Gliomas

LW

2023-09-06

Contents

INTRODUCTION

This data set was obtained from the UC Irvine Machine Learning Repository here[1]. There are 839 rows, each representing a patient. The 23 Attributes are tumor Grade, source Project, Case_ID, Gender, Age at diagnosis, Primary diagnosis, Race, and presence/absence of mutations on 15 Genes. The goal of this project was to build a classification tree model to predict tumor Grade based on Gene mutations. The two Grades are **High-Grade GBM**s and **Lower-Grade LGG**s. Glioblastoma multiforme (GBM) is a fast-growing brain/spinal cord tumor and is the most common type of primary malignant brain tumor in adults[2]. This data set has been used in numerous studies as prognostic and diagnostic molecular tests are often cost-prohibitive for patients. Data was first imported, cleaned, and split in to test and train sets. Four classification models were built and performance of each on the test set was evaluated.

```
genes<- read.csv("TCGA_GBM_LGG_Mutations_all.csv")
head(genes)
```

```
##   Grade  Project      Case_ID Gender Age_at_diagnosis       Primary_Diagnosis
## 1   LGG TCGA-LGG TCGA-DU-8164   Male             51.3  Oligodendroglioma, NOS
## 2   LGG TCGA-LGG TCGA-QH-A6CY   Male             38.7            Mixed glioma
## 3   LGG TCGA-LGG TCGA-HW-A5KM   Male             35.2        Astrocytoma, NOS
## 4   LGG TCGA-LGG TCGA-E1-A7YE Female             32.8 Astrocytoma, anaplastic
## 5   LGG TCGA-LGG TCGA-S9-A6WG   Male             31.5 Astrocytoma, anaplastic
## 6   LGG TCGA-LGG TCGA-DB-A4X9 Female             33.2            Mixed glioma
##    Race IDH1 TP53 ATRX PTEN EGFR CIC MUC16 PIK3CA NF1 PIK3R1 FUBP1 RB1 NOTCH1
## 1 white    1    0    0    0    0   0     0      1   0      0     1   0      0
## 2 white    1    0    0    0    0   1     0      0   0      0     0   0      0
## 3 white    1    1    1    0    0   0     0      0   0      0     0   0      0
## 4 white    1    1    1    0    0   0     1      0   0      1     0   0      0
## 5 white    1    1    1    0    0   0     0      0   0      0     0   0      0
## 6 white    1    0    1    0    0   0     0      0   0      0     0   0      0
##   BCOR CSMD3 SMARCA4 GRIN2A IDH2 FAT4 PDGFRA
## 1    0     0       0      0    0    0      0
## 2    0     0       0      0    0    0      0
## 3    0     0       0      0    0    0      0
## 4    0     0       0      0    0    1      0
## 5    0     0       0      0    0    0      0
## 6    0     0       0      0    0    0      0
```

METHODS

Data Cleaning and Feature Engineering

First, all character and integer attributes were converted to factors. There were also four rows for patients that had no Gender, Diagnosis, or Race which were removed.

```r
genes<- genes %>% mutate(across(.cols= where(is.character), .fns= as.factor))
genes<- genes %>% mutate(across(.cols= where(is.integer), .fns= as.factor))
str(genes)
```

```r
# remove four patients with empty Gender, Diagnosis, Race
genes<- genes %>% filter(Gender != "--")
```

```r
colSums(is.na(genes))
```

```r
str(genes)
```

```
## 'data.frame':    858 obs. of  27 variables:
##  $ Grade           : Factor w/ 2 levels "GBM","LGG": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Project         : Factor w/ 2 levels "TCGA-GBM","TCGA-LGG": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Case_ID         : Factor w/ 862 levels "TCGA-02-0003",..: 481 733 693 519 776 389 708 567 660 49
##  $ Gender          : Factor w/ 3 levels "--","Female",..: 3 3 3 2 3 2 2 2 2 3 ...
##  $ Age_at_diagnosis : num  51.3 38.7 35.2 32.8 31.5 33.2 35.2 44.7 34 87 ...
##  $ Primary_Diagnosis: Factor w/ 7 levels "--","Astrocytoma, anaplastic",..: 7 5 3 2 2 5 6 5 7 7 ...
##  $ Race            : Factor w/ 6 levels "--","american indian or alaska native",..: 6 6 6 6 6 6 6 6
##  $ IDH1            : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1 ...
##  $ TP53            : Factor w/ 2 levels "0","1": 1 1 2 2 2 1 2 2 2 1 ...
##  $ ATRX            : Factor w/ 2 levels "0","1": 1 1 2 2 2 1 2 2 2 1 ...
##  $ PTEN              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EGFR              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CIC               : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1 ...
##  $ MUC16             : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1 ...
##  $ PIK3CA            : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...
##  $ NF1               : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ PIK3R1            : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1 ...
##  $ FUBP1             : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...
##  $ RB1               : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ NOTCH1            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ BCOR              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CSMD3             : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SMARCA4           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ GRIN2A            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ IDH2              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ FAT4              : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1 ...
##  $ PDGFRA            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Then, the data was split to the default 75% train and 25% test. The strata argument ensures that the Grade levels are distributed evenly across data splits.

