Kaggle Shelter Animal Outcomes

Zach Tallevast, Hannah Snoke, Joe Helmer, Haley Bluhm

5/9/2018

# Competition Description

Every year, approximately 7.6 million companion animals end up in US shelters. Many animals are given up as unwanted by their owners, while others are picked up after getting lost or taken out of cruelty situations. Many of these animals find forever families to take them home, but just as many are not so lucky. 2.7 million dogs and cats are euthanized in the US every year. Using a dataset provided by the Austin Animal Shelter and ASPCA, intake information including breed, color, sex, and age are provided and we are tasked to predict the outcome for each animal. #Goal This dataset can help understand trends in animal shelter outcomes. These insights could help animal shelters focus their energy on specific animals who need a litte extra help finding a new home. #Loading the Dataset (Shelter Animal Outcomes)

setwd("/Users/zachtallevast/Box Sync/Kaggle Shelter Animal Outcomes/")  
full <- read.csv("Shelter Outcomes TRAIN.csv", header=T)  
#See First 3 Lines  
head(full, n=3L)

## AnimalID Name DateTime OutcomeType OutcomeSubtype  
## 1 A671945 Hambone 2/12/14 18:22 Return\_to\_owner   
## 2 A656520 Emily 10/13/13 12:44 Euthanasia Suffering  
## 3 A686464 Pearce 1/31/15 12:28 Adoption Foster  
## AnimalType SexuponOutcome AgeuponOutcome Breed  
## 1 Dog Neutered Male 1 year Shetland Sheepdog Mix  
## 2 Cat Spayed Female 1 year Domestic Shorthair Mix  
## 3 Dog Neutered Male 2 years Pit Bull Mix  
## Color  
## 1 Brown/White  
## 2 Cream Tabby  
## 3 Blue/White

#Show Columns Information  
str(full)

## 'data.frame': 26729 obs. of 10 variables:  
## $ AnimalID : Factor w/ 26729 levels "A006100","A047759",..: 5420 1604 11614 10239 3502 7396 17152 18183 5373 7574 ...  
## $ Name : Factor w/ 6375 levels ""," Joanie"," Mario",..: 2353 1849 4442 1 1 1837 2763 1 3458 1 ...  
## $ DateTime : Factor w/ 22918 levels "1/1/14 10:50",..: 9503 2487 1630 17531 5047 13126 11893 13464 10627 15017 ...  
## $ OutcomeType : Factor w/ 5 levels "Adoption","Died",..: 4 3 1 5 5 5 5 5 1 1 ...  
## $ OutcomeSubtype: Factor w/ 17 levels "","Aggressive",..: 1 17 8 14 14 14 14 14 1 13 ...  
## $ AnimalType : Factor w/ 2 levels "Cat","Dog": 2 1 2 1 2 2 1 1 2 2 ...  
## $ SexuponOutcome: Factor w/ 6 levels "","Intact Female",..: 4 5 4 3 4 2 3 6 5 5 ...  
## $ AgeuponOutcome: Factor w/ 45 levels "","0 years","1 day",..: 7 7 23 27 23 4 27 27 34 7 ...  
## $ Breed : Factor w/ 1380 levels "Abyssinian Mix",..: 1222 641 1067 641 915 370 641 641 46 365 ...  
## $ Color : Factor w/ 366 levels "Agouti","Agouti/Brown Tabby",..: 131 168 87 43 275 37 64 105 251 319 ...

# Loading Packages

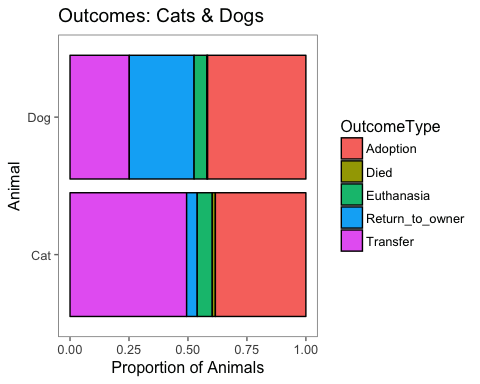
#Libraries Used in Analysis  
library(plyr)  
library(ggplot2)  
library(ggthemes)  
library(dplyr)  
library(lubridate)  
library(rpart)  
library(randomForest)  
library(Hmisc)

# Data Analysis

library(dplyr)  
library(ggplot2)  
library(ggthemes)  
summary(full$AnimalType)

## Cat Dog   
## 11134 15595

#There are 11,134 Cats and 15,595 Dogs in this dataset.   
  
#Create a Plot which shows the animal type, outcome type, and number in each category.   
outcomes <- full[1:26728, ] %>% group\_by(AnimalType, OutcomeType) %>%  
 summarise(num\_animals = n())  
ggplot(outcomes, aes(x=AnimalType, y=num\_animals, fill=OutcomeType))+  
 geom\_bar(stat='identity', position = 'fill', color='black')+  
 coord\_flip()+  
 labs(y='Proportion of Animals', x='Animal', title = 'Outcomes: Cats & Dogs')+  
 theme\_few()



#Notice that cats are twice as likely to be transferred to a new owner and dogs are four times as likely to be returned to their owner than cats.

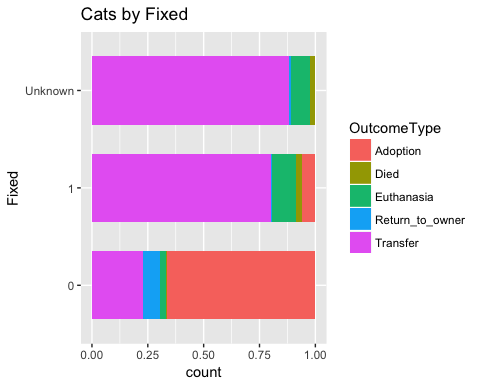
# Creation of Features: Fixed and Sex

Create feature called “Fixed” that distinguishes if the animal is intact or unknown. Intact means that the animal has not been fixed.  
Create a feature called “Sex” to provide solely the sex of the animal.

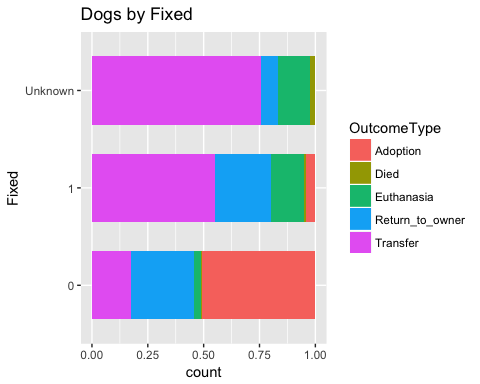
summary(full$SexuponOutcome)

## Intact Female Intact Male Neutered Male Spayed Female   
## 1 3511 3525 9779 8820   
## Unknown   
## 1093

full$SexuponOutcome = as.character(full$SexuponOutcome)  
full$SexuponOutcome <- ifelse(nchar(full$SexuponOutcome)==0, 'Unknown', full$SexuponOutcome)  
full$SexuponOutcome = as.factor(full$SexuponOutcome)  
full$Fixed <- ifelse(grepl('Intact', full$SexuponOutcome), 1, ifelse(grepl('Unknown', full$SexuponOutcome), 'Unknown', 0))  
full$Sex <- ifelse(grepl('Male', full$SexuponOutcome), 'Male', ifelse(grepl('Unknown', full$Sex), 'Unknown', 'Female'))  
#Make as.factors   
full$Fixed = as.factor(full$Fixed)  
full$Sex = as.factor(full$Sex)  
#Make A Plot of cats and dogs which shows how animal outcomes change when animal is fixed.   
cats = full[full$AnimalType == "Cat",]  
dogs = full[full$AnimalType == "Dog",]  
  
ggplot(cats, aes(x=Fixed, fill = OutcomeType))+  
 geom\_bar(stat="count", position = "fill", width = 0.7)+  
 coord\_flip()+  
 ggtitle("Cats by Fixed")



ggplot(dogs, aes(x=Fixed, fill = OutcomeType))+  
 geom\_bar(stat="count", position = "fill", width = 0.7)+  
 coord\_flip()+  
 ggtitle("Dogs by Fixed")

 Notice that once dogs are fixed, they are more adoptable. #Creation of Features: Breed (SimpleBreed) There are 1380 breeds in our dataset. For Prediction, we need to group the breeds into buckets. The way we did this was we cut the second breed in the mix out of the breed name, and took out the word mix.

nlevels(factor(full$Breed))

