Building a Classification Model to Predict Cancer Type Using miRNA Readings

Anubhav Roy Bhattacharya and **Yiyao Xie**
`{anroybhattacharya,yixie}@davidson.edu`
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

This paper focused on identifying a patient's cancer type based on the readings of microRNAs (miRNA) in the patient's body. Our data consisted of miRNA readings from 2895 patients who were diagnosed with 6 different cancer types. Our features included the normalized values of 1881 unique miRNAs. We used four different model types: Logistic Regression, k-nearest neighbors (kNN), decision trees, and random forests. Our logistic regression model with an l-1 penalty, a liblinear solver, and the over multiclass parameter performed the best, with a weighted F1 score of 0.97233.

1 Introduction

Learning how to diagnose cancer accurately and efficiently has been the focus of medical research for a long time. Diagnosing cancer early can save the lives of some patients. As such, it is critical that there be an accurate way of solving this problem. In this paper, we build a number of classification models in an attempt to diagnose cancer type using microRNA (miRNA) readings. This paper was inspired by a paper written by Telonis et al. (2017) (Telonis et al. 2017), and used a subset of their dataset (Institute 2018). The dataset we used contained collective information about 2895 patients regarding their miRNA readings and the specific cancer types they were diagnosed with. Here we looked at 6 different cancer types:

- Breast Invasive Carcinoma
- Kidney Renal Clear Cell Carcinoma
- Lung Adenocarcinoma
- Lung Squamous Cell Carcinoma
- Pancreatic Adenocarcinoma
- Uveal Melanoma

The readings for each miRNA level comprised the features for our dataset, while the target variable was the cancer type. Next, we will go over how we manipulated the dataset to transform it into a form that could be used by our models. After that, we will explain the background of our models and summarize the experiments we ran. Finally, we will go over our results and talk about possible future steps as well.

2 Background

Models

We tried the logistic regression, k-nearest neighbors (kNN), decision tree, and random forest models from Python's `scikit-learn` library on our dataset (Pedregosa et al. 2011). Our logistic regression model utilized a sigmoid function, defined as

$$\frac{1}{1 + e^{\sum_{i=1}^{n} \theta_i X_i}}$$

A sigmoid function takes a real value number and maps it to a value between 0 and 1.
For the kNN models, we varied the method used to calculate the distance between two nodes, and also the number of closest neighbors to consider. Thus, for any two data entries x and y, the method used to calculate the distance between them varied according to the formula

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

which is known as the Minkowski distance. Two special cases of this formula occur when $p = 1$, which results in Manhattan distance being used, and when $p = 2$, which results in Euclidean distance being used. Thus, p was the hyperparameter that we varied.
For the decision tree models, we used both the gini impurity criteron, which is defined as

$$Gini : Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

and entropy information gain, which is defined as

$$Entropy : H(E) = - \sum_{j=1}^{c} p_j \log p_j$$

These criteria are used to determine which node to install at each level of the decision tree.
The random forest model was `RandomForestClassifier` from `scikit-learn` and used the gini criterion as well.

Data Organization

The dataset could not be used directly in its original state, and had to go through some preprocessing and reformatting. Initially, the data was stored in 6 folders, one for each type of cancer. Each one of these folders contained further sub-folders that housed the details of a single patient with that type of cancer.

We wrote a Python script that checked all the files and made sure that each patient's data included miRNA features that were identical to and in the same order as every other patient's data. This was to ensure that no patient had any missing features, and that all features for each patient occurred in the correct order. Fortunately, our script revealed that all patients had the same features and in the same order.

Following this, it was necessary to pull all the patient data into one single file. This file would contain the miRNA types as the features, and the cancer type as the target variable. Another Python script was written to go through all the files and store the miRNA features and their normalized reads in a `pandas` DataFrame. Finally, the cancer type of each patient was encoded with an integer value and stored as the target variable in the DataFrame. The values assigned to each cancer type are enumerated below.

- Breast Invasive Carcinoma: 1
- Kidney Renal Clear Cell Carcinoma: 2
- Lung Adenocarcinoma: 3
- Lung Squamous Cell Carcinoma: 4
- Pancreatic Adenocarcinoma: 5
- Uveal Melanoma: 6

All the data from one patient constituted a single training example. The resulting dataset had dimensions of 2895x1883. Thus, there were a total of 2895 patients, with 1881 features, not including the target variable. We then wrote the `pandas` dataset to a file called `tcga_data.csv` to circumvent having to recreate the dataset for every model we ran. After this step, the data was read in directly from `tcga_data.csv`.

