Report: Developing a Prediction Algorithm to Assess Survival Time of Hematopoietic Stem Cell Transplant Recipients

Jackman Eschenroeder

2024-05-15

Introduction

Cancer is significantly less common in children than in adults. However, the incidence of cancer in young people is far from zero. In fact, at least 15,000 children and teens between 0 and 19 years of age were diagnosed with cancer in 2022 in the United States alone (Siegel et al. 2022). Fortunately, many childhood cancers may be treated with techniques such as hematopoietic stem cell transplant, and more than 85% of children diagnosed with cancer in the US will survive more than five years after their diagnosis (Russel et al. 2024). The increasing availability of healthcare data stands poised to provide a heretofore unprecedented ability to understand diseases and guide treatment, although leveraging these data is not without its challenges (Sweeney et al. 2023). The use of machine learning approaches may be of particular value for organizations seeking to optimize outcomes for transplant patients, as many factors related to the demographics of the donor and recipient, donor and recipient human leukocyte antigen (HLA) profiles, and stem cell sources may play a significant role in treatment success.

As the capstone project for the HarvardX Professional Certificate in Data Science, this report describes the application of machine learning approaches to predict transplant patient survival time. In this report, I detail the steps taken to develop a predictive algorithm using a dataset of stem cell transplants in child cancer patients. The data describes pediatric patients with a variety of hematologic diseases, including acute lymphoblastic leukemia, acute myelogenous leukemia, and chronic myelogenous leukemia, among others. The goal of my project is to investigate the importance of various factors influencing post-transplant survival time, and my primary objective was to develop a model to predict survival time with an RMSE $\leq$ 6 months (180 days) lower than the naive model using just the mean survival time.

Data Download and Preparation

As a first step, we will import the dataset, which is from Sikora et al. (2020) and is available on the Kaggle website at https://www.kaggle.com/datasets/adamgudys/bone-marrow-transplant-children/data. After ensuring the necessary R packages are installed, we will import the dataset and inspect it.

```r
# load packages #
library(openxlsx)
library(dplyr)
library(lubridate)
library(ggplot2)
library(reshape2)
library(caret)
library(gbm)
library(randomForest)
library(knitr)

# data from https://www.kaggle.com/datasets/adamgudys/bone-marrow-transplant-children
# Note that this directory must be set to the location where the data file is stored
dir<-"/Users/jackesch/projects/Bone_Marrow_Transplant_ML/"
setwd(dir)

data <- read.csv("bone_marrow_dataset.csv")

kable(str(data), format = "markdown")
```

```
## 'data.frame':    187 obs. of  38 variables:
##  $ patient_number           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ donor_age                : num  21.4 40 30.4 30 24.6 ...
##  $ donor_age_below_35       : chr  "yes" "no" "yes" "yes" ...
##  $ donor_ABO                : chr  "0" "A" "AB" "0" ...
##  $ donor_CMV                : chr  "present" "absent" "present" "absent" ...
##  $ recipient_age            : num  18.2 4.7 3.4 6.5 5 13.4 8.5 6.9 5 1.8 ...
##  $ recipient_age_below_10   : chr  "no" "yes" "yes" "yes" ...
##  $ recipient_age_int        : chr  "10_20" "0_5" "0_5" "5_10" ...
##  $ recipient_gender         : chr  "male" "male" "male" "female" ...
##  $ recipient_body_mass      : num  56 17 10 16 25.4 47 30 28.5 19.5 8.15 ...
##  $ recipient_ABO            : chr  "A" "B" "0" "A" ...
##  $ recipient_rh             : chr  "plus" "plus" "minus" "plus" ...
##  $ recipient_CMV            : chr  "absent" NA "present" "absent" ...
##  $ disease                  : chr  "nonmalignant" "ALL" "nonmalignant" "chronic" ...
##  $ disease_group            : chr  "nonmalignant" "malignant" "nonmalignant" "malignant" ...
##  $ gender_match             : chr  "other" "other" "other" "other" ...
##  $ ABO_match                : chr  "mismatched" "mismatched" "mismatched" "mismatched" ...
##  $ CMV_status               : int  1 NA 3 0 2 0 2 2 0 3 ...
##  $ HLA_match                : int  70 70 70 70 70 80 80 80 80 80 ...
##  $ HLA_mismatch             : chr  "mismatched" "mismatched" "mismatched" "mismatched" ...
##  $ antigen                  : int  2 2 2 1 2 2 3 2 1 2 ...
##  $ allel                    : int  3 3 3 4 3 2 1 2 3 2 ...
##  $ HLA_group_1              : chr  "mismatched" "mismatched" "mismatched" "mismatched" ...
##  $ risk_group               : chr  "low" "low" "high" "high" ...
##  $ stem_cell_source         : chr  "bone_marrow" "peripheral_blood" "peripheral_blood" "peripheral_bl
##  $ tx_post_relapse          : chr  "no" "no" "yes" "no" ...
##  $ CD34_x1e6_per_kg         : num  6.41 11.27 21.74 43.96 10 ...
##  $ CD3_x1e8_per_kg          : num  NA 3.48 7.32 7.32 4.09 6.45 0.73 4.15 8.71 5.24 ...
##  $ CD3_to_CD34_ratio        : num  NA 3.24 2.97 6 2.44 ...
##  $ ANC_recovery             : num  22 16 12 11 12 11 14 14 14 13 ...
##  $ PLT_recovery             : num  58 1000000 15 13 14 16 24 14 12 10 ...
##  $ acute_GvHD_II_III_IV     : chr  "yes" "yes" "no" "yes" ...
##  $ acute_GvHD_III_IV        : chr  "yes" "yes" "no" "yes" ...
##  $ time_to_acute_GvHD_III_IV: num  22 16 1000000 24 1000000 1000000 15 1000000 1000000 1000000 ...
##  $ extensive_chronic_GvHD   : chr  NA NA "no" "no" ...
##  $ relapse                  : chr  "no" "no" "no" "no" ...
##  $ survival_time            : int  45 35 1791 113 1692 2503 2354 2926 1424 1958 ...
##  $ survival_status          : int  1 1 0 1 0 0 0 0 0 0 ...
```

We can see that the dataframe consists of data from 187 patients, and includes 38 different variables. Because some of the machine learning approaches we will be implementing cannot handle missing data, we will inspect this data for missing values.

```
##              patient_number                donor_age        donor_age_below_35
##                           0                        0                         0
##                  donor_ABO              donor_CMV          recipient_age 
##                          0                      2                      0 
##    recipient_age_below_10      recipient_age_int       recipient_gender 
##                          0                      0                      0 
##       recipient_body_mass          recipient_ABO           recipient_rh 
##                          2                      1                      2 
##              recipient_CMV                disease          disease_group 
##                         14                      0                      0 
##               gender_match              ABO_match             CMV_status 
##                          0                      1                     16 
##                  HLA_match           HLA_mismatch                antigen 
##                          0                      0                      1 
##                     allel            HLA_group_1             risk_group 
##                          1                      0                      0 
##          stem_cell_source        tx_post_relapse       CD34_x1e6_per_kg 
##                          0                      0                      0 
##           CD3_x1e8_per_kg      CD3_to_CD34_ratio           ANC_recovery 
##                          5                      5                      0 
##               PLT_recovery    acute_GvHD_II_III_IV       acute_GvHD_III_IV 
##                          0                      0                      0 
## time_to_acute_GvHD_III_IV  extensive_chronic_GvHD                relapse 
##                          0                     31                      0 
##              survival_time        survival_status 
##                          0                      0
```

We see that some columns have missing data. In particular, extensive_chronic_GvHD is missing data for more than 16% of the rows. We will remove that column from the dataframe, and then all remaining rows with missing data before beginning our model development.

```r
# Remove extensive_chronic_GvHD column
data <- data[, !names(data) %in% "extensive_chronic_GvHD"]

# Remove extensive_chronic_GvHD column
data <- data[, !names(data) %in% "extensive_chronic_GvHD"]

# Remove all rows with missing values in any column
data <- data[complete.cases(data), ]
```

We will also round the continuous donor and recipient age variables to the nearest year, and convert all character variables to factors.