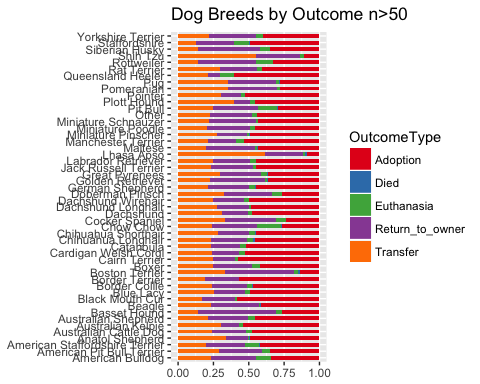
## [1] 1380

#First lets take out Mix Breeds and denote them as a feature called Mix.   
full$Mix <-ifelse(grepl('Mix', full$Breed),1,0)  
full$Mix = as.factor(full$Mix)  
#Simply Number of Breeds. Split on "/" and Remove "Mix"  
full$Breed = as.character(full$Breed)  
full$SimpleBreed <-sapply(full$Breed, function(x) gsub(' Mix', '', strsplit(x, split = '/')[[1]][1]))  
full$SimpleBreed = as.factor(full$SimpleBreed)  
nlevels(factor(full$SimpleBreed))

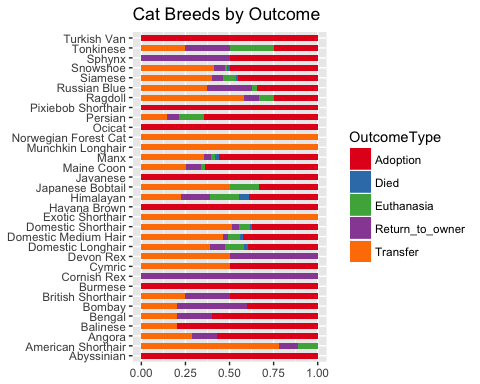
## [1] 220

Now we have 220 breeds. Lets Plot Breeds by Outcome for Cats and For Dogs

dogs = full[full$AnimalType == "Dog",]  
  
 dogs %>% count(SimpleBreed) %>% arrange(desc(n)) %>% filter(n>50) -> dog\_colors  
 dogs$SimpleBreed = as.character(dogs$SimpleBreed)  
 dogs$SimpleBreed[!(dogs$SimpleBreed %in% dog\_colors$SimpleBreed)] <-"Other"  
 ggplot(dogs, aes(x=SimpleBreed, fill=OutcomeType))+  
 geom\_bar(stat="count", position = "fill", width = 0.6)+  
 ggtitle("Dog Breeds by Outcome n>50")+  
 coord\_flip()+  
 scale\_fill\_brewer(palette = "Set1")+  
 theme(axis.title = element\_blank())



filter(full, AnimalType == "Cat" ) %>%   
 ggplot(aes(x=SimpleBreed, fill=OutcomeType))+  
 geom\_bar(stat="count", position = "fill", width = 0.6)+  
 ggtitle("Cat Breeds by Outcome")+  
 coord\_flip()+  
 scale\_fill\_brewer(palette = "Set1")+  
 theme(axis.title = element\_blank())



# Creation of Features: (Missing Values)

Now lets deal with missing values for Name. If the animal has no name, call them “Nameless”. Create a feature that indicates if the animal has a name or not.

full$Name = as.character(full$Name)  
full$Name <-ifelse(nchar(full$Name)==0, "Nameless", full$Name)  
  
full$HasAName[full$Name == 'Nameless'] <- 0   
full$HasAName[full$Name != 'Nameless'] <-1  
full$HasAName = as.factor(full$HasAName)

# Creation of Features: (Age)

Display age in Days across all animals. To do this, create a mulitiplier that corresponds to “Month”, “Days”, “Years”, “Weeks”, etc.

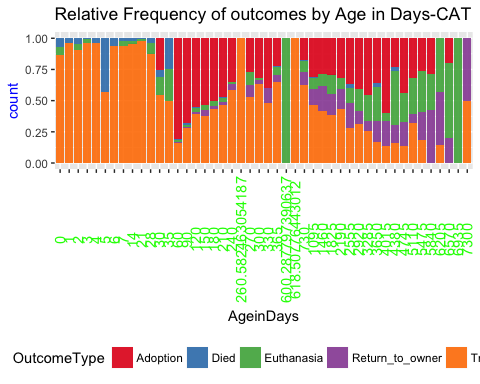
#Create a Time Value i.e. 1 or 12 or etc.   
#Create a Unit of Time i.e. "Year","Month", etc.   
full$AgeuponOutcome = as.character(full$AgeuponOutcome)  
full$TimeValue <-sapply(full$AgeuponOutcome, function(x) strsplit(x, split = ' ')[[1]][1])  
full$UnitofTime <- sapply(full$AgeuponOutcome, function(x) strsplit(x, split = ' ')[[1]][2])  
#Rid the plurals on the Units of Time   
full$UnitofTime <- gsub('s', '', full$UnitofTime)  
full$TimeValue <- as.numeric(full$TimeValue)  
#Standardize across ages for AgeinDays  
Multiplier <- ifelse(full$UnitofTime == "day", 1,   
 ifelse(full$UnitofTime == "week", 7,  
 ifelse(full$UnitofTime == "month", 30,   
 ifelse(full$UnitofTime == "year", 365, NA))))  
full$AgeinDays <- full$TimeValue \* Multiplier  
summary(full$AgeinDays)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 0.0 60.0 365.0 794.1 1095.0 7300.0 18

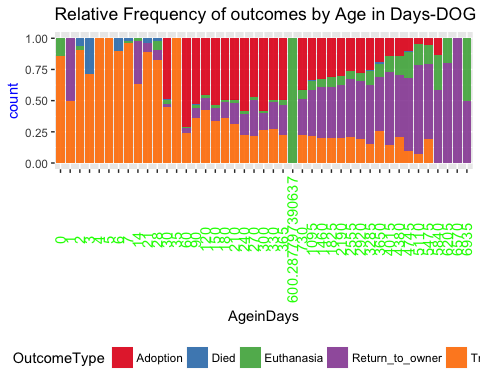
full$AgeinDays = as.factor(full$AgeinDays)  
#sum(is.na(full$AgeinDays))  
library(rpart)  
#Now there are 18 missing ages  
full$AgeinDays = as.factor(full$AgeinDays)  
full$AgeinDays = as.numeric(as.character(full$AgeinDays))  
#sum(is.na(full$AgeinDays))  
age\_fit = rpart(AgeinDays ~ AnimalType + Sex+ Fixed + SimpleBreed+ HasAName,   
 data = full[!is.na(full$AgeinDays), ],  
 method = 'anova')  
#Put the predicted values into the missing values   
full$AgeinDays[is.na(full$AgeinDays)] = predict(age\_fit, full[is.na(full$AgeinDays),])  
#sum(is.na(full$AgeinDays))  
#table(full$AgeinDays)  
full$AgeinDays = as.factor(full$AgeinDays)

Now lets plot the Ages by outcome type

cats = full[full$AnimalType == "Cat",]  
dogs = full[full$AnimalType == "Dog",]  
  
#Cats Plot   
ggplot(cats, aes(x=AgeinDays, fill=OutcomeType))+  
 geom\_bar(stat = "count", position = "fill", alpha = 0.9)+  
 scale\_fill\_brewer(palette = "Set1")+  
 ggtitle("Relative Frequency of outcomes by Age in Days-CAT")+  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, size=11, color = "green"),   
 legend.position = "bottom",   
 axis.title.y = element\_text(angle = 90, color = "blue"))



#Dogs Plot  
ggplot(dogs, aes(x=AgeinDays, fill=OutcomeType))+  
 geom\_bar(stat = "count", position = "fill", alpha = 0.9)+  
   
 scale\_fill\_brewer(palette = "Set1")+  
 ggtitle("Relative Frequency of outcomes by Age in Days-DOG")+  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, size=11, color = "green"),   
 legend.position = "bottom",   
 axis.title.y = element\_text(angle = 90, color = "blue"))



#head(full, n=5)

# Creation of Features:(Hypo-Allergenic)

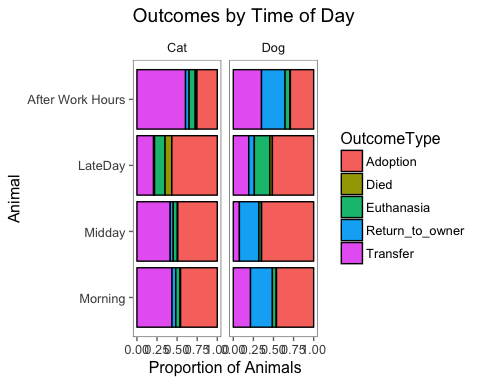
Cats =full[full$AnimalType == "Cat",]  
Dogs = full[full$AnimalType == "Dog",]  
Dogs$Hypo = Dogs$Breed  
Dogs = mutate(Dogs, Hypo = ifelse(Dogs$Hypo == 'Labrador' | Dogs$Hypo == 'Golden Retriever' | Dogs$Hypo == 'Siberian Husky', 0, Dogs$Hypo))  
  
