**Machine Learning - CS 7641 – Assignment 1**

Jeremy Martinez

Computer Science Dept., Georgia Institute of Technology

Abstract

Two common applications of machine learning, among many, are the ability to process natural language and improve health care. Natural Language Processing (NLP) and Medical data are two subjects this paper will explore. The data sets will examine yelp restaurant reviews and biopsies of cervical cancer, each of which have binary classifications. The data will be preprocessed and then run through a series of common machine learning algorithms to compare and contrast their performance on classifying these types of datasets. From this, we hope to gain an understanding of how these algorithms behave when presented with datasets that possess certain characteristics

**The Data**

**Restaurant Reviews – Natural Language Processing**

Natural Language Processing is a field that analyzes text in an attempt to classify it according to some set of hypotheses. This subsection of machine learning has received an immense amount of attention and is therefore an area where one can find an extensive amount of research that has already been conducted. It has a practical application for many businesses and as tools around NLP continue to grow, more companies will begin to explore it for their own use cases. It’s applicability in the real world makes it an interesting topic to explore for machine learning and when combined with other tools like voice recognition software, can accomplish impressive engineering feats.

NLP takes a large body of text and quantifies it into a “bag” of words. Analysis of this bag of words can be approached in different ways. The most common of which are either classifying each individual word as a feature in and of itself. Another would be to group words together in segments as to account for word order. This can play a significant role in instances where negation needs to be taken into account. Consider the restaurant review: “this meal was not very good.” This sentence, if quantifying each word individually, can be interpreted as a positive review if the word “not” is not weighted properly. However, if grouping these words in such a way that we label “not very good” as a standalone feature, it is clear that this is a negative review. There are several ways to approach preprocessing data in NLP, however, these are two of the most straightforward.

Our data for this project consists of yelp reviews that we have retrieved from a yelp database. Our data is in plain English, is purely categorical, has a binary classification, evenly distributed (50% positive classification, 50% negative), a maximum feature set of 1500, and has 1000 rows of data. When considering all of these features, this is quite an interesting dataset and one we can generate some powerful models from

It is important for your data to be raw enough to intrigue, though not too intimidating to extract some classification from. The fact that all reviews are in a consistent language makes classifying this data not too overwhelming. The data isn’t trying to do too much, which is really what we are going for here. It is purely categorical, which again, helps narrow the focus of the problem at hand (it also contrasts our next dataset in this respect). It is evenly distributed which is important to consider when analyzing the accuracy, precision and recall. It has a wide range of features, which is where it contrasts our other dataset the most. With this, we will get to see how certain algorithms handle large feature sets. This is a major contributing factor to the success rate of some machine learning algorithms and will offer a nice juxtaposition when comparing alongside the performance of our other dataset. The only downside to our dataset is that it has only 1000 rows. This is rather low, especially when considering how many features we have to compare to. It would be ideal to have a dataset exactly like this one, which more entries.

**Risk Factors of Cervical Cancer – Categorical and Discrete Data**

Medical care is field that, relatively speaking, is in earlier stages of artificial intelligence. This is in large part due to that fact that the medical industry, technologically speaking, moves slower than most. It could also be due to the sensitivity of medical data due to HIPAA laws. As artificial intelligence grows, it will transition from a technology revered as experimental, to a standard. The nature of medical care and the effect it has one people’s lives alone is what makes this topic so interesting. Few go into software engineering with the thought that they would have the ability to save someone’s life through health care. Now with how far ML/AI has come, that is a very real possibility.

Medical data can come in all shapes and sizes. Monitoring heart patterns is a real-world application for continuous classifiers. While predictive modeling of future health patterns based on previous health records almost always consists of the combination of categorical and continuous features. With our dataset in this project, we will be dealing with the latter.

Our data consists of health records, containing both discrete and categorical data features. The data determines, based on a limited health record, if a patient is a candidate for a positive biopsy of cervical cancer. Therefore, it is a binary classification, it is not evenly distributed (93.5% negative classification, 6.5% position), has a feature set of 46, and has 858 rows of data. This largely differs from our other dataset and will shed light on interesting strengths across algorithms.

