# Predicting FDA Risk Classification of Medical Devices Using Machine Learning & Public Data

Fahim Tajwar (tajwar93@stanford.edu), Zach Harned (zharned@stanford.edu)
CS 229A: Applied Machine Learning, Spring 2019, Stanford University

## Motivation

The Food and Drug Administration (FDA) classifies medical devices in order of increasing risk (risk level I, II and III), each with an attendant clearance process, becoming more onerous and expensive with the associated increase in risk. We experiment with different machine learning models, that will take various features of a medical device and predict its risk class. This can potentially save companies a lot of resources if they know their devices' risk classes and take measures accordingly to make sure they are safe and get clearance easily.

## Dataset

- We use a public dataset compiled by International Consortium of Investigative Journalists.
- The dataset contains 90,000 devices' information and risk classes, we randomly sampled 8079 of them.
- For each data point, we have six features and the risk class, which is our ground truth label.
- Generally there are three risk classes. For some models we also label our data into two classes – "Risky" (risk class III) and "Not Risky" (risk class I and II). There is **high class imbalance** in the data.

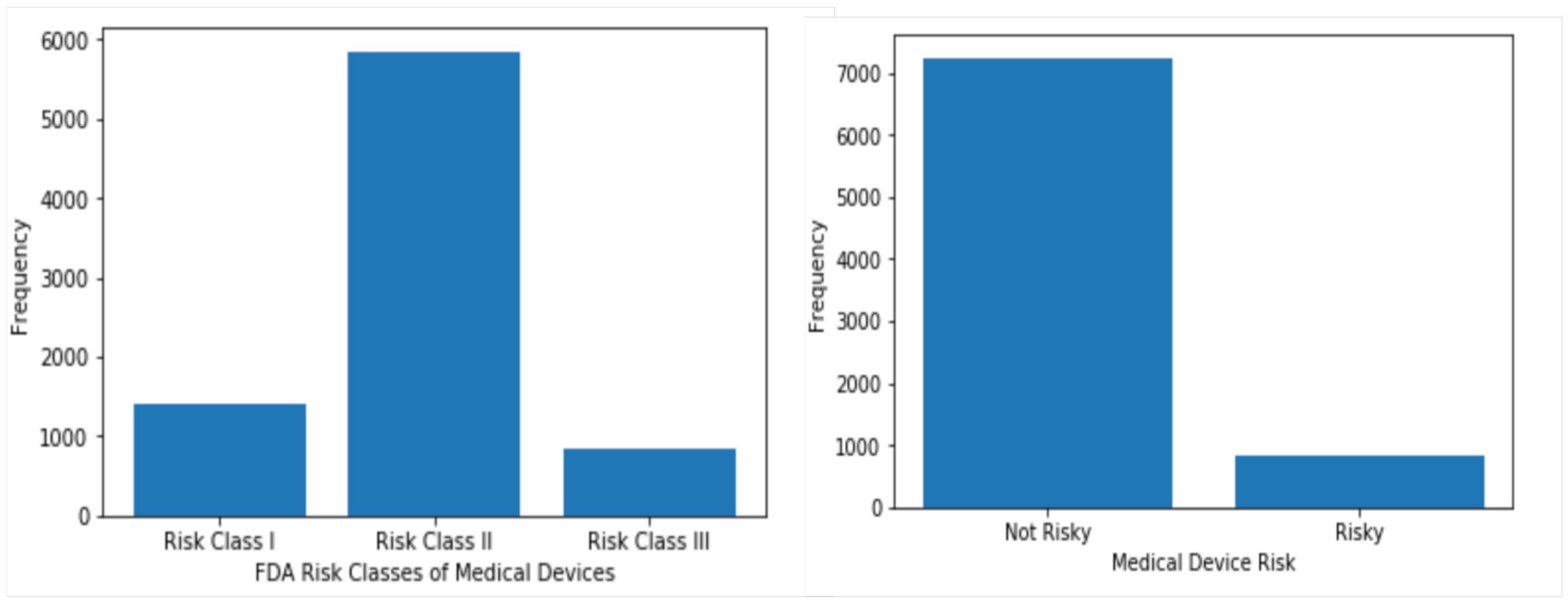

**Figure: Frequency distribution of different risk classes.**

- Features include quantity produced, distribution region, device recall, implantation, device type/medical setting and description

## Binary Classification – Methods and Results

- First, we try to solve an easier problem, trying to classify the devices using the numeric features we have into two classes – "Risky" (Risk level III) or "Not Risky" (Risk level I)
- We randomly divide our dataset into three separate datasets – with 60% data going to training set, and 20% going to validation and another 20% goes to test set.
- We implement and train different models for this classification job – including **weighted softmax classification** (due to class imbalance), **support vector machine** (with a linear kernel), and a **fully connected neural network** with two layers.

