Data Mining and Machine Learnig Project
Genetic engineering

Balázs Menkó - O67UT7

December 17, 2022

1 Introduction to the dataset

As the last homework I had to chose a theme from a list of project suggestions. I decided to chose a dataset about Genetic Engineering. The dataset was a challenge hosted by Altlabs [1].

After a long search I found the dataset on Kaggle [2]. According to the description of the promblem I have to predict the laboratory of origin for given DNA sequences.

1.1 Description of the original dataset

There are 41 columns in the training and also in the test dataset but most of columns are one-hot encoded labels.

- `sequence` and `sequence_id` (type: string): The tested DNA sequence and its id.
- `bacterial_resistance_ampicillin`, ∼`chloramphenicol`, ∼`kanamycin`, ∼`other`, ∼`spectinomycin` (type: binary): It shows the antibiotic resistance of the plasmid used for selecting during bacterial growth and cloning.
- `copy_number_high_copy`, ∼`low_copy`, ∼`unknown` (type: binary): The number of plasmids per bacterial cell.
- `growth_strain_dh10b`, ∼`dh5alpha`, ∼`neb_stable`, ∼`other`, ∼`stbl3`, ∼`top10`, ∼`xl1_blue` (type: binary): One-hot encoded columns that indicate the strain used to clone the plasmid.
- `growth_temp_30`, ∼`37`, ∼`other` (type: binary): One-hot encoded columns that indicate the temperature the plasmid should be grown at.
- `selectable_markers_blasticidin`, ∼`his3`, ∼`hygromycin`, ∼`leu2`, ∼`neomycin`, ∼`other`, ∼`puromycin`, ∼`trp1`, ∼`ura3`, ∼`zeocin` (type: binary): One-hot encoded columns that indicate genes that allow non-bacterial selection (for a plasmid used outside of the cloning organism).
- `species_fly`, ∼`human`, ∼`mouse`, ∼`mustard_weed`, ∼`nematode`, ∼`other`, ∼`rat`, ∼`synthetic`, ∼`zebrafish` (type: binary): The columns that indicate the species the plasmid is used in, after cloning.

Labels

The labels identify the laboratory of origin for each DNA sequence. They are one-hot encoded, meaning there is a column for each lab ID. The correct lab of origin for each `sequence_id` is indicated with a `1.0`, and the rest of the values in the row are `0.0`. After a reverse transform a created a historam about the distribution of labs.

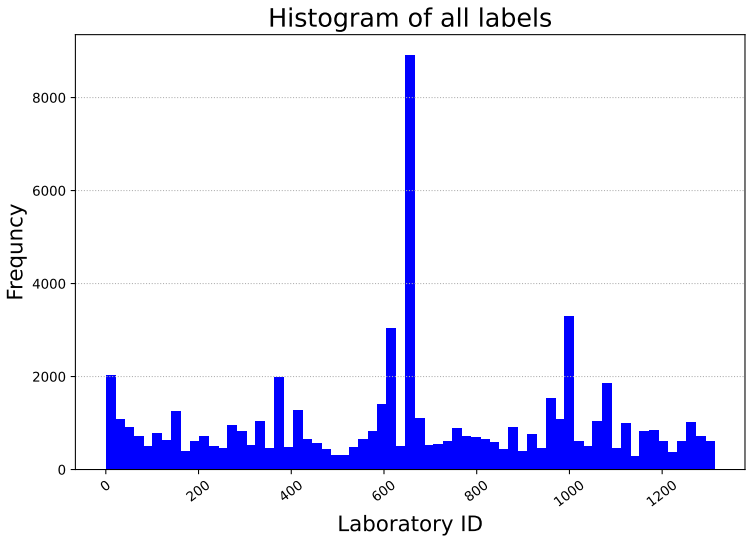

Figure 1: We can see this is not a uniform distribution.

There is a high peak in the distribution so models can predict this label. Sorted by frequency this the lab 'I7FXTVDP' and there are more than 8000 samples from there. From the next lab there are only 2732 samples so models can learn the frequent label and can predict that one.[1]

Preprocessing

First of all, I read the data into `pandas.DataFrame`, and I used my `df_info` function which is searching for missing values. However there van no missing values eiter in the dataset nor in the labels I should not clear it. This function showed us the shape of the raw data:

- `Train_values`: 39 columns and 63017 rows
- `Test_values`: 1314 columns[2] and 63017 rows

Lastly, I created a train (80%) and a test (20%) set and reverse transformed the one-hot encoded labels.