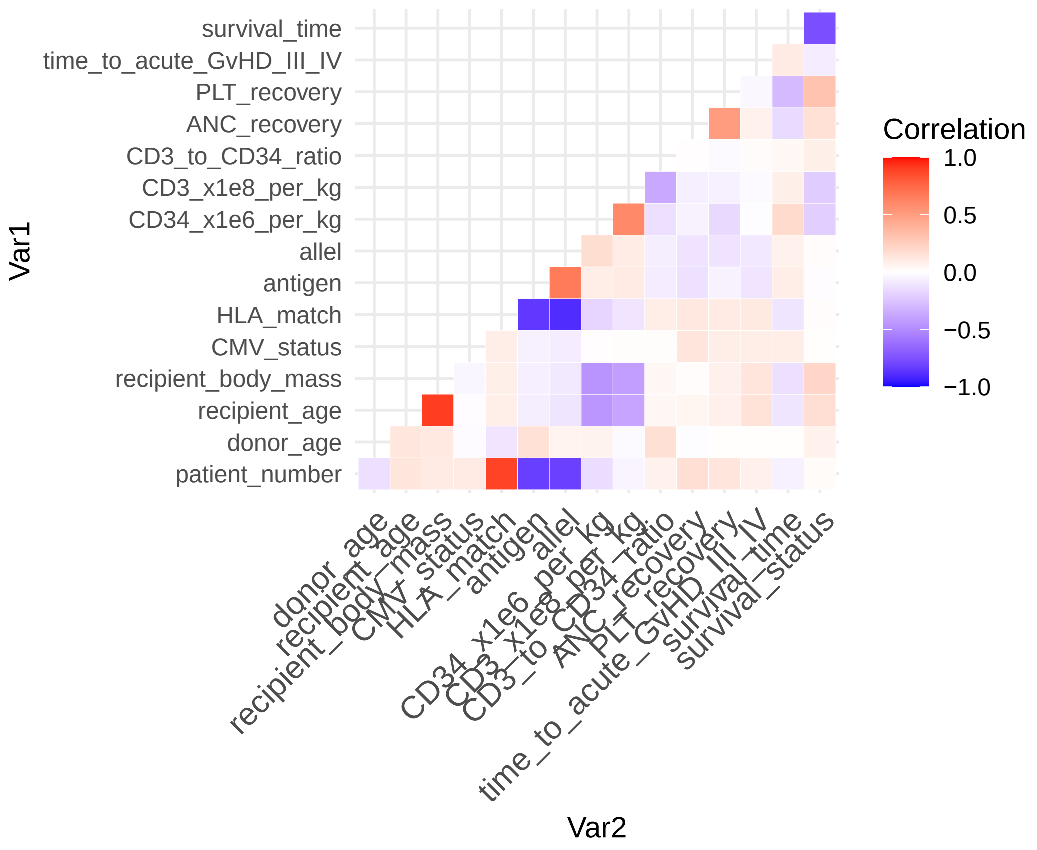
```r
# Round donor age to nearest year
data$donor_age <- round(data$donor_age)

# Round recipient age to nearest year
data$recipient_age <- round(data$recipient_age)

# Identify character variables
char_vars <- sapply(data, is.character)

# Convert character variables to factors
data[char_vars] <- lapply(data[char_vars], factor)
```

We can now examine the structure of the cleaned data. We see that it retains data for 164 patients across 37 variables, which include various factors related to the recipient, the donor, and treatment.

```
## 'data.frame':    164 obs. of  37 variables:
```

```
##  $ patient_number          : int  3 4 5 6 7 8 9 10 11 12 ...
##  $ donor_age               : num  30 30 25 37 30 33 40 31 47 32 ...
##  $ donor_age_below_35      : Factor w/ 2 levels "no","yes": 2 2 2 1 2 2 1 2 1 2 ...
##  $ donor_ABO               : Factor w/ 4 levels "0","A","AB","B": 3 1 4 2 1 1 2 3 2 2 ...
##  $ donor_CMV               : Factor w/ 2 levels "absent","present": 2 1 1 1 1 1 1 2 2 1 ...
##  $ recipient_age           : num  3 6 5 13 8 7 5 2 12 20 ...
##  $ recipient_age_below_10  : Factor w/ 2 levels "no","yes": 2 2 2 1 2 2 2 2 1 1 ...
##  $ recipient_age_int       : Factor w/ 3 levels "0_5","10_20",..: 1 3 1 2 3 3 1 1 2 2 ...
##  $ recipient_gender        : Factor w/ 2 levels "female","male": 2 1 2 1 1 2 2 2 1 2 ...
##  $ recipient_body_mass     : num  10 16 25.4 47 30 28.5 19.5 8.15 30 62.5 ...
##  $ recipient_ABO           : Factor w/ 4 levels "0","A","AB","B": 1 2 2 2 2 1 2 1 2 3 ...
##  $ recipient_rh            : Factor w/ 2 levels "minus","plus": 1 2 2 2 2 1 2 2 2 1 ...
##  $ recipient_CMV           : Factor w/ 2 levels "absent","present": 2 1 2 1 2 2 1 2 2 1 ...
##  $ disease                 : Factor w/ 5 levels "ALL","AML","chronic",..: 5 3 1 3 1 1 5 5 3 2 ...
##  $ disease_group           : Factor w/ 2 levels "malignant","nonmalignant": 2 1 1 1 1 1 2 2 1 1 ...
##  $ gender_match            : Factor w/ 2 levels "female_to_male",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ ABO_match               : Factor w/ 2 levels "matched","mismatched": 2 2 2 1 2 1 1 2 1 2 ...
##  $ CMV_status              : int  3 0 2 0 2 2 0 3 3 0 ...
##  $ HLA_match               : int  70 70 70 80 80 80 80 80 80 80 ...
##  $ HLA_mismatch            : Factor w/ 2 levels "matched","mismatched": 2 2 2 2 2 2 2 2 2 2 ...
##  $ antigen                 : int  2 1 2 2 3 2 1 2 2 3 ...
##  $ allel                   : int  3 4 3 2 1 2 3 2 2 1 ...
##  $ HLA_group_1             : Factor w/ 7 levels "DRB1_cell","matched",..: 3 3 3 7 7 6 6 7 7 7 ...
##  $ risk_group              : Factor w/ 2 levels "high","low": 1 1 2 1 1 2 2 2 1 1 ...
##  $ stem_cell_source        : Factor w/ 2 levels "bone_marrow",..: 2 2 2 2 1 2 2 2 2 2 ...
##  $ tx_post_relapse         : Factor w/ 2 levels "no","yes": 2 1 1 1 2 1 1 1 1 1 ...
##  $ CD34_x1e6_per_kg        : num  21.74 43.96 10 14.46 2.68 ...
##  $ CD3_x1e8_per_kg         : num  7.32 7.32 4.09 6.45 0.73 4.15 8.71 5.24 8.75 3.39 ...
##  $ CD3_to_CD34_ratio       : num  2.97 6 2.44 2.24 3.65 ...
##  $ ANC_recovery            : num  12 11 12 11 14 14 14 13 15 16 ...
##  $ PLT_recovery            : num  15 13 14 16 24 14 12 10 19 111 ...
##  $ acute_GvHD_II_III_IV    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 1 1 1 1 2 ...
##  $ acute_GvHD_III_IV       : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 1 1 1 1 ...
##  $ time_to_acute_GvHD_III_IV: num  1000000 24 1000000 1000000 15 1000000 1000000 1000000 1000000 100
##  $ relapse                 : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 1 1 1 2 ...
##  $ survival_time           : int  1791 113 1692 2503 2354 2926 1424 1958 606 767 ...
##  $ survival_status         : int  0 1 0 0 0 0 0 0 1 0 ...
```

