Analysis

Problem: Predict machine failure in a production facility.

Summary of approach:

Using k-nearest neighbor and naive Bayes classification algorithms, we investigate and attempted to find the optimal ML (machine learning) tool to predict machine failure. The KNIME Analytics Platform (Konstanz Information Miner) was utilized to deploy these algorithms and view their results.

General data observations:

The initial dataset used contained 350 observations which had an associated record number, hours run, average hours between maintenance, model version and if the machine failed. The last 50 rows did not have failure data and was excluded from the analysis. The failure rates associated with their model number and missing failure rates are listed below:

Model Number Failure (0 = no failure | 1 = failure | no info on failure)

1: 0=81 | 1=19 | 17 no info

19/100 = 19% failure rate with 17/117 = 15% no data

2: 0=79 | 1=10 | 14 no info

10/89 = 11% failure rate with 14/103 = 14% no data

3: 0=78 | 1=23 | 19 no info

23/101 = 23% failure rate with 19/120 = 16% no data

Note: Each model number had approximately the same percentage of missing data, which provides equal comparison when using only the data with failure rates.

K-Nearest neighbor

This algorithm finds the Euclidean distance from a data point to the its k closest neighbors and classifies the outcome based on a majority vote of the associated closest points.

The analysis was obtained reading in the data frame, used a column filter to remove the record number and model version, normalized hours run and average hours between maintenance columns using A-score method, changed the failure qualities from a number to a word to allow the KNN (K-nearest neighbor) algorithm to compute correctly, filtered out the last 50 rows that did not have failure data, partitioned the remaining 300 data fields into 70% for training and 30% for testing, displayed the results, and printed them to a CSV file.

Multiple iterations of this algorithm for k = 3 through k = 6 were completed using random sampling and random sampling with the same seed which can be noted below.

K results accuracy (no seed | seed)
k=3: 0.922, 0.967, 0.956, 0.911 | 0.944, 0944
k=4: 0.967, 0.944, 0.967, 0.944 | 0.933, 0.933
k=5: 0.944, 0.967, 0.956, 0.933 | 0.944, 0.944
k=6: 0.956, 0.967, 0.956, 0.956 | 0.922, 0.922

The accuracy of the non seeded values varied significantly and a value of **k = 5** with seed was ultimately decided based on the ability to reproduce results and it's consistent high accuracy. A value of k = 2 with seed also had the same accuracy, but the non seeded values (of k = 2) appeared consistently lower than that of k = 5, and thus (k = 5) appeared to be the best value for accuracy.

Naive Bayes

The overall flow of the naive Bayes functions were the same as KNN with a few exceptions. I decided to do three different data flow paths for this algorithm to see if removing the machine version data and normalizing the data before the ML process affected the outcomes. After processing each flow path, there was no difference in the accuracy between excluding the version and not, but there was an improvement in normalizing the data before using the naive Bayes learner.

Comparison of the two models

The full outputs of the scoring in each algorithm can be seen in the screen shots or CVS files created. After analyzing the outputs, the KNN model outperformed naive Bayes in pretty much every category. The accuracy of KNN was 94.4% compared to 78.9%; naive Bayes had high false positives at 19 and predicted failure for every test case (never predicted a failure); KNN's precision was high 94.6% with naive Bayes at 78.9%; and Cohen's kappa (the chance of not randomly predicting outcomes) was 82.3% for KNN and 0 for naive Bayes (not better than randomly guessing). The naive Bayes algorithm may have had trouble with this data set due to either zero probability factors influencing the overall probability, or because the naive Bayes algorithm assumes no correlations between the other variables and could be influencing the outcome if those correlations (like maintenance) are crucial.

Conclusion

With the different tests we have run so far, the KNN model is definitely going to predict – with a higher level of accuracy, specificity, and sensitivity, machine failure – and should be the algorithm we use to predict machine failure.

Future exploration may include investigating why model 2 has significantly fewer failure rates; and do increased maintenance intervals have a large enough impact to keep (or make sure happens consistently), or if it doesn't, we could save money by reducing maintenance frequency.