```r
# split data in to train and test sets
set.seed(1998)
genes_split<- initial_split(genes, strata=Grade) #strata ensures even distribution of Grade
genes_train<- training(genes_split)
genes_test<- testing(genes_split)
nrow(genes_train)/nrow(genes)
```

```
## [1] 0.7494172
```

Exploratory Data Analysis

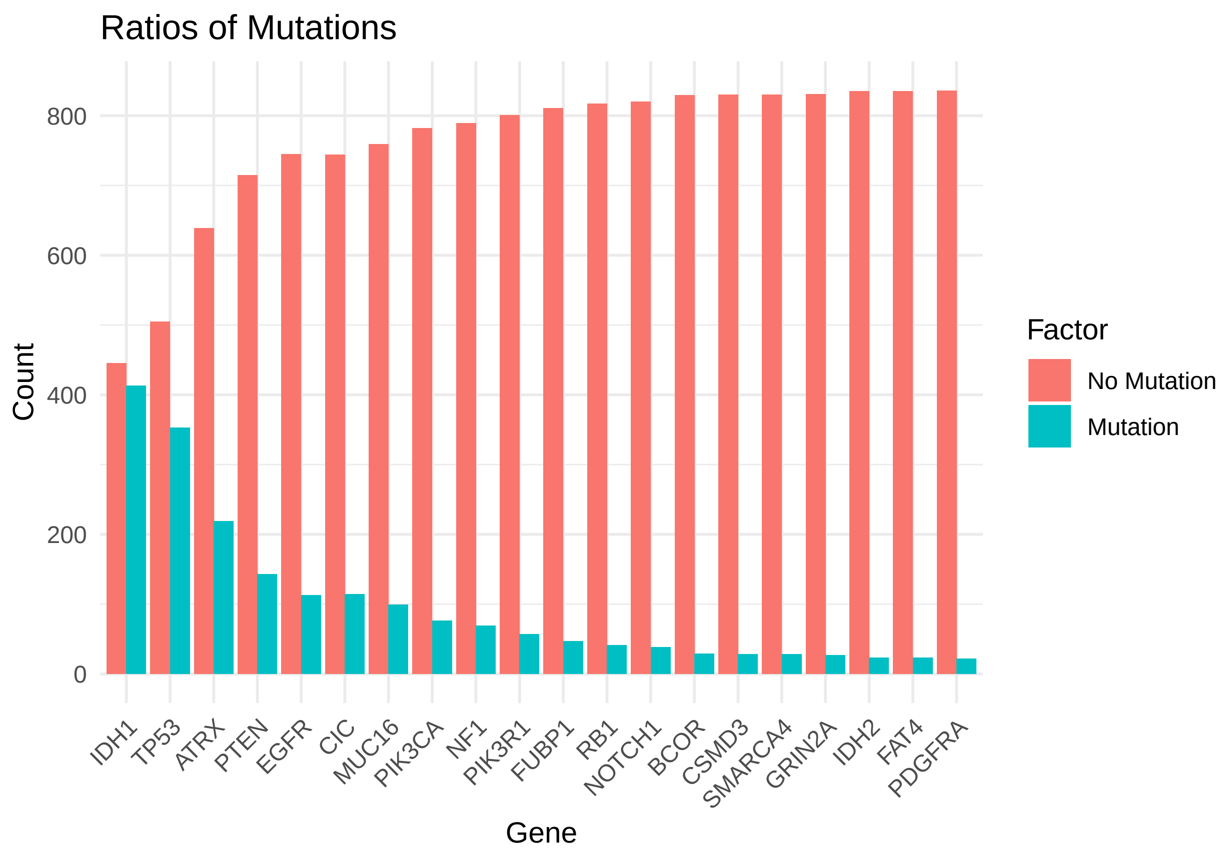
Looking at the factor counts of each factor attribute, we see that the two Grades are evenly split, with slightly less (43% of total) GBM than LGG.

