good summary paper in terms of how to use machine learning in social science and the existing norms and practices in this field.

Since we were expecting to use SVMs for our task, we were researching it, and found this paper by Xiaohong Wang et al [9] that gave us a very nice understanding of SVMs, and how to interpret the results of a SVM model. We were also looking for better update methods than Stochastic Gradient Descent, and found this paper by Mark Schmidt et al [7] that gave us a better update rule where we did not have to tune the learning rate. This gave us a choice to reduce the validation set size and increase the training set size if we wanted. (We found a scikit learn implementation of this update algorithm and used that)

Finally, we explored possible ways to extract information from text (as we shall see later), we explored several papers like the one by Patrick Tierney [8], on how to extract natural language information. In the end, the method we used was very simple, and did not give good results, however exploring text feature extraction methods was an important part of our project.

3. Dataset

3.1. Features and General Information

We use a public dataset compiled by International Consortium of Investigative Journalists - ICIJ (`https://www.icij.org/`); the downloadable database can be found at `https://medicaldevices.icij.org/p/download`. We started with the 'devices.csv' dataset.

Our dataset has information on around $90,000$ devices, we randomly sampled $8,079$ of them and pre-processed them to give us 6 features and the risk class of a particular device - our ground truth label. These six features are:

1. quantity of the devices produced
2. distribution of the devices
3. device recall
4. device implantation
5. medical setting for device usage
6. a string containing the description of the device

Feature (1) consists of whole numbers and is the only continuous variable among our features, hence it was the only one that was scaled. We scaled this feature using mean normalization. Feature (2) is a categorical variable consisting of three options: the device receives either:

(a) worldwide distribution

(b) US national distribution

(c) only state-by-state distribution

Feature (3) is a binary variable asserting that either the device has or has not ever been recalled before. Feature (4) is also a binary variable, asserting that the device either is or is not implanted into the patient's body. Feature (5) is a categorical variable asserting that the device will be deployed in one of 15 possible medical settings, which are the following: anesthesiology, cardiovascular, clinical chemistry and clinical toxicology, dental, ear nose and throat, gastroenterology-urology, general and plastic surgery, general hospital and personal use devices, hematology and pathology, immunology and microbiology, neurological, obstetrical and gynecological, ophthalmic, orthopedic, physical medicine, and radiology. Feature (6) is the technical description of the device in text.

We also process this dataset to form a modified one, where Risk Class I and II are classified with a the ground truth label Not Risky, and Risk Class III with label Risky, giving us a dataset with two classes.

3.2. Class Imbalance

One important thing to note is that the classes in our dataset are highly imbalanced - with Risk Class II (in our original dataset) and Not Risky (in our modified dataset) having about 3 times as many data points as the other classes combined. In our dataset of 8079 datapoints, the initial frequency distribution per class is as follows:

1. Risk Class I: 1393
2. Risk Class II: 5850
3. Risk Class III: 836

This highly influences all our later work, since all our models had to be built with the approach to minimize the effect of this class imbalance.

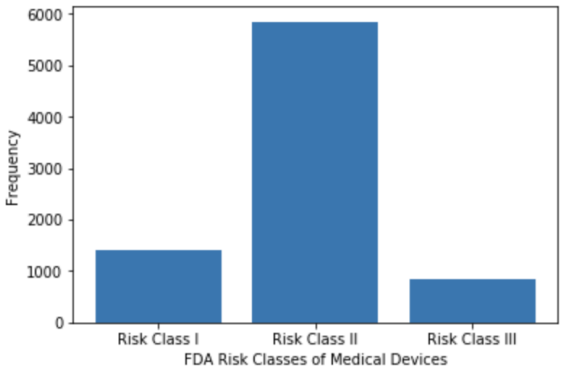

Figure 1. Histogram of risk class frequency distribution

3.3. Dataset split

We divide our dataset into the usual $60-20-20$ split, i.e. we randomly sample 60% of the data into the training set, 20% into validation set, and 20% into test set. All training and subsequent experimentation on hyperparameters is done using the training and validation dataset, and once we obtain the model best performing on the validation set, we test it on our test set and report the final results as our model's performance statistics.

