Jean de Malliard

CS-677: Final Project

Star classification

Preprocessing

Globally, the dataset was fairly clean, there was no missing values, and no strange values. However, the Color attribute had inconsistent values. It is a nominal attribute containing the color of the star, but I figured out that among the values, some were the same but considered as different values. For example, there were a value “blue-white” and a value “blue white”. So, using Python I modified the problems and made consistent values.

Then, I saw that among the attributes I had, 4 were numeric and 2 were nominal. Therefore, I had to transform the nominal attributes into numeric attributes, so I made new columns in the dataset, and each column was containing a binary value corresponding to one of the possible values of each attribute. This added about 15 columns to the dataset.

To use some classifiers using distances, I also had to scale the numerical attributes. Indeed, some attributes were on totally different scales. For example, the “L” attribute spans from 0 to 849,420, whereas the attribute spans from -11.92 to 20.06. Scaling is necessary.

Results

KNN:

For KNN, I tried different values of k, all the odds values from 1 to 21. I tried 2 different distances (Euclidean and Manhattan), as well as different weights for each neighbor (uniform vs weighted by distance).

The results I got were that Euclidean distance with k=3 and weighted votes depending on the distance of the neighbors was the best combination.

I got a training accuracy of 100% and a testing accuracy of 98.3%. It’s high scores, but we’ll see that other classifiers did even better.

I also looked at the performance using unscaled data and saw that even though the training accuracy remained at 100%, the testing accuracy dropped at 73.33%. This is quite obvious but was still interesting to note.

SVM:

For the SVM, I tried 3 different kernels: Linear, Gaussian (RBF) and Polynomial (degrees 2 to 5). I also tried 2 given values of the kernel coefficient.

I found that the best combination was the Linear kernel using the ‘scale’ value for the gamma (kernel coefficient). The scale value corresponds to 1/(n_features*X.var()).

Using those values, the SVM classifier got a training accuracy of 100% and a testing accuracy of 100%. I was surprised by such good results for unseen data, but this is also probably due to the fact that the dataset corresponds to meaningful data (the characteristics of a star and the type of star).

Furthermore, the confusion matrix shows that every tuple is correctly classified (the accuracy could be 100% with an “imperfect” confusion matrix).

Naïve Bayes:

Naïve Bayes gave me the worst results of all the classifiers. There are not a lot of hyperparameters to play with, I just tried different values of var_smoothing which is a smoothing technique in case there is a 0 probability that causes problems.

I found out that the best value was 10^{-9}, but I’m not sure that it changed a lot of things.

Using those values, the Naïve Bayes classifier got a training accuracy of 87.5% and a testing accuracy of 86.6%. That’s not too bad, but compared to other classifiers, it’s not interesting.

Decision Tree :

For the Decision Tree, I tried different values for the maximum depth, the minimum number of samples to have per leaf and I used the entropy.

I found that the best combination was a maximum depth of 3 with a minimum number of samples per leaf of 5.

Using those values, the Decision tree classifier got a training accuracy of 100% and a testing accuracy of 99.16%. Here again, the results are very good, but not as good as the SVM results.

Random Forest:

For the Random Forest, I played on a lot of parameters: the number of estimators, the minimum number of samples per leaf, the minimum number of samples needed to split a node, the maximum number of features considered to split a node, the maximum depth of each estimator and the criterion used was entropy again

I found that the best combination was using no maximum depth, a maximum number of features of $\sqrt{number\ of\ features}$, a minimum number of samples per leaf of 1, a minimum number of samples to split of 2 and number of estimators of 100.

Using those values, the Random Forest classifier got a training accuracy of 100% and a testing accuracy of 100%, which are the same results as the Linear SVM. As for the SVM, the confusion matrix shows that every tuple was correctly classified.

Conclusion

The best classifiers were Random Forest and SVM, with a 100% accuracy for both.