Stats 101C Final Project

Prediction of Diagnosis of Breast Cancer in Patients

August 4, 2021

Group Members:

Benjamin Gerber (bengerber13@gmail.com)

Kuiwen Ma (makuiwen1997@gmail.com)

Kaggle Score: 0.65420

Kaggle Rank: 20th in Professor's Table

Abstract

The object of the Kaggle Project was to create the best possible model that would predict a diagnosis for whether a person's tumor was cancerous or not. The classification of diagnosis was one of two categories, "benign" or "malignant" based on the classification model. We used training data to create the model, and then testing data through to submit to Kaggle and get ranked. Each group was ranked based on how accurately their model predicted whether an observation was a cancerous or benign tumor. We were also graded separately for how simple or complex our model was. Simpler models got a better score, since they could be much more easily explained as well as being computationally less expensive. We tested multiple types of categorization models, including logistic, QDA, LDA, random forest, but ultimately decided that logistic performed the best, and was easiest to breakdown.

The final model was a logistic model with five variables that were not transformed, so the model was extremely simple. The model was

glm(as.factor(diagnosis) ~ radius_mean + fractal_dimension_mean +
texture_se + area_worst + menopause.

1. Introduction

Unfortunately, breast cancer is an extremely prevalent form of cancer in the world, and medical experts are trying to find ways to combat it effectively and efficiently. Although this project did not have the intention of curing breast cancer, it helps in another way. The goal of this project was to use a training data set to create a model that would accurately predict whether a tumor was benign (cancerous) or malignant (non-cancerous). Creating a good model could help precisely determine whether or not a patient had a cancerous tumor and quickly get them the treatment that would be needed to get rid of that cancer.

The data set that we were given to create the model consisted of 40 variables: 30 numeric and nine categorical. The 10th categorical variable was the response variable, “diagnosis,” which was only included in the training data set since the goal was to predict the variable “diagnosis” for each observation in the testing data set. . The training data set was 748 observations and the testing data set was 321 observations. Each of the variables related to either observations about the actual tumor cells on the patient, or background on the patient themselves. Some variables about the cells included radius, texture, perimeter, and smoothness. The dataset information given to us said “the mean, standard error and “worst” or largest (mean of the three largest values) of these features were computed for each image.” Variables about the background of the person in each observation included age or if the person had reached menopause.

2. Data Analysis

A. Data Cleaning

First, it is important to check if the data is missing any variables. Missing values in a dataset create issues with some functions in R, so we need to check if any values are missing at all and what variables they come from. We can use the “is.na” and “colSums” functions in R to calculate the missing values of both the training and testing data sets. Figure 1 and 2 below show a bar chart that represents how many as well as which of the variables are missing from each data set. We can see that breast.quad and node.caps are the two variables with missing values in both the training and testing data. Figure 3 shows the ratio of missing data from each of the data sets in terms of total missing values rather than percentages.