Preprocessing

The data read in from `tcga_data.csv` had to undergo some further preprocessing before it was ready to be used by our models. First, we selected our target and feature columns using DataFrame slicing operations. Then, we split the dataset into a training set that included 80% of the data, and a test set that comprised of the remaining 20% of the data. For this, we used `scikit-learn`'s `train_test_split` method. Then, we used `scikit-learn`'s `StandardScaler` to standardize both the training and test sets. The test set was standardized according to the mean and standard deviation of the training set data, so as to prevent any information leaks.

Feature Selection

A manual glance at `tcga_data.csv` showed that a few of the features had very low variance. We decided to exclude all features with a variance of less than 0.1 using `VarianceThreshold` from the `scikit-learn` library.

3 Experiments

To evaluate the performance of all models, we used weighted F1 scores, since accuracy could be misleading and non-representative of our models' performances, given the imbalance of various cancer types in the dataset. Weighted F1 scores have the added advantage of taking label imbalances into account. To retrieve the scores, we used the `f1_score` function from `scikit-learn`.

Logistic Regression

We varied a number of parameters for our logistic regression model.

1. Penalty: This was varied between the l-1 penalty and the l-2 penalty.
2. Solver: The default solver of `scikit-learn`'s logistic regression model is liblinear. This solver serves well for small datasets. On the other hand, sag and saga work faster for larger data. There is also lbfgs, which together with sag and saga, can be used for multiclass problems. Note that not all solvers handle both l-1 and l-2 penalties. For example lbfgs and sag only work with the l-2 penalty, whereas the opposite is true for liblinear and saga.
3. Regularization Strength: In other words, the penalty for over-complex models. This number is a numeric value which we performed simple grid search on.
4. Multiclass: This can be *multinomial* or *ovr*. *Multinomial* views each target label as a separate classification label, while *ovr* creates a binary classification system for every label. Hence, *ovr* took longer to run in our case.

Moreover, we decided to look at both the macro and micro f-1 scores. Here is how they differ: macro means to take the average of the precision and recall of the system on different label sets; micro means that sum up the individual true positives, false positives, and false negatives of the system for different sets and the apply them to get the statistics. Hence, in other words, a macro-average will compute the metric independently for each class and then take the average (thus treating all classes equally), whereas a micro-average will aggregate the contributions of all classes to compute the average metric.

k-Nearest Neighbors

For our kNN models, we varied the distance method used, and the number of closest neighbors considered. Our distance method varied over $p = 1$ (Manahattan), $p = 2$ (Euclidean), and $p = 3$ (arbitrary Minkowski). We varied the number of closest neighbors considered over 3 and 5 neighbors.

Penalty	Solver	Multiclass	F1 score (weighted)
l-1	liblinear	ovr	0.97233
l-2	saga	ovr	0.97064
l-1	saga	ovr	0.96725
l-2	lbfgs	ovr	0.96542
l-2	saga	multinomial	0.96539
l-2	liblinear	ovr	0.96537
l-1	saga	multinomial	0.96370
l-2	lbfgs	multinomial	0.95692

Figure 1: Performance of logistic regression models (in order from best to worst).

Decision Tree

Due to the nature of decision trees, not much parameter tuning was needed. However, we did vary the criterion for node installation between gini and entropy. These criteria were described in more detail in the previous section.
One of the advantages of decision trees is that they are relatively easy to understand and visualize. After fitting our model, we used a visualization tool called `Graphviz` (Ellson et al. 2018), which was inspired by `scikit-learn`'s (Pedregosa et al. 2011) Decision Tree Classifier page. This allowed us to have a better understanding of the structure of our trees.

Random Forest

Given that `RandomForestClassifier` is a classifier that works relatively well without any tuning, for this paper, we didn't tune many parameters. We varied the criterion between entropy and gini. Apart from that, we used a standard implementation of random forests from `scikit-learn` with the default model parameters.

We also tried feature selection for all models with low variance (variance < 0.1) features removed.

4 Results

Logistic Regression

Since we performed grid search on the regularization strength, we will go over the best performing model for each penalty, solver, and multiclass. Our best performing logistic regression model had a weighted F1 score of 0.97233, and used the l-1 penalty, a liblinear solver, and the ovr multiclass parameter. In fact, our top 4 logistic regression models all used the ovr multiclass parameter. A summary of all our logistic regression models' performance can be seen in Figure 1.

k-Nearest Neighbors

Our kNN models performed decidedly worse than our logistic regression models. The kNN models that used Manhattan distance ($p = 1$) did better than models that used Euclidean distance ($p = 2$) or arbitrary Minkowski distance($p = 3$). However, even the best performing kNN model, with a weighted F1 score of 0.86538, did not do as well as the worst performing logistic regression model, with a weighted F1 score of 0.95692. A summary of the performance of all of our kNN models can be seen in Figure 2.

k	Distance Method	F1 score (weighted)
3	Manhattan	0.86538
5	Manhattan	0.85262
3	Euclidean	0.82993
5	Euclidean	0.82948
5	Minkowski	0.70945
3	Minkowski	0.69450

Figure 2: Performance of kNN models (in order from best to worst).