Dogs$Hypo = ifelse(grepl('Poodle', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Yorkshire', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Havanese', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Maltese', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Shih Tzu', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Schnauzer', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Basenji', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Silky Terrier', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Irish Water Spaniel', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Terrier', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Terr', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Afghan Hound', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Water Spaniel', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Spanish Water', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Water Dog', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Bichon Frise', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl(1, Dogs$Hypo), 1, 0)  
  
Cats$Hypo = Cats$Breed  
Cats$Hypo = ifelse(grepl('Siberian', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Balinese', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Bengal', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Burmese', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Siberian', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Cornish Rex', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Devon Rex', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Javanese', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Ocicat', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Russian Blue', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Siamese', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Sphynx', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl(1, Cats$Hypo), 1, 0)  
full = bind\_rows(Cats, Dogs)

# Creation of Features: (Time and Date Variables)

full$DateTime = as.character(full$DateTime)  
full$Date <- sapply(full$DateTime, function(x) strsplit(x,split = ' ') [[1]][1])  
full$Time <-sapply(full$DateTime, function(x) strsplit(x, split = ' ') [[1]][2])  
full$Date <-as.Date(full$Date, "%m/%d/%y")

## Warning in strptime(x, format, tz = "GMT"): unknown timezone 'zone/tz/  
## 2018c.1.0/zoneinfo/America/Chicago'

full$DateTime <-paste(full$Date, " ", full$Time)  
full$DateTime = as.character(full$DateTime)  
library(lubridate)  
full$Hour <- sapply(full$Time, function(x) strsplit(x, split = ':')[[1]][1])  
full$Hour = as.factor(full$Hour)  
#Establish a Time Of Day Covariate   
full$Hour = as.numeric(full$Hour)  
full$TimeOfDay <- ifelse(full$Hour > 5 & full$Hour < 11, 'Morning',  
 ifelse(full$Hour > 10 & full$Hour <16, 'Midday',  
 ifelse(full$Hour >15 & full$Hour <20, 'LateDay',  
 'After Work Hours')))  
full$TimeOfDay <- factor(full$TimeOfDay, levels = c('Morning', 'Midday', 'LateDay', 'After Work Hours'))  
full$TimeOfDay = as.factor(full$TimeOfDay)  
#Plot Time of Day by Outcomes   
daytimes <- full[1:26729, ] %>% group\_by(AnimalType, TimeOfDay, OutcomeType) %>%  
 summarise(num\_animals = n())  
ggplot(daytimes, aes(x=TimeOfDay, y=num\_animals, fill=OutcomeType, na.rm = TRUE))+  
 geom\_bar(stat = 'identity', position = 'fill', color = 'black')+  
 facet\_wrap(~AnimalType)+  
 coord\_flip()+  
 labs(y = 'Proportion of Animals', x = 'Animal', title = 'Outcomes by Time of Day')+  
 theme\_few()



#head(full, n=5)

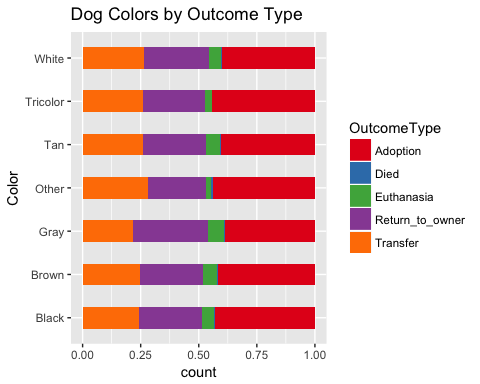
# Creation of Features: (Color)

There are 366 Colors in the color space. We will subset this to be 7 colors. Does color play a role in the outcome of an animal?

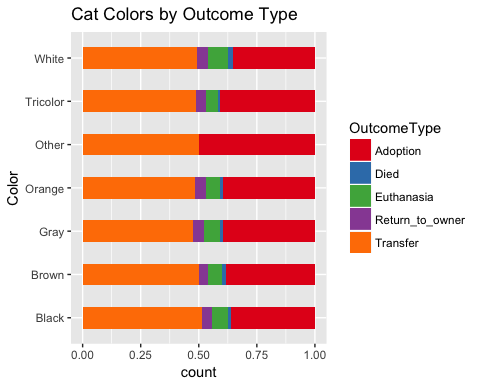
#Is the animal Multicolored?   
full$Multicolor = full$Color  
full$Multicolor = ifelse(grepl('/', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Tricolor', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Calico', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Tabby', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Point', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Brindle', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Tiger', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Merle', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Tick', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Tortie', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Torbie', full$Multicolor), 'Yes', full$Multicolor)  
full$Multicolor = ifelse(grepl('Yes', full$Multicolor), 'Yes', 'No')  
full$Multicolor = ifelse(grepl('Yes', full$Multicolor), 1, 0)  
  
full$Color = as.character(full$Color)  
full$Color <- sapply(full$Color,   
 function(x) strsplit(x, split = '/')[[1]][1])  
full$Color = ifelse(grepl('Blue Tick', full$Color), 'Brown', full$Color)  
full$Color <- sapply(full$Color,   
 function(x) strsplit(x, split = ' ')[[1]][1])  
full$Color = ifelse(grepl('Apricot', full$Color), 'Orange', full$Color)  
full$Color = ifelse(grepl('Agouti', full$Color), 'Gray', full$Color)  
full$Color = ifelse(grepl('Chocolate', full$Color), 'Brown', full$Color)  
full$Color = ifelse(grepl('Ruddy', full$Color), 'Brown', full$Color)  
full$Color = ifelse(grepl('Calico', full$Color), 'Tricolor', full$Color)  
full$Color = ifelse(grepl('Flame', full$Color), 'White', full$Color)  
full$Color = ifelse(grepl('Gold', full$Color), 'Yellow', full$Color)  
full$Color = ifelse(grepl('Lilac', full$Color), 'Gray', full$Color)  
full$Color = ifelse(grepl('Red', full$Color), 'Brown', full$Color)  
full$Color = ifelse(grepl('Pink', full$Color), 'Tan', full$Color)  
full$Color = ifelse(grepl('Seal', full$Color), 'Brown', full$Color)  
full$Color = ifelse(grepl('Blue', full$Color), 'Gray', full$Color)  
full$Color = ifelse(grepl('Lynx', full$Color), 'Gray', full$Color)  
full$Color = ifelse(grepl('Fawn', full$Color), 'Yellow', full$Color)  
full$Color = ifelse(grepl('Buff', full$Color), 'Orange', full$Color)  
full$Color = ifelse(grepl('Sable', full$Color), 'Brown', full$Color)  
full$Color = ifelse(grepl('Liver', full$Color), 'Gray', full$Color)  
full$Color = ifelse(grepl('Silver', full$Color), 'Gray', full$Color)  
full$Color = ifelse(grepl('Torbie', full$Color), 'Tricolor', full$Color)  
full$Color = ifelse(grepl('Tortie', full$Color), 'Tricolor', full$Color)  
full$Color = ifelse(grepl('Yellow', full$Color), 'Tan', full$Color)  
full = mutate(full, Color = ifelse(full$Color == 'Cream' & full$AnimalType == 'Cat', 'Orange', full$Color))  
full = mutate(full, Color = ifelse(full$Color == 'Cream' & full$AnimalType == 'Dog', 'Tan', full$Color))  
table(full$Color)