3.4. Text Feature Extraction

For our 3-class classification problem, in order to introduce new features for our dataset, we take the most frequent 30 words in our dataset's device descriptions, and add these words' frequency in a device description as additional features. This creates another 30 new features.

4. Methods

We implement different machine learning models for our classification task. First we try to solve our simpler binary classification task described before. For this, we implement a weighted softmax classifier, a support vector machine and a fully connected neural network with two layers. We train our models using a combination of the first five features described in section 3.1, since only they have numerical value. We achieve reasonable performance on the job, but these same models do not work well for our general 3-class classification task. We explore the possible reasons for this, and use two other methods - a support vector machine with a Gaussian kernel, and extracting additional text features from device description, described as feature (6) in section 3.1.

4.1. Dealing with Class Imbalance

For all of the methods described later, we have to take care of the class imbalance in our dataset described in section 3.2. To do so, we follow two approaches, in the first one, we implement weighted loss functions, so that misclassification for more populous classes are penalized less harshly than misclassification for less populous classes. For any class k, if the frequency of that class is f_k, then we choose the weight for that class as:

$$w_k = \frac{\sum_{i=1}^{K} f_i}{f_k} \tag{1}$$

where K is the total number of classes.

In the second method, we down-sample the populous classes, i.e. we randomly sample a subset from our most populous class so that our new dataset has even distribution of class frequency. In this case, the weight for each class is 1. We try both methods, and report the best results.

4.2. Weighted Softmax Classification

We implemented a general softmax classifier for K classes. The unit softmax function $S : \mathbb{R}^K \to \mathbb{R}^K$ is defined by:

$$S(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{2}$$

The parameter for our model is the weight matrix W and bias vector b. For any input x,

$$P = S(Wx + b) \in \mathbb{R}^K \tag{3}$$

is a vector, such that P_i represents the probability for input x to be in class i. For any input x, we classify it with the label that has the highest probability.

We use a weighted cross entropy loss function, i.e. for any training input $(x^{(i)}, y^{(i)})$, let $s = Wx + b \in \mathbb{R}^K$, then the loss per input is,

$$J_i = -b_{y^{(i)}}(s_{y^{(i)}} - \log(\sum_{k=1}^{K} e^{s_k})) \tag{4}$$

where b_i is the weight for class i, defined in section 4.1. The total training loss is the sum of this loss for all training input.

4.3. Support Vector Machine (with Linear Kernel)

We implement a general support vector machine for K classes. For this, our model has a weight matrix W and bias vector b as parameters. Consider any training input $(x^{(i)}, y^{(i)})$. Define the score vector $S^{(i)} \in \mathbb{R}^K$ as usual:

$$S^{(i)} = Wx^{(i)} + b$$

Then the SVM loss for this input is defined as:

$$L = b_{y^{(i)}}(\sum_{k \neq y^{(i)}} \max(0, S_k^{(i)} - S_{y^{(i)}}^{(i)} + \Delta))$$

where Δ is the margin of our support vector machine, usually set to 1, and b_i is the weight for class i, obtained as in section 4.1.

Using this loss and its gradient, we train our SVM classifier.

4.4. Fully Connected Neural Network

We implement and experiment with various simple architectures of fully connected neural networks. Generally such a model works on the multi-layer perceptron algorithm and is comprised of a input layer, various fully connected hidden layers, and a fully connected output layer.