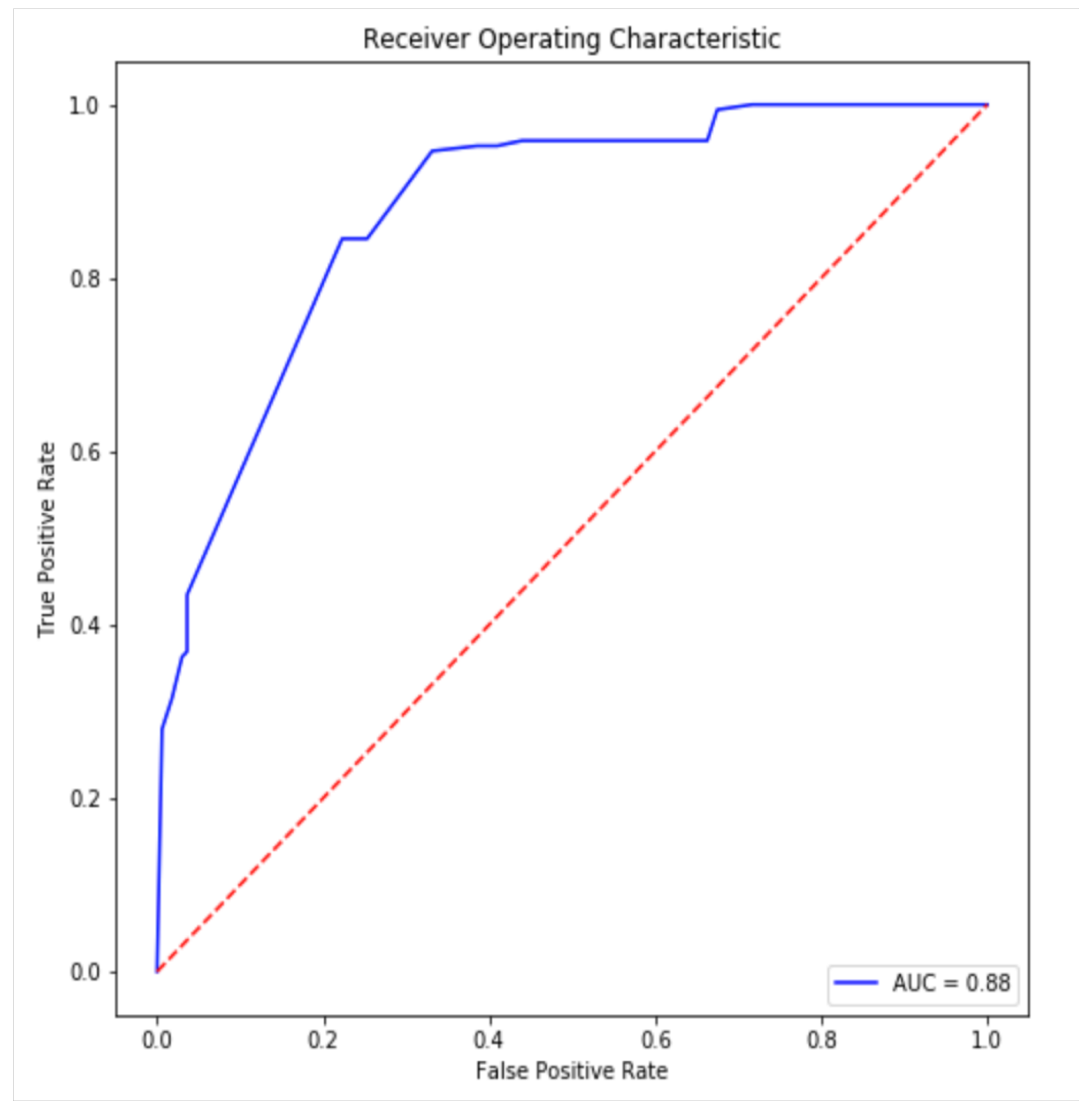

**Figure: ROC curve for weighted softmax classification**

| Model<br>Statistics | Softmax | SVM | Neural Network |
|---|---|---|---|
| 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% |
| Accuracy | 80.8% | 84.1% | 85.9% |