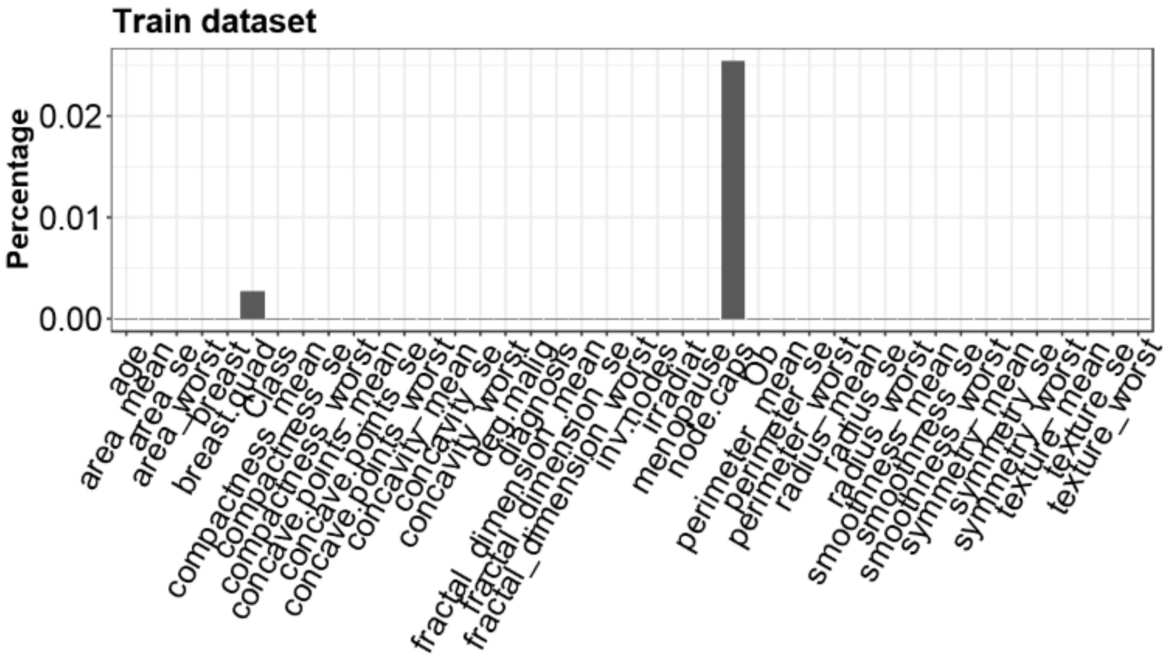


Figure 1: Missing Values from the Training Data Set

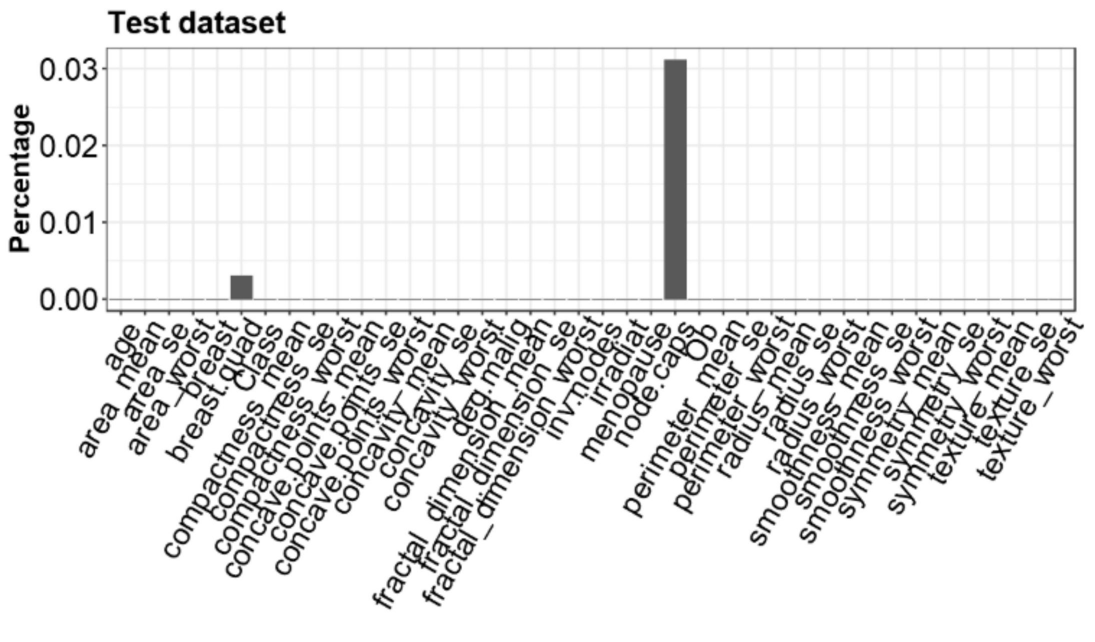


Figure 2: Missing Values from the Testing Data Set

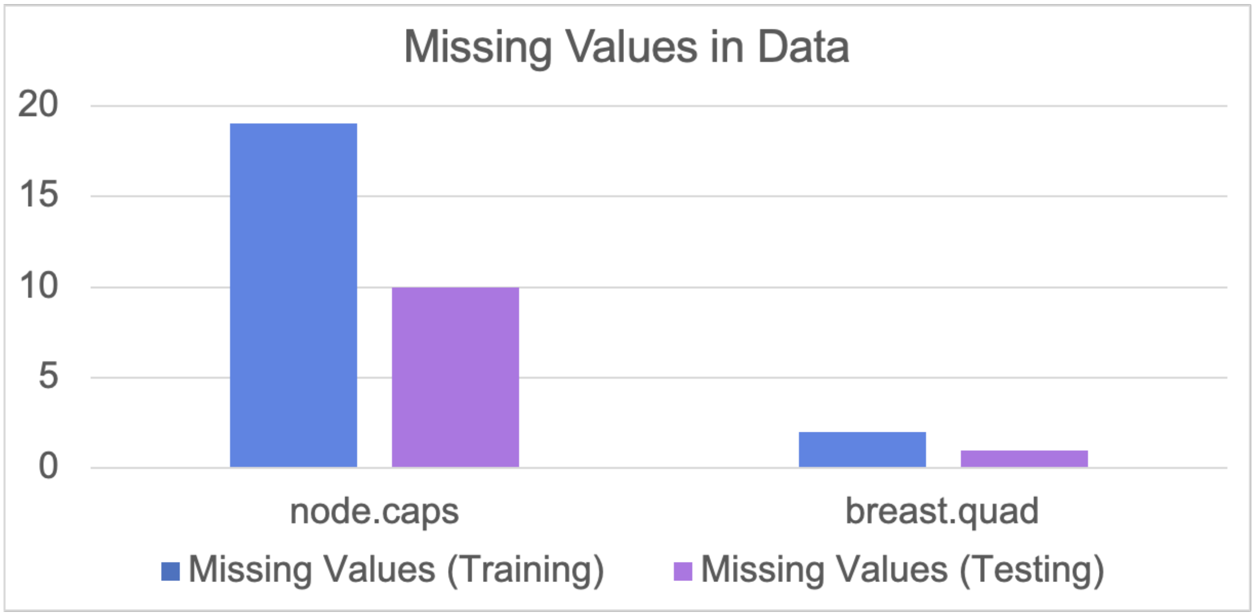


Figure 3: Missing Values as a Count from node.caps and breat.quad

Even though there are missing values for both the node.caps and breast.quad, there are only a few values missing for each variable. In total, the training data set is only missing 21 observations, which is only 2.8% of the full training data set. As for the testing data set, it is only missing data for 11 observations, or only 3.4% of the full testing data set. To combat the missing values in the training data set, we decided to use the function "na.omit" to get rid of the observations that had missing values. That way, we did not have to worry about functions not working while creating our model. We came to this decision since only 2.8% of the data was missing, so should not affect the final results by a significant amount. In addition, the variables with missing values (node.caps and breast.quad) were not significant when creating our final model, and so we did not have to worry about the observations that were missing them in the testing data.

B. Variable Analysis

The next important thing to do in creating our model was taking a look at some of the variables to see which ones looked more significant in the data than others. At first, we created density plots that showed the distributions for a certain variable and the diagnosis (benign or malignant). Figures 4 and 5 below show examples of the graphs we used in this analysis.