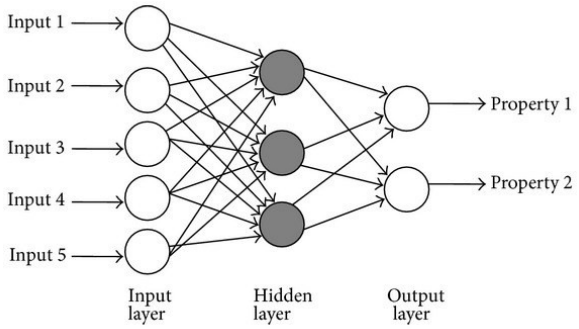

Figure 2. Fully Connected Neural Network [5]

For our model, we use a single hidden layer of size 3, and an output layer of size 2 or 3 depending on which classification task we are handling. In between the layers, we use ReLU non-linearity, so that our model can learn non-linear function. For any input $x \in \mathbb{R}$, we have

$$ReLU(x) = \max(0, x) \tag{5}$$

And in the last layer, we use a weighted cross entropy loss as described in section 4.2.

4.5. Support Vector Machine with Gaussian Kernel

The Gaussian kernel function between some arbitrary $x, x' \in \mathbb{R}^n$ is defined as:

$$K(x, x') = exp(-\frac{||x - x'||^2}{2\sigma^2}) \tag{6}$$

where σ is a free parameter.

Here instead of using the raw features directly, we define the similarity features f, which is equal to the Gaussian kernel of a landmark and another sample point. Then we train a support vector machine on these similarity features as described in section 4.3. This method implicitly maps our dataset into a higher dimensional feature space, thereby helps with the issue of under-fitting if there is any.

5. Experiments and Results

5.1. Hyperparameter Tuning

We use our validation dataset (as described in 3.3) to tune our hyperparameters. We mainly tune two hyperparameters - learning rate and regularization strength. For our neural network, we also tune the size of the hidden layer using our validation set. Some of our models are trained using Stochastic Average Gradient (SAG) method [7], we do not need to tune the learning rate for these models, and we only tune the regularization strength in these cases.

5.2. Examining the Feature Space

Training our models on the first five features of our original dataset (described in 3.1) kept giving very poor results. Upon plotting the dataset using various features, and training a simple model on different features, we found that the including quantity produced (feature (1) in section 3.1) deteriorated the performance significantly.

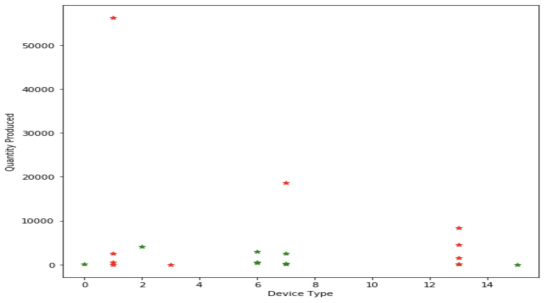

Figure 3. Quantity produced vs device type plot

We got visual confirmation that this feature has no correlation with device risk classes altogether (above is one such visualization, with green points being not risky and red points being from risky class). We decided not to use this feature at all for all of our future models.

5.3. Binary Classification Results

We train and test a weighted softmax classifier, support vector machine (with linear kernel), and a fully connected neural network, for our binary classification task. With some experimentation, we were able to achieve reasonable performance for this task. The summary of these models' performance, on our test set randomly sampled from our down-sampled dataset as described in section 3.3, is illustrated later in this section.

The neural network is our best model so far for the binary classification job, due to its ability to learn more complex decision boundaries, compared to the softmax and SVM classifiers, which can only form linear decision boundaries.

The area under the ROC curve for our SVM classifier is 0.88, which also shows that we have good models.

