Finding The Optimal Implementation of a Random Forest Algorithm For TCGA Cancer Data

Yusuf Mulyo
Department of Computer Science
Boston University
Boston, MA 02215
ymuluyo@bu.edu

Arundeep Singh
Bioinformatics
Boston University
asingh17@bu.edu

Parsa Shahbodaghi
Bioinformatics
Boston University
parsas@bu.edu

Abstract

The goal of this project was initially to take elements of the Extra Trees algorithm and make it more like the Random Forest. Upon further investigation, we realized that any meaningful modification to Extra Trees would essentially turn it into the Random Forest algorithm. We decided to take our project in a direction that would have more practical impact for individuals who would want a streamlined way to effectively run a Random Forest for this type of data. Our pipeline presents the output that would have the smallest mean absolute error for a range of possible combinations for the different hyperparameters and apply it to our dataset of lung cancer survival.

1 Introduction

The introduction will be laid out as follows. First, we are going to give some background on Random Forest algorithm itself. We will explain how it works, why it has been one of the most important algorithms in machine learning and explain some of its advantages and disadvantages.

We will briefly touch upon the initial conceit of the paper. How we figured out the direction we were going in wasn't productive and how that led us to change course. We will also go through what our previous goals were and the algorithms we developed before we decided to change course.

We will also describe the algorithm we used for the data analysis and the results of that analysis. We will go into further detail into how we accomplished such tasks in the methods section of the paper.

1.2 Random Forest Background

The Random Forest is an ensemble learner algorithm that grows and utilizes a large assortment of decision trees that vote for the best possible prediction. There are plenty of reasons for using an ensemble of decision trees as opposed to using one. The first is that decision trees are prone to overfitting and splitting at inappropriate sites. The solution to this problem, in the context of the Random Forest algorithm, would be to take the average output over many trees.

As a brief overview, here is generally how the code of the Random Forest operates in the general case. We have all our training data, our selected features that we will examine, and the number of trees that we will generate for our data. The first thing we will do is initialize our tree variable and for each tree we will generate a bootstrap replica sample from our sample space and we will call a function that generates a randomized decision tree for that bootstrap replica and all the features. We will add that decision tree to the ensemble of trees that we have generated and then repeat the process until we have the number of trees that has been previously specified in the algorithm.

As an aside, we thought we should briefly explain bootstrapping. Instead of sampling from the entire data set, bootstrapping allows for the sampling from a distribution that estimates the actual distribution of values. This bootstrap replica is made from samples drawn with replacement [2]. Bootstrapping can be disabled in the Random Forest, allowing from decision trees to be made from all the samples. Regardless of whether we decide to use bootstrap the algorithm takes a random set of features and decides which feature to split upon. This is perhaps the most computationally expensive part of the algorithm, because it requires a split to be made and determine the reduction in the mean square estimate, mean absolute estimate or some other metric of each split and the finalizing the split that has the greatest reduction in variance.

The function that we call to generate the tree uses a randomized smaller subset of the total number of features and takes a randomized split of the data for each feature throughout the decision tree [3][4]. It then returns that finalized decision tree that would be added to the ensemble of forests mentioned above [3][4].

The Random Forest has been one of the most effective and easily accessible algorithms in Machine Learning. In a ranking of the highest performing algorithms, the Random Forest was a close second to gradient boosted decision trees [7]. Since several small decision trees are being generated randomly into an ensemble, it does not require a tremendous amount of computational power to run the Random Forest algorithm.

In our case, we decided to use a Random Forest Regressor and not a classifier on our dataset. The reason for this is that we wanted to predict the probability of survival, which is a continuous value from 0 to 1. Because health data can be nonlinear we thought that the utilization of the Random Forest algorithm would be highly effective in predicting values since it has been known to work well with nonlinear data.

