# DSBI Final Project

Group 8

## 0. Install & Library Packages

```
#install.packages("tidyverse")
#install.packages("lubridate")
#install.packages("magrittr")
#install.packages("naniar")
#install.packages("VIM")
#install.packages("InformationValue")
#install.packages("caret")
#install.packages("tictoc")
#install.packages("caretEnsemble")
#install.packages("gbm")
#install.packages("kernlab")
#install.packages("nnet")
#install.packages("randomForest")
#install.packages("xgboost")
#install.packages("plyr")
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------
----------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0

## -- Conflicts -------------------------------------------------------------
----------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract

library(naniar)
library(VIM)

## Loading required package: colorspace

## Loading required package: grid

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at:
https://github.com/alexkowa/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep

library(InformationValue)
library(caret)

## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following objects are masked from 'package:InformationValue':
##
##     confusionMatrix, precision, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift

library(tictoc)
library(pROC)

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

##
## Attaching package: 'pROC'

## The following object is masked from 'package:colorspace':
##
##     coords

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

library(caretEnsemble)

##
## Attaching package: 'caretEnsemble'

## The following object is masked from 'package:ggplot2':
##
##     autoplot

library(gbm)

## Loaded gbm 2.1.5

library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library(nnet)
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

library(xgboost)

##
## Attaching package: 'xgboost'

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

Attention: 1. Please download the data from https://www.kaggle.com/c/shelter-animal-outcomes/overview, and put that into one suitable working directory 2. Please set your work directory

# 1. Read data & Take a glimpse

## 1.1 Training Data

```r
train <- read_csv("train.csv")

## Parsed with column specification:
## cols(
##   AnimalID = col_character(),
##   Name = col_character(),
##   DateTime = col_datetime(format = ""),
##   OutcomeType = col_character(),
##   OutcomeSubtype = col_character(),
##   AnimalType = col_character(),
##   SexuponOutcome = col_character(),
##   AgeuponOutcome = col_character(),
##   Breed = col_character(),
##   Color = col_character()
## )

glimpse(train)
```

```
## Observations: 26,729
## Variables: 10
## $ AnimalID       <chr> "A671945", "A656520", "A686464", "A683430", "A6...
## $ Name           <chr> "Hambone", "Emily", "Pearce", NA, NA, "Elsa", "...
## $ DateTime       <dttm> 2014-02-12 18:22:00, 2013-10-13 12:44:00, 2015...
## $ OutcomeType    <chr> "Return_to_owner", "Euthanasia", "Adoption", "T...
## $ OutcomeSubtype <chr> NA, "Suffering", "Foster", "Partner", "Partner"...
## $ AnimalType     <chr> "Dog", "Cat", "Dog", "Cat", "Dog", "Dog", "Cat"...
## $ SexuponOutcome <chr> "Neutered Male", "Spayed Female", "Neutered Mal...
## $ AgeuponOutcome <chr> "1 year", "1 year", "2 years", "3 weeks", "2 ye...
## $ Breed          <chr> "Shetland Sheepdog Mix", "Domestic Shorthair Mi...
## $ Color          <chr> "Brown/White", "Cream Tabby", "Blue/White", "Bl...
```

### 1.2 Submittable Data

```
test <- read_csv("test.csv")

## Parsed with column specification:
## cols(
##   ID = col_double(),
##   Name = col_character(),
##   DateTime = col_datetime(format = ""),
##   AnimalType = col_character(),
##   SexuponOutcome = col_character(),
##   AgeuponOutcome = col_character(),
##   Breed = col_character(),
##   Color = col_character()
## )

glimpse(test)

## Observations: 11,456
## Variables: 8
## $ ID             <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ...
## $ Name           <chr> "Summer", "Cheyenne", "Gus", "Pongo", "Skooter"...
## $ DateTime       <dttm> 2015-10-12 12:15:00, 2014-07-26 17:59:00, 2016...
## $ AnimalType     <chr> "Dog", "Dog", "Cat", "Dog", "Dog", "Dog", "Cat"...
## $ SexuponOutcome <chr> "Intact Female", "Spayed Female", "Neutered Mal...
## $ AgeuponOutcome <chr> "10 months", "2 years", "1 year", "4 months", "...
## $ Breed          <chr> "Labrador Retriever Mix", "German Shepherd/Sibe...
## $ Color          <chr> "Red/White", "Black/Tan", "Brown Tabby", "Trico...
```

### 1.3 Breed-Size Data (Dog)