**Figure: Summary Performance Statistics of the models on down-sampled data**

Best performing model for this classification task is our simple neural network.

## Three Class Classification – High Bias Problem

- Using the above models for our general 3 class problem kept giving bad performance.
- The train/test error remain very high **(~75%)** and does not change much as we increase the training dataset size. Also train and test errors are very close together, showing that we have a **high bias** issue.
- Our model is **underfitting** for the 3 class problem and we need more features!

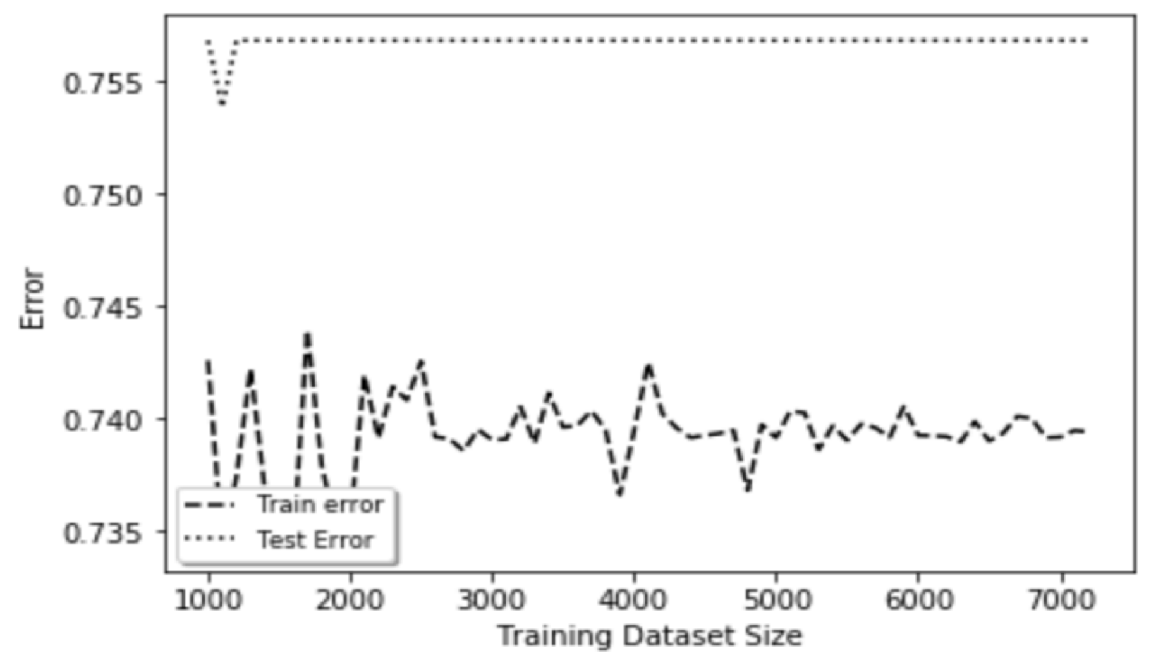

**Figure: Train/test error vs training dataset size**

## Gaussian Kernel and Device Description to the Rescue!

- **Support vector machine with Gaussian kernel** helps us with our model underfitting problem, since a Gaussian kernel implicitly maps the input to a higher dimensional feature plane.
- We also extract features from the device descriptions (word counts) and add them as features, and test our models on this new set of features.
- The result is promising – the error rate drops from **75%** to **~36%** using Gaussian Kernels, and adding the count of the top 30 most frequent words in device description to our feature space decreases the error by another **~3%.** We also get a pretty even accuracy per class distribution.