1.3 Initial Project Idea and Transition

Initially, we thought to find some way of modifying the source code to make the Random Forest more like Extra Trees. Since the only differences between the algorithms were that

Extra Trees does not use a bootstrap replica and that it makes its cut point choices randomly in growing the trees, we did not think that this would present a compelling project idea upon further examination. Because bootstrapping could easily be disabled in the Random Forest and we could pretty much randomize how the trees would be grown instead of looking for the optimal way to grow them we decided that this project idea would not be interesting and would not be particularly useful to anyone.

With that being the case, we wanted to describe some of what we did with the initial project idea. After realizing that the modification of the Random Forest into Extra Trees would not be effective, we thought of creating a better splitting algorithm than what was shown in the source code. Essentially, everything that we were trying to do with the goal of having a lower mean squared error per split could be done by changing the criterion hyperparameter and it would not be a productive use of time to alter the source code. After looking through the source code, we realized that it would be better to optimize the use of the Random Forest by looping through various hyperparameters to have the lowest mean squared error for the dataset. In the end we decided that it would be best to find an optimal implementation of the sci-kit learn Random Forest algorithm.

1.4 Background of Data

Data was collected from The Cancer Genome Atlas (TCGA), an initiative from The National Institute of Health (NIH) to centralize the clinical and genomic data relating to cancer. This has been an incredibly important tool in cancer research all over the world because it allows for researchers to conduct research on data that would be directly applicable to the work of those in the medical and cancer research community.

TCGA has been a highly reputable source for cancer data as it has been a collaboration of the government and nineteen other institutions. It contains roughly 2.5 petabytes of data and it has played a role in the discovery of the four different types of stomach cancer as opposed to simply one [1].

2 Methods

In this section, we will describe the methods that we have used in working with the Random Forest and the data processing and analysis. First, we will talk about the data processing. Second, we will explain the process of looping through the Random Forest by holding two highly predictive features constant and have an additional remaining feature that was the result of looping through our remaining features. As an aside, one of the features that we held constant in our Random Forest pipeline was one of the features used in the TCGA algorithm for predicting survivability, so decided not to test for it. Third, we will explain the process of looping through all the different hyperparameters to see which one gave us the lowest mean absolute error. We then recorded the output of the Random Forest in a csv file. We then plotted how different hyperparameters and features yielded different mean absolute errors and how we found the optimal result.

2.1 Data Processing and Reasoning Behind Algorithm's Use

We primarily processed the TCGA data from a JSON format into multidimensional dictionaries. Some information, specifically the data on simple somatic mutations, was processed from tab-separated-value files. We later turned the resulting dictionaries into csv files to work with our algorithm. When parsing the JSON file we filtered out values that we thought were redundant towards the research. The data was initially messy and there were multiple JSON files with different data associated with the same patient identification number. However, we eventually set up data processing algorithms that allowed us to pass in the patient identification into our dictionary and result in all the values associated with that

patient. After some data processing and intuitive understanding, we were also able to transform our categorical data with hierarchical values. For example, the different stages of cancer indicate cancer progression, the higher the number and letter associated with the stage, the more the cancer has progressed, and the more likely it will cause death. Ultimately, this data looked like a list with values ranging from 0 to 6, with the value of 1 corresponding to Stage 1a and 6 corresponding to Stage 4. We modified the data in such a way to take the patient identifications and the data associated with them (i.e. number of cigarettes they have smoked per day) from many files and process them into one comma separated value sheet for processing in our Random Forest algorithm.

Our data processing wasn't just limited to translating hierarchical values. We also needed to impute some missing values in the dataset. For categorical data we decided to use the mode because that would be the data that would have the highest probability of being the missing value given it was the most common value in our dataset. For our numerical data we decided to impute missing values with the median and not the mode. The reasoning for this is that the mean is more sensitive to change from outliers, so a dataset with a high variance could greatly skew the mean and imputing missing values with the mean may not be accurate. The median is much more resistant to outliers. By definition, half of the values in the dataset are higher and half the values in the dataset are lower, so the median is a better measure of central tendency. We therefore thought that the median would be the best option to replace our missing values.