```
dog_breed <- read_csv("dog_breed.csv") # outside source

## Parsed with column specification:
## cols(
##   Name = col_character(),
##   Breed = col_character(),
##   Breed_cl = col_character(),
##   MIX1 = col_character(),
```

```
##   MIX2 = col_character(),
##   Point1 = col_double(),
##   Point2 = col_double(),
##   Point = col_double()
## )
```

```
glimpse(dog_breed)
```

```
## Observations: 22,251
## Variables: 8
## $ Name     <chr> "Hambone", "Pearce", NA, "Elsa", "Lucy", NA, NA, "Roc...
## $ Breed    <chr> "Shetland Sheepdog", "American Pit Bull Terrier", "Lh...
## $ Breed_cl <chr> "shetlandsheepdog", "americanpitbullterrier", "lhasaa...
## $ MIX1     <chr> "shetlandsheepdog", "americanpitbullterrier", "lhasaa...
## $ MIX2     <chr> "shetlandsheepdog", "americanpitbullterrier", "miniat...
## $ Point1   <dbl> 3, 3, 2, 1, 3, 1, 2, 3, 1, 4, 3, 0, 0, 0, 3, 3, 3, 1,...
## $ Point2   <dbl> 3, 3, 2, 1, 3, 1, 2, 3, 1, 4, 3, 0, 0, 3, 3, 3, 3, 1,...
## $ Point    <dbl> 3.0, 3.0, 2.0, 1.0, 3.0, 1.0, 2.0, 3.0, 1.0, 4.0, 3.0...
```

### 1.4 Splitting & Cleaning the Data

```
tv_row <- 1:nrow(train)

# Make sure do AnimalID & ID contain any valuable information
sum(is.na(train$AnimalID))
```

```
## [1] 0
```

```
mean(str_detect(train$AnimalID, "A"))
```

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

```
mean(sapply(train$AnimalID, nchar) == 7)
```

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

```
sum(is.na(test$ID))
```

```
## [1] 0
```

```
# It seems like no information inside for those 2 features

# Omit AnimalID & ID because we think they tell us nothing
shelter <- bind_rows(train[,-1], test[,-1])
glimpse(shelter)
```

```
## Observations: 38,185
## Variables: 9
## $ Name           <chr> "Hambone", "Emily", "Pearce", NA, NA, "Elsa", "...
## $ DateTime       <dttm> 2014-02-12 18:22:00, 2013-10-13 12:44:00, 2015...
## $ OutcomeType    <chr> "Return_to_owner", "Euthanasia", "Adoption", "T...
## $ OutcomeSubtype <chr> NA, "Suffering", "Foster", "Partner", "Partner"...
## $ AnimalType     <chr> "Dog", "Cat", "Dog", "Cat", "Dog", "Dog", "Cat"...
```