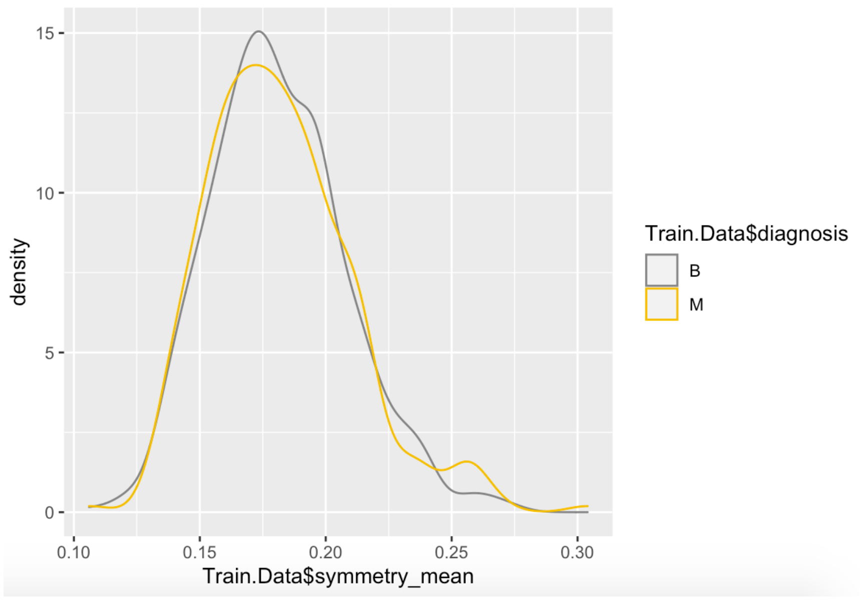


Figure 4: Distribution of symmetry_mean

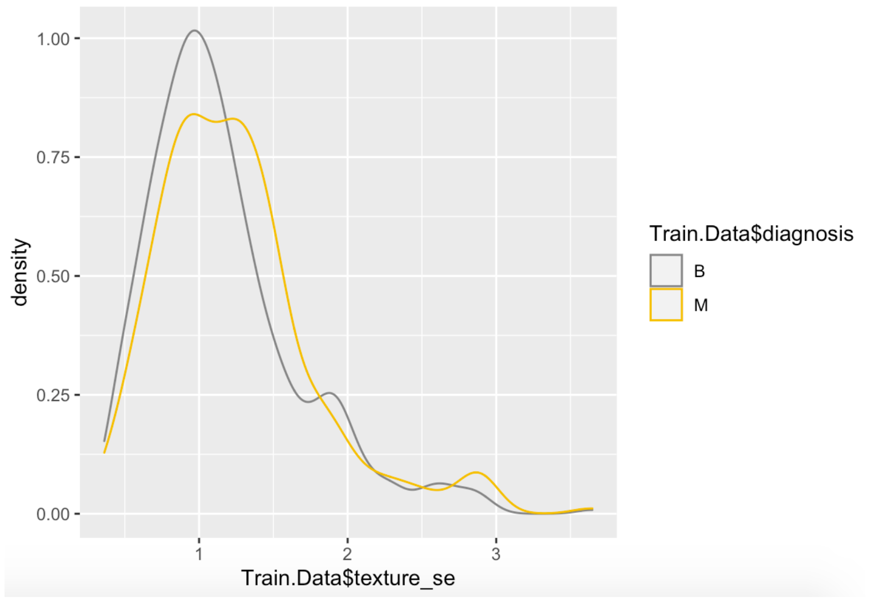


Figure 5: Distribution of texture_se

Figures 4 and 5 show that the distributions, like almost all the variables in the dataset, overlap a lot. This means that they are not great predictors in differentiating the diagnosis as benign or malignant. A perfect example of this is Figure 4, which shows the distribution of the variable symmetry_mean. The distributions differentiated for diagnosis are virtually the exact same and so we can say this variable does not help us to predict diagnosis. On the other hand, texture_se has a little bit of a better distribution as the curve for malignant is shifted a little to the right. This was a variable we ended up including in the final model, which makes sense because it initially looked to be a better predictor than most based on Figure 5.

Another method of variable analysis that we tried to use was boxplots. Figure 6 shows the boxplot that we created for radius_mean. Even though at first glance it looks like there is really no difference between the diagnosis for radius_mean, we used a Wilcoxon test to determine whether there was a significant difference. From the box plot, we can see that the radius of breast cancer diagnosed as B (benign) is mainly distributed around 13, and the radius of breast cancer diagnosed as M (malignant) is mainly distributed around 14. After Wilcoxon test, it is found that the difference between the two is significant, with a p value of 0.039.