We're getting survivability estimates to detect our accuracy. Predicting survivability and we're comparing it to their predictions. If our predictability is accurate we could then predict survivability using their function. We are trying to see which fields are most important in that prediction.

2.1 Train Test Split Function

Before we ran our Random Forest, we needed to employ a train test split function for our data set. This function essentially splits the whole data set into a training set and a testing set. We chose to train the model with 90% of the samples for our training dataset and 10% of the samples for our prediction dataset [6].

2.2 Running Random Forest With One Feature

We initially did a couple test runs of the Random Forest using only one feature. We wanted to see how well one feature would predict the survival estimate. We wanted to see how much our predictions would vary depending on the feature in the Random Forest (this is further elaborated in the figure).

2.3 Looping Through Features and Hyperparameters In Testing Random Forest

We created an array that contained all the features we decided to examine. For each hyperparameter, we created an array of the possible states for each hyperparameter. We were able to get all possible permutations by using a nested for loop. We used a data structure within the Random Forest that took the output of that nested for loop. We then ran the algorithm for every possible combination that was generated by that loop.

There were eleven features that we selected for our Random Forest: intermediate dimension of primary tumor sample, shortest dimension of the primary tumor sample, gender, year of birth, cigarettes per day, tissue or organ of origin (part of lung where the sample came from), age at diagnosis, tumor stage, number of simple somatic mutations, number of genes with

simple somatic mutations, the time since the patient had been diagnosed. The features: simple somatic mutations and number of genes with simple somatic mutations were added to our feature set after we completed five runs without them. We wanted to see how adding this genetic data would impact the accuracy of our model since genetic data tends to be highly predictive of whether one gets cancer we wanted to see if it would affect our model's ability to fit the data.

We ran the resulting pipeline from this feature set five times to account for variability and the mean absolute errors were returned in Figure 4. Since there some randomness in the way a Random Forest is run we examined the spread of the mean absolute error for the runs. Note we did not average the mean absolute errors from the five runs as that would have been overfitting (it would be the same as growing five times as many trees in one run of a random forest).

2.4 Running Same Pipeline In Breast Cancer

The same pipeline was run for breast cancer. The data was extracted from similar JSON files, as well as tab-separated-value files for the mutation information. We extracted the data from the TCGA website in a similar fashion to our lung cancer dataset. We decided to use the TCGA BRCA dataset with 1098 observations because of the completeness of the dataset. We made sure that the dataset only contained data relating to survivability.

A note about the data processing, the cancer stages had to be imputed in a similar way to the lung cancer dataset. There were a few exceptions though. There were Stage X's in the dataset, which didn't make sense to us, and we couldn't figure out the meaning of it through a Google search, so we decided to replace values of Stage X with "Not Reported," which won't affect the data one way or another.

One of the challenges associated with the data processing with the breast cancer dataset was finding the appropriate features for this dataset. Instead of gender and number of cigarettes per day, which were far less relevant to this dataset, we used race, ethnicity, days to last follow up appointment as features.