Immediately, the distribution of this dataset stands out. While most would consider this distribution poor, it introduces some interesting challenges. It is also important to note that in this medical industry, it is not uncommon to get data with this kind of distribution. It is reflective of the real population. For every 10 patients diagnosed with cervical cancer that you document health records of, you will undoubtedly have had to go through 200 to find. Due to the nature of this problem, we are forced to improvise. Also, the data has features that are both numeric, as well as categorical. This will be important to consider when analyzing the performance of each algorithm. The feature set is much smaller than our NLP problem (1454 less). It is also in a manageable range with respect to rows of data.

**Comparing Data Sets**

These two-dataset approach two very different issues, which will be intriguing if you are either interested in medical care or natural language processing. However, the implementation and performance of our data preprocessing and fitting our classifiers will vary greatly as well. Aspects such as number of features, categorical vs continuous, distribution and number of rows will play a large role in how we approach any machine learning problem. This will be interesting to compare and contrast and will shed light on when and where to use certain algorithms to solve certain problems. One thing that would be nice to have in this project is one dataset that is purely continuous. Having all three of these types of datasets, purely categorical, purely continuous, and a mixture of both, would be ideal when making these comparisons.

**Performance**

When evaluating the performance of a classifier, there are certain things to looks for that indicate whether or not the algorithm was able to provide any valuable information. If our data is evenly distributed (50% positive classifications, 50% negative), we would expect our classifier to be able to predict with an accuracy above 50%. The distribution dictates what a random guessing algorithm would perform at, so it is our threshold. Further, it needs to be defined how we want our classifier to predict certain classifications. When we are more concerned with accuracy in regard to positive classification, precision and true positive rate will help us analyze this performance. On the other hand, if negative classification accuracy is more of our concern, we will use specificity and false positive rate as our metrics.

Precision shows how accurate a given models positive predictions are. Specificity show how accurate a given models negative predictions are. Recall/true positive rate shows how often the model predicts positive when the classification actually is positive. On the contrary, false positive rate compares negative predictions over actual negative classifications. Accuracy shows us correct predictions over all predictions and error rate/misclassification rate is calculated incorrect predictions over all predictions.

**Restaurant Reviews**

For the restaurant review data, the distribution of label classification is 50/50. So we need our algorithm to be able to predict at a rate better than 50%. All five of our algorithms performed better than this, and so they are all preferable to randomly guessing. In this dataset, we are equally concerned with negative and positive reviews, so when selecting the most optimal algorithm, we will focus on accuracy first and foremost. All the metrics for each algorithm can be found in Table 1. Neural networks and support vector machines performed the best with an accuracy of 76% with a standard deviation of 1%. They each have an error rate of 24%. We can surpass this accuracy if we combine these algorithms to answer specific questions.

The amount of time it takes each algorithm to fit the training data can be seen in table 1. Neural networks and support vector machines took significantly longer than boosting, decision trees, and kNN.

|  |  |
| --- | --- |
| neural network | 13.85 |
| boosting | 1.91 |
| decision tree | 0.95 |
| k nearest neighbor | 2.08 |
| svm (linear kernel) | 8.09 |

Table 1: Restaurant Reviews - algorithm elapsed runtime (in seconds)

**Use Case 1**

Say, for a particular instance we are more concerned in finding out if a review is positive. We want to choose an algorithm that will classify positive labels most accurately. In this situation we will sacrifice overall accuracy to ensure that when we predict it is positive, we are most assured in our prediction. The precision metric will illustrate this best for us. Boosting will most accurately give us positive reviews. This is because boosting overall classifies far more reviews as negative than positive. So when it does classify as positive, we are more confident in our prediction.