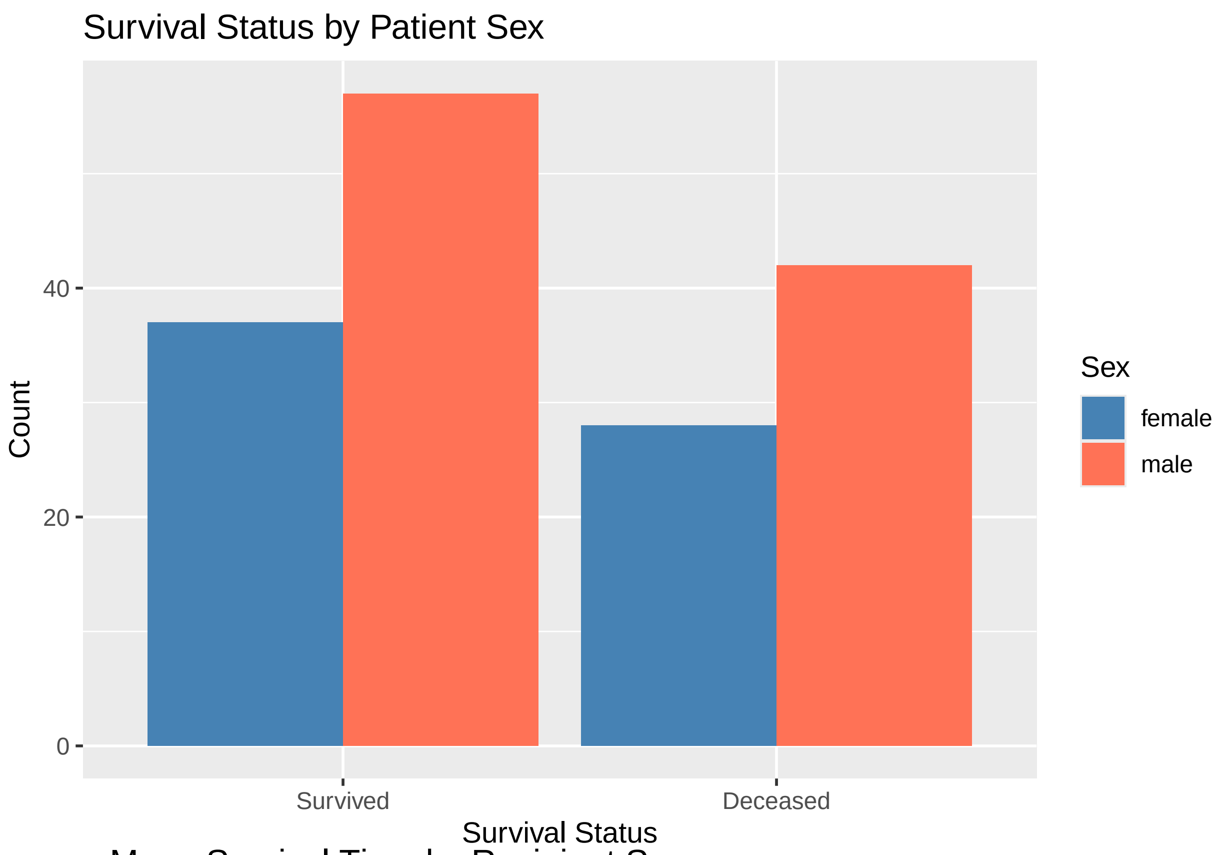
Our model will be designed to predict a continuous variable: survival_time. This variable refers to Time of observation (if alive) or time to event (if dead) in days. To begin with, we'll examine correlations among the numeric variables. We do this because highly correlated variables may lead to overfitting when using certain machine learning models.

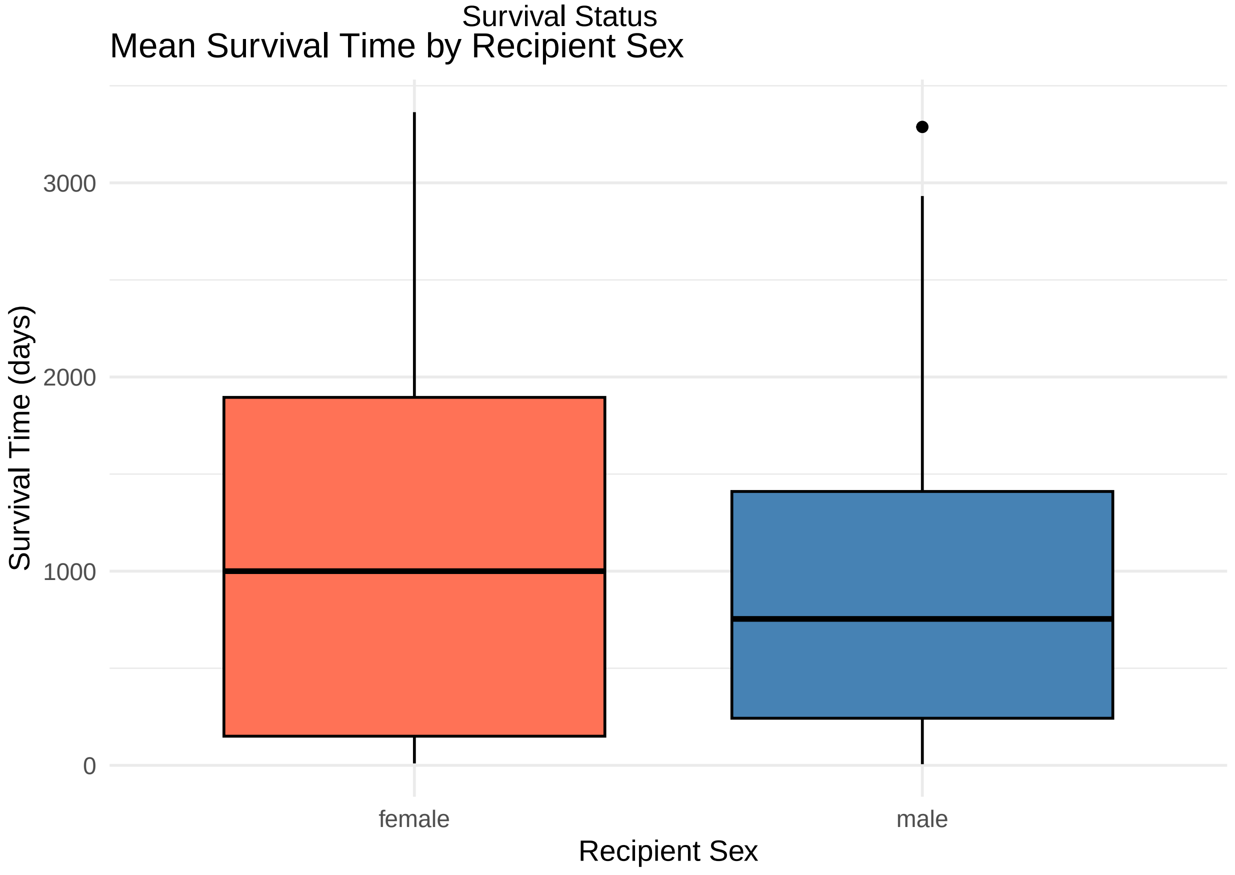


We can see that most variables have minimal correlation, though a few appear to be more highly correlated. We can double check to ensure than none are perfectly correlated.