3 Results

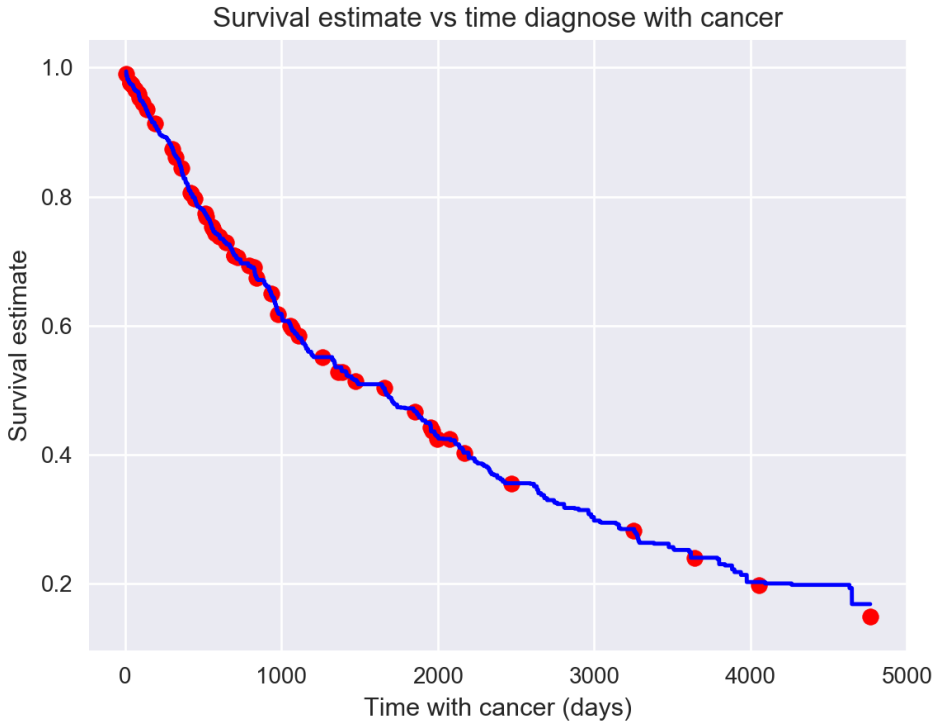

Figure 1: In this figure we ran a Random Forest with only one feature to look at how closely the predictions of the Random Forest match the testing data for lung cancer. This shows that the amount of time someone has cancer reduces the survival estimate. This was, of course, expected as this variable was directly incorporated into TCGA's determination of survivability (and thus, is why we held it as a constant feature throughout our tests). Our Random Forest depicts this nicely.

Figure 2: A similar run of the Random Forest in the lung cancer dataset. The feature of how much time a patient has spent with cancer has been changed to the shortest dimension of the Primary Tumor biospecimen. Unlike the time since diagnosis, we did not expect this feature to have much of a predictive effect. The Random Forest tries to fit this data, but there is too much spread to draw any meaningful relationship.

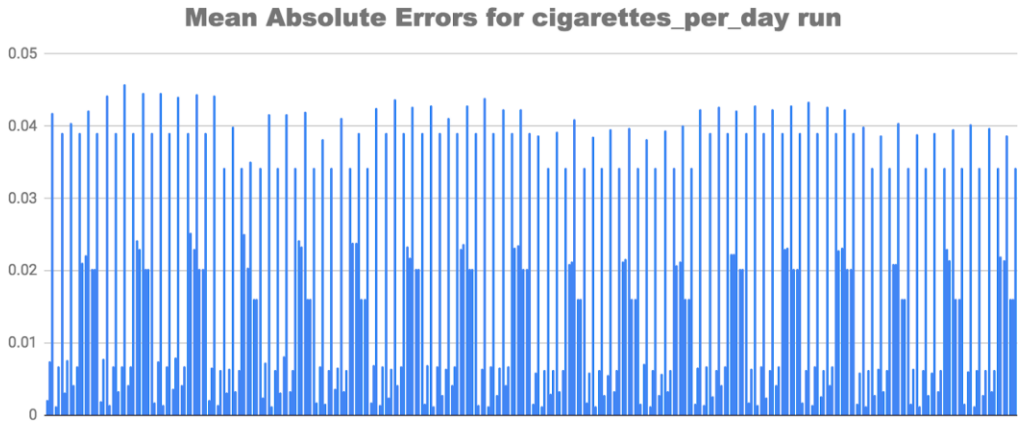

Figure 3: An example of the results on mean absolute errors in test of one feature (in addition to the constant features). This is a depiction of one full run on the cigarettes_per_day feature through every single parameter combination for a total of 324 combinations. Since there are seven features for each hyperparameter this graph represents 2268 total runs. The largest data spikes correspond to changes specifically in the minimum sample leaf hyperparameter indicating that the mean absolute error is very sensitive to changes in that hyperparameter. Our goal was to find the hyperparameters that result in the smallest bars on this figure.

