Tyler Percy

Homework 2 Analysis

Analysis Objective

Initially, I ran the KNN and Decision Tree classifiers with the 100 feature words I used to create the data matrix from the first homework. These feature words were too few and not distinct enough to get extremely accurate results though. So, I thought it would be good to see how much increasing the number of feature words increased the accuracy of the two classifiers. So for the second step of my homework, I ran the same classifiers on the same data but increased the number of feature words to 12,000.

Process Description

I used Python and sklearn packages for the decision tree and KNN classifiers and self-implemented a 5-fold cross validation to make sure the results would be more consistent for each run. I also used GraphViz to create a picture of a decision tree generated during testing. To further improve the consistency of my results, I averaged the results for 20 runs of both the decision tree and KNN for the best value of k. The data matrix from the last homework is loaded into cnnDecTree.py and all of the classifiers are run and results are output from there.

Decision Tree Results

For my decision tree classifier, I used the default values for minimum samples split and minimum samples per leaf. I also used the decision tree classifier to generate a single decision tree instead of a random forest. Although the accuracy could be increased by using a random forest instead, I went with the regular decision tree just to see how it would perform.

The decision tree yielded an accuracy of .529518 and an f1-score of .419026.

KNN Results

For the KNN, although there were 6 different categories, almost half were political articles and much of the 100 features words were related to politics. Due to KNN using Euclidean distance, which is not well suited for sparse nature of a document term frequency matrix, the results were not great. The highest accuracy was received from using a k=2. This is likely because of all the political articles could be identified since they all were closer to each other due to using a lot of the similar political feature words. Essentially the KNN recognized political and non-political, and just took guesses of which non-political category to assign. Increasing k towards 6 results in worse results and when k goes above 7 the accuracy drops quickly.

With k=2 (the most accurate) the KNN yielded an accuracy of .527619 and an f1-score of .3423515

Step 1 Analysis

Although KNN with k=2 had almost the same accuracy as the decision tree, its f1-score was much lower than the decision tree. Since the data is unbalanced in class size (mostly political articles) f1-score is more suited to truly revealing which classifier works better in this situation. The reason the accuracy was so close is because the knn just puts half of the articles into the political category mindlessly and gets a high accuracy because most of the articles are political. The f1-score detects these false positives and false negatives and adjusts the score accordingly. This results in a much higher f1-score for the decision tree because it actually has some predictive power, and not just guessing.

Step Two Predictions

For the next part of my analysis I think that increasing the number of feature words will increase the accuracy of the decision tree and shift the best value of k closer towards 6 because it will be able to detect the differences between groups more accurately. However, I don't think the accuracy for the KNN will increase as much as the decision tree because it is still using Euclidean distance which isn't as well suited for document matrix data.

Step 2 Results and Analysis

For the second step of my homework, I increased the number of feature words dramatically, from 100 to 12,000 to see the differences in accuracy and f1-score. My assumptions were all correct. With 12,000 feature words, KNN performed very poorly with k=2 usually returning an accuracy of .05. However, when I raised k towards 6, the accuracy rebounded a little, but remained in the 30% range. The decision tree rose a good amount in accuracy and f score due to more rules to look at and create. Although the results still aren’t perfect, it was a big step in the right direction to increase the number of feature words.

Here is a comparison of all the results obtained during the homework

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 100 Feature words | | | | 12,000 Feature words | | | |
| Decision Tree | | KNN | | Decision Tree | | KNN | |
| Accuracy | F1-Score | Accuracy | F1-Score | Accuracy | F1-Score | Accuracy | F1-score |
| .529518 | .419026 | .527619 | .3423515 | 0.70476 | 0.51681 | 0.35079 | 0.14682 |

As we can see, increasing the feature words was much needed to show the real differences between these two classifiers. With only 100 feature words, both the decision tree and KNN were both basically guessing whether the article was political or not and both returned an accuracy of around 50%. But when we increased the feature words, the Decision tree had some fairly accurate results, around 70% while the KNN fell because of its distance metric not really suited for this application.

Below is a picture of one of the decision trees generated with GraphViz to get an idea of what exactly is created by the decision tree algorithm.