```
## $ SexuponOutcome <chr> "Neutered Male", "Spayed Female", "Neutered Mal...
## $ AgeuponOutcome <chr> "1 year", "1 year", "2 years", "3 weeks", "2 ye...
## $ Breed          <chr> "Shetland Sheepdog Mix", "Domestic Shorthair Mi...
## $ Color          <chr> "Brown/White", "Cream Tabby", "Blue/White", "Bl...
```

# 2. Feature Engineering & Exploratory Data Analysis

## 2.1 OutcomeType

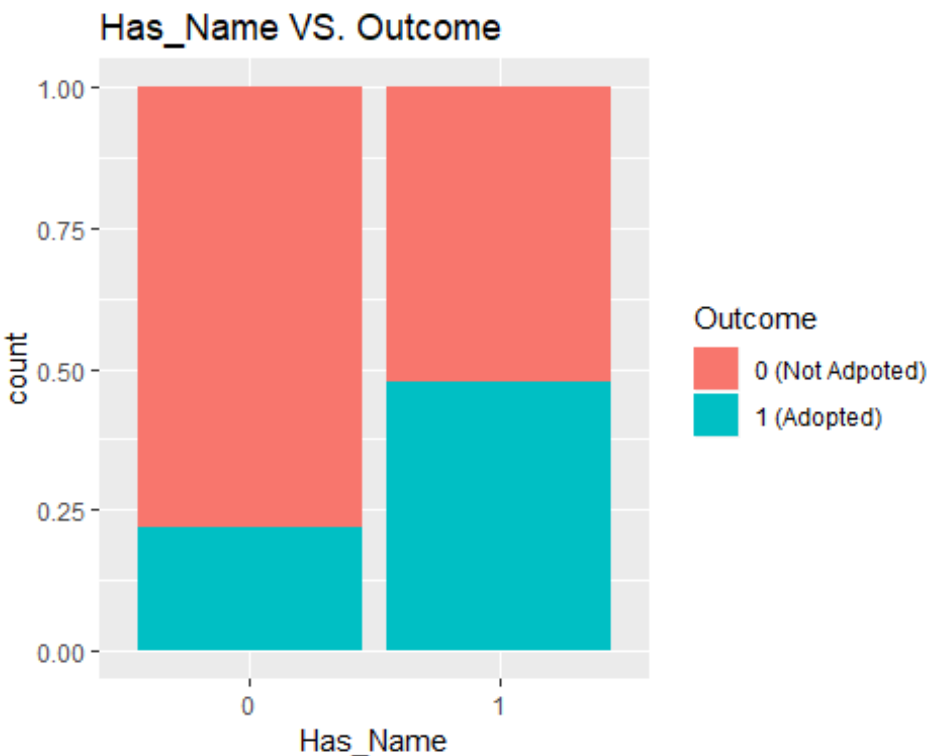
```
length(unique(shelter$OutcomeType))

## [1] 6

shelter <- shelter %>%
  mutate(Outcome = factor(ifelse(shelter$OutcomeType == "Adoption", 1, 0)))

shelter %$%
  summary(Outcome)

##     0     1  NA's
## 15960 10769 11456
```

## 2.2 Name

```
length(unique(shelter$Name)) # That's too many names!

## [1] 7967
```

### 2.2.1 Has_Name

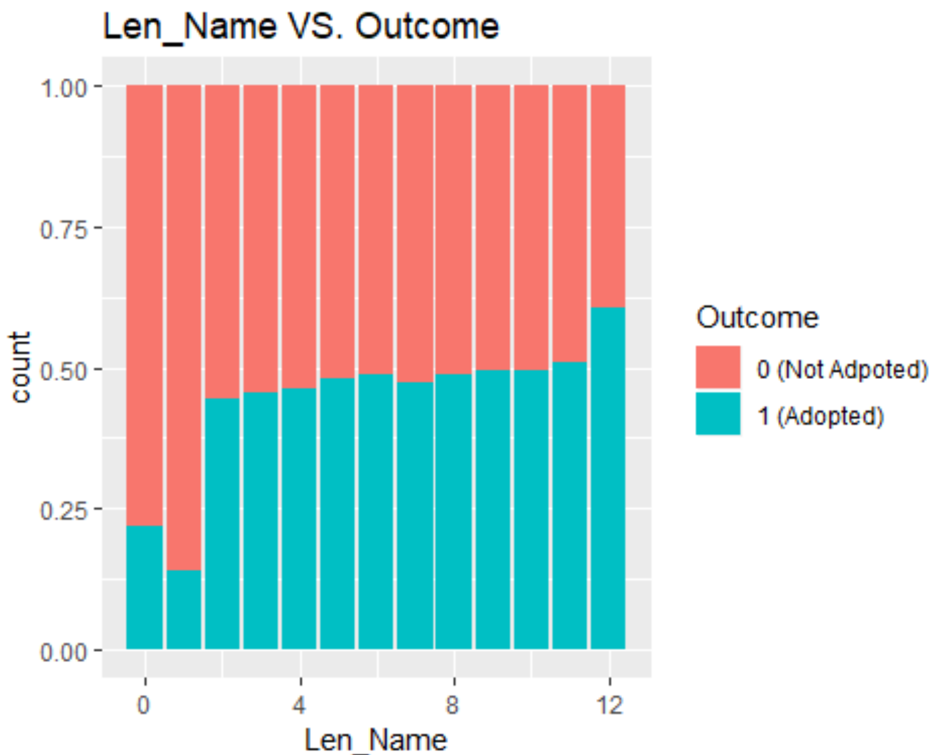
```
shelter <- shelter %>%
  mutate(Has_Name = factor(ifelse(is.na(shelter$Name), 0, 1)))

shelter[tv_row,] %>%
  group_by(Has_Name) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 2 x 3
##   Has_Name count  prob
##   <fct>    <int> <dbl>
## 1 0         7691 0.218
## 2 1        19038 0.478

ggplot(shelter[tv_row,], aes(x = Has_Name, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Has_Name VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)"))
```




### 2.2.2 Len_Name

