Predictive Modeling for Glioma Grading: Comparative Evaluation of Classification Models

Tarun Chikatipalli

Packages

```r
library(rpart)
library(rpart.plot)
library(readr)
library(archive)
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
library(class)
library(e1071)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
library(rpart)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(rpart)
library(tinytex)
```

Introduction

In this analysis, I will explore the Glioma Grading dataset obtained from the UCI Machine Learning Repository. This dataset contains clinical and mutation features for glioma grading, with the goal of pre- dicting the glioma grade. We will perform various data preprocessing, visualization, and modeling tasks to gain insights and build predictive models.

The goal of this project is to predict the glioma grade (LGG or GBM) of a patient based on their clinical and molecular/mutation features. It is a classification task since we are trying to categorize patients into two classes: LGG and GBM. To achieve this goal, I plan to use several classification algorithms, including lazy learning (e.g., k-Nearest Neighbors), Naïve Bayes, logistic regression, and decision trees. Each of these algorithms has its strengths and weaknesses, and by trying multiple methods, I aim to find the one that performs best on this specific dataset.

Before applying the algorithms, I will perform feature engineering and data shaping. As the dataset contains both clinical and molecular features, it is essential to identify the most informative subset of mutation genes and clinical features. Feature engineering will help me select the most relevant attributes, which can potentially lead to improved performance and reduced costs in the glioma grading process.

For evaluating the fit of the algorithms, I will split the data into training and validation sets. This step will allow me to train the models on one subset and test their performance on unseen data. I will use evaluation metrics such as accuracy, precision, recall, F1 score, and ROC-AUC to assess the models' performance. Since the dataset is imbalanced (as gliomas of different grades occur at different frequencies), I will also consider using techniques like oversampling or undersampling to handle this imbalance.

While similar analyses may have been done before, I aim to differentiate my approach by exploring various combinations of clinical and molecular features to determine the optimal subset for grading gliomas. Additionally, I will leverage the specific algorithms mentioned in the learning outcomes of the course, which will allow me to apply a diverse set of methods and compare their performance. The primary objective is to discover an accurate and cost-effective model that can aid in the glioma grading process and potentially assist medical practitioners in making informed decisions.

Data Preparation

We start by loading the necessary libraries and downloading the dataset from the provided URL using the archive and readr packages. I load two CSV files from the archive representing different aspects of the data.

I have loaded two datasets one is the original dataset which contains all the data without being processed. The other dataset is preprocessed and organized CSV dataset file consists of twenty-four fields per record. Each field is separated by a comma and each record is separated by a newline. Gender, Age_at_diagnosis, and, Race features are clinical factors, the remaining 20 molecular features consist of IDH1, TP53, ATRX, PTEN, EGFR, CIC, MUC16, PIK3CA, NF1, PIK3R1, FUBP1, RB1, NOTCH1, BCOR, CSMD3, SMARCA4, GRIN2A, IDH2, FAT4, PDGFRA. These molecular features can be mutated or not_mutated (wildtype) depending on the TCGA Case_ID.

```
# Data Loading
gli_data <- read.csv("/Users/tarun/Desktop/Glioma_classification/glioma+grading+clinical+and+mutation+fe

#Preprocessed data where all the factors are converted to numeric values and the data set has been clea
gli_data_num <- read.csv("/Users/tarun/Desktop/Glioma_classification/glioma+grading+clinical+and+mutatio
```