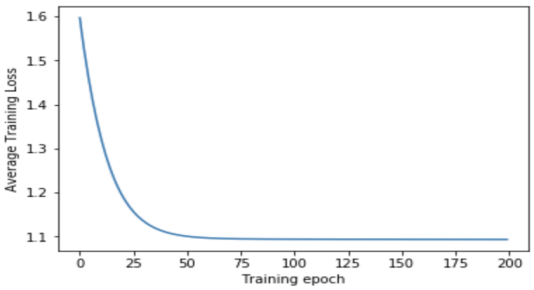

Figure 4. Training loss vs epoch for our neural network

Statistics	Softmax	SVM	Neural Net
Accuracy	80.8%	84.1%	85.9%
True Positive Rate	85.8%	88.9%	86.6%
False Positive Rate	24.2%	20.9%	14.9%
False Negative Rate	14.2%	11.1%	13.4%
True Negative Rate	75.8%	79.1%	85.1%
Precision	78.4%	81.7%	88.0%
Recall	85.8%	88.9%	86.6%
F1 Score	0.82	0.85	0.87

Table 1. Binary Classification Model Performance

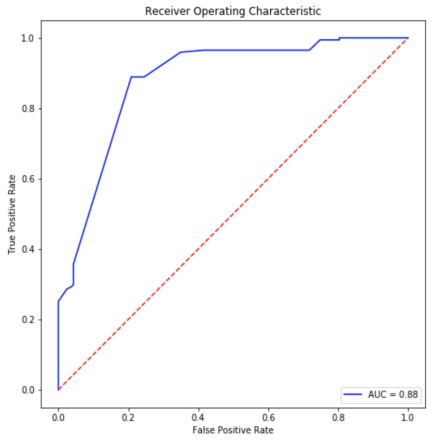

Figure 5. ROC curve for support vector machine for the binary classification task

5.4. Poor Performance on 3-class Classification and High Bias Problem

The models described in sections 4.2, 4.3 and 4.4, despite having good results on our two class classification, performs very poorly when they are trying to predict the FDA risk class of a device. Either their accuracies drop to approximately 30%, or their predictions are skewed towards the more populous class.

In order to understand the underlying problems that these models are facing, we decided to run the following diagnostic test: we would take our training and test dataset, as described in 3.3.

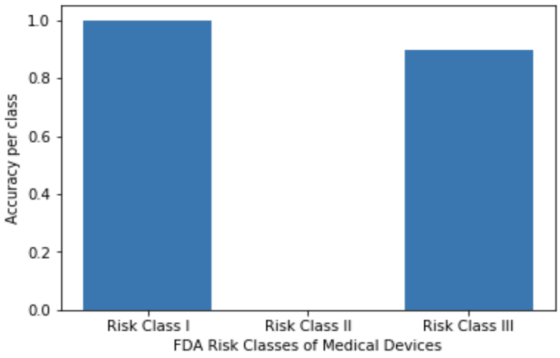

Figure 6. Accuracy per class for the weighted softmax classifier

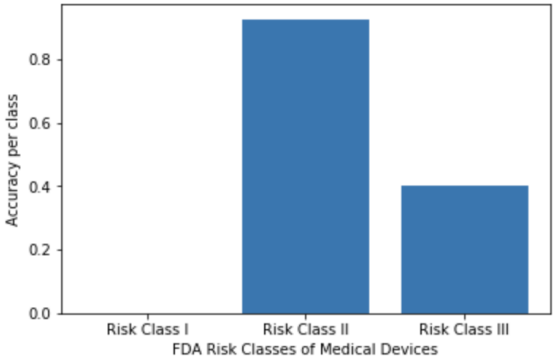

Figure 7. Accuracy per class for SVM classifier (linear kernel)

Then instead of training our model on the entire training set, we would train it on a small subset of the training set that is randomly sampled. Next we shall test this model on both our training and test dataset and record the errors. To not run into problems related to sampling, for any subset size, we will randomly sample a subset from our training set 5 times and record the average training and test error for that training subset size.