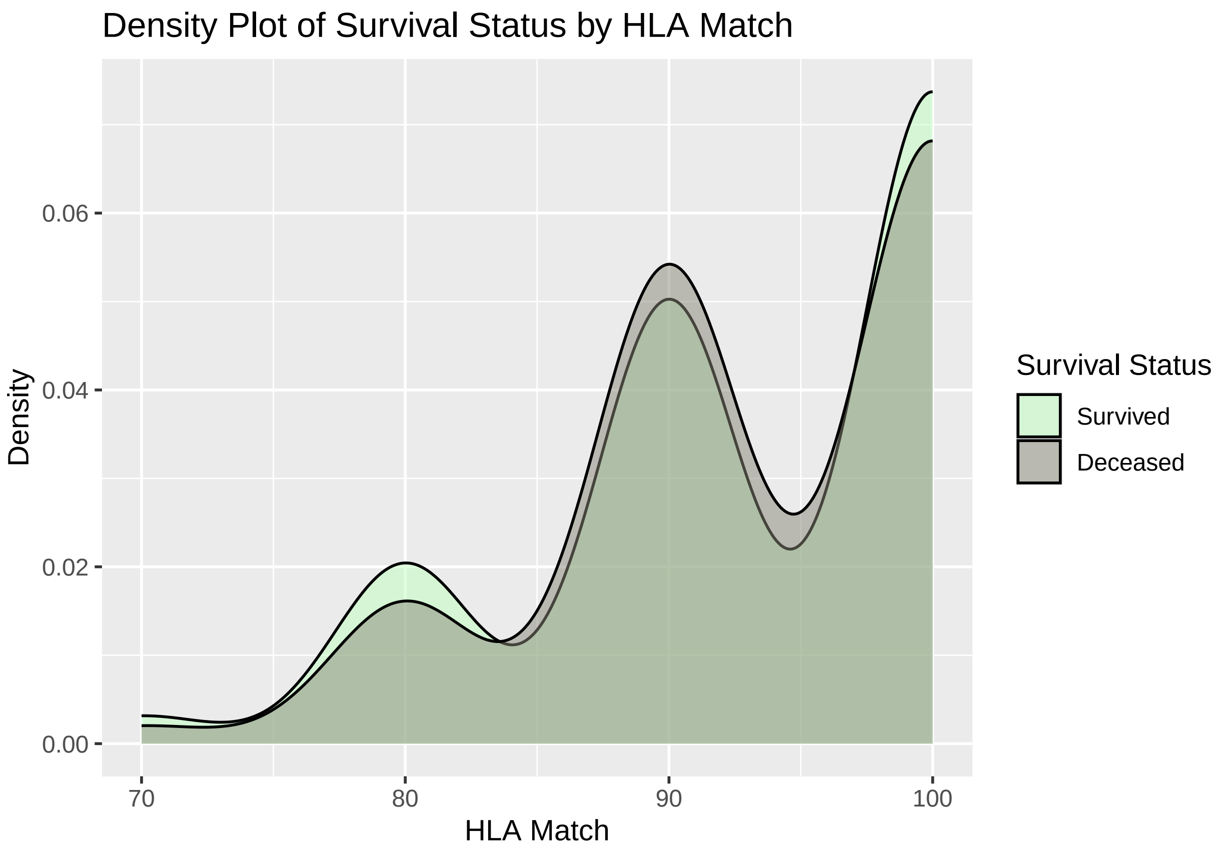
```
## [1] Variable1 Variable2
## <0 rows> (or 0-length row.names)
```

We see that none are, so we can retain all variables for our modeling. Before beginning, however, we can perform some basic visualizations to explore the data, starting with an examination of the trends between survival and patient sex.