```r
genes_count<- genes[,-2:-7] # exclude demographics
result<- apply(genes_count, 2, function(x) table(x)) # factor counts for each column
result
```

```
##      Grade IDH1 TP53 ATRX PTEN EGFR CIC MUC16 PIK3CA NF1 PIK3R1 FUBP1 RB1
## [1,]   360  445  505  639  715  745 744   759    782 789    801   811 817
## [2,]   498  413  353  219  143  113 114    99     76  69     57    47  41
##      NOTCH1 BCOR CSMD3 SMARCA4 GRIN2A IDH2 FAT4 PDGFRA
## [1,]    820  829   830     830    831  835  835    836
## [2,]     38   29    28      28     27   23   23     22
```

The Genes are ordered in the data set by mutation frequency, as mutation frequency decreases as the columns increase. IDH1, for example, has a mutation frequency of 414 of the 837 rows whereas PDGFRA has a mutation frequency of only 22 of the 837 rows.

```r
# Create a bar plot of the ratios
ggplot(result_long, aes(x = Gene, y = Ratio, fill = Factor)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Ratios of Mutations", x = "Gene", y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+ scale_fill_discrete(labels=c('No Mutation',
```



Variable Importance

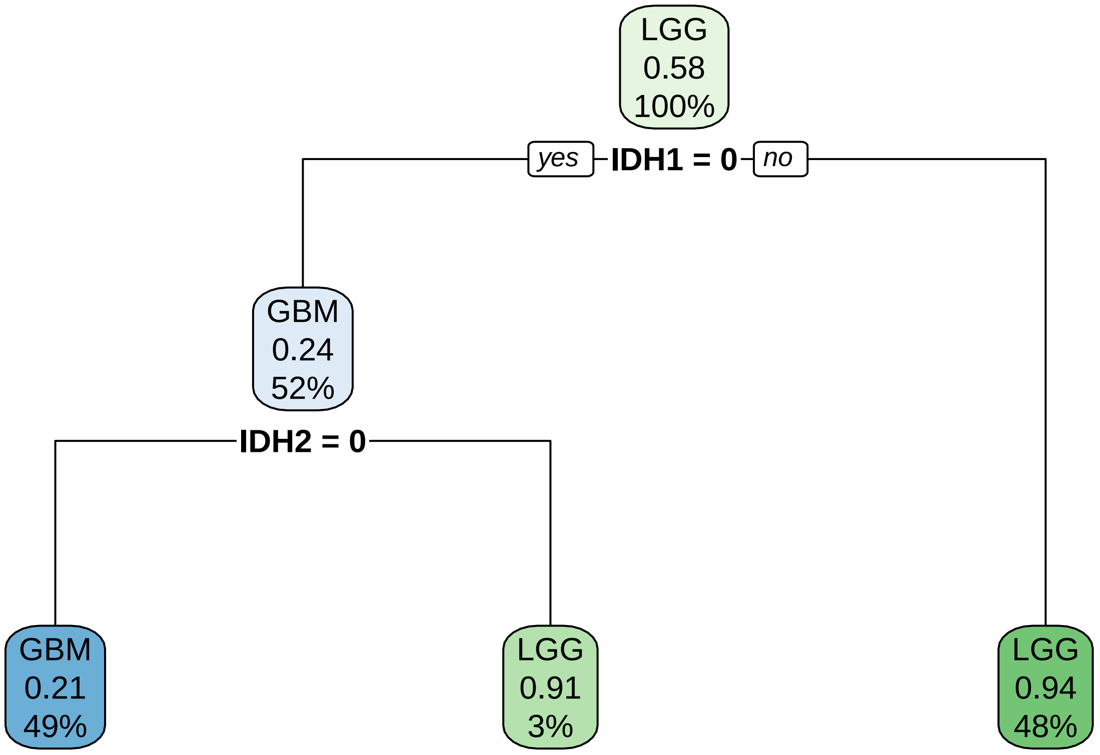
```r
# set specifications and fit to create Decision Tree model
model_base<- decision_tree() %>%
  set_mode("classification") %>%
  set_engine("rpart") %>%
  fit(Grade~.-Project-Case_ID-Primary_Diagnosis, data= genes_train)
```

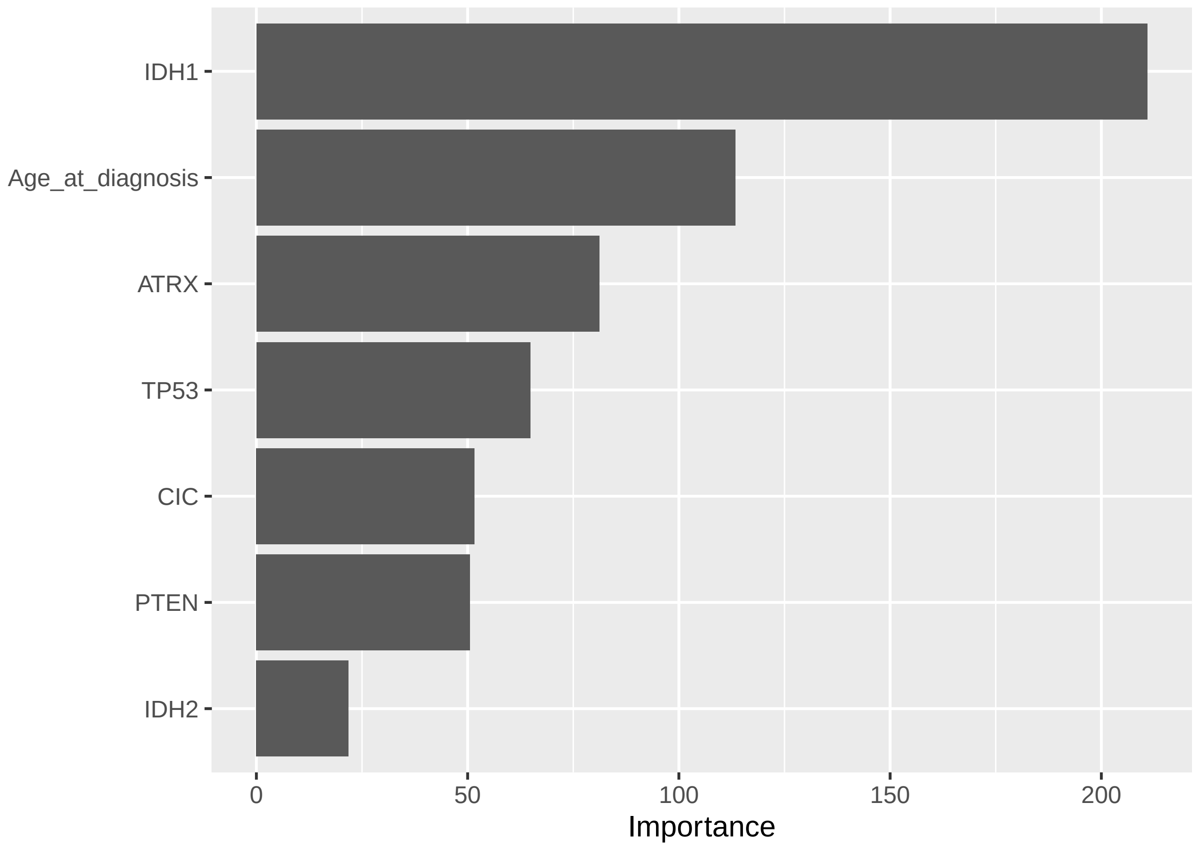
```r
# predict on test set
preds_base<- predict(model_base, genes_test, type="prob")
preds_base
```