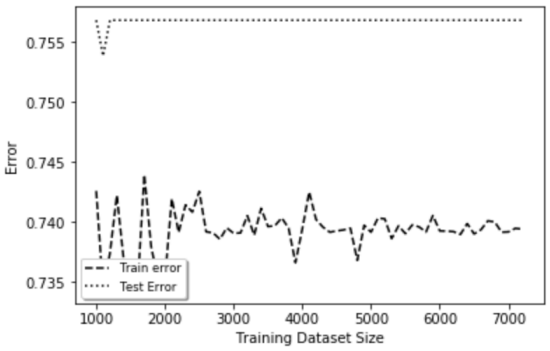

Figure 8. Training and test errors vs training dataset size, for weighted softmax classifier

We see that the training and test error both are very high

(75%), and they remain nearly constant as we increase the size of our training dataset. Training and test error are also quite close together. This shows that our models and dataset have high bias problem, the models are underfitting the dataset, and we need more features to improve our 3-class classification.

5.5. Support Vector Machine with Gaussian Kernel

Since our number of features is small (4), and size of dataset is intermediate (8079), we decide to train a support vector machine with a Gaussian Kernel (as described in 4.5) for our 3-class classification task. As expected, this improves the overall accuracy from around 30% to 64%, and gives us the best accuracy distribution per class for our 3-class classification task. We see that it accurately classifies any class at least 55% of the time, and risk class I and III are accurately classified in 87% and 86% test cases.

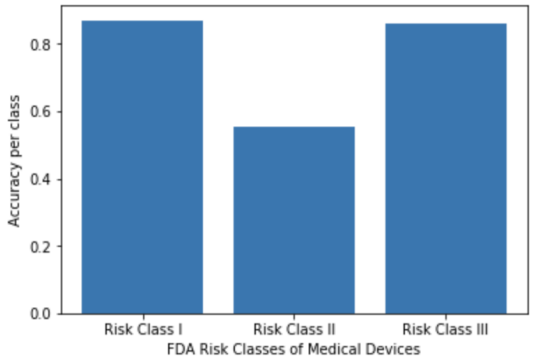

Figure 9. Per class accuracy distribution for SVM with Gaussian Kernel

In this way, using a Gaussian kernel helps us get better results for our 3-class classification, and also gives us further evidence that our original models were suffering from high bias (underfitting) problem and did not have sufficient features.

5.6. Word feature extraction

We extract word features from the device description in our dataset, as discussed in 3.4, and train our previous models on this augmented feature dataset. Unfortunately, this generally gave rise to decrease in performance - either in loss of total accuracy, or skewing the classifier towards more populous classes. This shows that either the simple word count features we extracted do not contain any additional information, or we had to properly extract the word features, i.e. consider a bigram model, or lemmatize/pre-process the words in the description. At any rate, this did not give any improvement in performance.

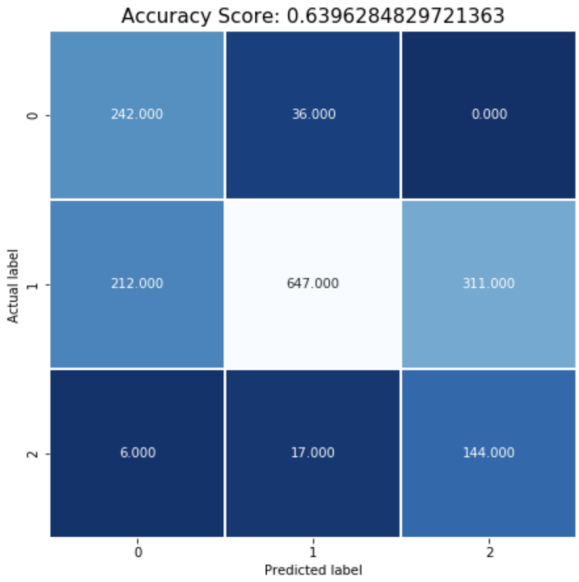

Figure 10. Confusion matrix for SVM with Gaussian kernel

5.7. Error analysis and further insight

For the error analysis, we simply printed a few misclassifications, and also explored whether a certain feature vector, x, has multiple truth values y for it in our dataset. We found that this is the case, there are 159 datapoints in our initial feature space that have multiple truth values for them. This means no matter which label our model predicts for such a feature vector, it always has a chance of having another label, and if multiple of these feature vectors are in our test set, we will classify the same feature vector correct a couple times and wrong another couple of times. Perfect classification on this dataset is thus impossible unless we happen to pick a very biased test set, which definitely will not represent a real world scenario.