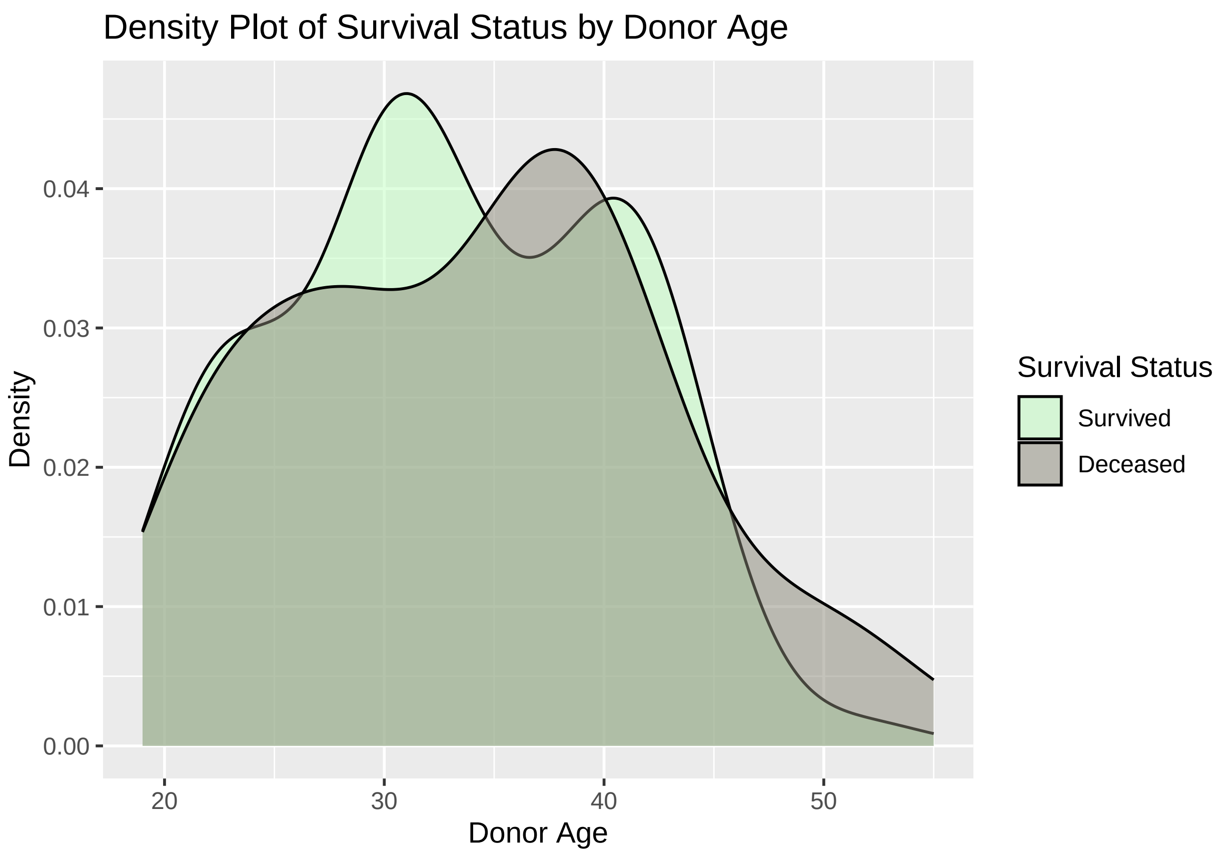




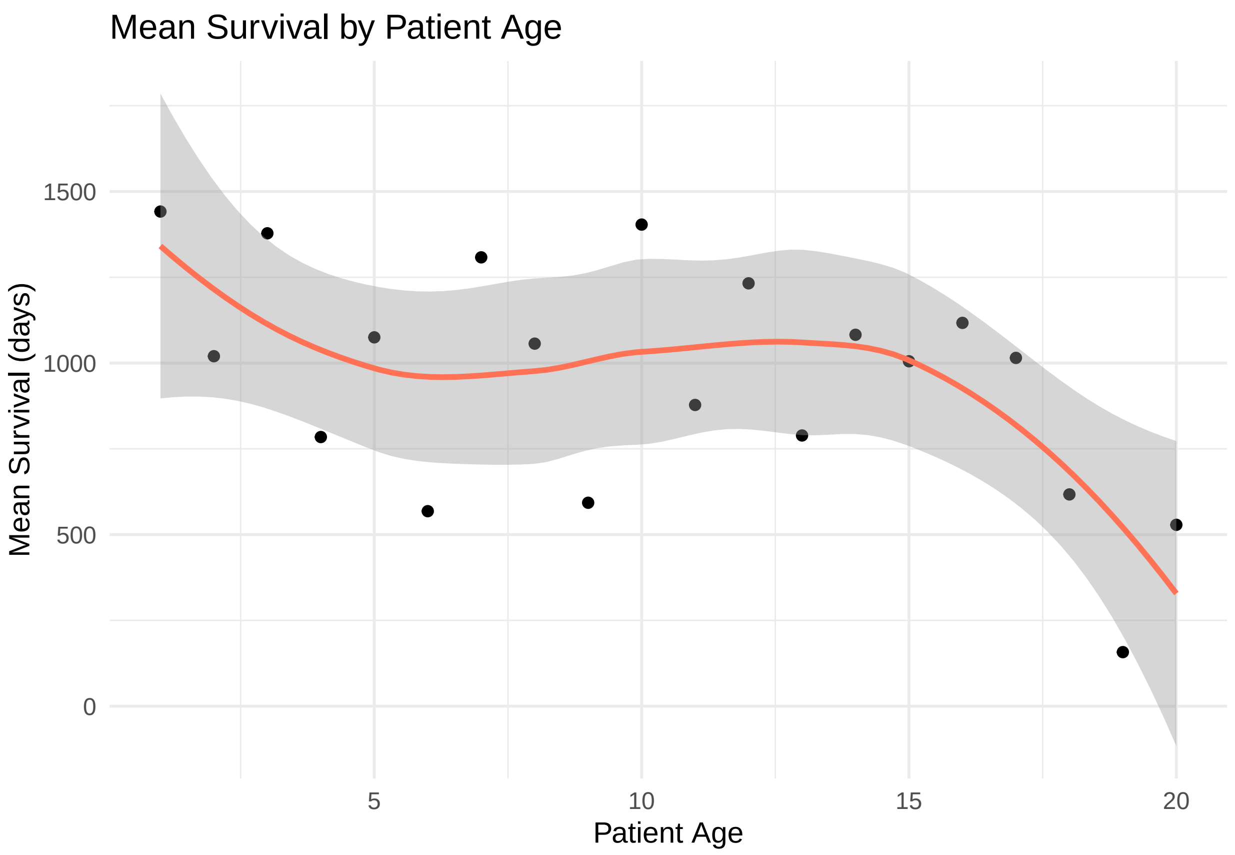
Survival status and survival time appear fairly evenly split by patient sex, suggesting that variable may not provide much predictive power in a machine learning algorithm focused on predicting survival time. We can continue to examine the relationship between survival and other potential predictor variables to see if there may be others which have more apparant trends, starting with HLA match. Human leukocyte antigens (HLA) are genes in major histocompatibility complexes (MHC) that help code for proteins that differentiate between self and non-self, and HLA match between donor and patient can be critical to treatment success. HLA match in this dataset is reported as a match out of ten (70%, 80%, 90%, or 100%).

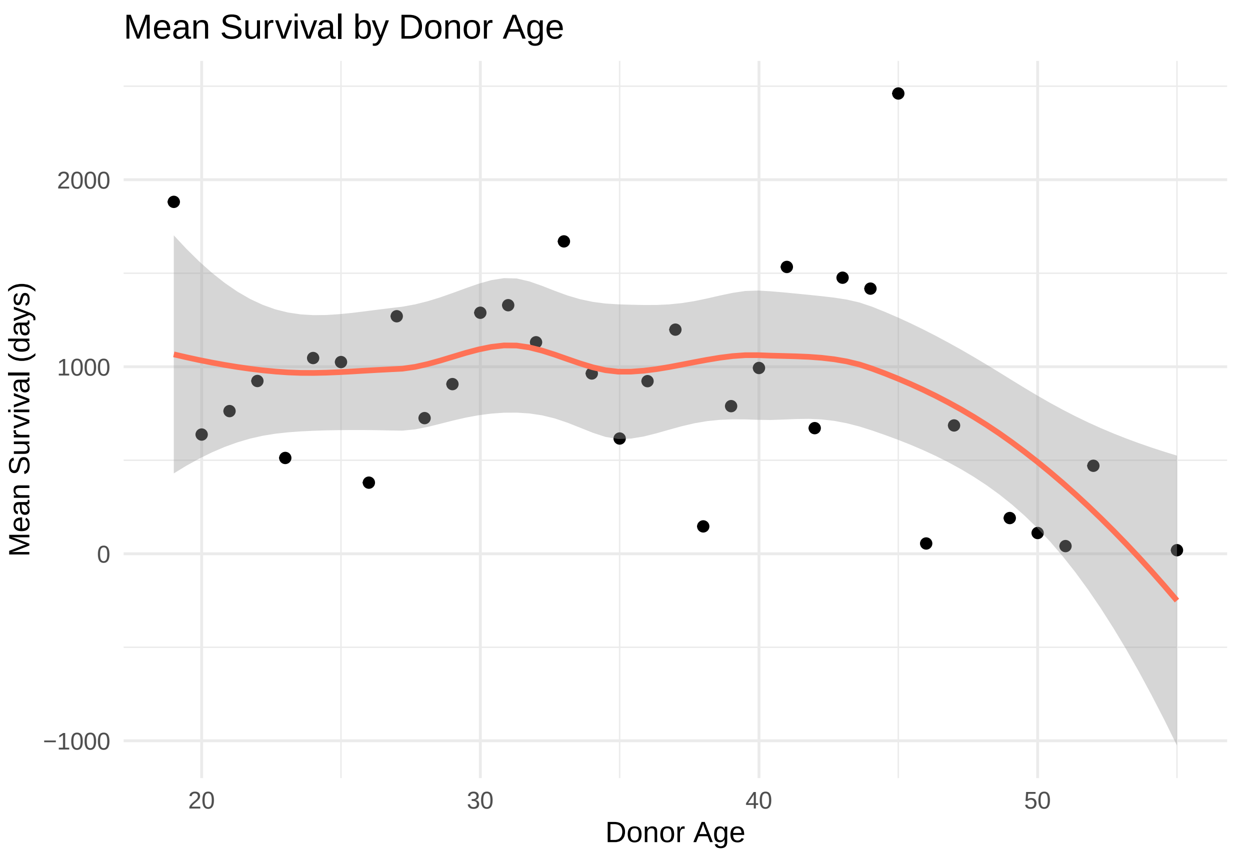


Survival does appear to vary somewhat by the percent HLA match, indicating this may be a useful predictor variable. Donor age is also known to be an important factor contributing to transplant success, so we can examine its relationship with survival.



Again, we see that survival also varies with the age of the stem cell donor. To look more closely at age effects, we can plot survival time by patient age, and by donor age.


Mean Survival by Patient Age
Mean Survival (days)
1500
1000
500
0
5
10
15
20
Patient Age


Mean Survival by Donor Age
Mean Survival (days)
2000
1000
0
−1000
20
30
40
50
Donor Age

Survival time does seem to vary with patient age, as well as with donor age. In particular, it seems younger donors lead to better survival.

Now that we've examined some of the relationships between predictor variables and survival time and have a better understanding of potential trends, we can begin developing our machine learning approaches.