```r
preds_base<- data.frame(Grade= genes_test$Grade, preds_base)
# label with new model column
preds_base<- preds_base %>% mutate(model= "Tree")
preds_base
```

The Decision Tree and Variable Importance Plot identify the Gene IDH1 to be the most influential factor in determining tumor Grade.

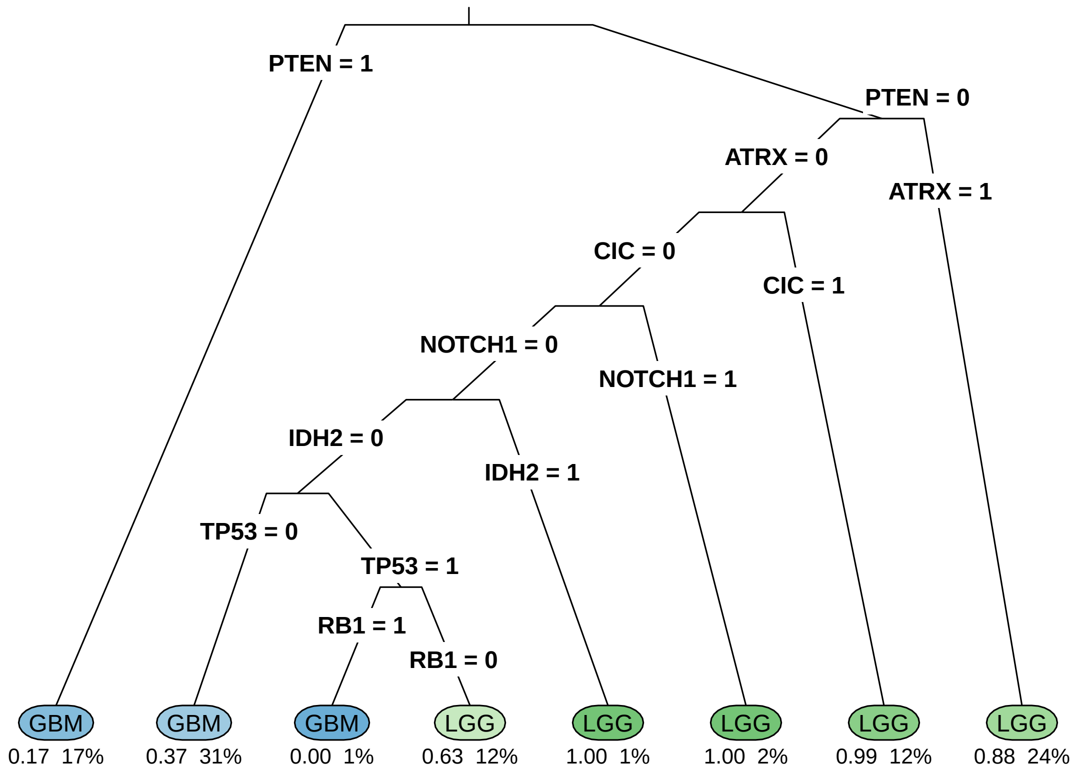
```r
rpart_model_base<- rpart(Grade~. -Project -Case_ID -Primary_Diagnosis, genes, cp=0.02)
rpart.plot(rpart_model_base)
```





It is well established that increased age is a determinant of cancer prevalence. Similarly, several types of cancers are widely attributed to missense mutations of the IDH1 Gene. These models will therefore zoom in on the role the other Genes play in the severity of brain gliomas in this data set. The appearance of the rpart tree can be customized, as detailed by Stephen Milborrow[3]. We can now see the relative importance of the remaining Genes in the probability of their mutations resulting in either a High-Grade GBM or Lower-Grade Glioma. Each leaf is labelled with a probability value on the left and a percent of the sample on the right. For example, from the bottom-right, 24% of the test set had no PTEN mutation with an ATRX mutation. Those patients have a 0.88 probability of having a Low-Grade Glioma.

```r
rpart_model<- rpart(Grade~.-Project-Case_ID-Primary_Diagnosis-IDH1-Age_at_diagnosis, genes, cp=0.02)
# customize appearance
rpart.plot(rpart_model, type=3, clip.right.labs= FALSE, branch= .3, under= TRUE)
```



Classification Tree

The first model built was a simple Classification Tree. Robust to outliers and relatively fast to train, these trees can have high variance and are prone to overfitting. The parsnip, rsample, and yardstick libraries in the tidymodels package were used for all of these models.

```
# model on just genes, removing IDH1 and Age, using the same process as above
model <- decision_tree() %>%
  set_mode("classification") %>%
  set_engine("rpart") %>%
  fit(Grade~.-Project-Case_ID-Primary_Diagnosis-IDH1-Age_at_diagnosis, data= genes_train)
preds_tprob<- predict(model, genes_test, type="prob")
preds_tree<- data.frame(Grade= genes_test$Grade, preds_tprob)
preds_tree<- preds_tree %>% mutate(model= "Tree")
```

The ROC AUC value for this model is `roc_auc, binary, 0.855688888888889`.