**Use Case 2**

Let us consider a scenario where a classifier is less concerned with accidentally classifying negative reviews inaccurately, but it is vital that it does not falsely classify a negative review to be positive. This situation would be relevant where a travel guide needs to classify restaurants for tourists to eat at. In order to keep up a reputation, they would never want to suggest a bad restaurant to eat at. In this scenario, we weigh specificity above everything else. If we focus most on specificity, everything that classifies as negative, we want to stay away from recommending. Boosting is also the optimal algorithm in this scenario and would be most useful for this use case.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | accuracy | precision | specificity | tpr | fpr |
| neural network | 76% +/- 1% | 73% +/- 1% | 70% +/- 4% | 81% +/- 5% | 30% +/- 4% |
| boosting | 73% +/- 3% | 81% +/- 3% | 86% +/- 2% | 60% +/- 5% | 14% +/- 2% |
| decision tree | 72% +/- 4% | 76% +/- 4% | 79% +/- 4% | 65% +/- 5% | 21% +/- 2% |
| k nearest neighbor | 67% +/- 1% | 68% +/- 4% | 68% +/- 10% | 67% +/- 10% | 32% +/- 10% |
| svm (linear kernel) | 76% +/- 2% | 76% +/- 2% | 76% +/- 3% | 75% +/- 4% | 24% +/- 3% |

Table 2: Restaurant Review - Training and testing metrics (tpr: true positive rate, fpr: false positive rate, +/- is standard deviation)

**Risk of Cervical Cancer**

For the medical care data, the distribution is much more extreme at 93:7 (primarily negative classification). In this scenario we need an algorithm to be able to predict at a rate of better than 93%. The reason this threshold is so high is because if we were just to guess a negative biopsy on each iteration, we would still have an accuracy of 93%, so we must improve upon this for any algorithm to be worth implementing. Given the nature of cervical cancer, it is rather important that we maximize our recall (true positive rate). It would be most detrimental to tell a patient they are not at risk, when in fact they are. To air on the side of safety, we aim to classify the most positive labels without letting too many through. We consider this in combination with accuracy. All the metrics for each algorithm can be found in Table 3. Neural Networks and decision trees performed the best with an accuracy of 96% and a standard deviation of 1%. They each have an error rate of 4%.

The amount of time it takes each algorithm to fit the training data can be seen in table 3. Neural networks took significantly longer than all of the other algorithms. Compared to the restaurant review data, all algorithms performed at a much quicker rate. This suggests that a high number of features has a negative impact on performance.

|  |  |
| --- | --- |
| neural network | 8.28 |
| boosting | 0.41 |
| decision tree | 0.01 |
| k nearest neighbor | 0.05 |
| svm (sigmoid kernel) | 0.09 |

Table 3: Cervical Cancer - algorithm elapsed runtime (in seconds)

**Use Case**

For cervical cancer, it is vital that we do not tell a patient they are not at risk of a positive biopsy when they could be since when dealing with cancer early detection and treatment is the largest contributor to overcoming the illness. Even though decision tree performs as well as neural networks, decision tree is significantly more valuable for our use case. Recall of decision tree outperformed neural networks 87% to 71%. For our test set, decision trees averaged 0.7 false negatives over 89 entries. Meaning less than 1% of patients would be told they are not at risk when they in fact are. This detail is most important to minimize and is why decision tree would be our algorithm of choice for this dataset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | accuracy | precision | specificity | tpr | fpr |
| neural network | 96% +/- 1% | 69% +/- 12% | 98% +/- 1% | 71% +/- 16% | 2% +/- 1% |
| boosting | 95% +/- 1% | 64% +/- 7% | 98% +/- 1% | 55% +/- 19% | 2% +/- 1% |
| decision tree | 96% +/- 1% | 66% +/- 8% | 97% +/- 1% | 87% +/- 7% | 3% +/- 1% |
| k nearest neighbor | 94% +/- 2% | 57% +/- 16% | 97% +/- 2% | 47% +/- 15% | 3% +/- 2% |
| svm (sigmoid kernel) | 95% +/- 2% | 62% +/- 12% | 97% +/- 2% | 73% +/- 16% | 3% +/- 2% |

Table 4: Cervical Cancer - Training and testing metrics (tpr: true positive rate, fpr: false positive rate, +/- is standard deviation)

**Algorithm Performance Analysis**

\*Note yellow line in graphs depicts training score

**Neural Networks**

The neural networks were among the most performant of all the algorithm for each dataset. It is clear that the neural network was not impacted (negatively) by the large number of features from the restaurant review dataset.