Data Exploration

We begin by exploring the dimensions, structure, and summary statistics of the datasets.

```
# Dataset Dimensions
dim(gli_data)
```

```
## [1] 862  27
```

```
# Dataset Structure
str(gli_data)
```

```
## 'data.frame':    862 obs. of  27 variables:
##  $ Grade            : chr  "LGG" "LGG" "LGG" "LGG" ...
##  $ Project          : chr  "TCGA-LGG" "TCGA-LGG" "TCGA-LGG" "TCGA-LGG" ...
##  $ Case_ID          : chr  "TCGA-DU-8164" "TCGA-QH-A6CY" "TCGA-HW-A5KM" "TCGA-E1-A7YE" ...
##  $ Gender           : chr  "Male" "Male" "Male" "Female" ...
##  $ Age_at_diagnosis : chr  "51 years 108 days" "38 years 261 days" "35 years 62 days" "32 years 283 
##  $ Primary_Diagnosis: chr  "Oligodendroglioma, NOS" "Mixed glioma" "Astrocytoma, NOS" "Astrocytoma, 
##  $ Race             : chr  "white" "white" "white" "white" ...
##  $ IDH1             : chr  "MUTATED" "MUTATED" "MUTATED" "MUTATED" ...
##  $ TP53             : chr  "NOT_MUTATED" "NOT_MUTATED" "MUTATED" "MUTATED" ...
##  $ ATRX             : chr  "NOT_MUTATED" "NOT_MUTATED" "MUTATED" "MUTATED" ...
##  $ PTEN             : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ EGFR             : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ CIC              : chr  "NOT_MUTATED" "MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ MUC16            : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "MUTATED" ...
##  $ PIK3CA           : chr  "MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ NF1              : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ PIK3R1           : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "MUTATED" ...
##  $ FUBP1            : chr  "MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ RB1              : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ NOTCH1          : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ BCOR            : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ CSMD3           : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ SMARCA4         : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ GRIN2A          : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ IDH2            : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
##  $ FAT4            : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "MUTATED" ...
##  $ PDGFRA          : chr  "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" "NOT_MUTATED" ...
```

```
# Dataset Summary Statistics
summary(gli_data)
```

```
##     Grade             Project            Case_ID             Gender
##  Length:862         Length:862         Length:862         Length:862
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  Age_at_diagnosis   Primary_Diagnosis      Race               IDH1
##  Length:862         Length:862         Length:862         Length:862
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##      TP53               ATRX               PTEN               EGFR
##  Length:862         Length:862         Length:862         Length:862
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##      CIC               MUC16              PIK3CA              NF1
##  Length:862         Length:862         Length:862         Length:862
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##     PIK3R1             FUBP1               RB1              NOTCH1
##  Length:862         Length:862         Length:862         Length:862
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##      BCOR              CSMD3             SMARCA4             GRIN2A
##  Length:862         Length:862         Length:862         Length:862
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##      IDH2               FAT4              PDGFRA
##  Length:862         Length:862         Length:862
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
# Numeric Dataset Dimensions
dim(gli_data_num)
```

```
## [1] 839  24
```

```
# Numeric Dataset Structure
str(gli_data_num)
```

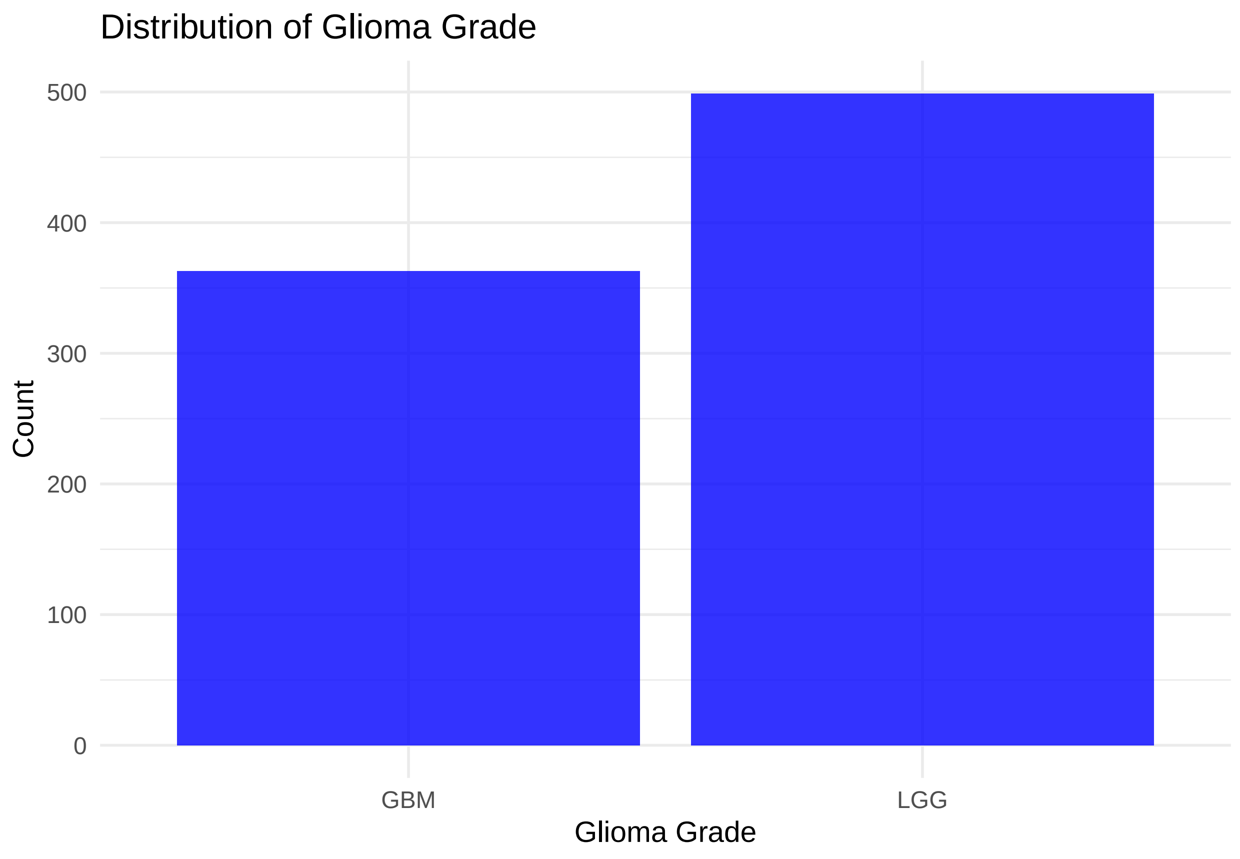
```
## 'data.frame':    839 obs. of  24 variables:
##  $ Grade           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Gender          : int  0 0 0 1 0 1 1 1 1 0 ...
##  $ Age_at_diagnosis: num  51.3 38.7 35.2 32.8 31.5 ...
##  $ Race            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ IDH1            : int  1 1 1 1 1 1 1 1 1 0 ...
##  $ TP53            : int  0 0 1 1 1 0 1 1 1 0 ...
##  $ ATRX            : int  0 0 1 1 1 1 0 1 1 0 ...
##  $ PTEN            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ EGFR            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ CIC             : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ MUC16           : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ PIK3CA          : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ NF1             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PIK3R1          : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ FUBP1           : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ RB1             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ NOTCH1          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BCOR            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ CSMD3           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SMARCA4         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GRIN2A          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ IDH2            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FAT4            : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ PDGFRA          : int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
# Numeric Dataset Summary Statistics
summary(gli_data_num)
```

```
##      Grade            Gender       Age_at_diagnosis      Race       
##  Min.   :0.0000   Min.   :0.0000   Min.   :14.42    Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:38.05    1st Qu.:0.0000  
##  Median :0.0000   Median :0.0000   Median :51.55    Median :0.0000  
##  Mean   :0.4195   Mean   :0.4184   Mean   :50.94    Mean   :0.1073  
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:62.80    3rd Qu.:0.0000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :89.29    Max.   :3.0000  
##       IDH1             TP53             ATRX             PTEN       
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000  
##  Mean   :0.4815   Mean   :0.4148   Mean   :0.2586   Mean   :0.1681  
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
##       EGFR             CIC              MUC16            PIK3CA       
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000  
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.00000  
##  Mean   :0.1335   Mean   :0.1323   Mean   :0.1168   Mean   :0.08701  
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000  
##       NF1              PIK3R1             FUBP1               RB1         
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000  
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000  
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000  
##  Mean   :0.07986   Mean   :0.06436   Mean   :0.05364   Mean   :0.04768  
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000  
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000  
##      NOTCH1              BCOR              CSMD3             SMARCA4
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.04529   Mean   :0.03456   Mean   :0.03218   Mean   :0.03218
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##      GRIN2A              IDH2               FAT4              PDGFRA
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.03218   Mean   :0.02741   Mean   :0.02741   Mean   :0.02622
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
```

