**CMPT 300 Final Paper**

**Zach Young, Serena Zi, Logan Gill, Evan Wang**

**12/4/17**

# Data mining: Classification of Handwritten Letters

**Table of Contents:**



## 1. Abstract.

Our task was to classify pixelated images representing handwritten letters as the appropriate letter from the english alphabet. We were given a dataset of 20000 images representing letters and we trained a model on the data in order to classify english letters. The

chief problem involved in the task was to find the best classification model and the best parameters for that model so we could obtain a robust and accurate classification system. In order to do this, we chose five classification algorithms: a decision tree classifier, random forest classifier, gradient boosting classifier, logistic regression classifier and k-nearest neighbors classifier. We ran cross validation and grid search on all five algorithms, then we experimented with training and test data allocation sizes to eliminate any possible overfitting. After running the grid search, we found that a random forest classifier worked the best for our task. With the parameters that we found from running grid search, we obtained a high accuracy on our test data.

## 2. Problem Description.

The problem that we chose was a letter classification problem. The data is comprised of black and white images that represent pixelated letters. The dataset came from https://archive.ics.uci.edu/ml/datasets/letter+recognition. The researchers who put the data together created data points derived from many different images of letters and broke all the data into 16 different features. Some of the features that the dataset measures are the average number of horizontal edges in the image, and the average number of vertical edges, as well as height of the image, and things of that sort. Once these features have been established, the data must be then cleaned and outliers removed, and the data trained. The data are then fit using a classification algorithm. In order to find the best algorithm, we use grid search and cross validation. We then train the best model, which was a random forest classifier. Once this has been completed, new data can be tested against the model and can be classified into what letter it most closely represents.

## 3. Collect and preprocess data.

The data consist of 26 capital letters from the English language that are based on 20 different fonts. Each letter within these fonts was randomly distorted to produce 20,000 unique stimuli. After this, each stimulus was converted into 16 primitive attributes that were normalized

between 0 and 15. There are no missing attribute values in this dataset, since the it was filtered beforehand. There are also no unusual points as we removed the outliers of the dataset by removing values that were further than 3 standard deviations away from the mean. Since our data was not gathered by some kind of measurement instrument, such as some kind of tool that measure water flow for example, we didn't have to worry about mechanical errors in any of the values. All of our data came from randomly distorted letters and there is therefore no noise.

**4. Formulate the task into a data mining problem.**

We determined that our dataset would be best explained by using multi class classification to better understand the meaning of the data. The task involved classification of each letter based off of the 16 attributes in the dataset, to obtain the best possible accuracy for each letter classification. The data mining task is to separate the points in the data set into the appropriate letter class, and visualize the output to better explain the results of our findings. We discovered information about each feature using the confusion matrix and feature importance plots after training our model. Determining the best algorithm was achieved by using Cross Validation and grid search. Based on our dataset we determined that certain algorithms worked better than others for the Classification task because of their ability to use each feature independently. The classification is superior in the random forest model because the large number of features for each data point are best utilized to classify each data point, as well as the fact that our data set has 26 features so the multi-tree approach of the random forest works well to separate each of those classes.

**5. Choose and evaluate multiple algorithms.**

The five algorithms that we use are Decision Tree Classifier, Gradient Boosting Classifier, Random Forest, Logistic Regression, and K-Neighbors. For each algorithm and their parameters we test them all using Grid Search to determine the best parameters and best scores,

then we print these values to reference for the rest of our project. In order to train each model during our search, we use Cross Validation with 5 folds. After running grid search, we analyzed the results and decided that the random forest classifier works the best for our data set. The gradient boosting classifier and K-neighbors algorithms are not as good of algorithms to classify the data based on their accuracies. Logistic regression got the lowest accuracy of all five. The decision tree classifier had a fairly good accuracy but random forest classifiers are better than decision trees in pretty much every way so it makes sense that random forest beat it out.

**6. Explain the best model based on your data and real-world examples.**

We chose the Decision Tree model, the Gradient Boosting Classifier, Random Forest, Logistic Regression, and K-Neighbors to build the model. After running Grid-search and cross validation on each model, we found that a Random Forest Classifier was our best model, narrowly beating out the K-nearest neighbors classifier. The best parameters for the random forest classifier were n_estimators = 30 (the number of trees the forest used), criterion = 'gini' and max_depth = none. In other words, the number of trees that our model used was 30. The model used the gini index of the data to build the trees and the trees had no max depth. Random forests perform better than binary trees as a general rule and since we have so many classes, having many different trees allowed us to calculate a better answer. The random forest classifies the data as the mode of whatever the trees return. Since we have so many classes, the random forest is the most robust of the models we ran.

Random forest classifiers are very powerful algorithms and there are are several reasons that we think the random forest classifier works the best out of the models that we tested. First and foremost, the random forest classifier has been popular among other classification algorithms because it is accurate and efficient. Our data set is very large and random forests run well on large dataset, so it is a good fit for our data in that regard. In addition, as random forests train, they give an estimate of what variables are important to the classification, and the prototypes of the trees that are generated during runtime can give us information about how the variables affect the classification process. All in all, random forest classifiers are powerful tools

for tackling multi-class classification problems and it makes sense that this is the best model we found.

**7. Explain and visualize results.**

Confusion matrix:

The confusion matrix allows us to visualize the error rate on the prediction of each class. From the matrix, we can see that, except for the diagonal center line, we get almost all zeros, which means that almost all the data points have the same actual class and predicted class. In other words, we correctly predict most of data with our model. This result also match with the prediction accuracy of 0.951812555261 pretty closely. Since most of the numbers lie along the central diagonal, we changed the color pallette of the matrix so that it has very high contrast and small differences in numbers are easy to see.

