Subreddit Classifier - Project 2

David Dunlap dad9489@rit.edu
Ian Randman ixr5487@rit.edu

Contents

Introduction

This is an implementation and analysis of the use of four different machine learning algorithms that are used to create a program to take in a post from Reddit and determine which subreddit it came from only based on its comments. The program can consider eight different subreddits and determine which of these it thinks is most likely for this post to have originated. To approach this, we gathered as much data as we could from these subreddits, separated it into 50% training data, 25% development data, and 25% testing data, and then used this to create our models. We completed a random search over different combinations of hyperparameters for each of the different classifiers used in order to maximize the accuracy of each classifier, and then compared these four different methods to create a program that has a relatively high accuracy.

Methods

The possible subreddits that this program can consider are: r/nba, r/nhl, r/nfl, r/mlb, r/soccer, r/formula1, r/CFB, and r/sports. These were chosen to be similar enough to challenge the model and provide interesting insights into where the model struggles, but also have enough differences to be possible to differentiate between with a relatively high accuracy.

To start, data from 1000 posts from the top of all time from each of the 8 subreddits was gathered. This data consists of a single string concatenating the text content of the post with the comments on the post. Using the Praw library to access the Reddit API, the program pulls posts from the top of all time from a given subreddit, and then completes a breadth first traversal of the comments of the post. It retrieves all nested comments except those that require clicking “more replies.” This is to prevent unnecessary information being gathered, as it is likely that a comment nested that far down would not provide much useful context. These 8000 labeled data points are then sanitized by removing all characters that are non-alphanumeric or an apostrophe (to preserve contractions), and are also stemmed to attain their root word. The stemming is done through the use of the nltk library (Natural Language Toolkit) and uses the Snowball stemming algorithm to strip all words down to their root word.

After stemming the words and getting their counts, a term frequency times inverse document frequency (TF-IDF) transformer was used. The TF part is to account for the fact that posts with more comments will have higher word counts than posts with lower word counts. Whether to use IDF, which reduces the weightage of more common words, was experimented with during the search for best hyperparameters. One observation to note is more popular subreddits (such as r/nfl) will have more comments on average than a smaller subreddit (r/mlb). Because of this using TF can actually be hindrance, as higher word counts can help classify a post. However,

since posts on all subreddits vary widely in number of comments (compared to generally higher comment counts when sorted by top of all time), we decided to stick with TF.

From here, we trained four different classifiers to try to fit the model: a Multinomial Naive Bayes Classifier, a Random Forest Classifier, a Stochastic Gradient Descent Classifier, and a Support Vector Classifier. In order to improve the performance of these classifiers, we also perform a random search over different combinations of hyperparameters for each of these classifiers. This allows us to achieve the highest possible accuracy for each of the classifiers used to be able to accurately compare them, and also get the greatest accuracy of the models overall.

