Identifying Cancer Types Given miRNA Profiles

Evan Blanpied and Michael Robertson

29 October 2019

1 Introduction

Our startup has worked to create a classifier that predicts a cancer type given a microRNA (miRNA) profile from a patient's blood sample. This will allow the patient's physician to tailor their immunotherapy treatment. The task was inspired by Telonis et al.'s paper "Knowledge about the presence or absence of miRNA isoforms (isomiRs) can successfully discriminate amongst 32 TCGA cancer types", in which the authors used support vector machines (SVMs) to successfully label tumor datasets with high accuracy. This report is an exploration into our own attempt to complete this task with new approaches.

2 Preprocessing

We received the data in the form of patient files grouped by type of Cancer, where each patient file contained a text file describing test results. In order to use this data for training, we first converted it to two csv files, one for features and one for targets. The targets file has a row for each patient, a column for each test, and the value of each test for each patient in the corresponding cell. The targets file is simply a list of the target values, where the patient in the i^{th} row of the features file has cancer in the i^{th} entry of the target file.

This form of the data allows us to conveniently call scikit learn's Decision Tree and Logistic Regression classifiers after we shuffle it and split it into training, validation, and testing sets. We used a 85%, 10%, 5% split for these three sets.

3 Classifier 1: Decision Tree

We first classified the processed data using a decision tree classifier. We used scikit-learn's decision tree library to create the decision tree, which could then be used to identify a cancer given an miRNA sample. Although the library creates the classifier itself, we were left with some options to make sure our classifier worked as well as possible.

First, we could select the criterion for our split decision, that is, we could pick which function should measure the quality of a split. We tested options, entropy, and Gini impurity, multiple times, and it was clear that using entropy as our split criterion yielded higher accuracy over several trials.

Then, we wanted to limit the growth of the tree in some way to avoid generating an unpruned tree. A tree that is allowed to grow to its maximum size will waste memory and likely overfit the data, so we wanted to control the size of the tree while maintaining its ability to correctly classify miRNA data. To do this, we set the `max_depth` parameter to 100, limiting the tree to a depth of 100 layers but not lowering its accuracy.

Our final classifier has an accuracy of around 91.5%, which is an increase of around 10 points from the classifier that was produced without any fine tuning. Telonis et al. created a classifier with "pan-cancer sensitivity $\geq$ 90%" and a false detection rate of about 3%, so our classifier is in line with these findings.

4 Classifier 2: Logistic Regression

For our second classifier, we used a logistic regression model. We again used scikit-learn's logistic regression classifier library to simplify the task. Again, we chose parameters to make our classifier as fast and accurate as possible.

The scikit-learn library gives several options for the `solver` parameter: "newton-cg", "lbfgs", "liblinear", "sag", and "saga". We tested each of these solvers several times to determine which one produced the most accurate classifier, which was "liblinear". This makes sense since our dataset is not huge and scikit's documentation states that this solver is good for small datasets. This also requires the `multi_class` paramater to be set to "auto". One can read more about "liblinear" at sci-kit learn's documentation[1], but

[1] `https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`

breifly, it differs from other Logistic Regression models by penalizing a y-intercept and penalizes the both the l_1 and l_2 norms of the weights.

Like the decision tree classifier, we wanted to ensure that our logistic regression classifier could be generated quickly without sacrificing accuracy. We set the `max_iter` parameter to 100. This caps the number of iterations that the classifier performs, limiting the amount of time spent creating it. When testing different values for this parameter, we found that 100 iterations was reasonably fast and did not lower the model's accuracy.

Our classifier had an accuracy of around 97.0%, and therefore had a false detection rate of around 3%. This is similar to Telonis et al., who also had a false detection rate of around 3%. However, our classifier was able to distinguish between the six types of cancer the model learned with greater accuracy than Telonis et al.

Both models worked well, but by examining the F1 scores in the table below, we can see that Logistic Regression clearly outperforms the Decision tree with an average F1 score of 97% compared to 86%.

F1 score for a Decision Tree using Scikit-Learn						
Cancer	0	1	2	3	4	5
F1 =	0.95	0.79	0.64	0.99	0.81	1.00

F1 score for a Logistic Regression using Scikit-Learn						
Cancer	0	1	2	3	4	5
F1 =	0.99	0.95	0.98	0.99	0.92	1.00

See the table below to interpret the above tables:

Key	Cancer
0	Breast Invasive Carcinoma
1	Lung Adenocarcinoma
2	Pancreatic Adenocarcinoma
3	Kidney Renal Clear Cell Carcinoma
4	Lung Squamous Cell Carcinoma 4
5	Uveal Melanoma

5 Contributions

Michael performed the cleaning of the data and Evan applied and tuned the models. Each partner offered the other help on their half and understood what the other was doing, but the work largely done separately.

6 Conclusion

Overall, we were successful in creating two models to classify cancer types given miRNA data. Our first model classified data using a decision tree, and the second used logistic regression. We report an average F1 score of 97% across the six cancers for our Logistic Regression models.