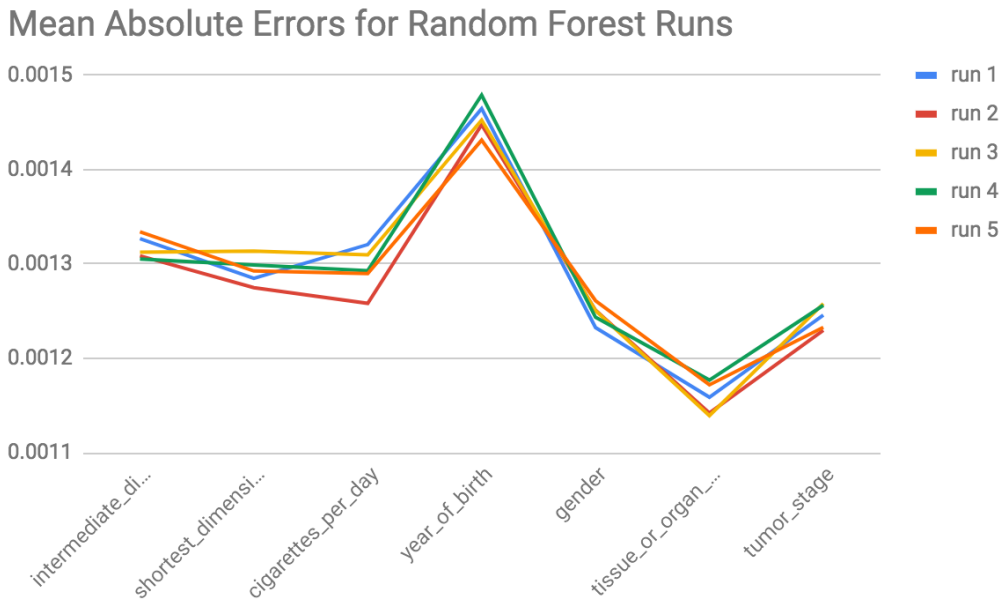

Figure 4: The minimum mean absolute error per feature across five runs of the Random Forest. This is done to show the spread of the predictions across and account for any variability between the runs of the Random Forest algorithm. At an absolute scale the variation in mean absolute error between each run, as well as each feature, is incredibly small.

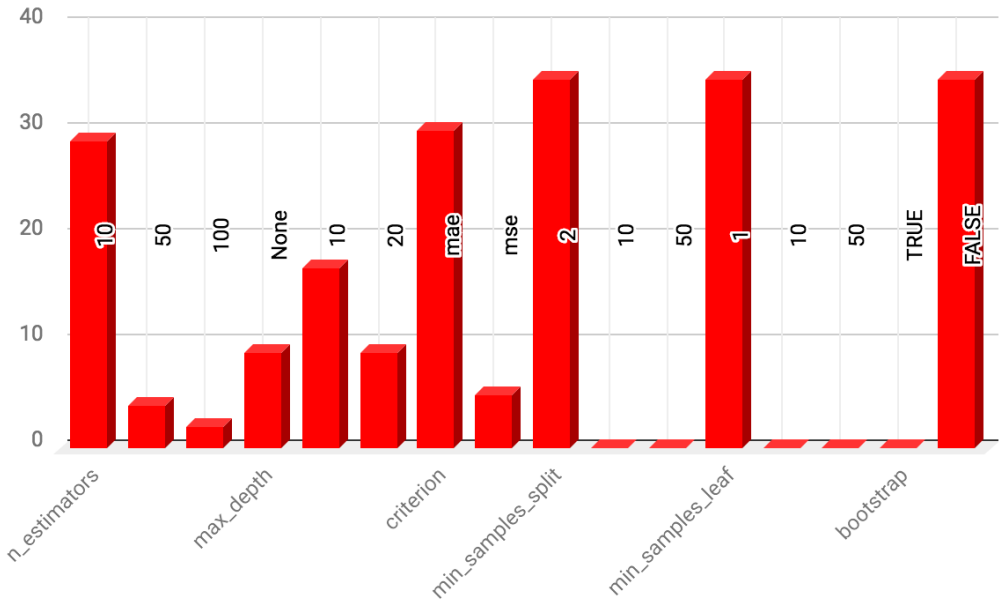

Figure 5: The representation for each of the different hyperparameters in the 5 runs (7 features * 5 runs = 35 possible occurrences) with the minimum absolute error across all five runs. Some interesting results are that having a lower number of trees generally corresponds to having a lower mean absolute error. A simple explanation for this is that there is