Customizing the neural network was trickier than all of the other algorithms because of the effectiveness and range of parameters. The input dimensions for cervical cancer and restaurant data were 46 and 1500, respectively. The only tweaking of parameters that seemed to have a positive impact that I exploited was the dimensionality of output in the input layer. For cervical cancer 600 was used, for restaurant reviews 400. The confusion matrix on the left pertains to cervical cancer, the left to the restaurant reviews.

[[78.4 1.9] [[34.3 15.7]

[ 1.6 3.9]] [ 8.5 41.5]]

[I was unable to get sklearn.model\_selection.learning\_curve to work with this classifier]

**Boosting**

Boosting proved to be quite a useful algorithm for our restaurant review data, however, it did not perform as well with the health care data. When analyzing the learning curves, for the restaurant reviews the training score and the test score converge nicely around 2.25 as we add training examples suggesting that we have a nice balance of training examples. For the cervical cancer, our test and training scores are far apart, suggesting that we have underfitting.

Due to the nature of boosting, it seems to handle datasets with a high number of features quite well. Our restaurant data was able to leverage this and could be a reason why boosting was a successful algorithm for that particular dataset. For the cervical cancer data, it seems for boosting to be more successful, we need more data. The confusion matrix on the left pertains to cervical cancer, the left to the restaurant reviews. The learning curves can be found in figure 1.

[[78.6 1.7] [[44.7 5.3]

[ 2.5 3. ]] [20.2 29.8]]

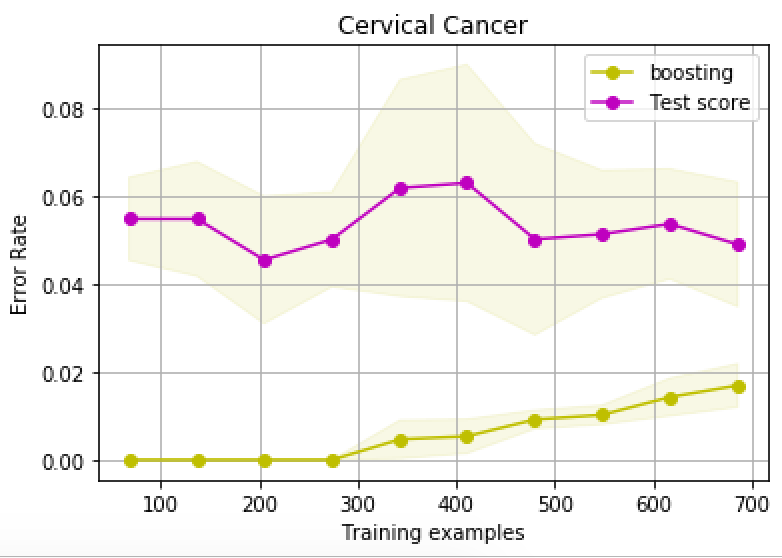
****

Figure 1: learning curve for training and test data (test scores in purple, training scores in yellow)

**Decision Trees**

Decision trees proved to be the most useful algorithm for our cervical cancer data. It performed decently for our restaurant review data, however, it is clear from the learning curve that we the algorithm suffered from overfitting (high variance) and we need to use fewer features. This should be addressed in the data preprocessing phase and would be a main point of revision when refining this codebase. This is also likely the main contributor to why decision tree performed poorly for the restaurant data. When analyzing the learning curve for cervical cancer data, we see the curves have a lower error but are close together, suggesting high bias.

When optimizing our decision tree classifier, no customization of parameters seemed to improve the curve for the restaurant data. However, for the cervical cancer data, assigned out leafs weights (min\_weight\_fraction\_leaf=0.05) and pre-pruning by assigning a maximum tree depth of 5 had a significant impact on performance. The criterion for information gain used was entropy. This was far more performant than gini. I chose specifically to avoid using the random forest tree solution and focused on just a single decision tree. This helped me analyze more closely the performance of just the pure decision tree algorithm. The confusion matrix of the test set along with the learning curves can be found below. The learning curves can be found in figure 2.