Distribution of Glioma grade

```r
library(ggplot2)
# Plotting Glioma Grade Distribution
ggplot(gli_data, aes(x = Grade)) + geom_bar(fill = "blue", alpha = 0.8) + labs(title = "Distribution of
       x = "Glioma Grade",
y = "Count") + theme_minimal()
```



Handling Missing Values

We check for missing values in the dataset and handle them by removing rows with missing values.

```
# Checking for Missing Values
any_missing <- any(is.na(gli_data))
 if (any_missing) {
cat("There are missing values in the dataset.\n")
   } else {
cat("No missing values found in the dataset.\n") }
```

```
## No missing values found in the dataset.
```

```
# Identifying and Removing Outliers
missing_by_column <- colSums(is.na(gli_data))
print(missing_by_column)
```

```
##             Grade           Project           Case_ID            Gender
##                 0                 0                 0                 0
##  Age_at_diagnosis Primary_Diagnosis              Race              IDH1
##                 0                 0                 0                 0
##              TP53              ATRX              PTEN              EGFR
##                 0                 0                 0                 0
##               CIC             MUC16            PIK3CA               NF1
##                 0                 0                 0                 0
##            PIK3R1             FUBP1               RB1            NOTCH1
##                 0                 0                 0                 0
##              BCOR             CSMD3           SMARCA4            GRIN2A
##                 0                 0                 0                 0
##              IDH2              FAT4            PDGFRA
##                 0                 0                 0
```

```
# Calculating Outlier Count by Column
gli_data <- na.omit(gli_data)
gli_data_num <- na.omit(gli_data_num)
```