generally less overfitting with fewer estimators. Having a low minimum for a split or samples in a leaf also corresponded to a low mean absolute error. This is an interesting result because it does not seem to overfit the data.

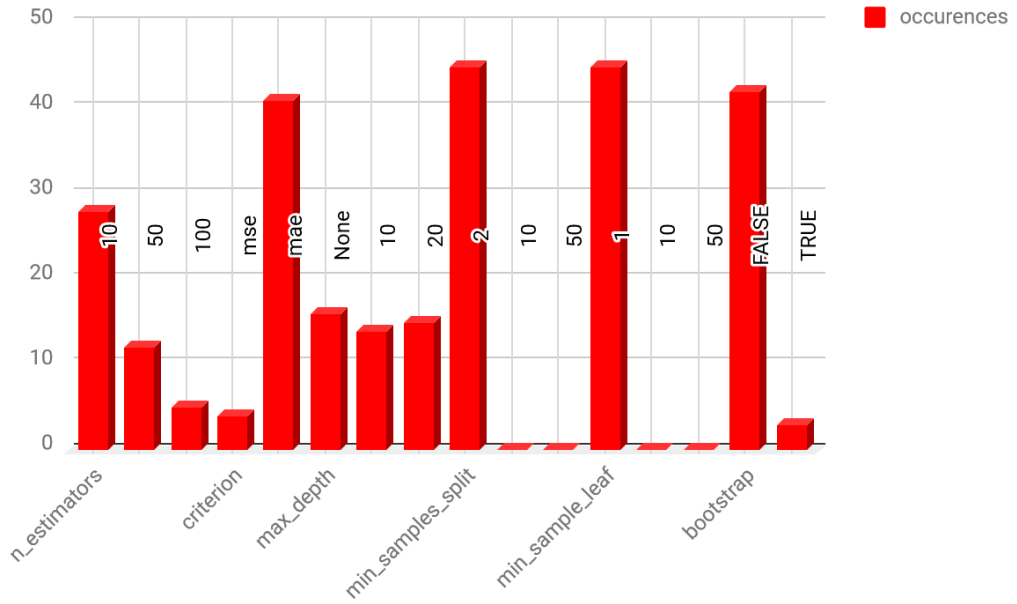

Figure 6: A representation of which hyperparameters occur within the runs yielding the minimum mean absolute errors when taking into account the additional genetic features. Mean absolute error is now much more represented for the criterion for a split. The minimum number of samples for a split and the minimum number of samples in a leaf remain the same.