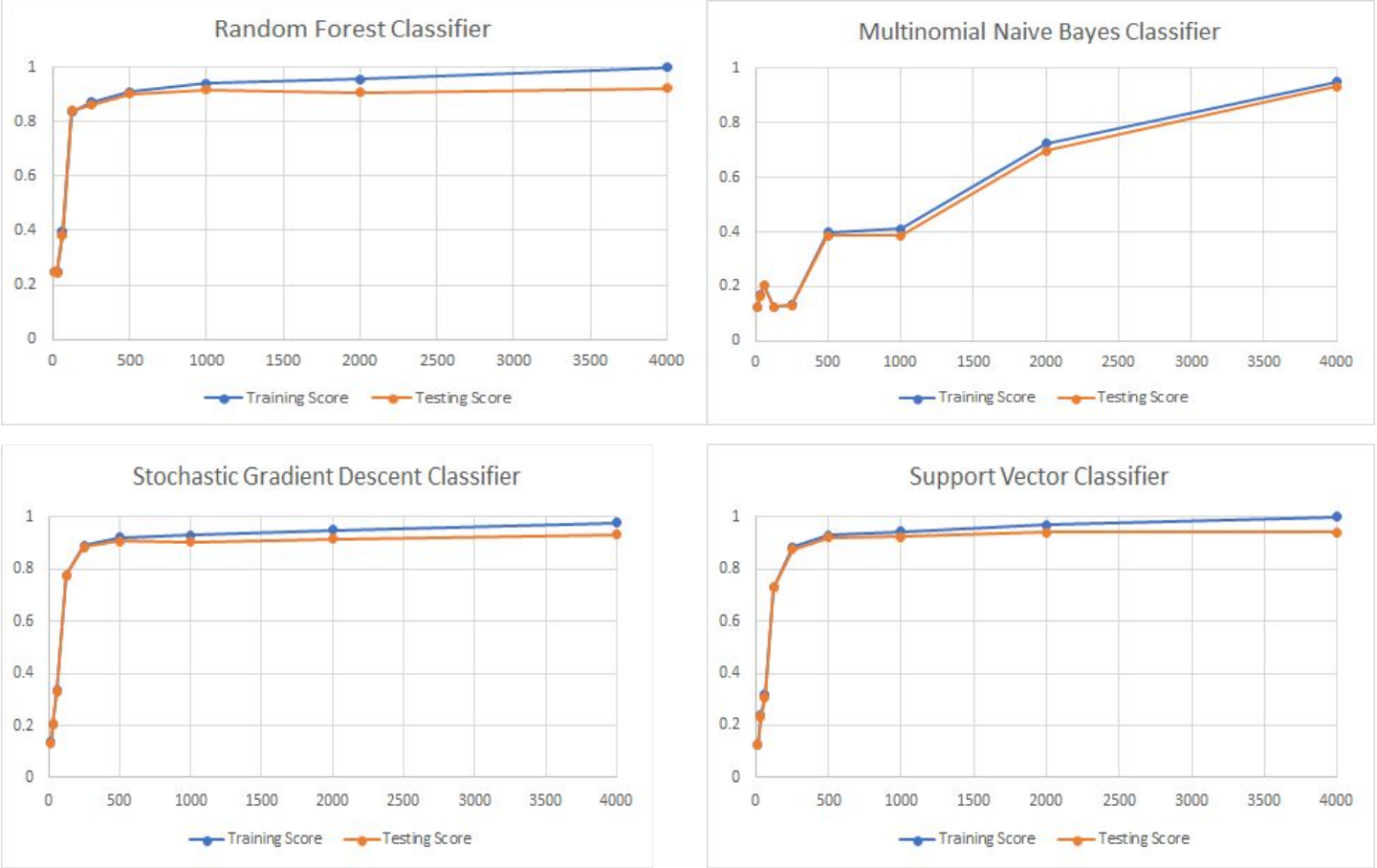
Because we split our data into training, development, and testing sets, scikit-learn could not help with cross-validation on our development data. Instead, we wrote the randomized search, and after training on a subset of the training data, a model would be scored on the development set. The model with the best score on the development set would have its hyperparameters saved for later training.

Hyperparameter combinations were generating using a logarithmic set of values for continuous hyperparameters and several options of discrete hyperparameters were used. An exhaustive grid search was considered, but time for computation would have been far too great. Instead, all parameter combinations were generated, and 500 were selected at random to test. Again to reduce time for computation, the search for hyperparameters was performed without stemming, and stemming was applied for the final training of each of the models.