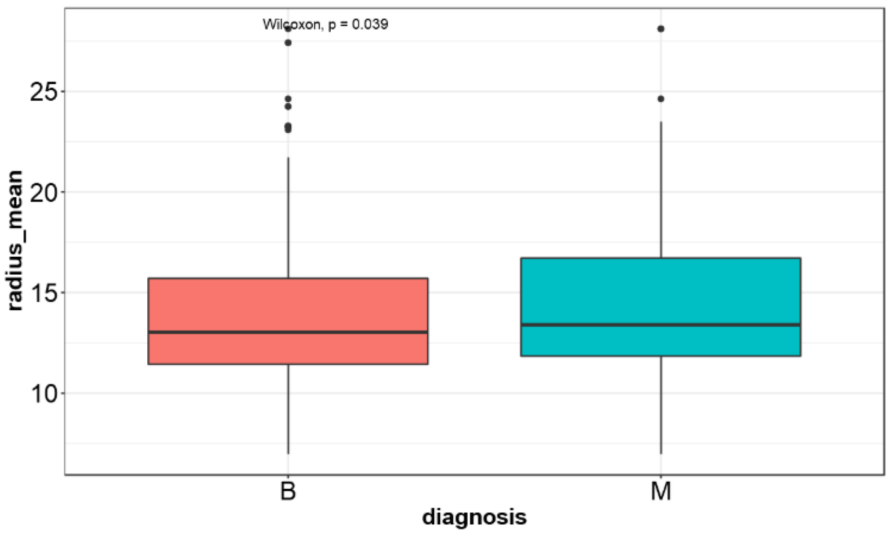


Figure 6: Box Plot of radius_mean with Wilcoxon Test

The Wilcoxon test helped to show how the variable radius_mean can be useful in predicting whether the tumor is benign or malignant. As a result, we figured that we would want to include it in our final model. Another example of us using boxplots and Wilcoxon tests to find good variables is shown in Figure 7. This analysis was of the variable texture_se and the Wilcoxon test shows that the difference in means of the variable is significant.

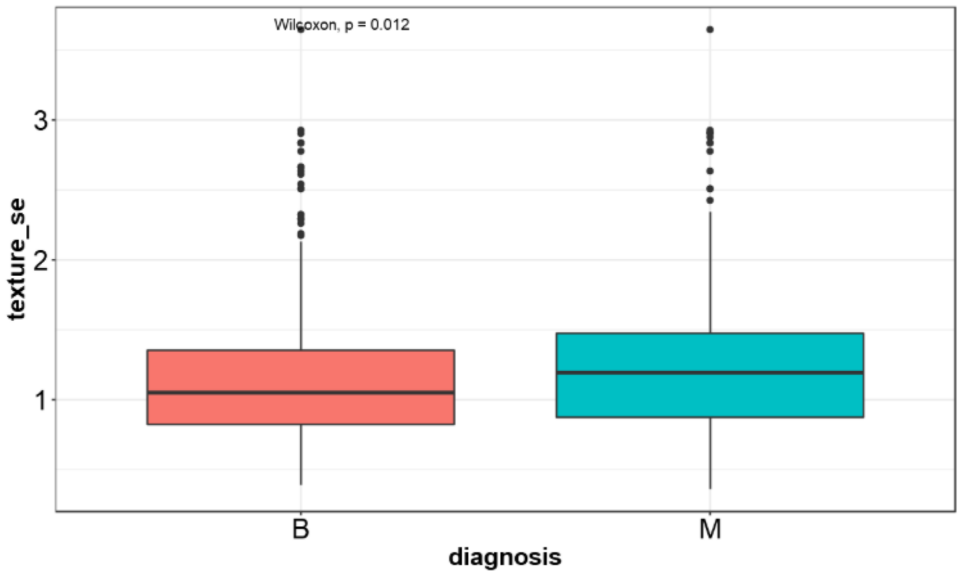


Figure 7: Box Plot of texture_se with Wilcoxon Test

3. Methods and Models

The next step in creating our model was to figure out the type of model we wanted to use. We tested methods like a logistic regression model, random forest model as well as QDA and LDA models on the training data.

The steps for testing each of the models was the following. First, we create and test the model's accuracy on the training data. Next, if we felt the model was good enough to test on the testing data, we would have our model make predictions on the testing data set and submit the results to the Kaggle website for scoring.

Most of the models that we tested on the data had training accuracy rates of around 0.60-0.65. But, the model that performed best on the training data was actually a QDA model with all of the variables included. It had a training accuracy rate of 0.8. We thought that this model would be great for the testing data set as well, so we tested it on Kaggle and got a score of 0.54. This score is clearly significantly worse than the training data set. This can be pinpointed to overfitting the training data. Therefore, we decided that a QDA model would not work well for us.

We then tried a random forest model. The random forest model was established with diagnosis as the dependent variable and the remaining variables as independent variables. It can be seen from Figure 8 below that the error rate of the model is the smallest when trees are selected as 100, and the random forest model is re-established through the selection of trees.

