Project 3: Applying SVMs to Breast Cancer Data due May 8, 2018

CSC 420 – Machine Learning

Jordan Turley and Eric Murrell

**Overview**

For this project, we were tasked with applying support vector machines and one other model to sklearn’s breast cancer dataset. We decided to apply k-nearest neighbors to the dataset as well. We tried to classify a tumor as benign or malignant, based on thirty different features, such as radius, perimeter, area, smoothness, compactness, and many others. With SVM, we were able to achieve an F1-score of 0.964. With k-nearest neighbors, we were able to achieve an F1-score of 0.957. Both models performed very well.

**Data Preparation**

As this was a dataset included with sklearn, there was not much to be done in terms of data preparation. All we had to do was call a method in python to load the dataset. However, it was very important that we rescaled the data. We used sklearn’s StandardScaler class to rescale our data. If the data was not rescaled, we encountered a major problem. Training took upwards of fifteen minutes when using the support vector machine with a polynomial kernel, before we killed our program, as we assumed something was wrong. However, after we rescaled our data, training took seconds.

**Methods**

As I mentioned earlier, we applied two different models to this problem: support vector machines and k-nearest neighbors. We randomly shuffled our data into training and test sets. We used sklearn’s StandardScaler class to rescale and normalize our data. After this, we used grid search to iterate over several combinations of parameters to find the parameters that perform the best. For support vector machines, we searched over several kernel choices: linear, sigmoid, rbf, and polynomial. We also searched over several values for C: 0.1, 0.2, … 1.9, 2.0. We used repeated k-folds cross validation to evaluate the performance, with ten folds and ten repeats. We found that the best performance was achieved when C = 0.1 and we use a linear kernel.

After applying support vector machines, we applied k-nearest neighbors to the problem as well. Our choice of k-nearest neighbors was because we wanted to see how well a simple model like k-nearest neighbors could do on a problem like this. We had seen that support vector machines did very well, but we were not sure how well a model like this would do. We used grid search again with repeated k-folds, and searched over different values of k: 1, 2, … 10. We achieved the best performance when k = 3.

**Results**

Both of our models were able to perform very well on this data, and performed very similarly. We evaluated their final performance on the test set. Our results are summarized in the following tables.

**Table 1: Success and Error Rates**

|  |  |  |
| --- | --- | --- |
| **Model** | **Score (Success Rate)** | **Error Rate** |
| Support Vector Machines | 0.965 | 0.035 |
| k-Nearest Neighbors | 0.958 | 0.042 |

**Table 2: Precision, Recall, F1-score, and Support of SVM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| Benign | 0.979 | 0.922 | 0.949 | 51 |
| Malignant | 0.958 | 0.989 | 0.973 | 92 |
| Average | 0.965 | 0.965 | 0.965 | 143 |

**Table 3: Precision, Recall, F1-score, and Support of kNN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| Benign | 1.000 | 0.882 | 0.938 | 51 |
| Malignant | 0.939 | 1.000 | 0.968 | 92 |
| Average | 0.961 | 0.958 | 0.957 | 143 |

**Table 4: Confusion Matrix for SVM**

|  |  |  |
| --- | --- | --- |
|  | Predicted Class: Benign | Predicted Class: Malignant |
| Actual Class: Benign | 47 | 4 |
| Actual Class: Malignant | 1 | 91 |

**Table 5: Confusion Matrix for kNN**

|  |  |  |
| --- | --- | --- |
|  | Predicted Class: Benign | Predicted Class: Malignant |
| Actual Class: Benign | 45 | 6 |
| Actual Class: Malignant | 0 | 92 |

**Analysis**

We can see that both of our models did a very good job classifying a tumor a benign or malignant. The SVM performed slightly better on the test set than kNN, but if we look at the confusion matrices, we see that while the SVM performed better overall, kNN classified all benign tumors correctly, but classified a few malignant tumors as benign. This could be a problem; classifying a tumor as malignant when it is benign could give a patient false security, and if it is not treated, this could be detrimental to the patient. However, classifying a benign tumor as malignant could result in the tumor being treated, just for the doctor to find out it is benign. The SVM has one more false positive, but fewer false negatives.

**Rescaling**

For SVMs to work well, it is very important to rescale and normalize the data. We used sklearn’s StandardScaler. At first, we tried training the SVM without training our data. Training took several minutes before we gave up. However, after scaling and normalizing the data, training took seconds. The results were also much better after scaling the data rather than not.

**Conclusion**

As we can see, SVMs can be applied to a problem and work very well. They can perform better than other models, such as kNN. In the future, I would like to apply other models to this problem, like logistic regression, naïve bayes, or even a neural network, and see if any of these perform better than SVMs. However, as we saw, this problem is relatively easy; a method as simple as kNN was able to achieve about 95% success rate. It would also be interesting to apply SVMs to a harder problem and see how it performed in comparison to other models.