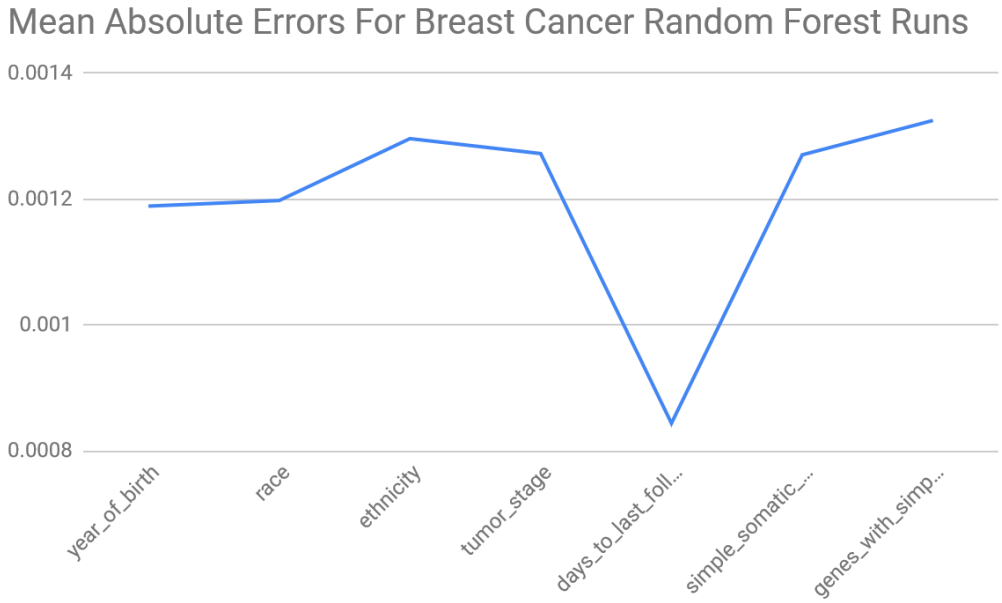

Figure 7: The mean absolute error for different features in the breast cancer dataset. The distribution of mean absolute errors in the breast cancer data seemed to be generally smaller

than those of the lung cancer data. This was likely the result of a much higher sample size for the training data as well as much more completeness in the individual parameters. The number of days to the patient's last follow-up appointment, in particular, was the most predictive feature based on having the lowest mean absolute error.

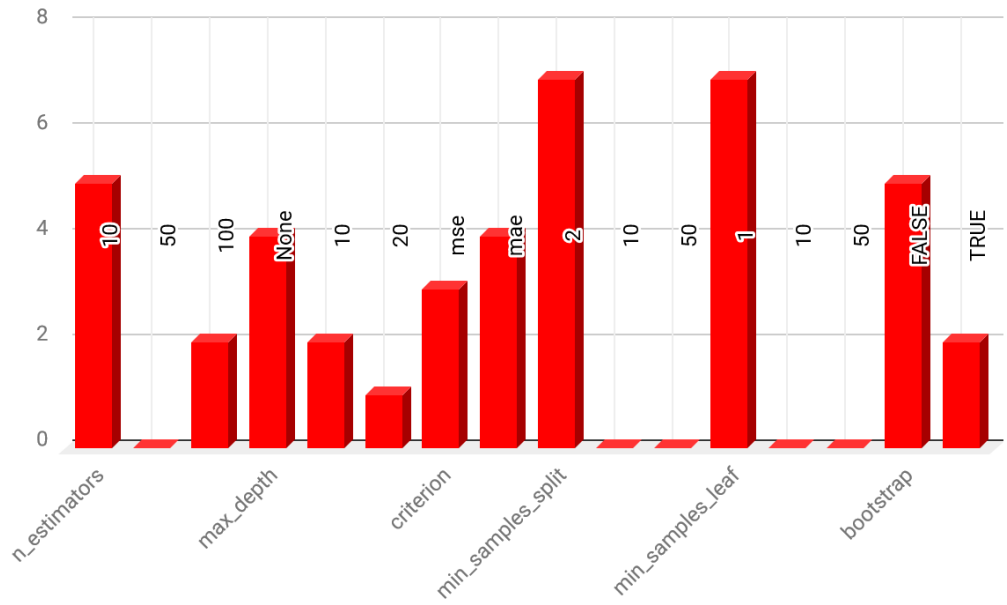

Figure 8: A similar bar showing which hyperparameters with the minimum absolute error across one run. Due to the lack of computational resources, time, and the breast cancer dataset being twice the size of the lung cancer dataset, we only have data representing one run, and thus a total of 8 occurrences (8 features * 1 run).