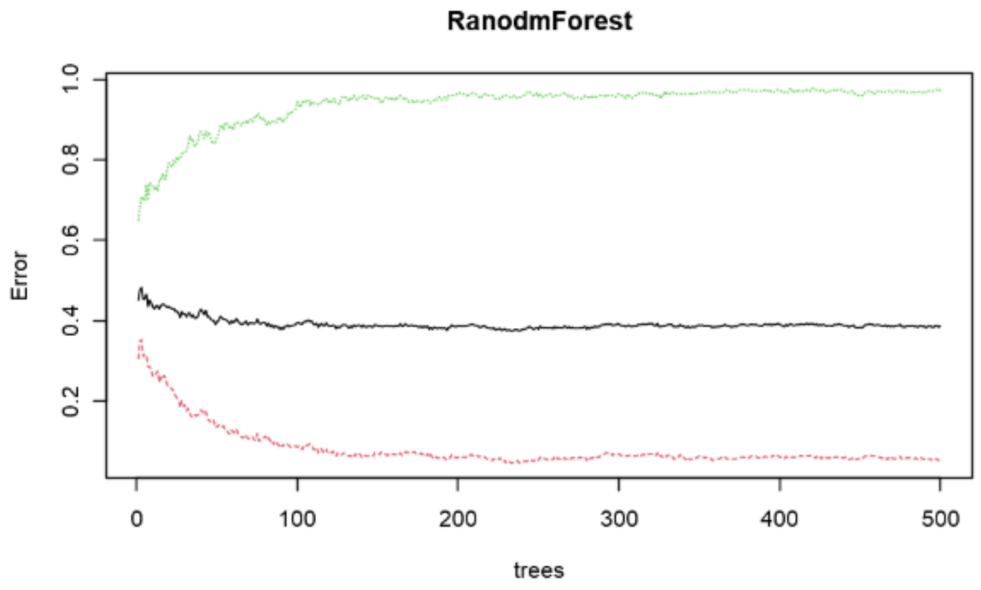


Figure 8: Error Rates versus number of Trees for Random Forest Model

From the results of the random forest model in Figure 9 below, we can see the importance of each feature variable in the model to predicting diagnosis. Compared with logistic regression, although some variables have insignificant correlations with diagnosis, they contribute a lot to the model.