```
roc_auc(preds_tree, Grade, .pred_GBM)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.856
```

Other performance metrics can be calculated on class, rather than probability, predictions.

```
# change to class predictions for additional metrics
preds_tclass<- predict(model, genes_test, type="class")
preds_tclass
# combine actual test Grades with class predictions
preds_tree_class<- data.frame(Grade= genes_test$Grade, preds_tclass)
preds_tree_class<- preds_tree_class %>% mutate(model= "Tree")
preds_tree_class
```

```
accuracy(preds_tree_class, Grade, .pred_class)
precision(preds_tree_class, Grade, .pred_class)
recall(preds_tree_class, Grade, .pred_class)
f_meas(preds_tree_class, Grade, .pred_class)
```

The diagonals of a confusion matrix identify the number of accurate predictions versus false positive and false negative predictions. In this case, the model accurately predicted Grade 81+86= 167 times, while there were 39+9= 48 incorrect predictions on the test set.

```
conf_mat(preds_tree_class, Grade, .pred_class)
```

```
##           Truth
## Prediction GBM LGG
##        GBM  81  39
##        LGG   9  86
```

In an attempt to improve performance, three more model types were built, tested, and compared.

Bagging

There was evidence of high variance exhibited in the difference between the original rpart tree and the classification tree when Age and IDH1 were dropped. The next model is an ensemble model, which essentially combines the predictions of several models, in this case via **B**ootstrap **Agg**regation, or Bagging. The baguette package functions were used for this method.

```
spec_bagged<- bag_tree() %>% set_mode("classification") %>% set_engine("rpart", times= 100)
```

```
model_bagged<- fit(spec_bagged, formula= Grade~.-Project-Case_ID-Primary_Diagnosis-IDH1-Age_at_diagnosi
```

```
preds_bag<- predict(model_bagged, genes_test, type="prob")
preds_bagging<- data.frame(Grade= genes_test$Grade, preds_bag)
preds_bagging<- preds_bagging %>% mutate(model= "Bagging")
preds_bagging
```

In this case, the performance of the model on the test set was perfect.

```
# calculate AOC
roc_auc(preds_bagging, Grade, .pred_GBM)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary             1
```

This is confirmed by the confusion matrix showing no incorrect predictions.

```
conf_mat(preds_bag_class, Grade, .pred_class)
```

```
##           Truth
## Prediction GBM LGG
##        GBM  90   0
##        LGG   0 125
```

Random Forest

Additional models were built to provide options for optimal performance on a larger data set when available. Random Forest is another ensemble of trees trained on bootstrapped samples, with a bit of extra randomness added to the model. Rather than considering all variables together, only a subset of predictors is considered for each split. The majority vote is the final prediction. This model was built using the rand_forest() function in the parsnip package.

```
spec<- rand_forest() %>% set_mode("classification") %>%
  # specify algorithm that controls node split
  set_engine("ranger", importance= "impurity")
```

```
forest_model<- spec %>% fit(Grade~.-Project-Case_ID-Primary_Diagnosis-IDH1-Age_at_diagnosis, data= gene
```

```
preds_f<- forest_model %>% predict(genes_test, type="prob")
```

```
# create data frame with actual test Grade and modeled test predictions
preds_forest<- data.frame(Grade= genes_test$Grade, preds_f)
preds_forest<- preds_forest %>% mutate(model= "Random Forest")
preds_forest
```

Again, the model built accurately predicted each of the test set's Grades.

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary             1
```

```
conf_mat(preds_forest_class, Grade, .pred_class)
```

```
##           Truth
## Prediction GBM LGG
##        GBM  90   0
##        LGG   0 125
```

Boosting

Boosting is another ensemble method that incorporates the Gradient Descent technique to Adaptive Boosting. After evaluating the first tree, the loss function is used to select and optimize predictions of observations that are difficult to classify. Subsequent trees then better classify those observations which were not initially well- classified. This was done using Tidymodels' xgboost package[3].

```
boost_spec<- boost_tree() %>% set_mode("classification") %>% set_engine("xgboost")
```

```
boost_model <- fit(boost_spec, Grade~.-Project-Case_ID-Primary_Diagnosis-IDH1-Age_at_diagnosis, genes_t
```

```
set.seed(1998)
folds<- vfold_cv(genes_train, v=10)
```