##   
## Black Brown Gray Orange Tan Tricolor White   
## 6646 7185 2722 1934 2422 2391 3429

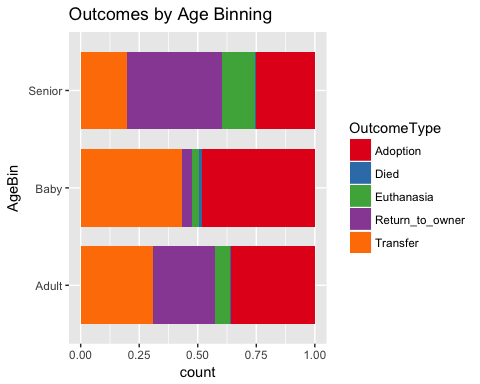
#Plot Color by Outcome Type   
cats =full[full$AnimalType == "Cat",]  
dogs = full[full$AnimalType == "Dog",]  
dogs %>% count(Color) %>% arrange(desc(n)) %>%  
 filter(n >400) -> dog\_colors  
dogs$Color = as.character(dogs$Color)  
dogs$Color[!(dogs$Color %in% dog\_colors$Color)] <- "Other"  
ggplot(dogs, aes(x=Color, fill=OutcomeType))+  
 geom\_bar(stat = "count", position = "fill", width = 0.5)+  
 coord\_flip()+  
 ggtitle ("Dog Colors by Outcome Type")+  
 scale\_fill\_brewer(palette = "Set1")



cats %>% count(Color) %>% arrange(desc(n)) %>%  
 filter(n >400) -> cat\_colors  
cats$Color = as.character(cats$Color)  
cats$Color[!(cats$Color %in% cat\_colors$Color)] <- "Other"  
ggplot(cats, aes(x=Color, fill=OutcomeType))+  
 geom\_bar(stat = "count", position = "fill", width = 0.5)+  
 coord\_flip()+  
 ggtitle ("Cat Colors by Outcome Type")+  
 scale\_fill\_brewer(palette = "Set1")

 #Creation of Features: (Age Binning) Create a feature that distinguishes between Senior, Adult, and Puppy/Kitten aged animals. In our analysis, a Puppy/Kitten is Under 1 years old, and a Senior animal is 7 or older.

full$AgeinDays = as.numeric(as.character(full$AgeinDays))  
full$AgeBin[full$AgeinDays <= 365] <- "Baby"  
full$AgeBin[full$AgeinDays < 2555 & full$AgeinDays >= 365] <- "Adult"  
full$AgeBin[full$AgeinDays >=2555] <- "Senior"  
full$AgeBin = as.factor(full$AgeBin)  
  
#Make A Plot that Shows Age Binning versus Outcome Type   
ggplot(data=subset(full, !is.na(full$AgeBin)), aes(x=AgeBin, fill = OutcomeType))+  
 geom\_bar(stat = "count", position = "fill", width = 0.8)+  
 coord\_flip()+  
 ggtitle("Outcomes by Age Binning")+  
 scale\_fill\_brewer(palette = "Set1")

 #Creation of Features: (Holiday Adoption Season) According to the ASPCA, more animals are adopted in the adoption season between October 1st through December 31. October is National Adopt a Dog Month

full$Month <- sapply(full$DateTime, function(x) strsplit(x,split = '-') [[1]][2])  
full$Month <-sapply(full$Month, function(x) strsplit(x,split = ' ') [[1]][1])  
full$HolidaySeason <- ifelse(full$Month <10, 'Non-Holiday', 'Holiday')  
full$Month = as.factor(full$Month)  
full$HolidaySeason = as.factor(full$HolidaySeason)  
#head(full, n=5)

# Creation of Features: (Top Ten Adopted Breeds)

According to data provided by the ASPCA, we established an indicator variable to distinguish if the breed is a top ten adopted breed for dogs and top five for cats.

full$TopAdopted <-ifelse(full$SimpleBreed == 'Bull Dog', 'Top 10',   
ifelse(full$SimpleBreed == 'Beagle', 'Top 10',ifelse(full$SimpleBreed == 'Bull Terrier', 'Top 10',  
 ifelse(full$SimpleBreed == 'Collie', 'Top 10',ifelse(full$SimpleBreed == 'Newfoundland', 'Top 10',  
 ifelse(full$SimpleBreed == 'Vizsla', 'Top 10',ifelse(full$SimpleBreed == 'Irish Setter', 'Top 10',  
 ifelse(grepl( 'Poodle',full$SimpleBreed), 'Top 10',ifelse(full$SimpleBreed == 'Labrador Retriever', 'Top 10',  
 ifelse(full$SimpleBreed == 'Golden Retreiver', 'Top 10', ifelse(full$SimpleBreed == 'Persian', 'Top 5',   
 ifelse(full$SimpleBreed == 'American Shorthair', 'Top 5', ifelse(full$SimpleBreed == 'Maine Coon', 'Top 5',   
 ifelse(full$SimpleBreed == 'Siamese', 'Top 5', ifelse(full$SimpleBreed == 'Abyssinian', 'Top 5', 'Not Top 10')))))))))))))))  
full$TopAdopted <- as.factor(full$TopAdopted)

# Creation of Features: (Top Ten Shelter Animal Breeds)

Although we have established a feature that teases out the top 10 adoptable breeds, in animal shelters the breeds may not fit this list. i.e. certain breeds may be more common in animal shelters than others. According to the ASPCA, we teases out the top 10 dog breeds in shelters and the top 5 cat breeds in shelters.

full$TopTenSheltered <-ifelse(full$SimpleBreed == 'Bulldog', 'Top 10',   
ifelse(full$SimpleBreed == 'Beagle', 'Top 10',ifelse(full$SimpleBreed == 'Boxer', 'Top 10',  
 ifelse(full$SimpleBreed == 'Chihuahua Shorthair', 'Top 10',ifelse(full$SimpleBreed == 'Dachshund', 'Top 10',  
 ifelse(full$SimpleBreed == 'German Shepherd', 'Top 10',ifelse(full$SimpleBreed == 'Labrador Retreiver', 'Top 10',  
 ifelse(full$SimpleBreed == 'Pit Bull', 'Top 10',ifelse(full$SimpleBreed == 'Bull Terrier', 'Top 10',  
 ifelse(full$SimpleBreed == 'Rottweiler', 'Top 10', ifelse(full$SimpleBreed =='Siamese', 'Top 5',   
 ifelse(full$SimpleBreed == 'Persian', 'Top 5',ifelse(full$SimpleBreed == 'Maine Coon', 'Top 5',   
 ifelse(full$SimpleBreed == 'Ragdoll', 'Top 5', ifelse(full$SimpleBreed == 'Abyssinian', 'Top 5','Not Top')))))))))))))))  
full$TopTenSheltered <- as.factor(full$TopTenSheltered)

# Creation of Features: (Average Lifespan by Breed)

setwd("/Users/zachtallevast/Box Sync/Kaggle Shelter Animal Outcomes/")  
Lifespan <-read.csv("Breed Lifespans.csv", header=T)  
Lifespan = unique(Lifespan)  
Lifespan$AVERAGE.LIFESPAN..YEARS. = ifelse(Lifespan$Breed == 'Yorkshire Terrier', 13, Lifespan$AVERAGE.LIFESPAN..YEARS.)  
Lifespan$AVERAGE.LIFESPAN..YEARS. = ifelse(Lifespan$Breed == 'Tonkinese', 15, Lifespan$AVERAGE.LIFESPAN..YEARS.)  
Lifespan = unique(Lifespan)  
Lifespan = Lifespan[-c(72),]  
full <-merge(x=full, y=Lifespan, by.x = "SimpleBreed", by.y = "Breed", all.x=TRUE, incomparables =='Unknown')  
full$AVERAGE.LIFESPAN..YEARS.[is.na(full$AVERAGE.LIFESPAN..YEARS.)] <- 10  
full$AVERAGE.LIFESPAN..YEARS. <- as.factor(full$AVERAGE.LIFESPAN..YEARS.)  
#sum(is.na(full$AVERAGE.LIFESPAN..YEARS.))