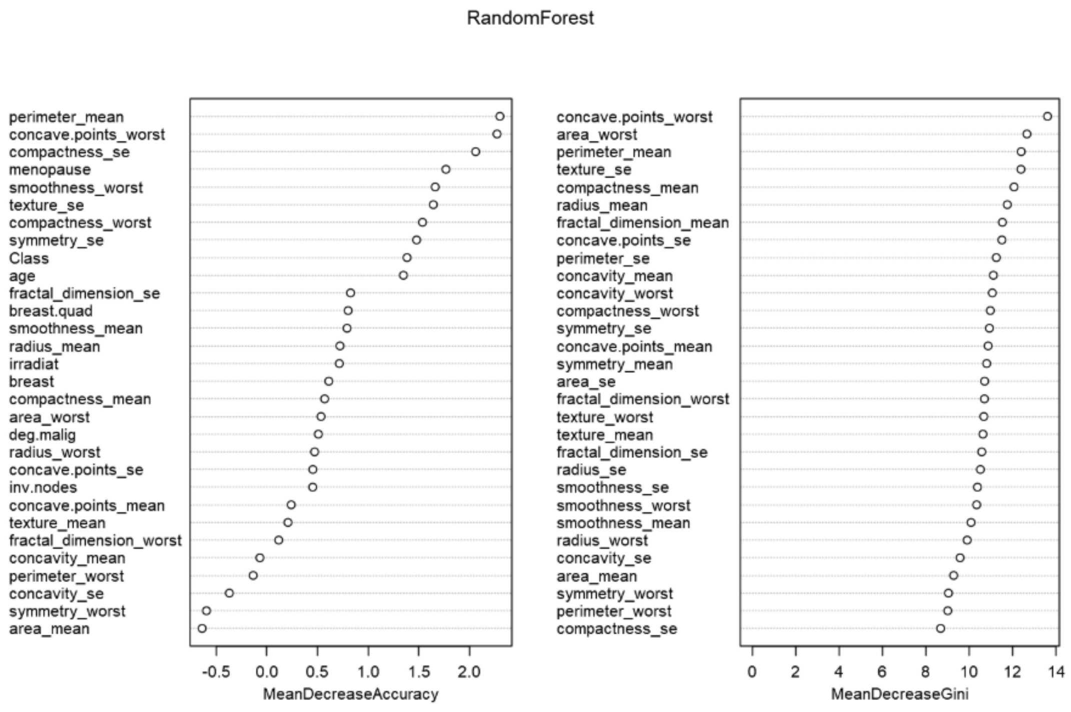


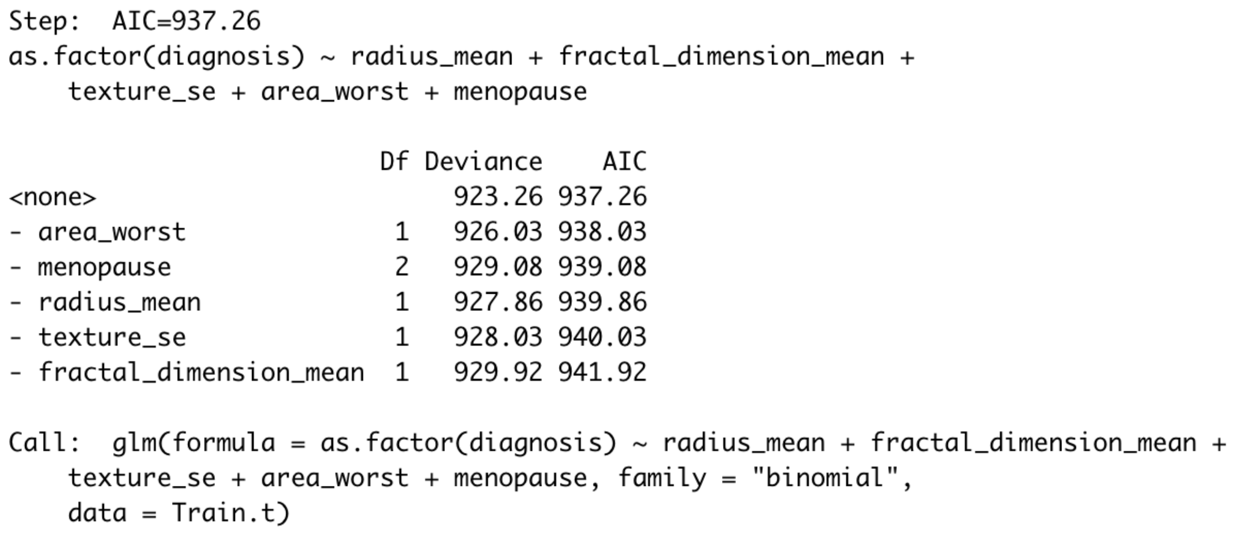
Figure 9: Random Forest Model on Training Data

We decided then that it would be a good idea to test the model with the test data and submit to Kaggle. The prediction result of the test set submitted to the Kaggle website scored 0.64485. But we ultimately decided to go a different route because we wanted our model to be simple. Oftentimes with the random forest model, the limitation is that the model is overfitted and too complicated. We did not want a complex model as our final model because we wanted to be able to easily pinpoint where the variation in diagnosis came from.

4. Results and Final Model

In the end, we decided to use a GLM, or logistic regression model as our final model to submit on Kaggle. This is because the logistic model is much simpler compared to the other types of models that we were testing. Not only that, but the logistic model does a great job of handling variables that have non-linear relationships with each other. Most importantly, though, it consistently performed well on the Kaggle competition site as well as with the training data.

The next important step in going forward in creating our final model was choosing which variables to include in the final model. We decided that the best way to do so was by creating a logistic model that would predict diagnosis and then use a backwards step function to determine which variables were most important in predicting diagnosis. The step function used AIC as the determinant in deciding when to cut off variables. Below, in Figure 10, we can see the results of the step function.



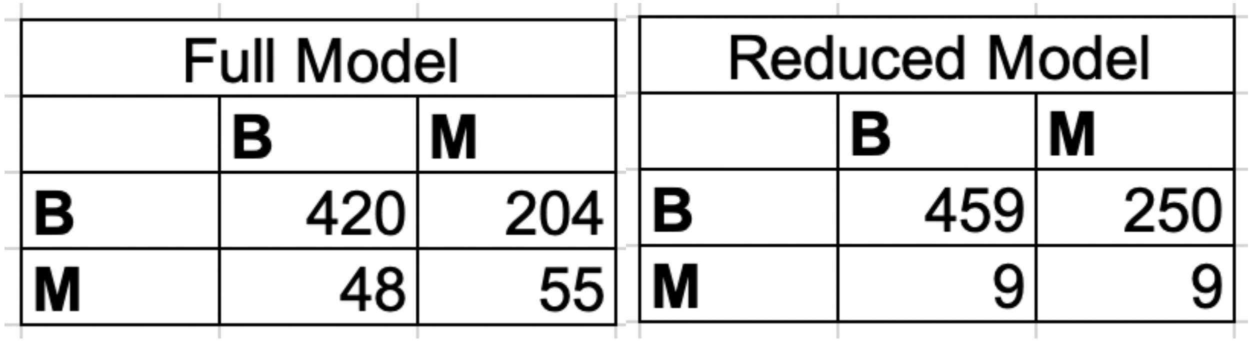
```
Step:  AIC=937.26
as.factor(diagnosis) ~ radius_mean + fractal_dimension_mean +
    texture_se + area_worst + menopause

                         Df Deviance    AIC
<none>                        923.26 937.26
- area_worst              1   926.03 938.03
- menopause               2   929.08 939.08
- radius_mean             1   927.86 939.86
- texture_se              1   928.03 940.03
- fractal_dimension_mean  1   929.92 941.92

Call:  glm(formula = as.factor(diagnosis) ~ radius_mean + fractal_dimension_mean +
    texture_se + area_worst + menopause, family = "binomial",
    data = Train.t)
```