Feature importance:

The feature importance describes the meaningfulness of each attribute or dimension of our data. The dataset we are using has 16 dimensions so it is useful to visualize the importance of each dimension in ascending importance. The plot describes how informative each feature is in the classification task from most informative to least informative, and also shows the inter-dimensional differences of importance between attributes. From this plot, we can see that xedge and yedge are the most important per the reason above. The least important features are boxheight and xbox, which represent the height of the box and the horizontal position of the box. Presumably, this would mean the where the letter was drawn in the space provided.

Algorithm accuracy:

This graph displays the accuracy of each model using a bar plot. For each model, we graph a bar with height representing the accuracy of our model. It is a Bokeh plot, so you can

zoom in to see the heights of the bars with more accuracy and pan around the plot. This plot gives an alternate way of viewing the accuracies of our models without simply reading a list of numbers. From it, we can see that Random Forest has a higher accuracy (as stated earlier) than K-Nearest Neighbors, though it is very close. Second is Decision Tree, then Gradient Boosting and finally Logistic Regression with a performance that is quite lackluster in comparison to the other 4.

Feature Distribution:

This box plot shows the shape of the distribution and variability of our data set for each of the 16 features. This boxplot shows that the xbox feature has the most variance in its data at 13, while the feature xedge has the least amount of variance in its data at 4. The features xbox and ybox have the largest interquartile range at 6.5 while xedge has the lowest interquartile range at 2. Most of the features have a mean between 5 and 6.5. The most common means are 5, 5.5, 6 and 6.5. The lowest mean is 2, while the highest is 7.5.

**8. How did you improve the model in step 3 from step 2?**

In step 2, our model got 100% accuracy at first, because we ran cross validation and grid search on our full dataset for all of our models. When we found the best model, random forest, we trained a model with the optimal parameters on the full data set. This means when we tested, we got 100% accuracy. Clearly, 100% accuracy is overfitted, which is not what we want. In order to fix this issue, we took out 2000 data points as well as about 1800 outliers, which left us about 16000 data points that we trained on. After training the model on this separate training data, we tested the model with with the 2000 data points we removed. On this data, our model performed with about 95% accuracy. It wasn't 100% correct but it was still quite accurate on data we had never seen. This accuracy is still rather high, so in an attempt to lower the accuracy to obtain a more generalized model, we decided adjust our training and test data allotments. After taking out about 4000 points to test on and leaving 1400 to train, we still got a high

accuracy. It dropped below 95%, but barely. After several runs, it looks like it averaged around 94%, rarely falling below that. We went as far as to try training the model on half the data and testing on the rest and the lowest accuracy that we saw was around 93%. Going any farther would mean training on less data than we test, which seems like it would be counter productive. From this, we can conclude that our algorithm is most likely not overfit.

Graph wise, we changed out confusion matrix to make it simpler and easier to read. We originally had it compute the matrix with all 26 classes included which made the plot quite large and nearly impossible to read. In step 3, we created the plot using only the 10 most common letters in our data set so it was not quite so large. Matplotlib also created the color scheme as a default 'cool' pallet, with blue and white, that made it very difficult to distinguish differences in cells. We changed that to a color palette with high contrast to make it easier to read. In addition, after working on reducing any possible overfitting in our algorithm, we created more variety in the confusion matrix. Originally, since our algorithm had such a high accuracy, the confusion matrix was 0 basically everywhere except on the diagonals. This simply means essentially every data point was classified correctly. After partitioning the test and target data differently, we see more cells not on the diagonals with values besides zero. Since the model accuracy is still high, the numbers aren't large but there are quite a few more values ranging from 0 to 4.

Other than these two changes, we did not have to change much about our model our project between steps 2 and 3. There were no glaring issues that had to be fixed and the model worked well on our data.

**9. Summarize what you have done and any thoughts in your model. Refer some answers in step3. Give some future work you may improve.**

In summary, we have utilized letter data to train a classification algorithm that can predict letters with high accuracy. We tested five different models and found that the random forest classifier fits our data the best, giving us a robust model that performs well. Using this model, we ranked the importances of the features and analyzed the results, explaining why some features are important and why others aren't. Namely, the variables xedge and yedge were the most

important. As these represent the number of horizontal and vertical edges, respectively, our model seems to suggest that the most important feature of a letter is its width and its height. After training the model, we evaluate its testing results using ROC and AUC as well as creating numerous visualizations to display the information that we obtained from the model. Ultimately, we trained and tested a model that performs well on large amounts of test data. Since it performed well even on lots of test data, we concluded that the model is most likely not overfit. Rather, the model that we trained simply works well for our data. Still, there are always things we could improve. For example, In order to be even more certain that the model is not overfit, we could create a process to generate more data, or collect or obtain more data, in order to test our algorithm even more thoroughly. In addition, in the future we could research more why our ROC curve didn't come out as a curve, instead of a straight line and determine if it is some strange characteristic of our data set. One thing that could be useful for improving our model's generalization abilities is to look into a more sophisticated way of preprocessing. The only preprocessing we performed was filtering out data outside of three standard deviations from the mean and it is possible with more work along those lines our model would improve further. In summary, we trained a random forest classifier on data representing handwritten letters and obtained a high-accuracy, effective model.