Programming 3 - Final Assignment

Project: Predict large InterPro annotations accession by small InterPro annotations accession.

Student: Kai-Kai Lin

Student number: 435543

Supervisor: Dr. Martijn Herber

Keywords: PySpark, Machine learning, Decision Tree, Naïve Bayes, Random Forest

1. Goal

The main purpose of the project is to predict large InterPro annotations accession by small InterPro annotations accession. The large and small InterPro annotations accession are defined by the ratio of feature and sequence as below.

$$\frac{\text{feature length}}{\text{sequence length}} = \frac{(\text{stop location} - \text{start location})}{\text{sequence length}}$$

If the ratio is greater than 0.9 it is defined as large InterPro annotations accession, vice versa is small annotations accession.

2. Inspect the data and pre-process the data

➢ Data loading

The tsv file in "/data/dataprocessing/interproscan/all_bacilli.tsv" was loaded by SparkSession with 128g driver memory and 128g executor memory to prevent out of memory.

➢ Data inspection

There are 4,200,590 data in the original data. If the InterPro annotations accession does not exist, it should be removed. Furthermore, large and small InterPro annotations accession must match each other. That means one large InterPro annotations accession must match at least one small InterPro annotations accession. After filtering the data is reduced to 1,198,607 with 413,683 large InterPro annotations accession and 784,924 small InterPro annotations accession.

➢ Data processing

Due to small InterPro annotations accession is almost two times more than large InterPro annotations accession. One large InterPro annotations accession will match around 2 small InterPro annotations accession. The correct count of small InterPro annotations accession in each large InterPro annotations accession is calculated in Figure 1, so using group by and pivot the small InterPro annotations accession to distinguish large InterPro annotations accession will be useful. There are several InterPro annotations accession with the same Protein accession at large InterPro annotations accession, I only remained the largest large InterPro annotations accession for each Protein accession.

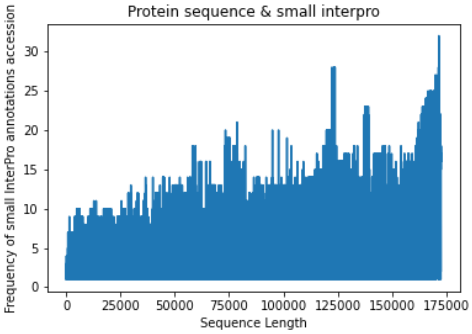


Figure 1: Different proteins and the frequency of small InterPro annotations accession

3. Define X(features) and y(labels)

The largest large InterPro annotations accession in each Protein accession will be set as y values. The counted small InterPro annotations accession for each Protein accession will be set as X values.

4. Set machine learning pipeline

- **Catalogize X and y variable**

y variables are string should be changed to integral. X contains too much data, using one hot encoder is a nice choice to reduce the compute time. However, it contains too many zeros, it is not suitable for large datasets. Thus, I used VectorAssembler to keep the value and index more efficient.

- **Create model**

I tried the Random Forest classifier, Decision Tree classifier, and Naïve Bayes models for the classifying. After that, I try to use different parameters for tuning.

5. Model performance

Model	Random Forest	Decision Tree	Naïve Bayes
Accuracy (without tuning)	0.30	0.09	0.78
Tuning accuracy	0.30	0.14	0.85
Time (hr)	6.25	25.63	3.24

6. Output

Training data: /students/2021-2022/master/Kai_DSLS/trainData.pkl

Test data: /students/2021-2022/master/Kai_DSLS/testData.pkl

Random Forest model: /students/2021-2022/master/Kai_DSLS/RandomForestModel

Random Forest best model: /students/2021-2022/master/Kai_DSLS/RandomForestBestModel

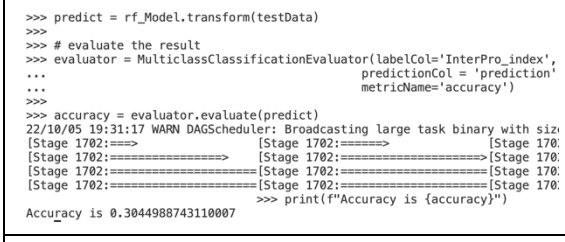
Decision Tree: /students/2021-2022/master/Kai_DSLS/DecisionTreeModel