Partitioning the Data - Test Set and Training Set

Before developing the model, the dataset must be split into a "test set" and a "train set." This was achieved by partitioning 10% of the data as the test set, and having the remainder housed in the training set. These two sets were created to allow for subsequent model development and testing. We can achieve this partition with the following code:

```
set.seed(1701, sample.kind="Rounding")
test_index <-createDataPartition(y = data$survival_status, times = 1, p = 0.2, list = F)
train <-data[-test_index,]
test <-data[test_index,]
```

Before beginning our experimentation with different modeling approaches, we have to define root-mean-square deviation (RMSE). This is also referred to as the loss function, and, as noted above, RMSE is the approach we use for evaluating the accuracy of the predictions our model generates compared to true survival time values. The equation for RMSE is:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}(\hat{y}_i - y_i)^2}$$

Where: N is the number of observations, $\hat{y}_i$ is the predicted value (in other words, the predicted survival time of the patient), and y_i is the true value (in other words, the actual survival time of the patient).

As a benchmark model, we can first calculate RMSE based on the mean survival time of the train data set. We will call this the "Naive" model. The formula for this is as follows:

```
# Apply naive model
mu_hat <- mean(train$survival_time)
mu_hat
```

```
## [1] 979.6947
```

```
naive_rmse <- RMSE(train$survival_time, mu_hat)
naive_rmse
```

```
## [1] 843.5371
```

```
results_table <-tibble(Model_Type = "Naive RMSE", RMSE = naive_rmse) %>%
  mutate(RMSE = sprintf("%0.4f", RMSE))

kable(results_table, format = "markdown")
```

Model_Type	RMSE
Naive RMSE	843.5371

This naive model generates an RMSE value of 843.5370813, which is equivalent to 28.1179027 months. As stated above, our goal for this modeling excerise will be to impove upon this RMSE value by at least six months (180 days). Therefore, our target RMSE is 663.5370813.

As a next step, we can try to improve upon this naive modle by adding HLA match effects. Again, this refers to the compatibility of antigens of the main histocompatibility complex of the donor and the recipient of hematopoietic stem cells, and includes values of 10/10, 9/10, 8/10, and 7/10. We can incorporate this into our model using the code below:

```r
mu <- mean(train$HLA_match)
b_HLA_match <- train %>%
  group_by(HLA_match) %>%
  summarize(b_HLA_match = mean(survival_time - mu))

predicted_b_HLA_match <- mu + test %>%
  left_join(b_HLA_match, by='HLA_match') %>%
  pull(b_HLA_match)
b_HLA_match_rmse<-RMSE(predicted_b_HLA_match, test$survival_time)
b_HLA_match_rmse
```

```
## [1] 840.7002
```

```r
results_table <-tibble(Model_Type = c("Naive RMSE", "HLA_match"),
                       RMSE =  c(naive_rmse,b_HLA_match_rmse)) %>%
  mutate(RMSE = sprintf("%0.4f", RMSE))

kable(results_table, format = "markdown")
```

Model_Type	RMSE
Naive RMSE	843.5371
HLA_match	840.7002

This leads to a slight improvement in RMSE, but only by 2.8368987. This is still a long way off from our target. As such, we'll need to incorporate other predictor variables, starting with antigens (i.e., how many antigens there is a difference between the donor and the recipient (0-3)).

```r
# Adding Antigen Effect
b_antigen <-train %>% left_join(b_HLA_match, by = "HLA_match") %>% group_by(antigen) %>%
  summarize(b_antigen = mean(survival_time - mu - b_HLA_match))

predicted_b_antigen <- test %>%
  left_join(b_HLA_match, by='HLA_match') %>%
  left_join(b_antigen, by='antigen')%>%
  mutate(predictions = mu + b_HLA_match + b_antigen) %>% .$predictions

b_antigen_rmse<-RMSE(test$survival_time, predicted_b_antigen)
b_antigen_rmse
```

```
## [1] 828.0364
```

```r
results_table <-tibble(Model_Type = c("Naive RMSE", "HLA_match",
                                      "HLA_match + antigen"),
                       RMSE =  c(naive_rmse, b_HLA_match_rmse, b_antigen_rmse)) %>%
  mutate(RMSE = sprintf("%0.4f", RMSE))

kable(results_table, format = "markdown")
```

Model_Type	RMSE
Naive RMSE	843.5371
HLA_match	840.7002

Model_Type	RMSE
HLA_match + antigen	828.0364

This further improves RMSE over the naive model by a total of 15.5006434. However, we need to continue to make improvements to reach our target RMSE. Next, we'll add in relapse status, meaning whether the patient's disease reoccurred.

```
# Adding Relapse status

b_relapse <-train %>% left_join(b_antigen, by = "antigen") %>%
  left_join(b_HLA_match, by = "HLA_match")%>%
  group_by(relapse) %>%
  summarize(b_relapse = mean(survival_time - mu - b_HLA_match - b_antigen))

predicted_b_relapse <- test %>%
  left_join(b_HLA_match, by='HLA_match') %>%
  left_join(b_antigen, by='antigen')%>%
  left_join(b_relapse, by='relapse')%>%
  mutate(predictions = mu + b_HLA_match + b_antigen + b_relapse) %>% .$predictions

b_relapse_rmse<-RMSE(test$survival_time, predicted_b_relapse, na.rm = TRUE)
b_relapse_rmse
```

```
## [1] 804.4859
```

```
results_table <-tibble(Model_Type = c("Naive RMSE", "HLA_match",
                                      "HLA_match + antigen",
                                      "HLA_match + antigen + relapse"),
                       RMSE =  c(naive_rmse, b_HLA_match_rmse, b_antigen_rmse, b_relapse_rmse)) %>%
  mutate(RMSE = sprintf("%0.4f", RMSE))

kable(results_table, format = "markdown")
```

Model_Type	RMSE
Naive RMSE	843.5371
HLA_match	840.7002
HLA_match + antigen	828.0364
HLA_match + antigen + relapse	804.4859

We continue to improve the predictive ability of our model, and have now reduced RMSE by 39.0511722 over the naive model. Now we can incorporate the time to acute GVHD variable, which refers to the time elapsed after transplant to development of acute graft versus host disease (GvHD) stage III or IV.

```
# Adding time to acute GVHD

b_time_to_acute_GvHD <-train %>% left_join(b_relapse, by = "relapse") %>%
  left_join(b_antigen, by = "antigen") %>%
  left_join(b_HLA_match, by = "HLA_match")%>%
  group_by(time_to_acute_GvHD_III_IV) %>%
  summarize(b_time_to_acute_GvHD = mean(survival_time - mu - b_HLA_match - b_antigen - b_relapse))

predicted_b_time_to_acute_GvHD <- test %>%
  left_join(b_HLA_match, by='HLA_match') %>%
  left_join(b_antigen, by='antigen')%>%
  left_join(b_relapse, by='relapse')%>%
  left_join(b_time_to_acute_GvHD, by = "time_to_acute_GvHD_III_IV")%>%
  mutate(predictions = mu + b_HLA_match + b_antigen + b_relapse + b_time_to_acute_GvHD) %>% .$predictio

b_time_to_acute_GvHD_rmse<-RMSE(test$survival_time, predicted_b_time_to_acute_GvHD, na.rm = TRUE)
b_time_to_acute_GvHD_rmse
```

```
## [1] 789.2042
```

```
results_table <-tibble(Model_Type = c("Naive RMSE", "HLA_match",
                                      "HLA_match + antigen",
                                      "HLA_match + antigen + relapse",
                                      "HLA_match + antigen + relapse + time to acute GvHD"),
                       RMSE =  c(naive_rmse, b_HLA_match_rmse, b_antigen_rmse, b_relapse_rmse,b_time_to
  mutate(RMSE = sprintf("%0.4f", RMSE))

kable(results_table, format = "markdown")
```

Model_Type	RMSE
Naive RMSE	843.5371
HLA_match	840.7002
HLA_match + antigen	828.0364
HLA_match + antigen + relapse	804.4859
HLA_match + antigen + relapse + time to acute GvHD	789.2042

By incorporating more variables as predictors, we've been able to continue to reduce RMSE by 54.3328722 compared to the naive model. However, adding single predictor variables to the model one at a time in this fashion is time consuming and may involve significant trial and error. As an alternative, implementing more sophisticated machine learning approaches such as k-Nearest Neighbor, Gradient Boosting Machine, and Random Forest may leverage all of the available data and generate models with improved predictive accuracy and reduced RMSE.