The out-of-the-box performance as measured by AUC for the untuned model is 85%.

```
# out-of-the-box performance for untuned model
cv_results<- fit_resamples(boost_spec, Grade~.-Project-Case_ID-Primary_Diagnosis-IDH1-Age_at_diagnosis,
collect_metrics(cv_results, summarize= FALSE)
```

```
## # A tibble: 10 x 5
##    id     .metric .estimator .estimate .config
##    <chr>  <chr>   <chr>          <dbl> <chr>
##  1 Fold01 roc_auc binary         0.790 Preprocessor1_Model1
##  2 Fold02 roc_auc binary         0.906 Preprocessor1_Model1
##  3 Fold03 roc_auc binary         0.9   Preprocessor1_Model1
##  4 Fold04 roc_auc binary         0.793 Preprocessor1_Model1
##  5 Fold05 roc_auc binary         0.851 Preprocessor1_Model1
##  6 Fold06 roc_auc binary         0.773 Preprocessor1_Model1
##  7 Fold07 roc_auc binary         0.9   Preprocessor1_Model1
##  8 Fold08 roc_auc binary         0.875 Preprocessor1_Model1
##  9 Fold09 roc_auc binary         0.876 Preprocessor1_Model1
## 10 Fold10 roc_auc binary         0.849 Preprocessor1_Model1
```

```
collect_metrics(cv_results)
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 roc_auc binary     0.851    10  0.0157 Preprocessor1_Model1
```

Through trial and error, 500 trees and 4 levels were settled upon as providing the best performance.

Trees	Levels	AUC
200	5	0.843
300	4	0.865
100	3	0.866
500	3	0.870
500	4	0.882

```
# create tuning spec with placeholders
boost_spec<- boost_tree(trees= 500, learn_rate= tune(), tree_depth= tune(), sample_size= tune()) %>% se
```

```
tunegrid_boost<- grid_regular(extract_parameter_set_dials(boost_spec), levels= 4)
```

The tune_grid function computes performance metrics for the tuning parameters.

```
tune_results<- tune_grid(boost_spec, Grade~.-Project-Case_ID-Primary_Diagnosis-IDH1-Age_at_diagnosis, f
```

The select_best() function specifies optimal values for the placeholder parameters that were set to tune in boost_spec.

```
best_params<- select_best(tune_results)
best_params
```

```
## # A tibble: 1 x 4
##   tree_depth learn_rate sample_size .config
##        <int>      <dbl>       <dbl> <chr>
## 1          2      0.190       0.677 Preprocessor1_Model10
```

```
final_spec<- finalize_model(boost_spec, best_params)
```

```
final_model<- final_spec %>% fit(Grade~.-Project-Case_ID-Primary_Diagnosis-IDH1-Age_at_diagnosis, data=
```

```
preds_b<- predict(final_model, genes_test, type="prob")
```

```
# create data frame with predictions and column for model type
preds_boosting<- data.frame(Grade= genes_test$Grade, preds_b)
preds_boosting<- preds_boosting %>% mutate(model= "Boosting")
preds_boosting
```

Tuning the tree depth, learn rate, and sample size improved the AUC by 0.028 to 0.879.

```
# AUC prob metric
roc_auc(preds_boosting, Grade, .pred_GBM)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.863
```

```
conf_mat(preds_boost_class, Grade, .pred_class)
```

```
##           Truth
## Prediction GBM LGG
##        GBM  68  32
##        LGG  22  93
```