Decision Tree best model: /students/2021-2022/master/Kai_DSLS/DecisionTreeBestModel

Naïve Bayes: /students/2021-2022/master/Kai_DSLS/NaiveBayesModel

Naïve Bayes best model: /students/2021-2022/master/Kai_DSLS/NaiveBayesBestModel

7. Appendix



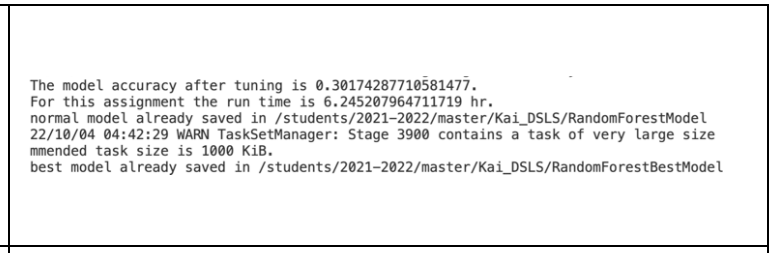
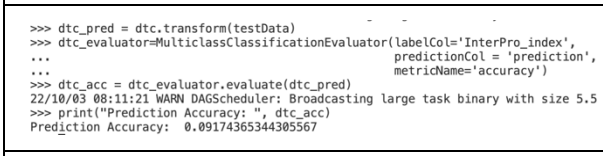


Figure 2: The result for Random Forest classifier

Figure 3: The result for Random Forest after tuning



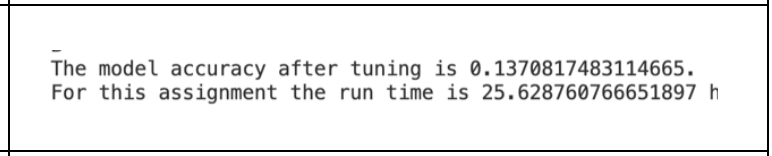
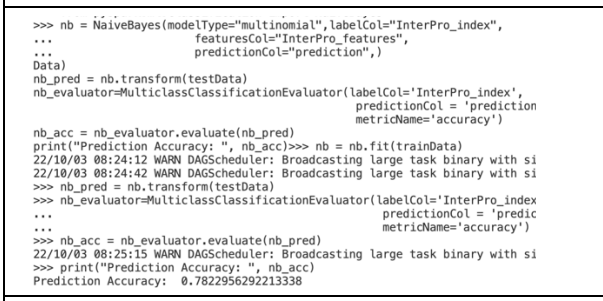


Figure 4: The result for Decision Tree classifier

Figure 5: The result for Decision Tree after tuning



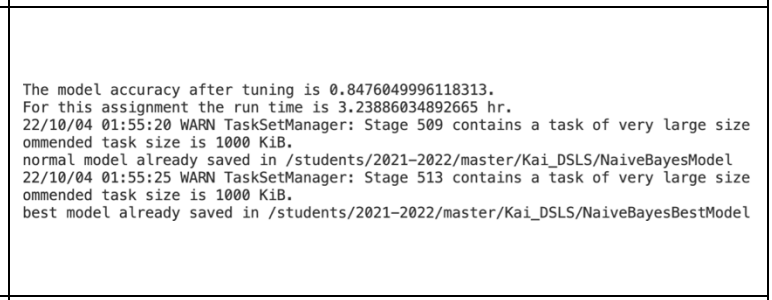


Figure 6: The result for Naïve Bayes

Figure 7: The result for Naïve Bayes after tuning

8. Conclusion

In the beginning, I tried to use Random Forest. However, it did not perform well. I can only get approximately 3% accuracy. I try to use different parameters and also use PySpark instead of half PySpark and half sklearn. I also search about whether there were some common methods for multiple labels and multiple features. I tried it one by one. Finally, get the accuracy of approximately 85% accuracy for training 3 hours.

9. Discussion

Although random forest is the most common model we use for machine learning, it did not perform well in this assignment. I believe training for higher depth and trees can make it better. However, I did not succeed in this model. Without tuning Naïve Bayes, it can get nearly 78%. If tuning for fewer parameters and putting the code for tuning to the pipeline module will optimize the process.