Outlier Detection and Removal

We identify outliers in the numeric columns of the dataset using z-scores and remove them.

```
# Calculate z-scores
z_score <- data.frame(sapply(gli_data_num, function(x) (abs(x - mean(x)) / sd(x))))
# Identify outliers using z-scores
outliers <- z_score[rowSums(z_score > 3), ]
removed_outliers <- z_score[!rowSums(z_score > 3), ]
# Dimensions of data before and after outlier removal
dim(gli_data_num)
```

```
## [1] 839  24
```

```r
dim(z_score)
```

```
## [1] 839  24
```

```r
dim(removed_outliers)
```

```
## [1] 467  24
```

Outlier Count by Column

We calculate and print the count of outliers for each numeric column.

```r
# Initialize a vector to store the count of outliers for each column
outliers_count <- numeric(length(colnames(gli_data_num)))
# Loop through the column names and find outliers
for (col in colnames(gli_data_num)) {
z_col <- z_score[, col]
outliersA <- gli_data_num[z_col > 3, col]
outliers_count[col] <- length(outliersA)
print(head(outliersA))
}
```

```
## integer(0)
## integer(0)
## numeric(0)
## [1] 2 2 2 2 2 2
## integer(0)
## integer(0)
## integer(0)
## integer(0)
## integer(0)
## integer(0)
## integer(0)
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
## [1] 1 1 1 1 1 1
```

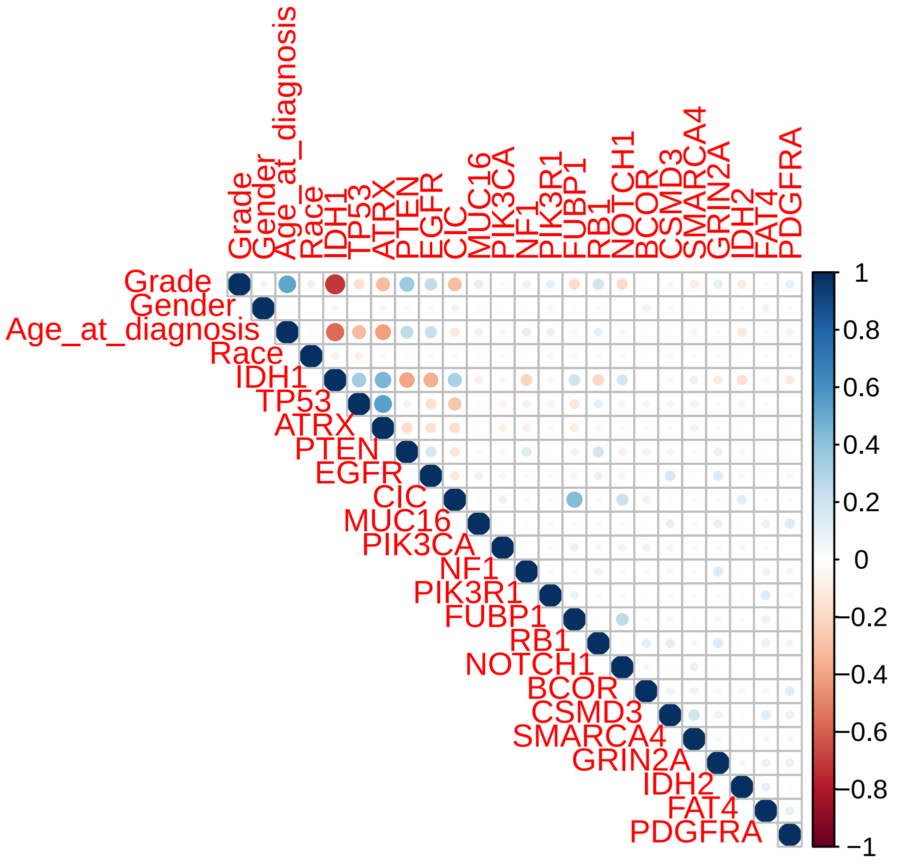
```r
# Print the count of outliers for each column
print(outliers_count)
```

```
##
##                0                0                0                0
##
##                0                0                0                0
##
##                0                0                0                0
##
##                0                0                0                0
##
##                0                0                0                0
##
##                0                0                0                0
##            Grade           Gender Age_at_diagnosis             Race
##                0                0                0               15
##             IDH1             TP53             ATRX             PTEN
##                0                0                0                0
##             EGFR              CIC            MUC16           PIK3CA
##                0                0                0               73
##              NF1           PIK3R1            FUBP1              RB1
##               67               54               45               40
##           NOTCH1             BCOR            CSMD3          SMARCA4
##               38               29               27               27
##           GRIN2A             IDH2             FAT4           PDGFRA
##               27               23               23               22
```

Correlation Analysis

We compute the correlation matrix and visualize it using a correlation plot.

```r
library(corrplot)
# Compute correlation matrix
correlation_matrix <- cor(gli_data_num)
# Plot correlation matrix
corrplot(correlation_matrix, method = "circle", type = "upper")
```