```
shelter <- shelter %>%
  mutate(Len_Name = ifelse(is.na(shelter$Name), 0, nchar(shelter$Name)))

shelter[tv_row,] %>%
  group_by(Len_Name) %>%
  summarize(count = n(), prob = mean(Outcome == 1)) # too similar with
Has_Name

## # A tibble: 13 x 3
##    Len_Name count  prob
##       <dbl> <int> <dbl>
##  1        0  7691 0.218
##  2        1    36 0.139
##  3        2    81 0.444
##  4        3   953 0.454
##  5        4  3788 0.464
##  6        5  5578 0.479
##  7        6  4265 0.489
##  8        7  2203 0.472
##  9        8  1024 0.489
## 10        9   502 0.494
## 11       10   302 0.493
## 12       11   189 0.508
## 13       12   117 0.607
```

```r
ggplot(shelter[tv_row,], aes(x = Len_Name, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Len_Name VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)"))
```

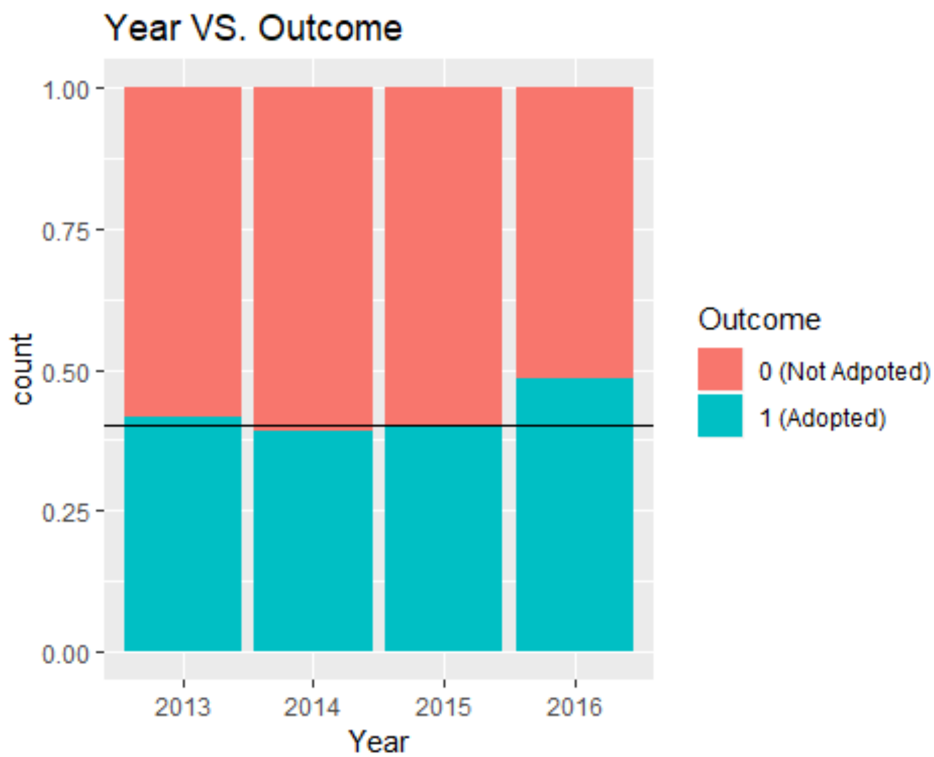



## 2.3 Datetime

### 2.3.1 Year

```r
shelter <- shelter %>%
  mutate(Year = factor(year(shelter$DateTime)))

shelter[tv_row,] %>%
  group_by(Year) %>%
  summarize(count = n(), prob = mean(Outcome == 1)) # no help