We will begin by evaluating the k-Nearest Neighbors (kNN) approach:

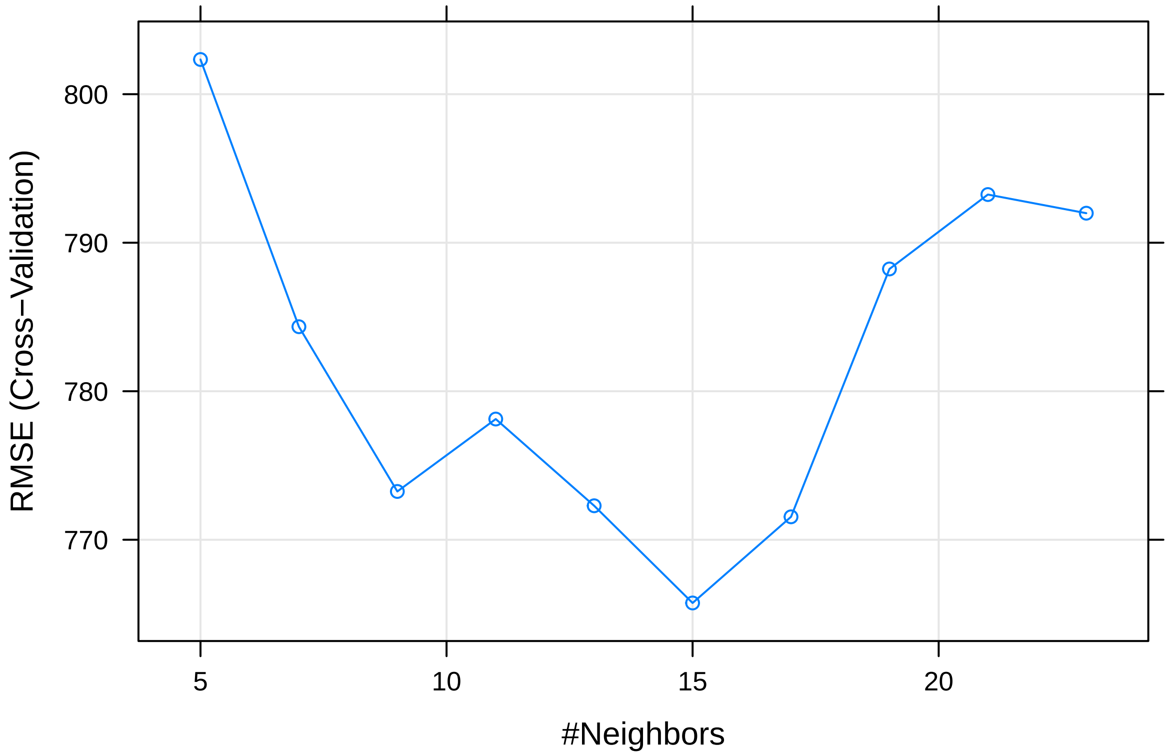
```
# Fit the model on the training set
set.seed(1701)
model <- train(
  survival_time ~., data = train, method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 10
)
# Plot model error RMSE vs different values of k
plot(model)
# Best tuning parameter k that minimize the RMSE
model$bestTune
```

```
##    k
## 6 15
```

```
# Make predictions on the test data
predictions <- model %>% predict(test)
head(predictions)
```

```
## [1] 1147.067 1189.200 1266.733 1067.733 1453.067 1464.533
```

```
# Compute the prediction error RMSE
kNN_rmse<-RMSE(predictions, test$survival_time)
```



The kNN model suggests that 15 is the optimal number of neighbors (k) to include in the model. However, when we look at our results we see that this model has failed to improve RMSE, and instead has taken us a step backwards in terms of predictive accuracy:

Model_Type	RMSE
Naive RMSE	843.5371
HLA_match	840.7002
HLA_match + antigen	828.0364
HLA_match + antigen + relapse	804.4859
HLA_match + antigen + relapse + time to acute GvHD	789.2042
kNN	803.7338

We see that use of the kNN model actually leads to a higher RMSE, and therefore a poorer prediction. There are several reasons this may happen. First, kNN models are known to suffer from overfitting when the number of neighbors (k) is very small, as may be the case for some of the more noisy predictor variables. Second, kNN considers all features equally when computing distances, which can lead to poorer prediction if some variables are not related to the variable that the model is trying to predict. And finally, the limited size of this training set (only 131 individuals) may mean that kNN does not have enough data to accurate estimate the underlying distribution.

Although this model did not perform as well, we can attempt to use a different method: Gradient Boosting Machine, or GBM. This machine learning algorithm builds an ensemble of weak learners (typically decision trees) in a sequential manner to improve predictive performance. It's known for its high predictive accuracy and robustness against overfitting. We can apply this model to our data to see if it allows us to achieve a better RMSE value.

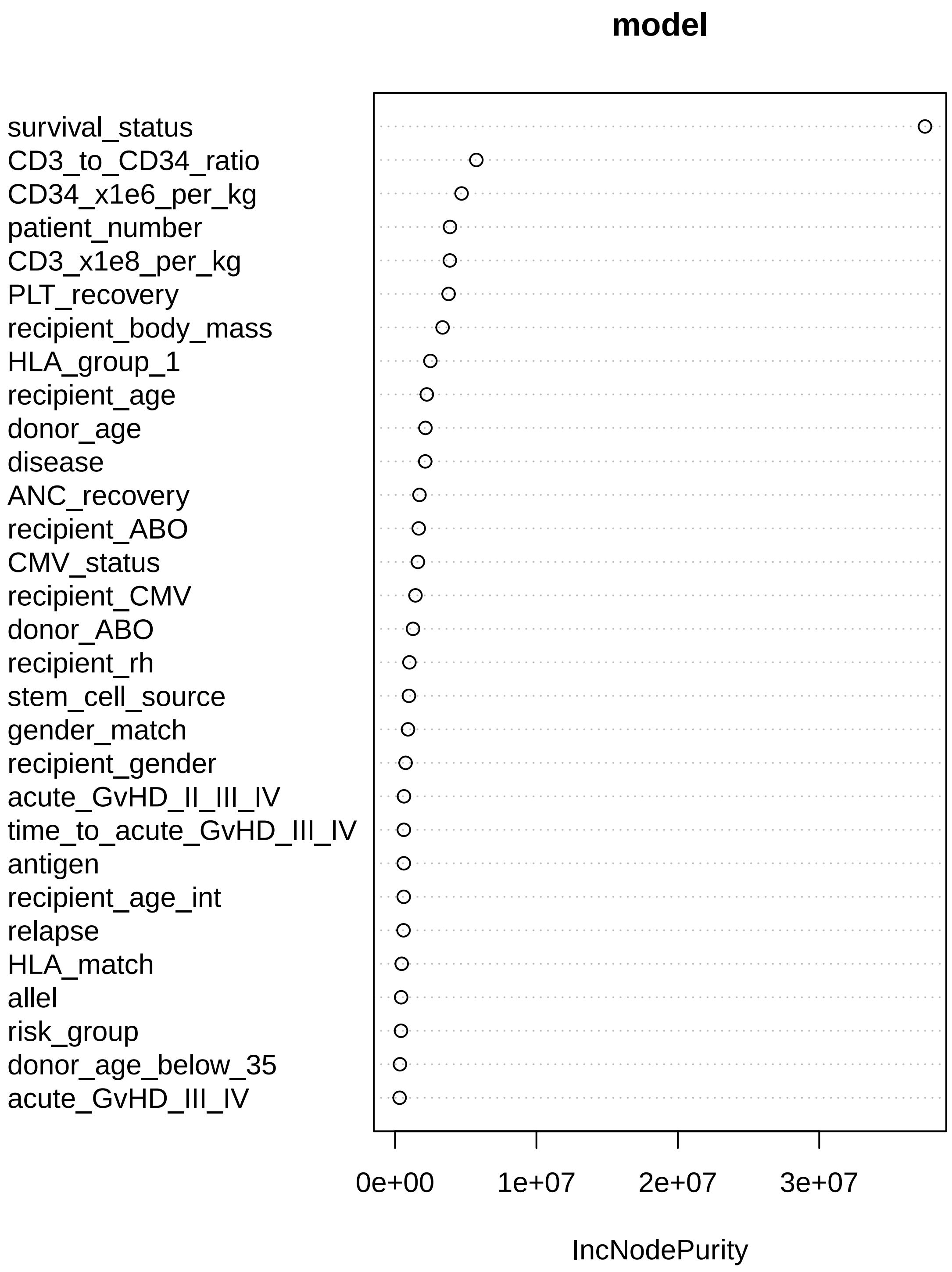
Model_Type	RMSE
Naive RMSE	843.5371
HLA_match	840.7002
HLA_match + antigen	828.0364
HLA_match + antigen + relapse	804.4859
HLA_match + antigen + relapse + time to acute GvHD	789.2042
kNN	803.7338
GBM	547.4943