Data Normalization

Min-Max normalization technique.

```r
# Min-Max Normalization and visualization
feature_columns <- setdiff(names(gli_data_num), "Grade")
features <- gli_data_num[, feature_columns]

min_max_normalize <- function(x) { (x - min(x)) / (max(x) - min(x))
}

normalized_features <- as.data.frame(lapply(features, min_max_normalize))
# Combine normalized features with Glioma_Grade column
normalized_data <- cbind(Glioma_Grade = gli_data_num$Grade, normalized_features)
```

Log Transformation of Continuous Features

We perform log transformation on continuous features to stabilize variance.

```r
# Select continuous features for log transformation
continuous_features <- c("Age_at_diagnosis")
# Perform log transformation
transformed_data <- gli_data_num
for (feature in continuous_features) {
```

```
if (any(gli_data_num[, feature] <= 0)) {
constant = 1 # Add constant for non-positive values transformed_data[, feature] <- log(gli_data_num[, f
} else {
transformed_data[, feature] <- log(gli_data_num[, feature])
} }

# Display of transformed values
head(transformed_data)
```

```
##   Grade Gender Age_at_diagnosis Race IDH1 TP53 ATRX PTEN EGFR CIC MUC16 PIK3CA
## 1     0      0         3.937691    0    1    0    0    0    0   0     0      1
## 2     0      0         3.656356    0    1    0    0    0    0   1     0      0
## 3     0      0         3.560193    0    1    1    1    0    0   0     0      0
## 4     0      1         3.489819    0    1    1    1    0    0   0     1      0
## 5     0      0         3.450305    0    1    1    1    0    0   0     0      0
## 6     0      1         3.502851    0    1    0    1    0    0   0     0      0
##   NF1 PIK3R1 FUBP1 RB1 NOTCH1 BCOR CSMD3 SMARCA4 GRIN2A IDH2 FAT4 PDGFRA
## 1   0      0     1   0      0    0     0       0      0    0    0      0
## 2   0      0     0   0      0    0     0       0      0    0    0      0
## 3   0      0     0   0      0    0     0       0      0    0    0      0
## 4   0      1     0   0      0    0     0       0      0    0    1      0
## 5   0      0     0   0      0    0     0       0      0    0    0      0
## 6   0      0     0   0      0    0     0       0      0    0    0      0
```

Principal Component Analysis (PCA)

We perform PCA on selected continuous features to reduce dimensionality.

```
library(MASS)
# Select continuous features for PCA
continuous_features <- c("Age_at_diagnosis") # Subset data
data_subset <- gli_data_num[, continuous_features] # Perform PCA
pca_result <- prcomp(data_subset, scale = TRUE) # Access principal component scores
pc_scores <- as.data.frame(pca_result$x)
# Display first few rows of principal component scores
head(pc_scores)
```

```
##           PC1
## 1  0.02321876
## 2 -0.77793579
## 3 -1.00401676
## 4 -1.15622339
## 5 -1.23710306
## 6 -1.12883893
```

Principal component scores are the transformed values of your original data points projected onto the principal component axes. Each value in the PC1 column represents how much the corresponding data point contributes to the first principal component. From the given values, we can make some initial observations: 1. The values appear to be numeric, suggesting that they represent the extent to which each data point contributes to the variation captured by PC1. 2. Negative values (like in rows 2 to 6) suggest that these data points have an opposite orientation in relation to the PC1 axis compared to the positive value (in row 1). 3. The magnitude of the values indicates the strength of the contribution of each data point to PC1.

Larger magnitudes typically indicate a stronger influence on the principal component. 4. Since PC1 captures the most significant variance in the data, the patterns in the scores of PC1 may reveal important trends or groupings within your data.

Interaction Feature Creation

We create a new derived feature by multiplying Age_at_diagnosis and Gender.

```
# Create interaction feature
transformed_data$Age_Gender_interaction <- transformed_data$Age_at_diagnosis * transformed_data$Gender
# Display first few rows of transformed data with new feature
head(transformed_data)
```

```
##   Grade Gender Age_at_diagnosis Race IDH1 TP53 ATRX PTEN EGFR CIC MUC16 PIK3CA
## 1     0      0         3.937691    0    1    0    0    0    0   0     0      1
## 2     0      0         3.656356    0    1    0    0    0    0   1     0      0
## 3     0      0         3.560193    0    1    1    1    0    0   0     0      0
## 4     0      1         3.489819    0    1    1    1    0    0   0     1      0
## 5     0      0         3.450305    0    1    1    1    0    0   0     0      0
## 6     0      1         3.502851    0    1    0    1    0    0   0     0      0
##   NF1 PIK3R1 FUBP1 RB1 NOTCH1 BCOR CSMD3 SMARCA4 GRIN2A IDH2 FAT4 PDGFRA
## 1   0      0     1   0      0    0     0       0      0    0    0      0
## 2   0      0     0   0      0    0     0       0      0    0    0      0
## 3   0      0     0   0      0    0     0       0      0    0    0      0
## 4   0      1     0   0      0    0     0       0      0    0    1      0
## 5   0      0     0   0      0    0     0       0      0    0    0      0
## 6   0      0     0   0      0    0     0       0      0    0    0      0
##   Age_Gender_interaction
## 1               0.000000
## 2               0.000000
## 3               0.000000
## 4               3.489819
## 5               0.000000
## 6               3.502851
```