# Creation of Features: (Hunting)

setwd("/Users/zachtallevast/Box Sync/Kaggle Shelter Animal Outcomes/")  
Hunting <- read.csv("Hunting Dog.csv", header=T)  
Hunting = unique(Hunting)  
full <- merge(x=full, y = Hunting, by.x = "SimpleBreed", by.y ="Breed", all.x=TRUE, incomparables = 'Unknown')  
#head(full, n=5)

# Creation of Features:(Hypo-Allergenic)

Cats =full[full$AnimalType == "Cat",]  
Dogs = full[full$AnimalType == "Dog",]  
Dogs$Hypo = Dogs$Breed  
Dogs = mutate(Dogs, Hypo = ifelse(Dogs$Hypo == 'Labrador' | Dogs$Hypo == 'Golden Retriever' | Dogs$Hypo == 'Siberian Husky', 0, Dogs$Hypo))  
Dogs$Hypo = ifelse(grepl('Poodle', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Yorkshire', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Havanese', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Maltese', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Shih Tzu', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Schnauzer', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Basenji', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Silky Terrier', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Irish Water Spaniel', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Terrier', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Terr', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Afghan Hound', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Water Spaniel', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Spanish Water', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Water Dog', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl('Bichon Frise', Dogs$Hypo), 1, Dogs$Hypo)  
Dogs$Hypo = ifelse(grepl(1, Dogs$Hypo), 1, 0)  
  
Cats$Hypo = Cats$Breed  
Cats$Hypo = ifelse(grepl('Siberian', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Balinese', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Bengal', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Burmese', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Siberian', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Cornish Rex', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Devon Rex', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Javanese', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Ocicat', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Russian Blue', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Siamese', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl('Sphynx', Cats$Hypo), 1, Cats$Hypo)  
Cats$Hypo = ifelse(grepl(1, Cats$Hypo), 1, 0)  
full = bind\_rows(Cats, Dogs)

# Creation of Features: (Animal Size)

According to the ASPCA, we created a Size variable according to the typical breed’s size. Our sizes are: 1. Small <= 25 lbs 2. Medium 25 lbs < x < 50 lbs 3. Large 50 lbs < x < 100 lbs 4. XL >= 100 lbs

##   
## Large Medium Small Unknown XL   
## 7097 2244 5852 2 400

##   
## Large Medium XL   
## 38 11037 59

# Creation of Features: (Animal Hair Length)

## Length Class Mode   
## 26729 character character

# Creation of Features: (Near Death)

full$AgeinDays = as.numeric(as.character(full$AgeinDays))  
full$AVERAGE.LIFESPAN..YEARS. = as.numeric(full$AVERAGE.LIFESPAN..YEARS.)  
full$AboutToDie = (full$AgeinDays/365)  
full$AboutToDie = as.integer(full$AboutToDie)  
full$AboutToDie = full$AVERAGE.LIFESPAN..YEARS. - full$AboutToDie  
full$AboutToDie = ifelse(full$AboutToDie <= 3, 1, 0)

# Part 2: Preparing Data for Model Building

Create training and test sets. Training is 70% of the data. Test is 30% of the data. This is in accordance with Kaggles Guidelines

#Make Variables into Factors   
full$AVERAGE.LIFESPAN..YEARS. = as.factor(full$AVERAGE.LIFESPAN..YEARS.)  
full$Multicolor = as.factor(full$Multicolor)  
full$AboutToDie = as.factor(full$AboutToDie)  
full$AgeinDays = as.factor(full$AgeinDays)  
full$Hypo = as.factor(full$Hypo)  
full$Month = as.factor(full$Month)  
full$Hour = as.factor(full$Hour)  
full$Hair = as.factor(full$Hair)  
full$Size = as.factor(full$Size)  
full$Color = as.factor(full$Color)  
#Create training and test sets.   
Sample = sample(c(TRUE, FALSE), nrow(full), replace = T, prob = c(0.7, 0.3))  
Train.animals = full[Sample,]  
Test.animals = full[!Sample,]  
#head(full, n=10)  
#str(full)

# Part 3: Unsupervised Learning

#Convert categorical variables into numeric variables.   
library(dummies)

## dummies-1.5.6 provided by Decision Patterns

full$TimeOfDay.factor <-factor(full$TimeOfDay)  
full$TimeOfDay.factor <-as.numeric(full$TimeOfDay.factor)  
full$Hair.factor <-factor(full$Hair)  
full$Hair.factor <-as.numeric(full$Hair.factor)  
full$Size.factor <- factor(full$Size)  
full$Size.factor <-as.numeric(full$Size.factor)  
full$Mix.factor <-factor(full$Mix)  
full$Mix.factor <- as.numeric(full$Mix.factor)  
full$Fixed.factor <-factor(full$Fixed)  
full$Fixed.factor <- as.numeric(full$Fixed.factor)  
full$Sex.factor <-factor(full$Sex)  
full$Sex.factor <- as.numeric(full$Sex.factor)  
full$AVERAGE.LIFESPAN..YEARS.factor <-factor(full$AVERAGE.LIFESPAN..YEARS.)  
full$AVERAGE.LIFESPAN..YEARS.factor <-as.numeric(full$AVERAGE.LIFESPAN..YEARS.factor)  
full$AboutToDie.factor <-factor(full$AboutToDie)  
full$AboutToDie.factor <-as.numeric(full$AboutToDie.factor)  
full$Hypo.factor <-factor(full$Hypo)  
full$Hypo.factor <-as.numeric(full$Hypo.factor)  
full$AnimalType.factor <-factor(full$AnimalType)  
full$AnimalType.factor <- as.numeric(full$AnimalType.factor)  
full$HasAName.factor <-factor(full$HasAName)  
full$HasAName.factor <-as.numeric(full$HasAName.factor)  
full$Multicolor.factor <- factor(full$Multicolor)  
full$Multicolor.factor <- as.numeric(full$Multicolor.factor)  
  
numeric=full[,c(36,37,38,39,40,41,42,43,44)]  
  
# Supply names of columns that have 0 variance  
names(numeric[, sapply(numeric, function(v) var(v, na.rm=TRUE)==0)])

## character(0)

#summary(is.na(numeric))  
  
#str(numeric)  
#Try Principal Components Analysis  
prin\_comp <-prcomp(numeric, scale. = T)  
  
prin\_comp$center #These are the means for each column

## TimeOfDay.factor Hair.factor   
## 2.096300 2.494444   
## Size.factor Mix.factor   
## 2.003666 1.834225   
## Fixed.factor Sex.factor   
## 1.345093 1.579595   
## AVERAGE.LIFESPAN..YEARS.factor AboutToDie.factor   
## 7.722773 1.228404   
## Hypo.factor   
## 1.116540

prin\_comp$scale #These are the SD for each column

## TimeOfDay.factor Hair.factor   
## 1.4019127 0.7508208   
## Size.factor Mix.factor   
## 0.8004610 0.3718855   
## Fixed.factor Sex.factor   
## 0.5548640 0.5705571   
## AVERAGE.LIFESPAN..YEARS.factor AboutToDie.factor   
## 1.9404652 0.4198119   
## Hypo.factor   
## 0.3208775

prin\_comp$rotation

## PC1 PC2 PC3  
## TimeOfDay.factor 0.00652919 -0.1730076 0.27718030  
## Hair.factor 0.42905728 -0.2268375 -0.30189840  
## Size.factor 0.46971350 0.3364459 0.17068639  
## Mix.factor 0.06641263 -0.3610836 -0.27103303  
## Fixed.factor 0.14797218 -0.5016225 0.43015867  
## Sex.factor 0.07737120 -0.3514083 0.57132734  
## AVERAGE.LIFESPAN..YEARS.factor 0.59013407 0.1203684 -0.05665292  
## AboutToDie.factor -0.44725833 0.1503757 0.13942921  
## Hypo.factor 0.12101861 0.5126612 0.44196631  
## PC4 PC5 PC6  
## TimeOfDay.factor -0.87222088 0.28352165 -0.05365885  
## Hair.factor 0.02523041 0.33467151 -0.30538356  
## Size.factor -0.08071663 -0.20739634 -0.25329954  
## Mix.factor -0.24489824 -0.82461239 -0.02280660  
## Fixed.factor 0.03794155 -0.05643170 0.37275693  
## Sex.factor 0.37971806 -0.03952822 -0.43308564  
## AVERAGE.LIFESPAN..YEARS.factor -0.07111071 -0.05003719 -0.15223680  
## AboutToDie.factor -0.13825036 -0.16681338 -0.63778094  
## Hypo.factor -0.04804778 -0.22256275 0.28762710  
## PC7 PC8 PC9  
## TimeOfDay.factor -0.21411212 -0.05216661 -0.024462312  
## Hair.factor 0.10006123 0.62432650 -0.260029610  
## Size.factor 0.24119669 -0.39709677 -0.554516698  
## Mix.factor -0.17125067 0.13431900 -0.062448494  
## Fixed.factor 0.62777143 0.04959381 0.036252868  
## Sex.factor -0.45783026 -0.03301964 0.010033780  
## AVERAGE.LIFESPAN..YEARS.factor 0.05167739 -0.12361670 0.765085311  
## AboutToDie.factor 0.47471273 0.21272979 0.183292417  
## Hypo.factor -0.15783811 0.60636848 0.005569968

dim(prin\_comp$x)