We assume that there may be some other factors behind the human classification decision of these data points in FDA that is not present in this feature space. For example, an implanted orthopaedic device can have different degrees of risk levels depending on where it is implanted. In this case, our device description should contain more information, but we could not extract any meaningful information from there. In future, one might try to look for/collect information on these additional risk factors of a device as features as well.

6. Conclusion and Future Work

We managed to obtain satisfactory performance on our 2-class classification task, and reasonable performance on our 3-class classification task. We explored how lack of features might hinder one from making a good machine learn-

ing model for a particular dataset, and also explored some ways one can try to solve this issue. We saw that in case where our number of features is small but size of dataset is intermediate, using a Gaussian kernel improves the performance of our machine learning models. We also explored the importance of taking the class imbalance into consideration, and saw how methods like using a weighted loss function, down-sampling etc. can help us get a balanced classifier for these cases.

For future, we would suggest looking for and gathering additional features on these medical devices that would help one understand the classification task better. We would also suggest looking into better natural language feature extraction, since we believe the device description might have key insights behind a device's risk classification. And finally, we would look for different neural network architectures, and optimization of our existing models to ensure better performance.

Code

The code for this project is in the public github repository: `https://github.com/tajwarfahim/CS-229A-Project-Machine-Learning.git`

All the results are there as well. The code was mainly written using python. Each code file contains the name of its author, and also any additional references (i.e. code taken from other public repositories). Most of the code was written by the author of each file, and not much outside code was taken (not enough to have a co-authorship of this paper). The initial experiment results have all the models coded in numpy, no additional library was used. Subsequent models often use scikit-learn and other libraries for better performance and speed. The neural network was coded using pytorch.

Contribution

Zach Harned worked on finding relevant works, gathering and cleaning up the dataset, inspecting the dataset. Fahim Tajwar worked on building and testing the models, and interpreting the results.

Acknowledgement

We would like to thank the teaching team of CS 229A, Spring 2019, Stanford University for their valuable advice and help on this project. We also would like to thank Juan Carlos Sosa Hernandez, Stanford University, and Rittik Basak, Bangladesh University of Engineering and Technology, for their help in reviewing and proof-reading this paper, as well as giving valuable advice.

References

[1] N.-C. Chen, M. Drouhard, R. Kocielnik, J. Suh, and C. R. Aragon. Using Machine Learning to Support Qualitative Coding in Social Science: Shifting The Focus to Ambiguity, March 2018.

[2] C. for Devices and R. Health. About the Center for Devices and Radiological Health - CDRH Mission, Vision and Shared Values, accessed September 17, 2018.

[3] C. for Devices and R. Health. Overview of Device Regulation, accessed September 13, 2018.

[4] A. S. for Public Affairs (ASPA). About HHS, February 3, 2015.

[5] P. Mathur. A Simple Multilayer Perceptron with TensorFlow, May 1, 2016.

[6] O. of the Commissioner. FDA Basics - Fact Sheet: FDA at a Glance, accessed September 13, 2018.

[7] M. Schmidt, N. L. Roux, and F. Bach. Minimizing Finite Sums with the Stochastic Average Gradient, 2017.

[8] P. Tierney. A qualitative analysis framework using natural language processing and graph theory, 2012.

[9] X. Wang, S. Wu, X. Wang, , and Q. Li. SVMV–a novel algorithm for the visualization of SVM classification results, 2006.