blah

The thing is, thyroid diseases are often misdiagnosed or missed since their symptoms manifest in many ways and they often overlap with other mundane problems. And even blood work results can be obscured by other factors easily.

So I thought how about we just use a model to classify the disease?

Based on

blood test results.

So I got the dataset from the UCI machine learning repository,

the dataset includes some basic information like age and sex of the patients, but also whether they have had surgery before, and whether they are on medication. And most importantly, it has their levels of a couple blood markers, including TSH, T3, TT4, et cetera.

Also alongside those it also has their diagnoses, so whether they have thyroid diseases, and what variation they have.

So the raw dataset is a 9172×31 matrix. I had to do some basic sanitisation and augmentation for it to be usable. So first I purged the useless data columns like patient ID and referral source.

There are also quite a few non-numeric data, so I also have to change them into 0s and 1s.

Then there are some empty values where a certain blood marker was not tested, so I also have to fill those with 0s. Doing this should be fine since one of the non-numeric data from before is whether each marker was tested.

The interesting thing about this dataset is that each patient can have more than one diagnoses.

Each diagnosis is represented by a character, so like A-S. And there are 20 diagnoses in total, including healthy. And since each diagnosis is not mutually exclusive, this is actually a multi-label dataset,

---

and I have to do multi-label binary-sation instead of one-hot encoding.

So in the top example, the diagnosis is M and K, so the M and K columns are both set to 1. Same thing in the bottom.

And because some diagnoses are rarer than others, I did data augmentation by duplicating some data until they reach the mean value, just so each diagnosis is represented in the training data evenly. This ended up doubling the dataset size.

For classification, I use a random forest ensemble with 100 decision trees to each generate a class, then to congregate the results via majority voting.

Since the dataset is multi-class, I had to use one vs rest classification. That means I have to use 20 different classifiers, each looking at only one class. So 20 different random forest ensembles. They are like the red line in the right diagram, which only distinguishes between yes red triangle and not red triangle but not between the square and the X.

To combine the results from the 20 classifiers into one output, classifier chaining is used. So this is where each classifier is sequentially applied to answer yes/no for one class and that answer is fed to the next classifier as a feature, and so on.

For example, my dataset has 28 features, and the first random forest ensemble will decide whether the patient has diagnosis A based on the 28 features and it will give a yes/no answer. Then, that yes/no answer will be fed to the next random forest ensemble, and it will decide whether the patient has diagnosis B based on the original 28 feature plus the yes/no answer from the first classifier, totalling 29 features, and so on.

So I used 60% of the dataset for training and it got 100% accuracy.

And for testing, I used 40% of the dataset and got a 98% accuracy. You can also see the confusion matrix on the right. Interestingly, the diagnoses with fewer data points don’t seem to have suffered from that.

Here are the confusion matrices for each diagnosis if you are interested.

So in conclusion, it can help classify thyroid diseases with 98% accuracy.

And I think for a future direction, more features and more diagnoses can be included to discriminate differential diagnoses to help solve the misdiagnosis problem. And maybe also give more information about the disease.