## [1] 26729 9

prin\_comp$rotation = -prin\_comp$rotation  
prin\_comp$x = -prin\_comp$x  
#biplot(prin\_comp, scale=0)  
  
prin\_comp$sdev

## [1] 1.4478345 1.2358861 1.0857115 0.9987233 0.9509417 0.8619168 0.7839633  
## [8] 0.7624236 0.5975487

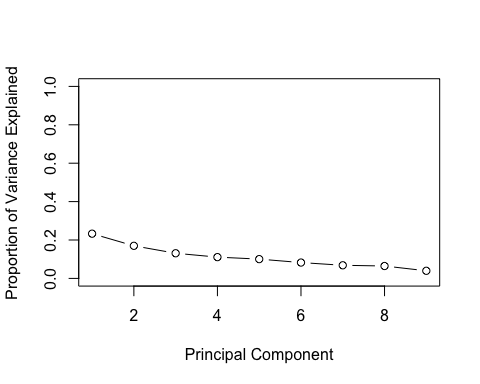
prin\_var = prin\_comp$sdev^2  
prin\_var

## [1] 2.0962248 1.5274144 1.1787694 0.9974481 0.9042901 0.7429005 0.6145984  
## [8] 0.5812897 0.3570645

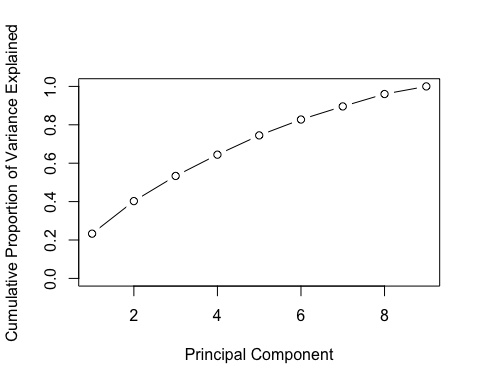
pve = prin\_var/sum(prin\_var)  
pve

## [1] 0.23291387 0.16971271 0.13097438 0.11082757 0.10047667 0.08254450  
## [7] 0.06828872 0.06458775 0.03967383

#Plot the proportion of variance explained by each principal component.   
  
plot(pve, xlab="Principal Component", ylab = "Proportion of Variance Explained", ylim=c(0,1), type='b')



plot(cumsum(pve), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explained", ylim=c(0,1), type='b')



#Try Clustering   
  
set.seed(2)  
km.out=kmeans(numeric, 4, nstart = 50)  
#km.out  
#K means clustering with 4 clusters of sized 9755, 7007, 3972, 5995  
km.out$tot.withinss

## [1] 80805.49

#plot(numeric, col=km.out$cluster, main = "k-means with 4 clusters")  
km.data = data.frame(km.out$cluster, full$OutcomeType)  
table(km.data)

## full.OutcomeType  
## km.out.cluster Adoption Died Euthanasia Return\_to\_owner Transfer  
## 1 4613 42 422 1300 3378  
## 2 2430 17 468 1750 1330  
## 3 1752 32 206 668 1314  
## 4 1974 106 459 1068 3400

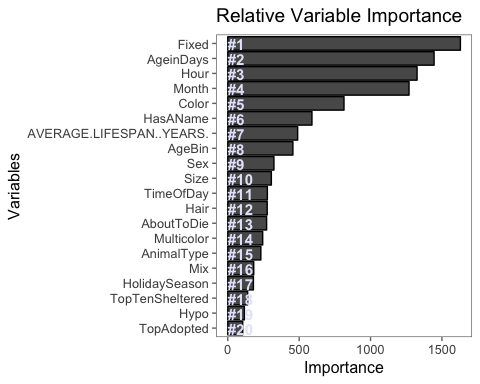
# Hierarchial Clustering (Not Enough Computing Power)  
#hc.complete = hclust(dist(km.data), method='complete')  
#hc.clusters = cutree(hc.complete, 4)  
#hc.data = data.frame(full$OutcomeType, hc.clusters)  
#table(hc.data)  
#hc.avg = hclust(dist(km.data), method='average')  
#hc.clusters.avg = cutree(hc.avg, 4)  
#hc.data.avg = data.frame(full$OutcomeType, hc.clusters.avg)  
#table(hc.data.avg)  
#hc.centroid = hclust(dist(km.data), method='centroid')  
#hclust.centroid = cutree(hc.centroid, 4)  
#hc.data.centroid = data.frame(full$OutcomeType, hclust.centroid)  
#table(hc.data.centroid)

# Part 4: Model Building

We fit 4 models in this analysis. Random Forest, LDA, Ridge Regression, and K-Nearest Neighbors.

# Random Forrest Model

library(randomForest)  
#Cannot predict with SimpleBreed   
RF\_Model1 <- randomForest(OutcomeType~AnimalType+Sex+Fixed+Mix+HasAName+AgeinDays+AVERAGE.LIFESPAN..YEARS.+Hypo+AboutToDie+TopTenSheltered+TopAdopted+HolidaySeason+Month+AgeBin+TimeOfDay+Multicolor+Hour+Hair+Size+Color, data=Train.animals, ntree=500, mtry=4, importance = TRUE, na.action = na.exclude)  
#Variable Importance Plot  
importance <- importance(RF\_Model1)  
varImportance <- data.frame(variables = row.names(importance), Importance = round(importance[ , 'MeanDecreaseGini'], 2))  
  
rankImportance <-varImportance %>%   
 mutate(Rank = paste0('#', dense\_rank(desc(Importance))))  
ggplot(rankImportance, aes(x=reorder(variables, Importance),   
 y=Importance)) +   
 geom\_bar(stat='identity', color = 'black')+  
 geom\_text(aes(x=variables, y=0.5, label = Rank),   
 hjust = 0, vjust = 0.55, size=4, color = 'lavender',   
 fontface = 'bold')+  
 labs(x='Variables', title = 'Relative Variable Importance') +   
 coord\_flip()+  
 theme\_few()



head(getTree(RF\_Model1, 1, labelVar =TRUE),10)

## left daughter right daughter split var split point  
## 1 2 3 TimeOfDay 7  
## 2 4 5 AVERAGE.LIFESPAN..YEARS. 7301  
## 3 6 7 Hour 795105  
## 4 8 9 Fixed 1  
## 5 10 11 Fixed 1  
## 6 12 13 AgeBin 3  
## 7 14 15 Sex 3  
## 8 16 17 AgeBin 2  
## 9 18 19 AgeinDays -2147483648  
## 10 20 21 AboutToDie 1  
## status prediction  
## 1 1 <NA>  
## 2 1 <NA>  
## 3 1 <NA>  
## 4 1 <NA>  
## 5 1 <NA>  
## 6 1 <NA>  
## 7 1 <NA>  
## 8 1 <NA>  
## 9 1 <NA>  
## 10 1 <NA>

#Now Predict on the Test Set   
preds <- predict(RF\_Model1, Test.animals, type = 'vote')  
solution <-data.frame(Test.animals$AnimalID, preds)  
  
#Obtain Misclassification Rate  
solution[, "max"] <- apply(solution[, 2:6], 1, max)  
solution$Adopt = ifelse(solution$Adoption == solution$max, 1, 0)  
solution$Die = ifelse(solution$Died == solution$max, 1, 0)  
solution$Ethanise = ifelse(solution$Euthanasia == solution$max, 1, 0)  
solution$Trans = ifelse(solution$Transfer == solution$max, 1, 0)  
solution$RTO = ifelse(solution$Return\_to\_owner == solution$max,1,0)  
  
Test.animals$Adopted =ifelse(Test.animals$OutcomeType == 'Adoption',1,0)  
Test.animals$Died1 = ifelse(Test.animals$OutcomeType == 'Died',1,0)  
Test.animals$Euthanasia1 = ifelse(Test.animals$OutcomeType == 'Euthanasia', 1, 0)  
Test.animals$Return\_to\_owner1 = ifelse(Test.animals$OutcomeType == 'Return\_to\_owner', 1, 0)  
Test.animals$Transfer1 = ifelse(Test.animals$OutcomeType == 'Transfer',1,0)  
  
actual = Test.animals[, c(36, 37, 38, 39, 40)]  
Eval = data.frame(cbind(solution, actual))  
  
x= sum(ifelse(Eval$Adopted != Eval$Adopt, 1, 0))+  
sum(ifelse(Eval$Die != Eval$Died1, 1, 0))+  
sum(ifelse(Eval$Ethanise != Eval$Euthanasia1, 1, 0))+  
sum(ifelse(Eval$Trans != Eval$Transfer1, 1, 0 ))+  
sum(ifelse(Eval$RTO != Eval$Return\_to\_owner1, 1, 0))  
  
x/8050

## [1] 0.6639752

In this Random Forest Model we fit 500 decision trees and split on 4 random predictors at each split. We chose 4 because it is the square root of the number of features and it performed the best in the model. The misclassification rate achieved by this model was 67.27% . We believe this model performed poorly due to the large disparity between relative importance of features in this dataset. Age in Days was by far the most important feature in this analysis.