[[77.7 2.6] [[37. 13. ]

[ 0.7 4.8]] [15.4 34.6]]

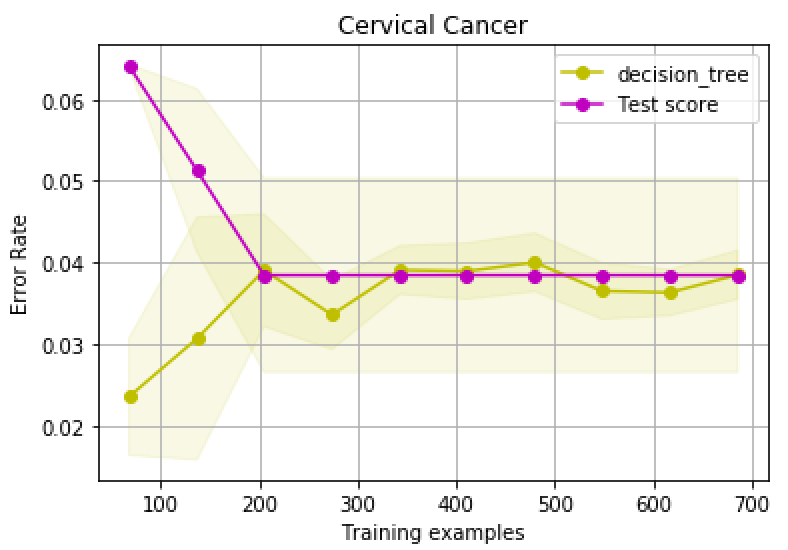
****

Figure 2: learning curve for training and test data (test scores in purple, training scores in yellow)

**K Nearest Neighbors**

K nearest neighbors in both scenario was the least performant across the board, most important being accuracy. This is likely due to the nature of the data since the restaurant reviews were purely categorical and the cervical cancer was partially continuous and partially categorical.

For each dataset, a k value of 3 was most performant. K values up to 11 were tested, but accuracy gradually declined with higher values. The distance function used was Euclidean. Neighbors were also weighted based on closeness as to not treat all nearest neighbors equally. This suggests that the “closeness” of data points was not very indicative of similarity. Moving forward, it would be worth exploring a custom function for determining closeness, especially with the restaurant review data. The confusion matrix on the left pertains to cervical cancer, the left to the restaurant reviews.

[[78.2 2.1] [[34.3 15.7]

[ 2.9 2.6]] [18.3 31.7]]

[I was unable to get sklearn.model\_selection.learning\_curve to work with this classifier]

**Support Vector Machines**

Support vector machines proved to be one of the most useful algorithms for our restaurant data. For the cervical cancer data, it fell short of the other algorithms. It is hard to draw a conclusion from the learning curve for the restaurant review data. It appears that at about 720 training examples, the algorithm stopped being as performant. Moving forward with this codebase, I would train the support SVM’s with the optimal amount of training examples per dataset. The learning curve for cervical cancer suggests underfitting (high bias) because they have low error rates, and converge very closely at the far right. The learning curves can be found in figure 3.

For the restaurant data, the sigmoid kernel was used. For the cervical cancer, the linear kernel was used. I also tested these SVM algorithms with the radial based function and polynomial kernels, though they were not as performant. For the sigmoid kernel function, a coefficient of 1.25 was used. This had a significant impact on the performance of the SVM. The confusion matrix of the test set along with the learning curves can be found below. The confusion matrix on the left pertains to cervical cancer, the left to the restaurant reviews.

[[77.6 2.7] [[78.6 1.7]

[ 1.5 4. ]] [ 2.2 3.3]]