This model performed much better than kNN, and also better than all of our previous linear models. This decreased RMSE by 296.0428014 days compared to the naive model, which has already achieved our stated objective of reducing RMSE by six month over the naive approach. However, we can attempt to use one more machine learning approach- random forest - to see if we can further improve upon this. Random forest is an ensemble method that works by building multiple decision trees during training and outputting the mean prediction of the individual trees. We can implement this approach to see if it improves our predictive accuracy.

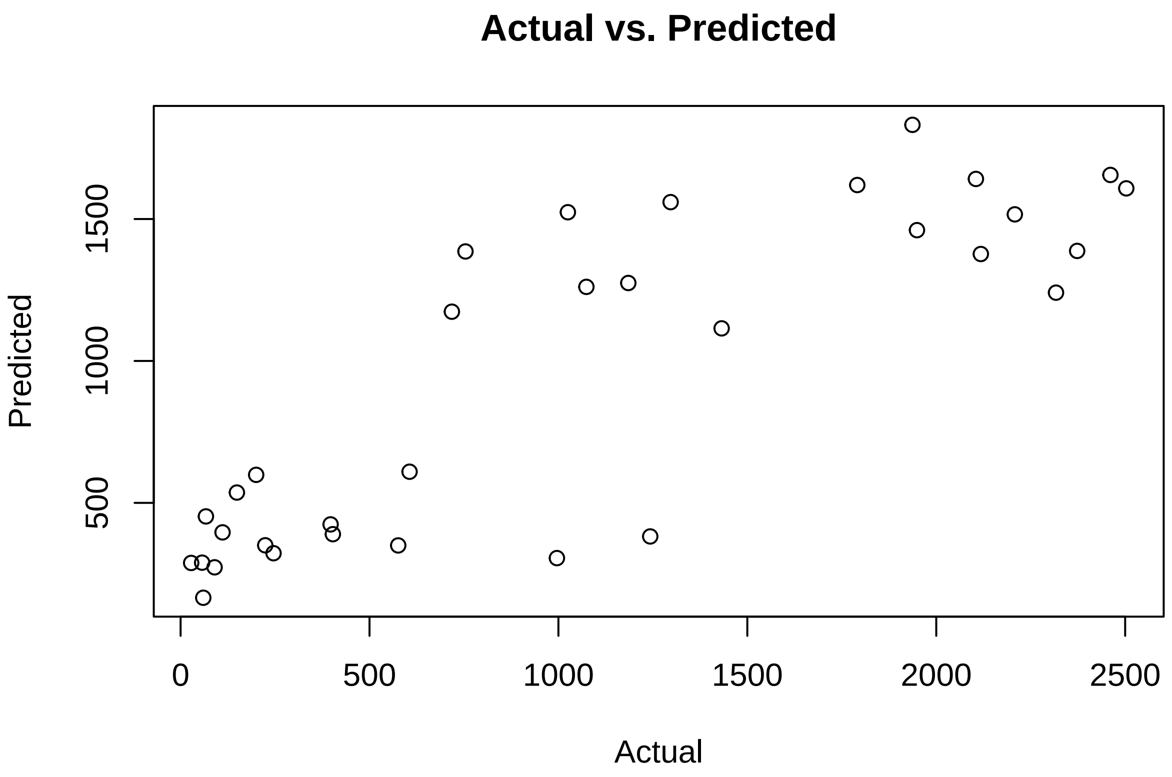
```r
# Use the Random Forest Approach
model <- randomForest(survival_time ~ ., data = train, ntree = 1000)
predictions <- predict(model, newdata = test)
```

Using this approach also allows us to generate a plot of relative importance of the predictor variables being used in the model:

```r
varImpPlot(model)
```



We can also plot the predicted versus actual survival time values:



We can now evaluate the RMSE generated by the random forest approach:

Model_Type	RMSE
Naive RMSE	843.5371
HLA_match	840.7002
HLA_match + antigen	828.0364
HLA_match + antigen + relapse	804.4859
HLA_match + antigen + relapse + time to acute GvHD	789.2042
kNN	803.7338
randomForest Model	497.4907

We can see that the random forest approach resulted in a substantial improvement in our ability to predict survival time. The RMSE of the random forest model is more accurate by 346.0463749 days, nearly a full year. This is sufficient to achieve our primary objective of achieving an RMSE that is more accurate than the naive model by more than six months.

Conclusion

Ultimately, the random forest model yielded the lowest RMSE, at a value of 497.4907064. Although this achieves the goal set forth for this project, it is likely that this RMSE value could be further improved by expansion of the dataset through the inclusion of more individuals and more predictor variables, which may be achieved by ongoing clinical research. In addition, alternative modeling approaches could be leveraged to further improve model predictions, including the use of matrix factorization. Although larger datasets and more complex models may be more computationally intensive, they have the potential to yield valuable insight that will help achieve better outcomes for pediatric cancer patients. As the fields of cancer treatment and machine learning continues to advance, it is likely that more and more sophisticated approaches will be developed and allow for much more accurate predictions of patient outcomes, which will in turn greatly improve treatment processes.

Literature Cited

Russell, H., Hord, J., Orr, C. J., & Moerdler, S. (2024). Child Health and the Pediatric Hematology-Oncology Workforce: 2020–2040. Pediatrics, 153(Supplement 2).

Siegel, R. L., Miller, K. D., Fuchs, H. E., & Jemal, A. (2022). Cancer statistics, 2022. CA: a cancer journal for clinicians, 72(1).

Sikora, M., Wróbel, Ł., and Gudyś, A. (2020). Bone marrow transplant: children. UCI Machine Learning Repository. https://doi.org/10.24432/C5NP6Z.

Sweeney, S. M., Hamadeh, H. K., Abrams, N., Adam, S. J., Brenner, S., Connors, D. E., … & Srivastava, S. (2023). Challenges to using big data in cancer. Cancer research, 83(8), 1175-1182.