# K-Nearest Neighbors Model

full$TopAdopted.Factor <- factor(full$TopAdopted)  
full$TopAdopted.Factor <- as.numeric(full$TopAdopted.Factor)  
full$TopTenSheltered.Factor <- factor(full$TopTenSheltered)  
full$TopTenSheltered.Factor <- as.numeric(full$TopTenSheltered.Factor)  
full$HolidaySeason.Factor <- factor(full$HolidaySeason)  
full$HolidaySeason.Factor <- as.numeric(full$HolidaySeason.Factor)  
full$AgeBin.Factor <- factor(full$AgeBin)  
full$AgeBin.Factor <- as.numeric(full$AgeBin.Factor)  
full$Color.Factor <- factor(full$Color)  
full$Color.Factor <- as.numeric(full$Color.Factor)  
full$OutcomeType.Factor <- factor(full$OutcomeType)  
full$OutcomeType.Factor <- as.numeric(full$OutcomeType.Factor)  
full$AgeinDays.Factor <- factor(full$AgeinDays)  
full$AgeinDays.Factor <-as.numeric(full$AgeinDays.Factor)  
#str(full)  
Sample = sample(c(TRUE, FALSE), nrow(full), replace = T, prob = c(0.7, 0.3))  
Train.animals = full[Sample,]  
Test.animals = full[!Sample,]  
library(class)  
  
Test.animals1 =Test.animals[, c(36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54)]  
Train.animals1 = Train.animals[, c(36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54)]  
  
sum(is.na(Test.animals1))

## [1] 0

sum(is.na(Train.animals1))

## [1] 0

#str(Train.animals1)  
set.seed(1)  
actual = Test.animals1$OutcomeType.Factor  
knn.animals = knn(train = Train.animals1, test = Test.animals1, Train.animals1$OutcomeType.Factor, k = 10)  
  
Eval = data.frame(cbind(knn.animals, actual))  
Eval$Error = ifelse(Eval$knn.animals == Eval$actual, 0, 1)  
sum(Eval$Error)/length(Eval$Error)

## [1] 0.09935016

The K Nearest Neighbors approach works by comparing each test observation to a given number n most similar training observations and assigning it to which ever class the most of the n training observations belong to. Multiple values of n were tested, but n=14 yielded the lowest misclassification rate and was used in the final model. The main advantages of the KNN approach for this data are that it makes no assumptions about the distribution of the data and works well for multiple classes. The final model predicted which animals would be adopted and transferred relatively accurately, but did not perform well on observations in the other classes and did not predict that any animals would die in the shelter. Because the model compares observations with similar training observations, it would be expected that the most common outcomes are predicted most often.

# Linear Discriminant Analysis Model

set.seed(1)  
Sample = sample(c(TRUE, FALSE), nrow(full), replace = T, prob = c(0.7, 0.3))  
Train.animals = full[Sample,]  
Test.animals = full[!Sample,]  
Train.animals$AVERAGE.LIFESPAN..YEARS. = as.factor(Train.animals$AVERAGE.LIFESPAN..YEARS.)  
Train.animals$AboutToDie = as.factor(Train.animals$AboutToDie)  
Train.animals$Hypo = as.factor(Train.animals$Hypo)  
  
#Pick the columns We Want  
library(dplyr)  
Train.animals\_LDA <- Train.animals[,c(5,7,11,12,13,14,15,18,19,21,22,23,24,25,26,27,28,30,31,33,34,35)]  
Test.animals\_LDA <- Test.animals[,c(5,7,11,12,13,14,15,18,19,21,22,23,24,25,26,27,28,30,31,33,34,35)]  
  
library(MASS)  
#Cannot Predict with Age in Days   
#Collinearity: Fixed,Mix   
lda.fit = lda(OutcomeType~AnimalType+Sex+Fixed+Mix+HasAName+AVERAGE.LIFESPAN..YEARS.  
 +Hypo+AboutToDie+TopTenSheltered+TopAdopted+HolidaySeason+Month+AgeBin+TimeOfDay  
 +Multicolor+Hour+Hair+Size+Color, data=Train.animals)

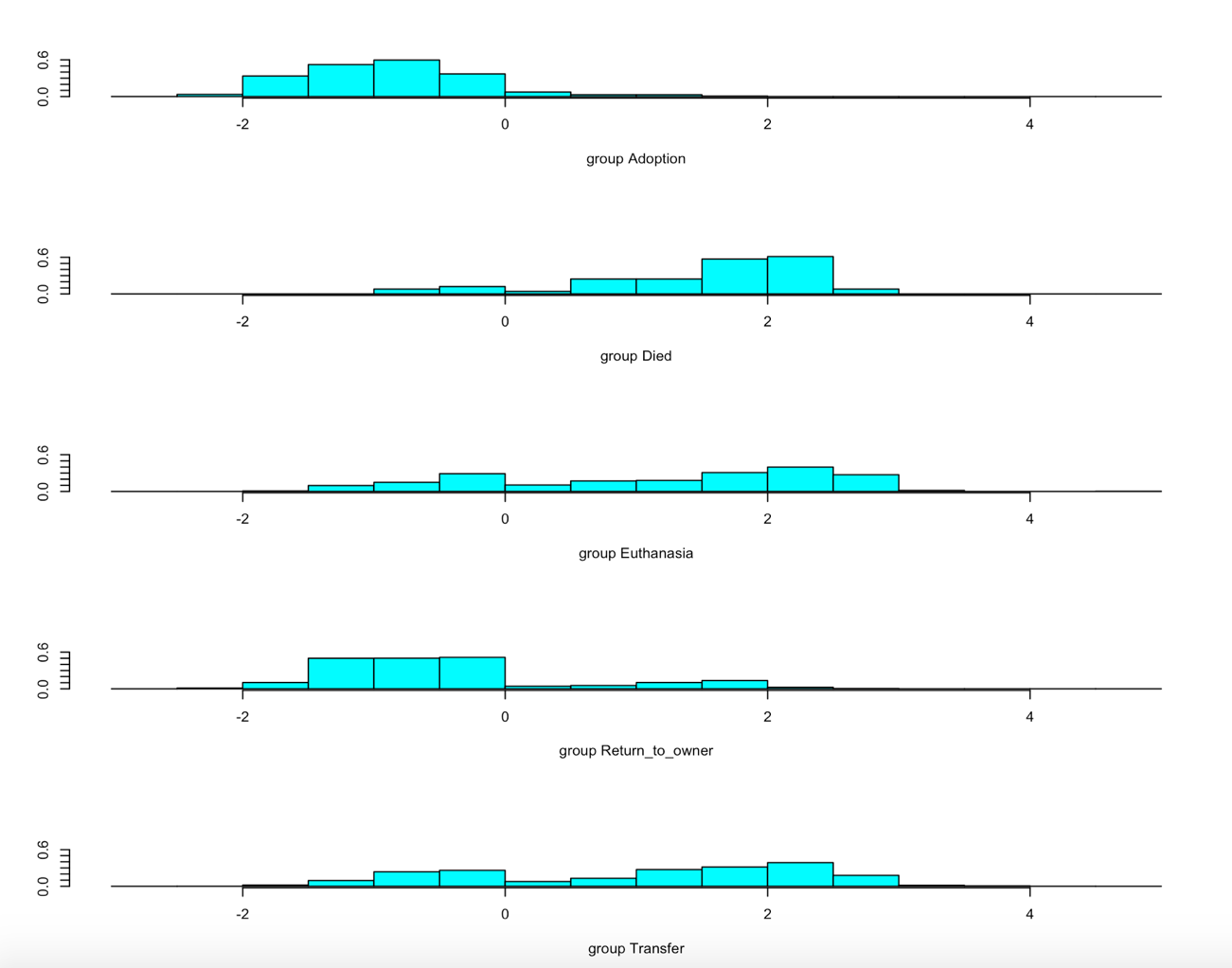
## Warning in lda.default(x, grouping, ...): variables are collinear

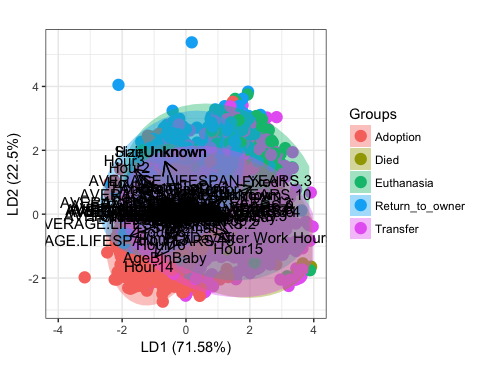
summary(lda.fit)