MODEL A: k-Nearest Neighbors (k-NN) Classification

We perform k-NN classification and evaluate the model's performance.

```
library(class)
gli_data_num$Glioma_grade <- ifelse(gli_data_num$Grade == 0, "LGG", "GBM")
# Set a seed for reproducibility
set.seed(142)
# Perform 80/20 split for training and test data
train_indices <- sample(1:nrow(gli_data_num), 0.8 * nrow(gli_data_num))
train.data <- gli_data_num[train_indices, ]
test.data <- gli_data_num[-train_indices, ]
library(class)
class_knn_pred <- knn(train=train.data[,-25], test=test.data[,-25], cl=train.data$Glioma_grade, k=5, )
class_knn_pred
```

```
##   [1] LGG LGG GBM LGG LGG LGG LGG LGG LGG GBM LGG LGG LGG LGG LGG LGG LGG LGG
##  [19] LGG LGG LGG LGG LGG LGG LGG GBM LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG
##  [37] LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG
##  [55] LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG
##  [73] LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG
##  [91] LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG LGG GBM GBM GBM GBM GBM GBM GBM
## [109] GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM
## [127] GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM LGG GBM GBM GBM GBM GBM
## [145] GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM GBM LGG GBM GBM GBM GBM GBM GBM
## [163] GBM GBM GBM GBM GBM GBM
## Levels: GBM LGG
```

```
# Confusion matrix
conf_matrix <- table(Actual = test.data$Glioma_grade, Predicted = class_knn_pred)
conf_matrix
```

```
##       Predicted
## Actual GBM LGG
##    GBM  65   2
##    LGG   3  98
```

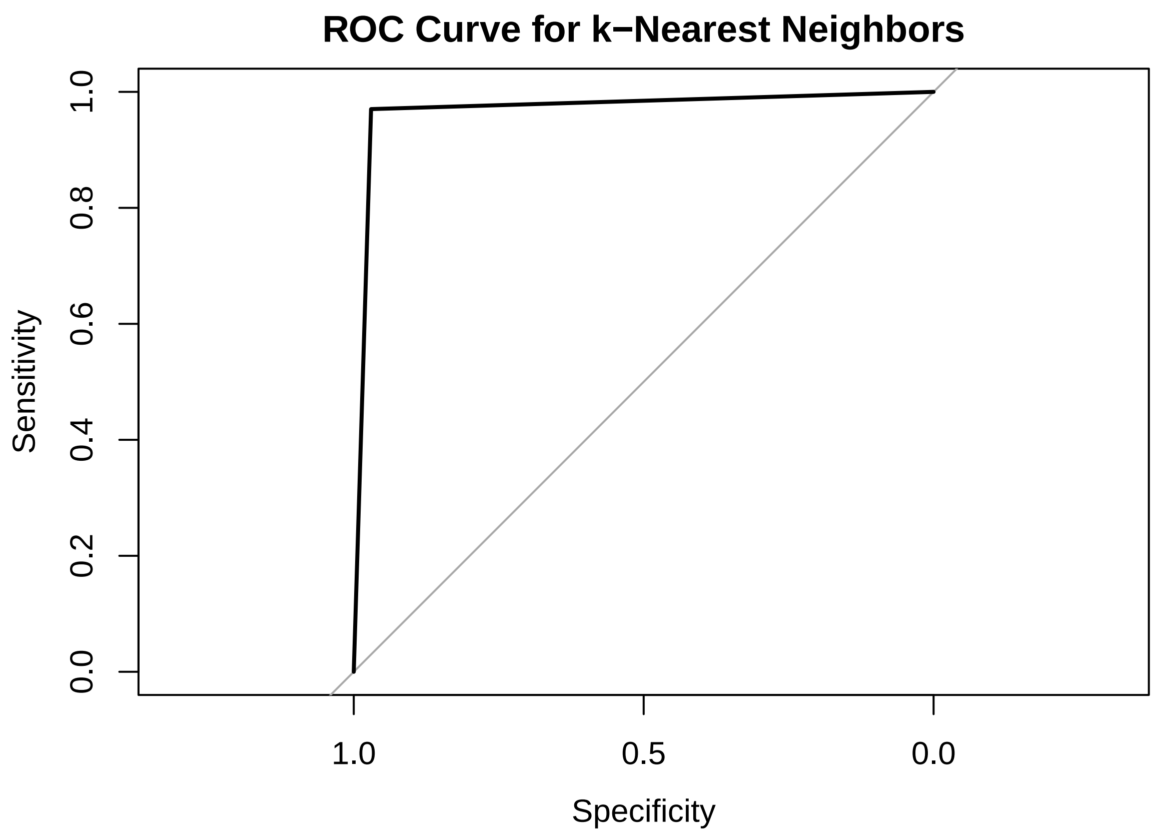
```
# Define true_labels (replace with actual true labels)
true_labels <- test.data$Glioma_grade
# Calculate accuracy, precision, recall, etc. using confusion matrix
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
precision <- conf_matrix[2, 2] / sum(conf_matrix[, 2])
recall <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
f1_score <- 2 * (precision * recall) / (precision + recall)
# Calculate ROC curve and AUC
roc_curve <- roc(true_labels, as.numeric(class_knn_pred))
```

```
## Setting levels: control = GBM, case = LGG
```

```
## Setting direction: controls < cases
```

```
auc_score <- auc(roc_curve)
roc_auc <- auc(roc_curve)
```

```
# Plot ROC curve
plot(roc_curve, main = "ROC Curve for k-Nearest Neighbors", colorize = TRUE)
```



```
# Print evaluation metrics
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.9702381
```

```
cat("Precision:", precision, "\n")
```

```
## Precision: 0.98
```

```
cat("Recall:", recall, "\n")
```

```
## Recall: 0.970297
```

```
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 0.9751244
```

```
cat("ROC AUC:", roc_auc, "\n")
```

```
## ROC AUC: 0.9702231
```