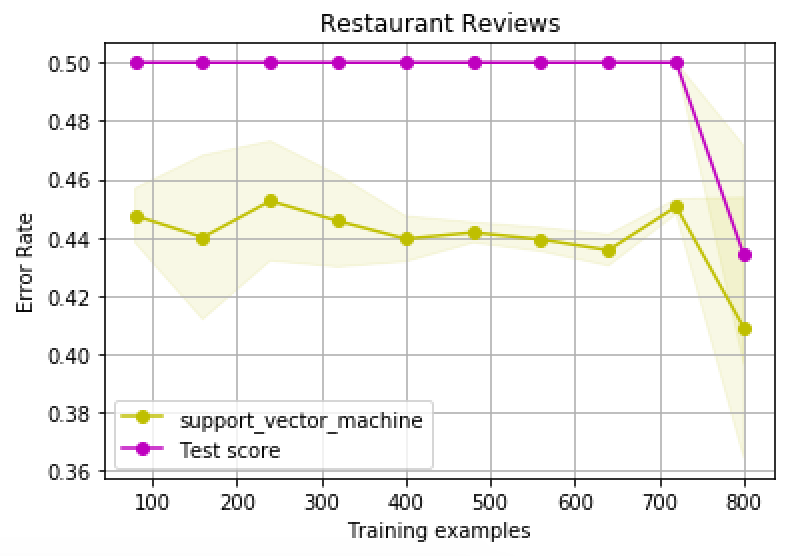
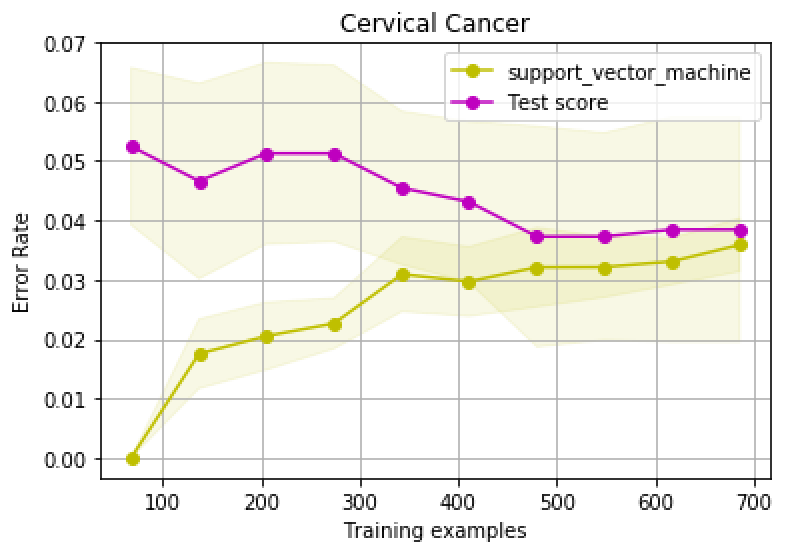


Figure 3: learning curve for training and test data (test scores in purple, training scores in yellow)

**Summary**

For all algorithms a train/test split of the data was done at an 80/20 split. I used the sklearn library StratifiedKFold which uses shuffles the data and does this split five (n\_splits) separate times and fits the classifier each time. This yields five confusion matrices which I then sum together to get the final confusion matrix to calculate the metrics from. This form of cross validation is far more thorough than just the one time split of 80/20, training, and testing. It also provided us with standard deviation between our confusion matrices.

Due to the categorical nature of the data, it appeared that classification algorithm performed much better than linear regression algorithms (kNN). This was not really a surprise by any measure, but that notion was further solidified by these results.

When analyzing the results for the restaurant review dataset, it seemed to suffer in nearly every scenario from the curse of dimensionality. The data preprocessing stage should address this and moving forward would be the primary issue to address. Another thing that would help this dataset would be to have more data. With any classifier, as the number of features grows, the amount of data we need to generalize accurately grows exponentially.

A clear issue with the cervical cancer dataset is the lack of even distribution. Positive labeled data only made up 7% of our data. This would clearly be an issue with any dataset. It narrows our margin of analyzing our performance to that 7%. If our generalization is perfect, we will see a 7% increase in our accuracy. This is not a lot of room to work with. Our most optimal algorithm (decision trees) was able to cover 3.15% of that ground, which is decent. However, learning curve suggested underfitting which further proves the point that we simply need more (positively labeled) data.

The neural networks took (by far) the longest amount of time to fit. It would be useful to thread the fitting of these algorithms so that we can reduce the time complexity of computation from 5\*O(time to fit a classifier) to just O(time to fit a classifier).

I ran into unexpected issues with the code in that sklearn.model\_selection.learning\_curve was not compatible with the library I used with for neural networks and kNN. This was unfortunate because I did not think this was going to be an issue since there were so many examples online, but then ran into this limitation last minute and had no way to graph my results. If I had more time, I would consider using different libraries for these algorithms (that are compatible with sklearn’s learning\_curve function) so that I could map these and analyze what my bias is, and whether I am underfitting/overfitting my data.