Results

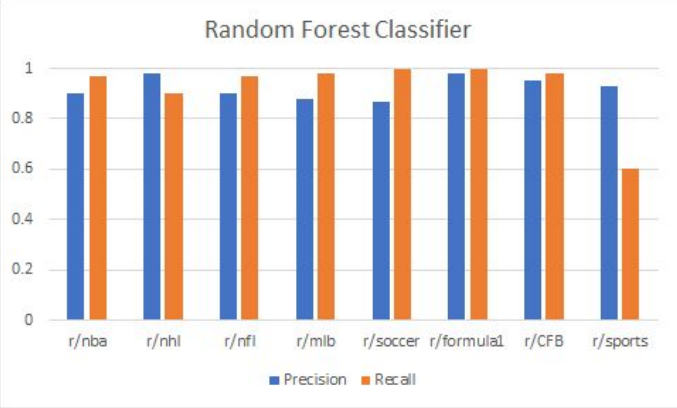
With the large amount of data collected, as well as the search over the hyperparameters for each classifier, we were able to achieve fairly high levels of accuracy. While developing, the classifiers were evaluated based on the percentage of the development data that was correctly classified. By the end, when evaluated using the testing data, all of the classifiers were able to get greater than 90% accuracy. Out of 2000 examples in the testing data, the Random Forest Classifier got 92.3% correct, both the Multinomial Naive Bayes Classifier and the Stochastic Gradient Descent Classifier got 93.2% correct, and the Support Vector Classifier got 94.2% correct. In terms of runtime, the Multinomial Naive Bayes Classifier was almost always the fastest to complete, and the Support Vector Classifier took several times longer to complete than any of the other models.

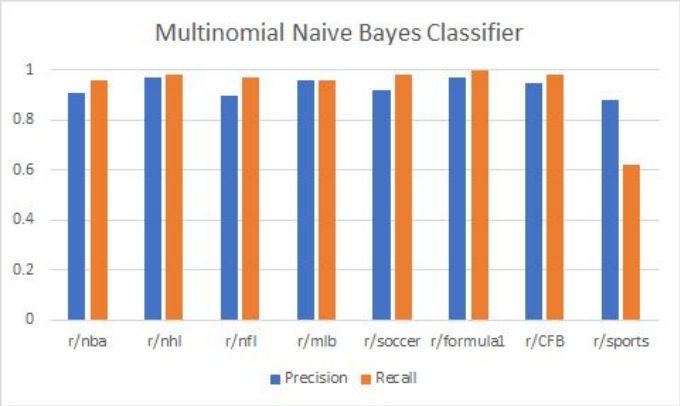
The following charts show the learning curves for each of the classifiers as they see more training examples. On the x axis are the number of training examples that the model has seen, and on the y axis is the proportion of the data that was correctly classified. Each graph shows the learning curve for the model's prediction of both the training data and also the test data.

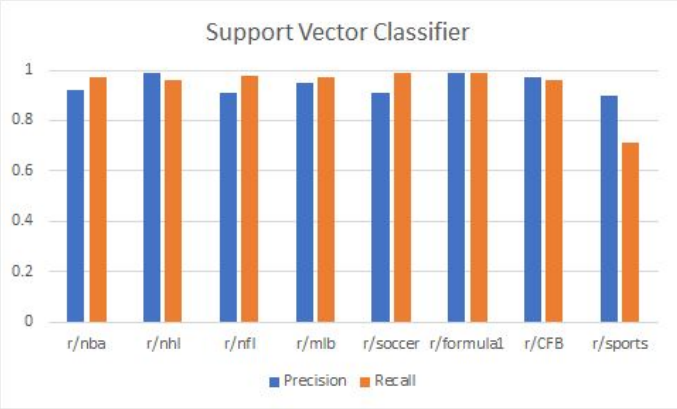


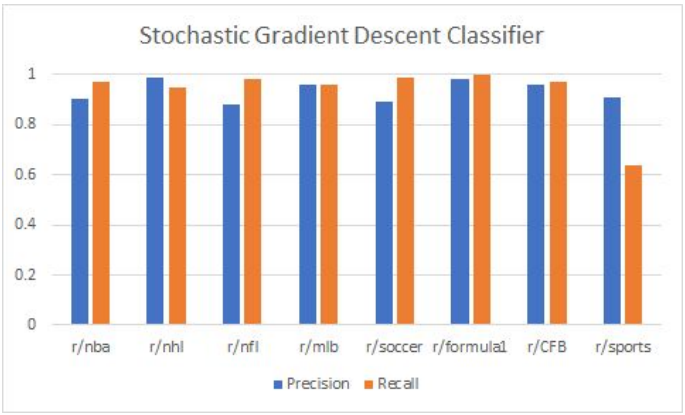
From these learning curves, we can see that most of these models learn the data fairly quickly. By 500 training examples, 3 out of the 4 models were predicting the testing data with 90% accuracy or above. However, the Multinomial Naive Bayes Classifier took much longer to learn the model. It learned it fairly linearly, and took somewhere around 3,500 training examples to be able to predict the test data with 90% accuracy. These graphs also show information about whether the model is overfitting the data. The training score is very close to the testing score for most of the time for all of the models. This shows that there is not much overfitting occuring, as it is not significantly more confident about the data that it is training on as compared to data that it has never seen before. However, by the end, most models get close to 100% of the training data correct, indicating some amount of overfitting.