MODEL B: Naïve Bayes Classification

We perform Naïve Bayes classification and evaluate the model's performance.

```
library(e1071)
features <- c("Gender", "Age_at_diagnosis", "Race", "IDH1", "TP53", "ATRX", "PTEN", "EGFR", "CIC", "MUC
target <- "Glioma_grade"
# Select data for modeling
train_x <- train.data[, features]
train_y <- train.data[, target]
valid_x <- test.data[, features]
valid_y <- test.data[, target]
# Naïve Bayes
model_nb <- naiveBayes(train_x, train_y)
# Evaluate model_naive_bayes and calculate metrics...
# Predict using the trained model on validation data
predictions_nb <- predict(model_nb, newdata = valid_x) # Convert predictions_nb to numeric if it's a fa
predictions_nb <- as.numeric(predictions_nb) # Calculate metrics for Naive Bayes
conf_matrix_nb <- table(Actual = test.data$Glioma_grade, Predicted = predictions_nb)
# Calculate accuracy, precision, recall, etc. using confusion matrix
accuracy <- sum(diag(conf_matrix_nb)) / sum(conf_matrix_nb)
precision <- conf_matrix_nb[2, 2] / sum(conf_matrix_nb[, 2])
recall <- conf_matrix_nb[2, 2] / sum(conf_matrix_nb[2, ])
f1_score <- 2 * (precision * recall) / (precision + recall)
# Calculate ROC curve and AUC
roc_curve <- roc(true_labels, as.numeric(predictions_nb))
```

```
## Setting levels: control = GBM, case = LGG

## Setting direction: controls < cases
```

```
roc_auc <- auc(roc_curve)
```

```
# Print evaluation metrics
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.8333333
```

```
cat("Precision:", precision, "\n")
```

```
## Precision: 0.9506173
```

```
cat("Recall:", recall, "\n")
```

```
## Recall: 0.7623762
```

```
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 0.8461538
```

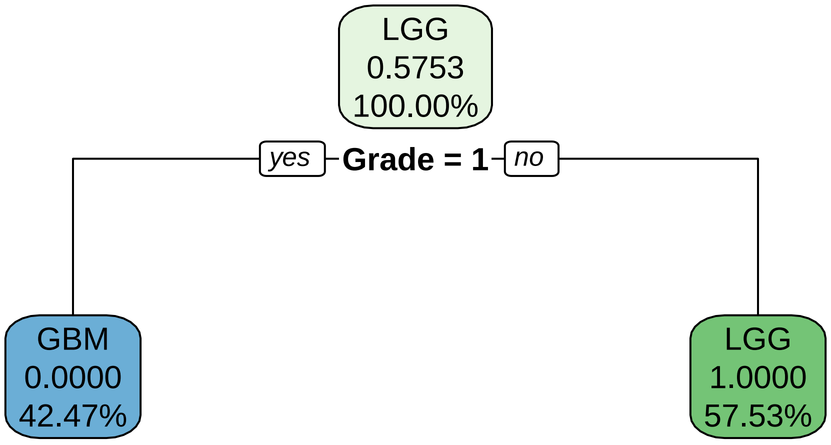
```
cat("ROC AUC:", roc_auc, "\n")
```

```
## ROC AUC: 0.8513374
```