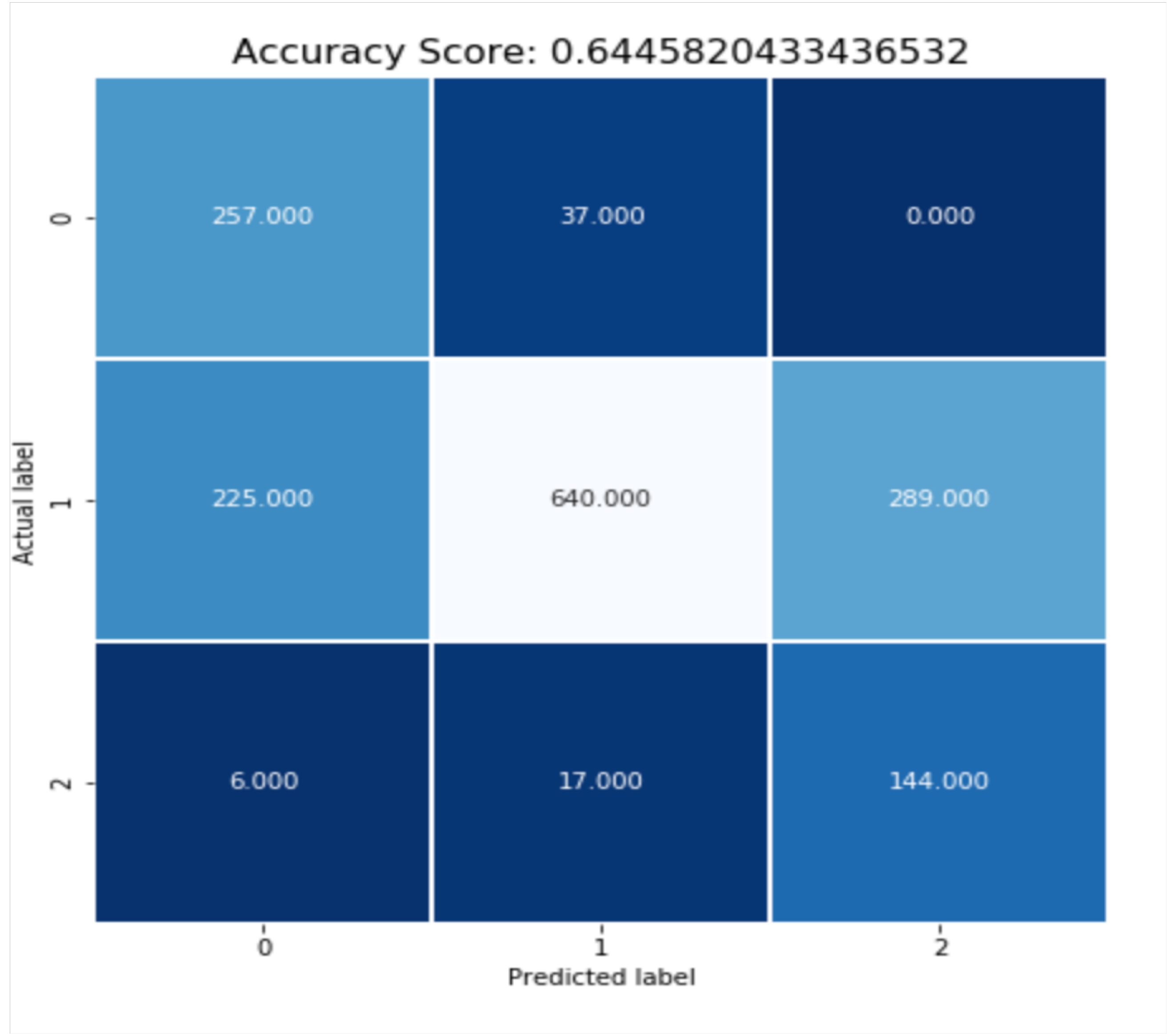

**Figure: Confusion Matrix for SVM with Gaussian Kernel**

## Insight and Future Work

- It is impossible to have perfect performance from our original features, we found 159 feature vectors that are labelled with multiple classes, so they can't be predicted perfectly.
- There might be some other human motivations and reasons for these classifications, not explicitly in the features of our dataset.
- We suggest working on getting more features for our dataset, and work on models to generate better features/ make models that are able to do better prediction based on the features we have.