## # A tibble: 4 x 3
##   Year  count  prob
##   <fct> <int> <dbl>
## 1 2013   2702 0.417
## 2 2014  11179 0.390
## 3 2015  11481 0.402
## 4 2016   1367 0.485

shelter[tv_row,] %>%
  ggplot(aes(x = Year, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Year VS. Outcome") +
```

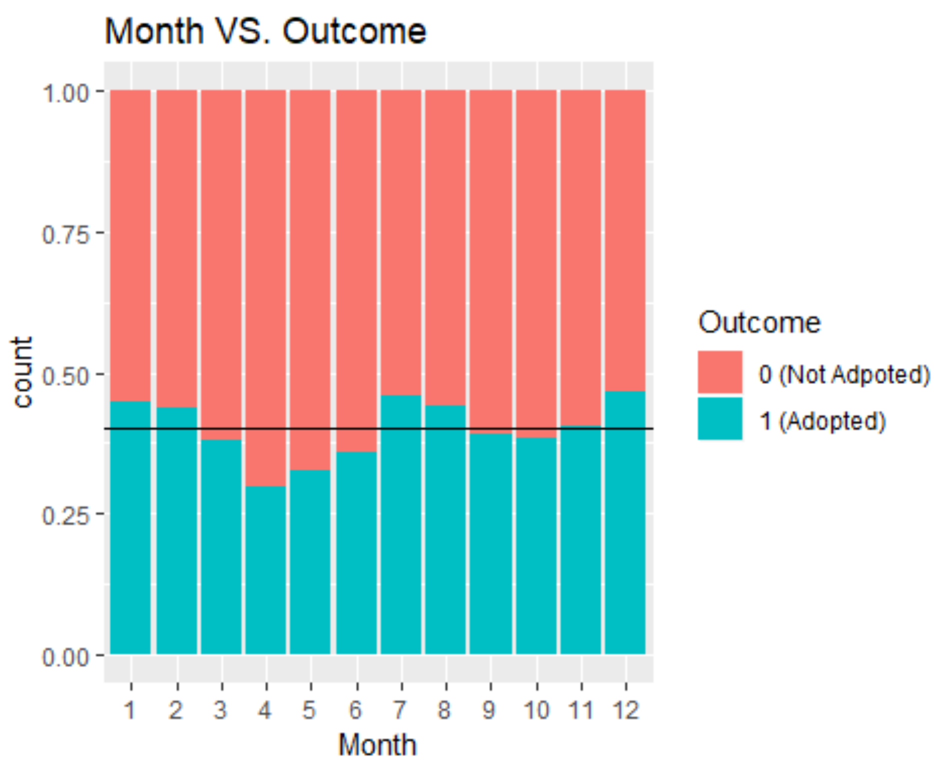
```
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  geom_hline(aes(yintercept = mean(shelter[tv_row,]$Outcome == 1))) # see the
big pic
```




### 2.3.2 Month & Season(Normal Seasons)

```
shelter <- shelter %>%
  mutate(Month = factor(month(shelter$DateTime)))

shelter[tv_row,] %>%
  group_by(Month) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 12 x 3
##    Month count  prob
##    <fct> <int> <dbl>
##  1 1      2334 0.448
##  2 2      1873 0.437
##  3 3      1498 0.381
##  4 4      1689 0.296
##  5 5      2094 0.328
##  6 6      2319 0.358
##  7 7      2506 0.459
##  8 8      2172 0.440
##  9 9      2004 0.389
## 10 10     2881 0.382
## 11 11     2668 0.405
## 12 12     2691 0.465
```

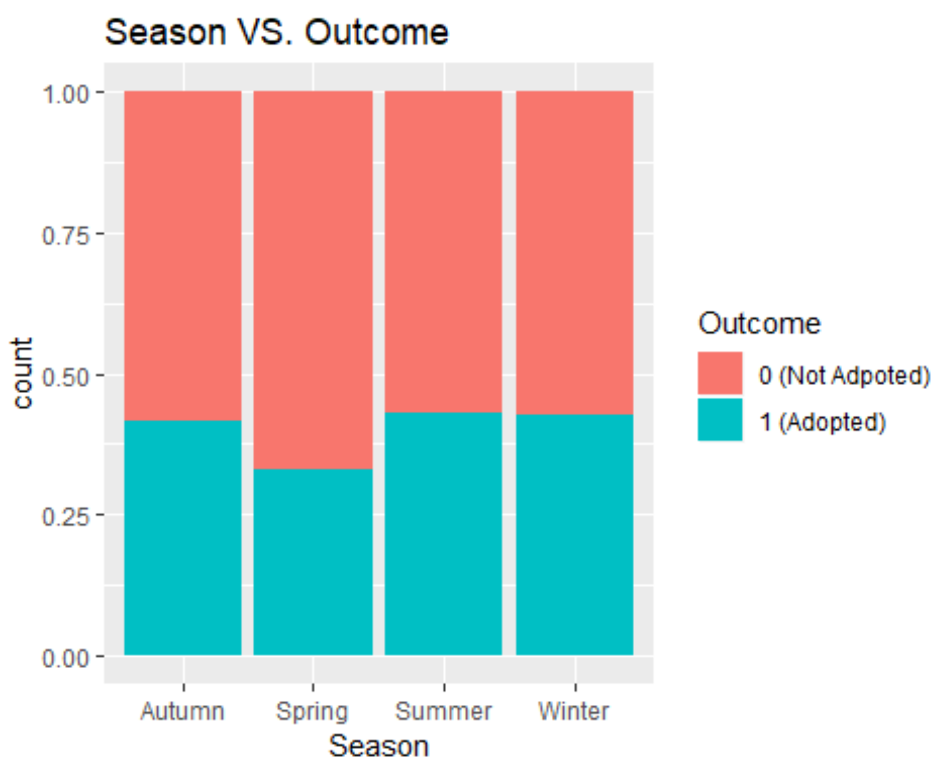
```r
shelter[tv_row,] %>%
  ggplot(aes(x = Month, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Month VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  geom_hline(aes(yintercept = mean(shelter[tv_row,]$Outcome == 1))) # see the
big pic
```




