CMSC 435

Introduction to Data Science

11-30-2017

Ninja Turtles

Aditya Vadrevu

Jordan Mays-Rowland

Andrew Bailey

Ahmed Soliman

*Description of the design of the prediction system*

The design of the prediction system involved us using multiple algorithms in RapidMiner that allowed us to see the level of accuracy in which each algorithm produced. Here we discuss the algorithms we thought were the most accurate according to the results given. Each proposed algorithm below is a representation of each member’s strongest model with regards to accuracy.

First and the best model we used was the *Gradient Boosted Tree*, with all parameters in default but with changing Minimum Rate parameter from 10 to 15. We chose this algorithm because it provided the best accuracy at 50.2%. We also tried playing with the learning rate by increasing and decreasing the default value, but we found that changing it lowered the accuracy. We did not perform feature selection on this algorithm because the accuracy when we got rid of the values ended up dropping. When we did try feature selection, we got rid of columns with 2000 or more 0’s and also tried imputing columns with less than 1000 0’s.

The second algorithm used was the *k-NN* algorithm, with changing the only given parameter from k = 1 to k = 10 causing an increase in accuracy. We chose this algorithm because in assignment 3, the dataset we used provided an excellent accuracy we thought it would translate well into this project as well, also it provided us with the fourth best accuracy of 41.1%. No feature selection was done on this algorithm because when we attempted to do it the accuracy was dropping, just like what occurred in the *Gradient Boosted Tree* algorithm.

The third algorithm we chose is the *Deep Learning* algorithm that provided us with the second best accuracy of 49.96%, we chose this algorithm because it uses unsupervised machine learning algorithm that learns by examples and it is known, because of the approach, that it shows very high accuracy. We kept most parameters at default and changed *L2* from 0 to 0.1 and *rho* from 0.99 to 0.996. We chose those parameters because *L2* produces substantial gains in modeling, according to RapidMiner. Again, no feature selection was done after we attempted it because it causes a decrease in accuracy.

Fourth algorithm was used was the Vote method with Decision tree, k-NN, and Generalized Linear Mode. We used the Vote method because it uses a majority vote for classification and the average for regression, we thought that approach should give us a high accuracy. After attempting the algorithm we got an accuracy of 46.56%. Since this vote method is based on votes, there were no parameters to play around with in order to change the accuracy. We tried to use feature selection but, similar to the other algorithms, the accuracy when we got rid of the values, the accuracy was dropping. the accuracy was dropping because there wasn’t enough data. In an attempt to use different different design options, we tried discretizing by size, binning, and frequency, however, those did not help out values improve by much so we didn’t actually use them.

*Results*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Outcome | Quality measure | ***Gradient Boosted***  ***Tree*** | k-NN | Deep Learning | Vote |
| material production failed | Sensitivity | 65.9 | 50.6 | 74.6 | 83.6 |
| Specificity | 62.6 | 62.1 | 53.6 | 38.1 |
| PredictiveACC | 63.7 | 58.1 | 60.9 | 53.8 |
| **MCC** | **0.270** | **0.122** | **0.271** | **0.224** |
| purification failed | Sensitivity | 10.2 | 10.2 | 0.6 | 0.3 |
| Specificity | 94.9 | 92.3 | 99.6 | 99.97 |
| PredictiveACC | 78.0 | 75.9 | 79.8 | 80.1 |
| **MCC** | **0.085** | **0.036** | **0.008** | **0.034** |
| crystallization failed | Sensitivity | 2.6 | 5.2 | 0.0 | 0.0 |
| Specificity | 99.0 | 97.2 | 100.0 | 100.0 |
| PredictiveACC | 87.6 | 86.3 | 88.2 | 88.2 |
| **MCC** | **0.047** | **0.046** | **0** | **0** |
| crystallizable | Sensitivity | 74.9 | 62.3 | 71.5 | 52.3 |
| Specificity | 69.5 | 61.5 | 70.8 | 80.5 |
| PredictiveACC | 71.3 | 61.8 | 71.0 | 71.1 |
| **MCC** | **0.421** | **0.225** | **.403** | **0.337** |
| **averageMCC** | | **0.206** | **0.107** | **0.171** | **0.149** |
| *accuracy* | | 50.2 | 41.1 | 49.96 | 46.56 |

*Conclusions*

Our results compared to table 1 follow the same pattern where the material production failed and crystallizable produced a much higher MCC than the purification failed and the crystallization failed. Our max MCC was for crystallizable with a value of .421 and our lowest was .047. These values represent that we can calculate crystallizable outcomes with relative accuracy, whereas our prediction for crystallization, while better than a coin flip, is still pretty inaccurate.

An advantage of our model is that it is easy to reproduce. This simple design approach lends itself to the concept of maintaining the greatest results compared to the amount of work required to reach a desired accuracy. Another advantage is that we are using a gradient boosted tree which means it constructs a tree in a greedy manner. This means that we have greater accuracy for some values. A disadvantage is that this algorithm is pretty slow because it generates 20 trees for each of the outcomes thus creating 80 trees.

While working on this project we have noticed that usually when we get the accuracy of a method without changing any parameters we could only get up to +10% accuracy after changing parameters. We feel that these results fairly decent for the time we had. Producing a model that does better than random was difficult but once we got accustomed to using RapidMiner it became simpler to produce better models. In the future if we had more time we would use use the voting method with our other top algorithms to improve our results. We believe that if we found this method earlier it would have given us a better solution than the gradient boosted tree alone. Using the tree and other methods to vote on which outcome the prediction would be a better way to improve the purification failed and the crystallization failed predictions, thus increasing our overall accuracy and MCC.