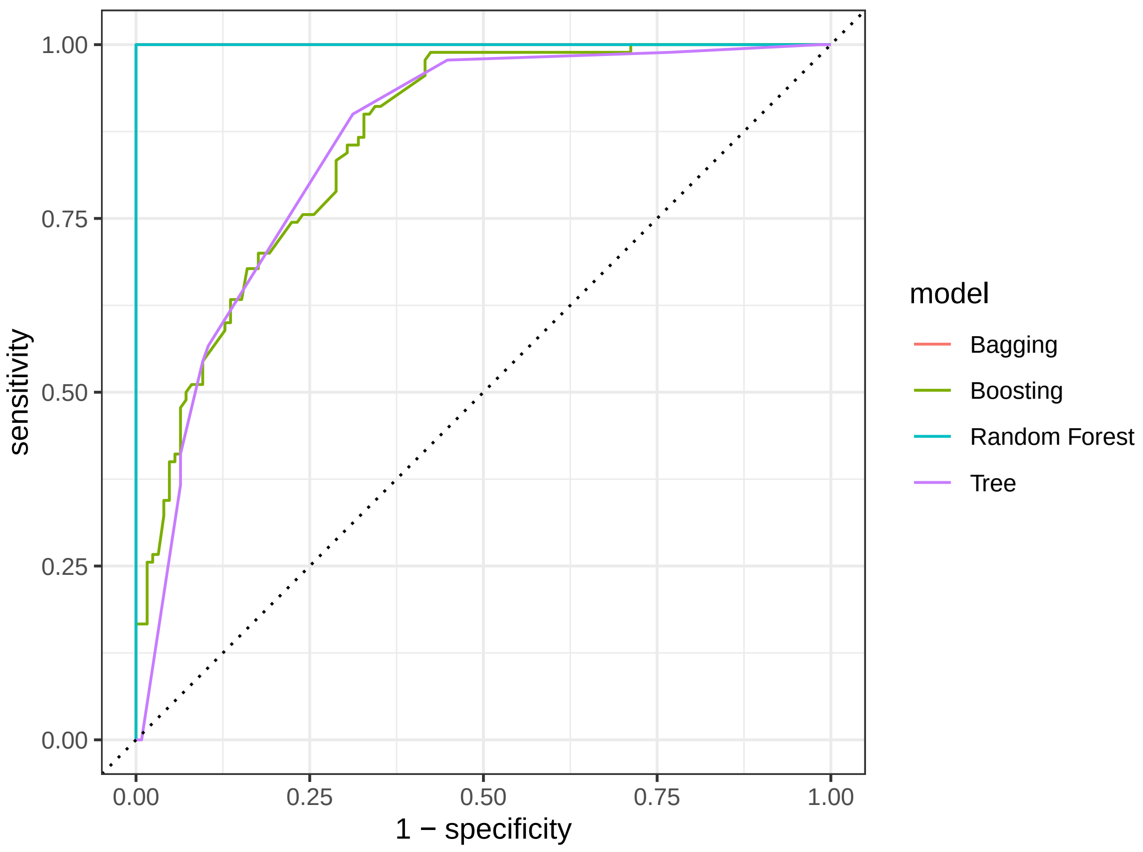
RESULTS

The metrics indicate that, for this data set, the Bagging and Random Forest models performed best. With a larger data set, that may not be the case. Boosting with tuning performed only slightly better than the simple Classification Tree.

```
## # A tibble: 4 x 4
##   .metric .estimator .estimate model
##   <chr>   <chr>          <dbl> <chr>
## 1 roc_auc binary         0.856 Tree
## 2 roc_auc binary         1     Bagging
## 3 roc_auc binary         1     Forest
## 4 roc_auc binary         0.863 Boosting
```

All model predictions were combined to construct ROC Curves for each model. This is a visualization of the performance (sensitivity vs specificity) summarized by the AUC values of each model.

```
cutoffs<- preds_combined %>% group_by(model) %>% roc_curve(Grade, .pred_GBM)
autoplot(cutoffs)
```



CONCLUSION

The Bagging and Random Forest models performed without fault on this data set, however that may not be the case with a larger data set. Of note is that the Bagging and Random Forest models outperformed the Boosting model. This may be attributable to outliers or the high dimensionality of the data. Boosting works particularly well with unbalanced data sets, but the Grades were fairly well balanced in this data. It is also possible that the hyperparameter tuning was not fully optimized. Perhaps a Boosted model with other parameters tuned may perform better. It is also important to consider the nature of these Gene mutations, not just their presence. Future investigations could investigate molecular profiles available on these mutations to further elucidate the predictive capabilities of these mutations for prognostic and diagnostic purposes. In conjunction with stacking ensemble methods, this could lead to very strong real-world models and increased accessibility.

REFERENCES

[1] Tasci,Erdal, Camphausen,Kevin, Krauze,Andra Valentina, and Zhuge,Ying. (2022). Glioma Grading Clinical and Mutation Features. UCI Machine Learning Repository. https://doi.org/10.24432/C5R62J.

[2] https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/idh1

[3] http://milbo.org/rpart-plot/prp.pdf

[4] https://tune.tidymodels.org/