```r
season <- function(x){
  s <- month(x)
  if(between(s, 4, 6)){
    return("Spring")
  } else if(between(s, 7, 9)) {
    return("Summer")
  } else if(between(s, 10, 12)){
    return("Autumn")
  } else {
    return("Winter")
  }
}

shelter <- shelter %>%
  mutate(Season = factor(sapply(shelter$DateTime, season)))

shelter[tv_row,] %>%
  group_by(Season) %>%
  summarize(count = n(), prob = mean(Outcome == 1))
```

```
## # A tibble: 4 x 3
##   Season count  prob
##   <fct>  <int> <dbl>
## 1 Autumn  8240 0.417
## 2 Spring  6102 0.330
## 3 Summer  6682 0.432
## 4 Winter  5705 0.427
```

```
ggplot(shelter[tv_row,], aes(x = Season, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Season VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)"))
```



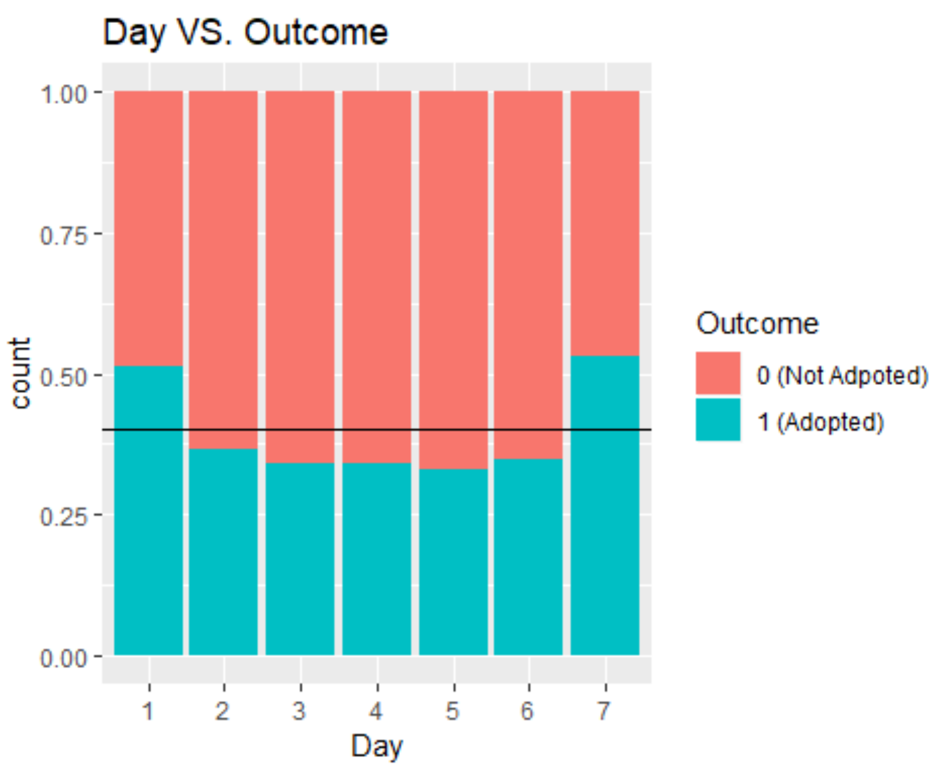

### 2.3.3 Day & Wday(Weekdays or Weekends)