These next charts show the precision and recall for each subreddit by the end of training. The precision value represents the proportion of predictions for that subreddit that are correct, while the recall value represents the proportion of posts from that subreddit that were identified correctly.









While there are minor differences in the precision and recall for a given subreddit between the different classifiers, they are all relatively the same. However, one major point of interest that comes from this data lies with the subreddit r/sports. When picking the subreddits, we were interested to see how the models would deal with this subreddit, as all of the others are subreddits dedicated to specific sports, while r/sports is about all sports in general. This ambiguity is reflected in the recall value for r/sports, which is on average only 0.6425, meaning out of all the posts that should have been classified as coming from r/sports, only 64% were classified correctly. This is consistent with our expectations, as it means that many of the posts from r/sports are being classified as being from a more specific subreddit — most likely the subreddit corresponding to the sport discussed in the post.

Overall, our algorithms worked better than we expected. We were very happy to end up with results of over 90% accuracy on new posts, and to see that when presented with a random post from one of the subreddits, the model would almost always get it correct. The places where the

models struggle are usually understandable; they often have trouble when the post has very few comments, or the comments are discussing topics that overlap between the two different sports. It also struggles with r/sports, which was hypothesized from the beginning.

In the future, this can mainly be improved by improving the search of hyperparameters. If k-fold cross-validation were used in the search for hyperparameters instead of using the development set, it would allow for more data to be used for training, and would provide a nonstatic set on which to validate. This could improve the search for hyperparameters that optimize the function overall for new posts. We could also consider different hyperparameters to search over, and search over more hyperparameters overall since the random search is not exhaustive. However, while there is still room for improvement in the models, we are very satisfied with the results. A large majority of the time, our model will accurately classify the post, and better optimization would simply close the gap for the few examples that it still gets wrong.

Conclusion

This project sought to create a model to accurately predicts which of 8 subreddits a post came from based solely on its comments. Models were trained using four different classifiers available in scikit-learn. Data was pulled from Reddit — 1,000 posts from the top of all time of each subreddit. This data was then split 50%, 25%, and 25% into training, development, and testing tests, respectively. During development, a randomized search over a grid of possible hyperparameters was performed to determine the best hyperparameters for each model, while balancing performance vs. time for computation. After this, an extra transformation from natural language toolkit (nltk) was performed for the final training.

Learning curves were made, showing that for three out of the four models, their scores began to level off after about 500 posts, while for the Multinomial Naive Bayes Classifier, it took about 3,500 posts to become over 90% accurate. Looking at precision and recall for each subreddit for each model, values seem to be over 90% overall. It seems that most models perform fairly similarly after training. However, one suspicion is confirmed when looking at the recall values of r/sports, which is far lower than the other values, sitting around 60%-70%. This is likely because posts that originated from r/sports were misclassified into the subreddit of the specific sport being discussed in the post.

Improving future models would see a revised method of tuning hyperparameters. This would include either an exhaustive search or more combinations to try in the random search. In addition, tuning hyperparameters would be done with the inclusion of stemming. Also, different hyperparameters can be experimented with, and the method of validation for each model would

be changed to k-fold cross-validation. Finally, we can experiment with classifiers outside of the four used in this project.