4 Discussion

The first takeaway for this project is that the Random Forest was highly effective at predicting the training data in the testing set for lung squamous cell carcinoma. The property that seemed to yield the smallest mean absolute error was tissue or organ of origin. It is interesting that this was the most predictive feature because the place where a tumor is found could determine how fatal it would be. For example, if the tissue of origin were somewhere near a lymph node or major blood vessel it would mean that the cancer could easily metastasize and spread to other tissues. Metastatic cancers are the ones that are most responsible for the deaths associated with the disease. The gender and the number of cigarettes smoked per day are also closely linked to the survival rate in general, so it is worth noting that our Random Forest pipeline's results match our intuition about the disease.

Running our pipeline on the lung cancer dataset did yield some surprising results for the mean absolute error. Even though the magnitude of the mean absolute error was small for the year an individual was born, the results for that data yielded generally higher errors than with the other features. One idea we might entertain is that even though age plays a factor in our ability to survive, it could be that it can be more limited in how it plays as a factor in comparison to lifestyle choices such as whether an individual is a cigarette smoker. Another possibility is that our sample size would need to be bigger for our model to better approximate which features would be the most predictive in our survival estimate.

Some insights that we found from the lung cancer dataset was that having the lowest minimum samples returned in a leaf were very sensitive hyperparameters, meaning that any

alteration to them could greatly alter the resulting mean absolute error. Another interesting insight is that given the larger dataset, bootstrapping became a hyperparameter that was less sensitive to alteration, which is likely due to the fact that sampling with replacement more closely approximates the actual distribution of data.

In the future, we hope to have more computing power to be able to run the Random Forest many more times across much larger datasets to find other ways to best optimize its use. We could work on implementation of this pipeline on a computing cluster so that we can get our results even faster. We hope that individuals will look at the implementation of our algorithm and find ways to optimize the use of the Random Forest.

For our breast cancer data, the year of birth again did not show to be particularly predictive. The days_to_last_follow_up was the most predictive of the survival estimate. As this is represents the number of days since the patient's last doctor visit, we might entertain the notion that breast cancer patients who do not follow regular appointments with their doctors may have a lower survivability. However, since the metric we used was the mean absolute error which does not indicate direction of trend, the contrary might also be the case. In other words, patients that stop returning to their doctor might not need to anymore as a result of remission. According to the TCGA database, the majority of the patients in our dataset were alive. This is important to note, as otherwise this data may not be as predictive of survival as our results would indicate.

We have a few important takeaways that go beyond the scope of the pipeline or the use of the algorithm itself. Running these tests, we realized the hyperparameters can be altered and the features can be changed, but the quality and the extent of the data matter greatly in making predictions. The Random Forest had a lower overall mean absolute error with the breast cancer dataset than the lung cancer dataset and this is likely attributed to the breast cancer dataset having more than twice as many samples. There also tended to be very few missing values that needed to be imputed, so the overall integrity of our dataset was maintained. If that data were less reliable then the Random Forest algorithm would have been less effective.

Acknowledgements

We would like to acknowledge Professor Sang "Peter" Chin, Yida Xin, and Xiao Zhu for all their help throughout the project and this course. We greatly appreciate it.

References

[1] The Cancer Genome Atlas. (2017) *TCGA LUSC. TCGA BRCA*. www.cancergenome.nih.gov.

[2] Breiman, Leo & Culter, Adele. (1995) *Random Forests*. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#intro. Retrieved Aug 1. 2018.

[3] Koehrsen, William. (2017) *Random Forest Simple Explanation* pages.cs.wisc.edu/~matthewb/pages/notes/pdf/ensembles/RandomForests.pdf

[4] "Random Forests" http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/ensembles/RandomForests.pdf. Retrieved Aug 1. 2018.

[5] "sklearn.ensemble.RandomForestRegressor" http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Retrieved Aug 1. 2018.

[6] "sklearn.ensemble.model_selection_train_test_split" http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html Retrieved Aug 1. 2018.

[7] Caruana, Rich & Nicelescu-Mizil, Alexandru. (2006) *An Empirical Comparison of Machine Learning Algorithms* www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf. Retrieved Aug 1. 2018