2 PCA – Principal Component Analysis

Principal Component Analysis shows us the directions with the highest variances. These directions are orthogonal to each other so it can give us a new

[1] As I wrote in the *Conclusion* I did not recognize this, only after I ran my models.
[2] Because of one-hot encoding.

coordinate system. As we learned on another course PCA can be easily code in python, but I used sklearn package. As the first three component are the most important in many cases, so I create a plot the three embedded 2D combination.

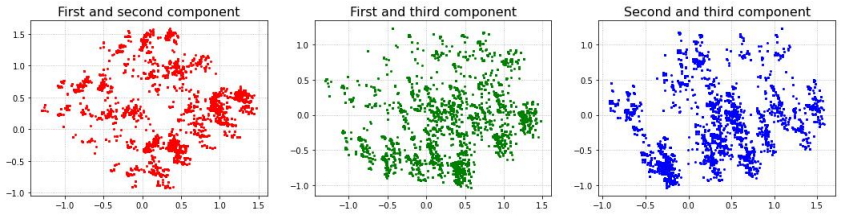

Figure 2: First three 2D combination

PCA shows us the relevance of the dataset's columns. `componens_` attribution give as a score which means the importance of a column. Ordering it a descending order we can see the most significant columns. In this situation the first three were `bacterial_resistanceq_ampicillin`, `copy_number_high_copy` and `bacterial_resistance_kanamycin`.

3 Logistic regression

Logistic regression can be used as a model to predict probabilities when there are more than two classes in the set. I used the methon from `sklearn` and the the `C` parameter to 0.01 and 1000. This parameter is a regularization parameter. It is the inverse of regularization strength so smaller values specify stronger control. With higher C parameter runtime was more longer, and accuracy was better. As I thought stronger control gave lower accuracy: 29.499%, but in the another case accuracy were 10% more. I create two plots about the distribution of predictions.

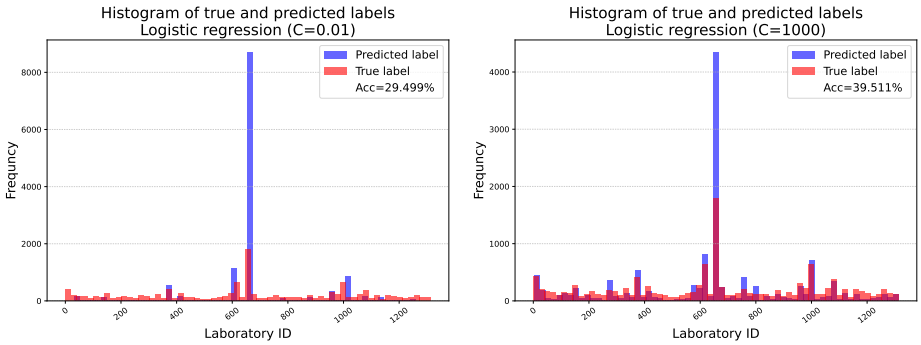

Figure 3: Histograms of the distribution of predictions by LogReg models

4 Decision tree & Random forest

Decision tree

In my project I used decision tree from sklearn. I set the criterion parameter to 'gini', and maximalized the features wiht $\log_2()$ function. By accuracy it turned better than logistic regression. On the next figure we can see the distribution of this model's prediction.

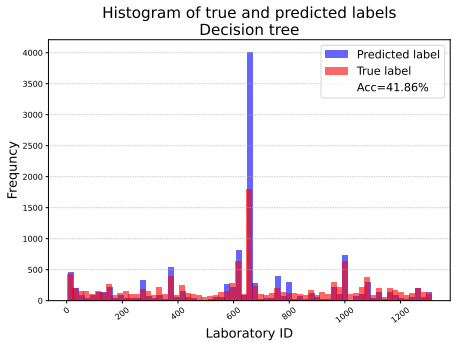

Figure 4: Histogram of the distribution of predictions by decision tree

This model can show us the features importance. Based on this `selectable_markers_puromycin` feature is the most important one with 9.3%. And lasty I create a plot with `plot_tree` function to show the decision tree.