Figure 10: R Output of Step Function for Logistic Model

As shown in Figure 10, the best variables for the model were area_worst, menopause, radius_mean, texture_se, and fractal_dimension_mean. So, we decided to create a simplified model with these five variables and see how it compared to the model with all the variables. After doing so, we created accuracy tables comparing the full and reduced model for the training data set. They are shown below in Figure 11.



Full Model		
	B	M
B	420	204
M	48	55

Reduced Model		
	B	M
B	459	250
M	9	9

Figure 11: Accuracy Tables of Full vs Reduced Models

The accuracy rate of the full model was 65.337% while the accuracy rate of the reduced model was 64.374%. Although the accuracy rate of the reduced model is less than the full model, it was only about 1% less accurate on the training data. Even more, the reduced model is much simpler than the full model since it was made up of only five variables rather than 39, respectively. The reduced model allowed us to understand what correlated with variation in diagnosis much more easily than the full model. As a result, we decided to overlook the fact that the accuracy rate of the reduced model was a little less than the accuracy rate of the full model.

So, the final model was the following:

glm(as.factor(diagnosis) ~ radius_mean + fractal_dimension_mean + texture_se + area_worst + menopause

5. Discussion/Limitations

This model gave us a Kaggle score of 0.65420 which was better than the training accuracy rate for both the full and reduced models. Our Kaggle score placed us in 20th on the professor's final table of scores. The best score was 0.67601 on the Kaggle website which was not too much larger than our score. In fact, our score was 96.8% of the top score which means we were very close in accuracy to that top model. But, we decided to go with a model that was very simple and yet our model was very close in percentage to the top of the leaderboard on Kaggle.

As a group, the thing that hindered us most was time. Since we were learning all these methods for regression and categorization as we continued the Kaggle project, we were constantly updating our model and did not have time to slow down and process everything. It would have been great if we also had time to test variables that we created out of the other variables to get a better fit on the model without overfitting. A variable that we did not use that seemed important in conceptually but was not according to the step function was Age. It is obvious that as women get older they are more likely to develop breast cancer and so it would make sense that age would be in our model. But, the way age was set as a variable in this dataset was as a categorical variable, with a range of ages for each category. If we could have figured out a way to transform the variable into a numeric, maybe take the average for the range and use that, it would have helped to create a better model.

6. Conclusion and Recommendation

After going through all the data, we determined that the best form of regression/categorization on the data was a logistic model. In variable selection, we ultimately used a backwards step function with AIC as the determinant for how to get rid of unnecessary variables. There is not one way to determine what is the best model, it is a long process with many routes and ultimately comes down to what the statisticians value. Do they want a model with a specific variable? Do they want to have a specific type of model because the variables seem like they follow a normal distribution? We decided that our focus was to create a simple, yet accurate model, and we feel as though we have achieved that goal. Our logistic regression model did not have any transformations on the variables that we used, and we only used five variables. We did this all while producing a testing Kaggle score that was 96.8% of the top score on Kaggle. We believe that this project was a great way to showcase many of the regression techniques that we learned in Stats 101A, B and C and will be beneficial for our data analysis in real-world situations going forward.

References

"Breast Cancer Data Analysis (Classification)." *Kaggle*, www.kaggle.com/c/breast-cancer-data.

RStudio Team (2020). RStudio: Integrated Development for R. RStudio, PBC, Boston, MA URL http://www.rstudio.com/.