MODEL C: Decision Tree Classification

We build and evaluate a decision tree classification model.

```
# Assuming 'target' is the dependent variable and other columns are independent variables
model_decision_tree <- rpart(Glioma_grade ~ ., data = train.data, method = "class")
rpart.plot(model_decision_tree, digits = 4)
```



```
# Perform predictions using the fitted model
predictions_decision_tree <- predict(model_decision_tree, newdata = test.data, type = "class")
conf_matrix_dt <- table(Actual = test.data$Glioma_grade, Predicted = predictions_decision_tree)

# Calculate accuracy, precision, recall, etc. using confusion matrix
accuracy <- sum(diag(conf_matrix_dt)) / sum(conf_matrix_dt)
precision <- conf_matrix_dt[2, 2] / sum(conf_matrix_dt[, 2])
recall <- conf_matrix_dt[2, 2] / sum(conf_matrix_dt[2, ])
f1_score <- 2 * (precision * recall) / (precision + recall)

# Calculate ROC curve and AUC
roc_curve <- roc(true_labels, as.numeric(predictions_nb))
```

```
## Setting levels: control = GBM, case = LGG
```

```
## Setting direction: controls < cases
```

```r
roc_auc <- auc(roc_curve)

# Print evaluation metrics
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 1
```

```r
cat("Precision:", precision, "\n")
```

```
## Precision: 1
```

```r
cat("Recall:", recall, "\n")
```

```
## Recall: 1
```

```r
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 1
```

```r
cat("ROC AUC:", roc_auc, "\n")
```

```
## ROC AUC: 0.8513374
```

Comparision of the models

Based on the provided evaluation metrics for the three models, we can compare and conclude their performance as follows:

MODEL A: - Accuracy: 0.9702381 - Precision: 0.98 - Recall: 0.970297 - F1 Score: 0.9751244 - ROC AUC: 0.9702231

MODEL B: - Accuracy: 0.8333333 - Precision: 0.9506173 - Recall: 0.7623762 - F1 Score: 0.8461538 - ROC AUC: 0.8513374

MODEL C: - Accuracy: 1 - Precision: 1 - Recall: 1 - F1 Score: 1 - ROC AUC: 0.8513374

From the metrics provided, we can make the following observations:

1. Accuracy: Model C has the highest accuracy of 1, which indicates that it correctly predicts all instances in the validation set. However, perfect accuracy can also be a sign of overfitting, so it's important to consider other metrics.
2. Precision: Model C has the highest precision of 1, indicating that when it predicts a positive class (LGG or GBM), it is almost always correct. Model A also has a very high precision of 0.98.
3. Recall: Model C has the highest recall of 1, which means it correctly identifies all instances of the positive class. Model A has a recall of 0.970297, which is also very high.

4. F1 Score: Model C has the highest F1 score of 1, which is a balance between precision and recall. This indicates a good balance between correctly identifying positive instances and minimizing false positives.
5. ROC AUC: Model A and Model B have similar ROC AUC values, but Model A's value is slightly higher. ROC AUC measures the model's ability to distinguish between classes and ranks them accordingly.

Based on these observations, Model C appears to perform exceptionally well with perfect accuracy, precision, recall, and F1 score. However, such perfect performance might suggest overfitting or issues in the evaluation process, especially if the dataset is small or imbalanced.

Both Model A and Model B seem to have strong performance across various metrics. Model A has higher accuracy and slightly better ROC AUC than Model B. If we consider the balance between precision and recall, Model A seems to be a well-rounded choice.

In conclusion, while Model C's perfect performance might raise questions, Model A seems to be the better choice due to its high accuracy, balanced precision and recall, and good ROC AUC. However, it's important to validate the findings on additional datasets and perform further analysis before making a final decision. An accuracy of 0.3988095 for the bagged ensemble means that the ensemble model correctly predicted the target variable (in this case, glioma grade) for approximately 39.88% of the validation set instances.

Conclusion

In this analysis, I performed comprehensive data preprocessing, explored the dataset, handled missing values, detected and removed outliers, normalized features, transformed continuous features, conducted PCA, and built classification models using k-NN, Naïve Bayes, and decision trees. The evaluation of each model's performance using various metrics provided insights into their strengths and weaknesses.

The comparison of the models revealed that while Model C exhibited perfect performance, potential overfitting or evaluation issues must be considered. Models A and B displayed strong performance, with Model A appearing as a well-rounded choice due to its high accuracy, balanced precision and recall, and good ROC AUC.

However, it's imperative to validate these findings on additional datasets and conduct further analysis before making a final decision.