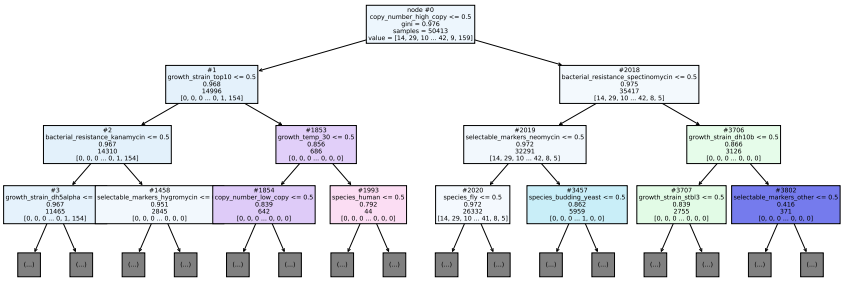

Figure 5: Decision tree

Random forest

The next model I used was Random forest. I set 140 tree in the forest, and maximalize the features with $\log_2()$ function. With this model I exceeded 42% accuracy. The mostly predicted label were still the most common label.

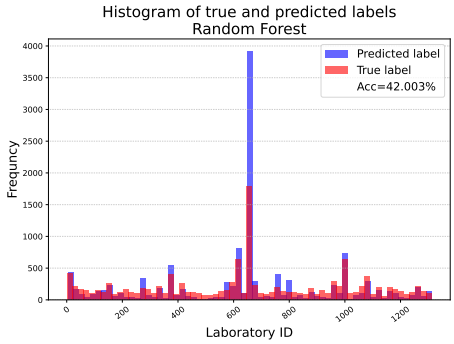

Figure 6: Histogram of the distribution of predictions by random forest

Sorting the features importances of this model give us the same as in the previous: `selectable_markers_puromycin`, but lower value with 1.05%.

5 Neural network

In this case I create a basic neural network. After the imput layer I put four dense layers with $600, 800, 1000, 1300$ neurons, and `relu` activation function. The output layer was a dense layer with 1314 neurons because of the number of predictable label, and I set the activation function to `softmax` to get probability as a predictions.I compiled the model with `categoricalCrossentropy()` as loss function, and I used `Adam()` optimizer. I fitted the model for 20 epochs and I set the bachsize to 64. After one and half minute runtime I got I little bit worse accuracy compared to the previous model: 41.98% for the test set. I created the histogram like before.

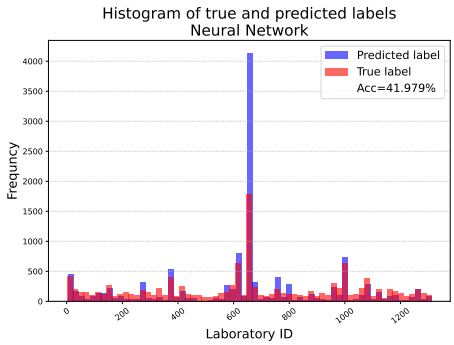

Figure 7: Histogram of the distribution of predictions by neural network

On the next two figures we can see the history of the model. We can see the loss on the test set is growing a little after the eighth epoch, and the accuracy fluctuates.

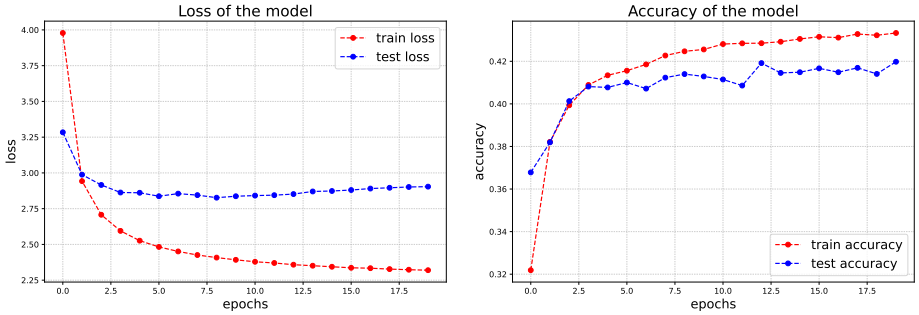

Figure 8: History of the neural network model

6 Conclusion

As we can see the best model could reach 93.9% accuracy in the competition. My models could reach about 40%. The main reason in the difference is I did not notice enougt soon the peak in the label's distribution and I assumed uniform distribution. It could be the reason of my accuracy can not exceeded this value.

But on the other hand I tried some different models and those gave me similar accuracy. It means the could be better considering the not uniform distribution.

References

[1] DrivenData – AltLabs: *Genetic Engineering Attribution Challenge*
`https://www.drivendata.org/competitions/63/genetic-engineering-attribution/page/165/#features_list`

[2] *The dataset from Kaggle*
`https://www.kaggle.com/datasets/darisdzakwanhoesien2/genetic-engineering-attribution-challenge`

[3] The author: *Python notebook and generated files*
`https://drive.google.com/drive/folders/1OS9BNZLAOWDQgKTgHu-w96RB_uM96L78?usp=share_link`