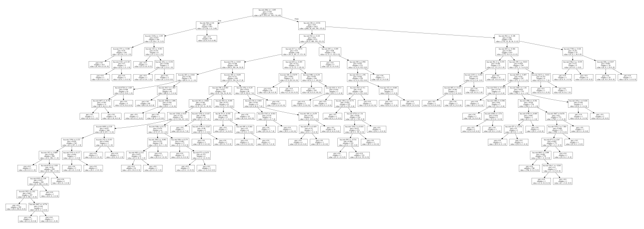

Figure 3: Decision tree with Gini installation criteria.

Decision Trees

For our decision tree models, we tried two installation criterion: gini and entropy. Our decision tree model using gini returned a weighted F1 score of 0.91197, while the model that used entropy returned a weighted F1 score of 0.92238. We expected that both these criteria would yield similar results, but it turned out that using entropy yielded a higher F1 score. We thought this was because gini is essentially a measure of how often a random sample would get categorized incorrectly, hence minimizing misclassification. However, entropy penalizes small probabilities, and is more useful for exploratory building of the tree. For decision trees, performance improved slightly when the dataset had low variance features deleted from it.

An example visualization of a gini decision tree can be seen in Figure 3. This tree has a maximum depth of 16. Moreover, the first pure node appears in $height = 2$ with 69 samples being categorized to Uveal Melanoma cancer types. This type of visualization provides an easy walk-through of the algorithm. Further, decision trees are useful because predicting the cancer type for a new example is $O(logn)$, where n is the number of nodes in the tree.

Random Forests

Similar to our decision tree models, we varied the node installation criterion: gini and entropy. Our random forest model that used gini had a weighted F1 score of 0.94453, while the model that used entropy had a weighted F1 score of 0.94464. These models both ran on datasets that had had low variance features deleted, and performed better than when no features were deleted from the dataset.

Thus, logistic regression seemed to perform the best overall, followed by random forests, decision trees, and finally, kNN. Deleting low variance features did not seem to have too much of an effect on the weighted F1 scores of any models, except the decision trees and the random forests. However, since deleting low variance features didn't adversely affect the performance of kNN and logistic regression models, the results reported earlier were all on datasets that had low variance features deleted.

Even though our decision tree models did not perform as well compared to our logistic regression models, there were still many advantages to trying them:

1. No heavy tuning of parameters involved.
2. Work for both continuous and numerical data.
3. Prediction of new data is logarithmic of the data points encountered during training.
4. Easy to understand and visualize.

These advantages made it easier to run many of our experiments.

5 Conclusions

In this paper, our goal was to build a classification model that could accurately predict cancer type based on the normalized levels of certain miRNA. The models we tried included logistic regression, k-nearest neighbors, decision trees, and random forests. Our logistic regression model that utilized an l-1 penaltywith a liblinear solver and ovr multiclass managed to perform the best, with a weighted F1 score of 0.97233. Given that our top 4 logistic regression models used ovr as the multiclass parameter, it seems that a one-vs-rest classification system worked best for this problem.

In the future, perhaps it would be beneficial to spend some more time tuning parameters of the random forest models. The random forest models were our next best-performing models. They work relatively well even with a standard implementation and no parameter tuning, and future explorations of this problem could involve further investigation into tuning parameters of the random forest.

6 Contributions

Y.X. built the logistic regression and decision tree models, while A.R.B. built the kNN and random forest models. Y.X. and A.R.B. both worked on the experiments and the results section. Y.X. worked on the abstract and the introduction, while A.R.B worked on the conclusion and also proofread the paper.

7 Acknowledgements

We would like to thank Dr. Ramanujan for his help with the background information and knowledge used in this paper. We would also like to extend our gratitude to Phillipe Loher, of Thomas Jefferson University, who first introduced this problem to us.

References

Ellson, J.; Gansner, E.; Hu, Y.; Janssen, E.; and North, S. 2018. Graphviz software. `https://graphviz.gitlab.io/credits/`. Retrieved on Feb. 27, 2018.

Institute, N. C. 2018. Tcga data. `https://portal.gdc.cancer.gov/`. Retrieved on Feb. 22, 2018.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Telonis, A. G.; Magee, R.; Loher, P.; Chervoneva, I.; Londin, E.; and Rigoutsos, I. 2017. Knowledge about the presence or absence of mirna isoforms (isomirs) can successfully discriminate amongst 32 tcga cancer types. *Nucleic acids research* 45(6):2973–2985.