## Length Class Mode   
## prior 5 -none- numeric   
## counts 5 -none- numeric   
## means 375 -none- numeric   
## scaling 300 -none- numeric   
## lev 5 -none- character  
## svd 4 -none- numeric   
## N 1 -none- numeric   
## call 3 -none- call   
## terms 3 terms call   
## xlevels 19 -none- list

#lda.fit  
lda.pred = predict(lda.fit, Test.animals)  
p <-predict(lda.fit, Test.animals)  
#ldahist(data=p$x[,1], g=Test.animals\_LDA$OutcomeType)  
  
#Bi-Plot   
library(devtools)  
#install\_github("fawda123/ggord")  
library(ggord)  
ggord(lda.fit, Train.animals\_LDA$OutcomeType, xlim=c(-4,4))

## Warning: Removed 3 rows containing missing values (geom\_point).





eval = data.frame(lda.pred$class, Test.animals\_LDA$OutcomeType)  
  
eval$misclass = ifelse(eval$lda.pred.class == eval$Test.animals\_LDA.OutcomeType, 0, 1)  
x=sum(eval$misclass)  
x/8001

## [1] 0.359955

Linear Discriminant Analysis is a classification method that searches for a linear combination of variables that separates them into the best classes. LDA tries to maximize separability. This model assumes that the observations within each class are drawn from a multivariate Gaussian distribution.

Here you can see the proportion of trace figures, which are the proportions of the between-class variance explained by their linear combinations. The scatterplots here show that as the proportion of trace decreases, the separability worsens. This is indicative of a poor model. Ideally you would want to see the colors of these dots separate to show a more distinct class. Instead, we see a lot of overlap. This is due to multicollinearity issues, since some of our variables are highly correlated to each other. This suggests that LDA is a poor approach for predicting the outcomes of Shelter Animals. We found the error rate to be 35.9%, but as we said, due to multicollinearity issues, this model is still a poor estimator.

# Ridge Regression Model

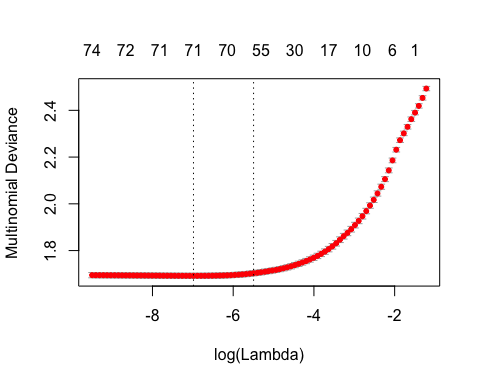
library(glmnet)  
set.seed(1)  
Sample = sample(c(TRUE, FALSE), nrow(full), replace = T, prob = c(0.7, 0.3))  
train.Master= full[Sample,]  
test.Master = full[!Sample,]  
  
train\_matrix<- model.matrix(OutcomeType~AnimalType+Sex+Fixed+Mix+HasAName+AVERAGE.LIFESPAN..YEARS.  
 +Hypo+AboutToDie+TopTenSheltered+TopAdopted+HolidaySeason+Month+AgeBin+TimeOfDay  
 +Multicolor+Hour+Hair+Size+Color, data=train.Master)  
test\_matrix <- model.matrix(OutcomeType~AnimalType+Sex+Fixed+Mix+HasAName+AVERAGE.LIFESPAN..YEARS.  
 +Hypo+AboutToDie+TopTenSheltered+TopAdopted+HolidaySeason+Month+AgeBin+TimeOfDay  
 +Multicolor+Hour+Hair+Size+Color, data=test.Master)  
y <- as.factor(train.Master$OutcomeType)  
  
lambdas <- 10^seq(10,-2, length = 100)  
  
ridge.glmnet <- glmnet(train\_matrix, y, family = "multinomial", alpha = 0, lambda = lambdas)  
summary(ridge.glmnet)

## Length Class Mode   
## a0 500 -none- numeric   
## beta 5 -none- list   
## dfmat 500 -none- numeric   
## df 100 -none- numeric   
## dim 2 -none- numeric   
## lambda 100 -none- numeric   
## dev.ratio 100 -none- numeric   
## nulldev 1 -none- numeric   
## npasses 1 -none- numeric   
## jerr 1 -none- numeric   
## offset 1 -none- logical   
## classnames 5 -none- character  
## grouped 1 -none- logical   
## call 6 -none- call   
## nobs 1 -none- numeric

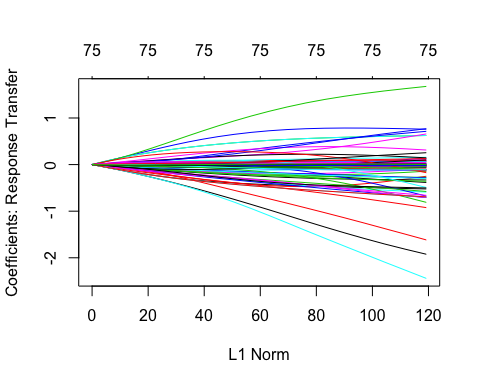
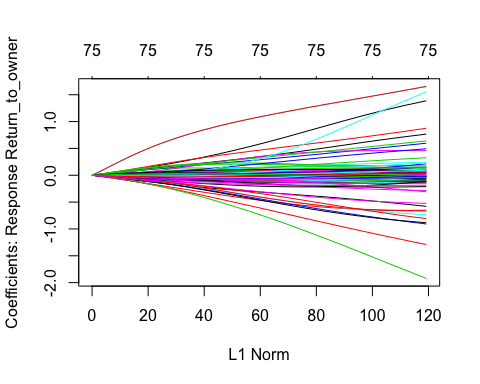
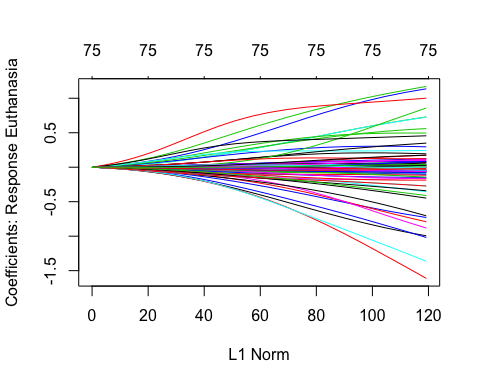
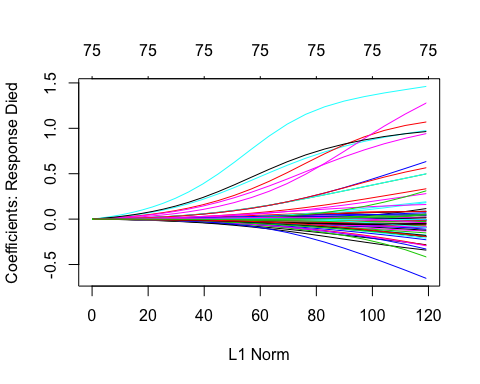
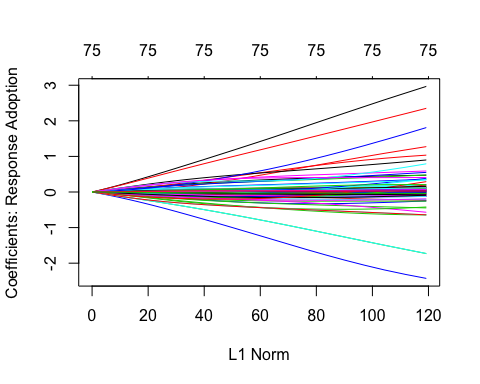
cv2 <- cv.glmnet(train\_matrix, y, family = "multinomial", type.multinomial = "grouped")  
sum(resid(ridge.glmnet)^2)

## [1] 0

plot(cv2)



mod <- plot(ridge.glmnet)



min.lambda <- cv2$lambda.min  
min.lambda

## [1] 0.0009268137

#min lambda is .001022479  
  
#What is the test Misclassification Rate associated with the best lambda under ridge regression?  
ridge\_prediction <-predict(ridge.glmnet, newx = test\_matrix, s=min.lambda)

Ridge regression works by reducing model variance while introducing a small amount of bias. This is done through L1 normalization which determines a penalty term that shrinks the coefficients of the ordinary least squares equation. This technique is especially useful when there is disparity in the size of the coefficients. After running the model with our data we determined that the optimal lambda value was .00102. This value resulted in a 48.70% misclassification rate for our data. (Although this value was low, it may be a more appropriate way to model the data because ridge regression, by nature, makes it difficult to over fit the data.)