```
shelter <- shelter %>%
  mutate(Day = factor(wday(shelter$DateTime)))

shelter[tv_row,] %>%
  group_by(Day) %>%
  summarize(count = n(), prob = mean(Outcome == 1))
```

```
## # A tibble: 7 x 3
##   Day   count  prob
##   <fct> <int> <dbl>
## 1 1      4317 0.514
## 2 2      3696 0.366
## 3 3      3896 0.341
```

```
## 4 4      3510 0.340
## 5 5      3376 0.331
## 6 6      3586 0.348
## 7 7      4348 0.531
```

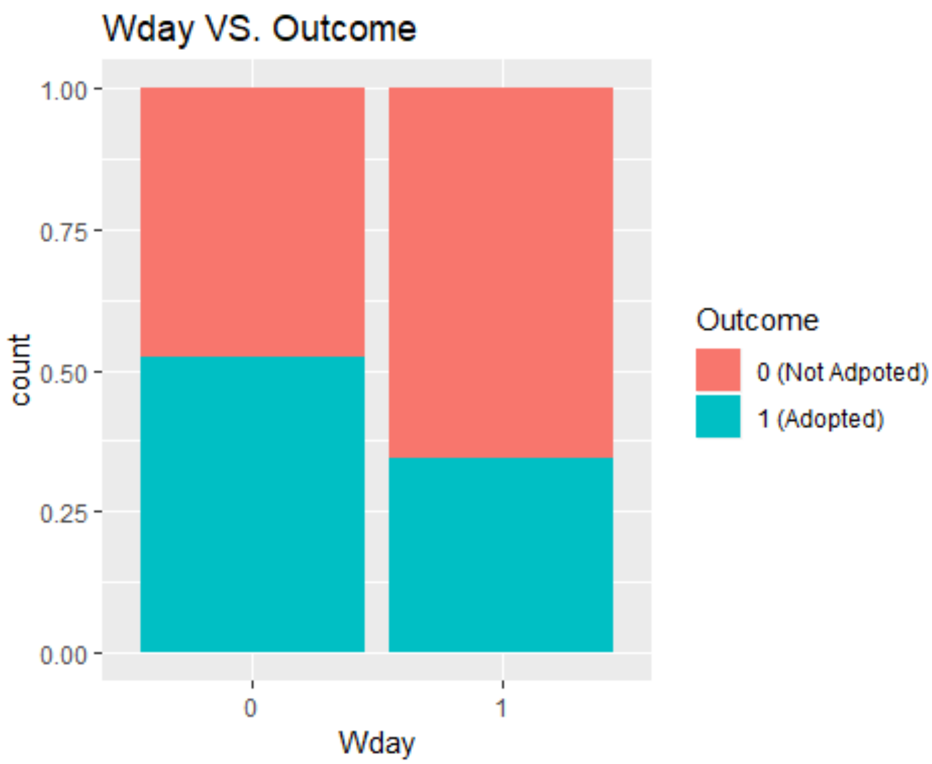
```
shelter[tv_row,] %>%
  ggplot(aes(x = Day, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Day VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  geom_hline(aes(yintercept = mean(shelter[tv_row,]$Outcome == 1))) # see the
big pic
```




```
shelter <- shelter %>%
  mutate(Wday = factor(ifelse(wday(shelter$DateTime) %in% c(1, 7), 0, 1)))

shelter[tv_row,] %>%
  group_by(Wday) %>%
  summarize(count = n(), prob = mean(Outcome == 1))
```

```
## # A tibble: 2 x 3
##   Wday  count  prob
##   <fct> <int> <dbl>
## 1 0      8665 0.522
## 2 1     18064 0.346
```

```r
ggplot(shelter[tv_row,], aes(x = Wday, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Wday VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)"))
```

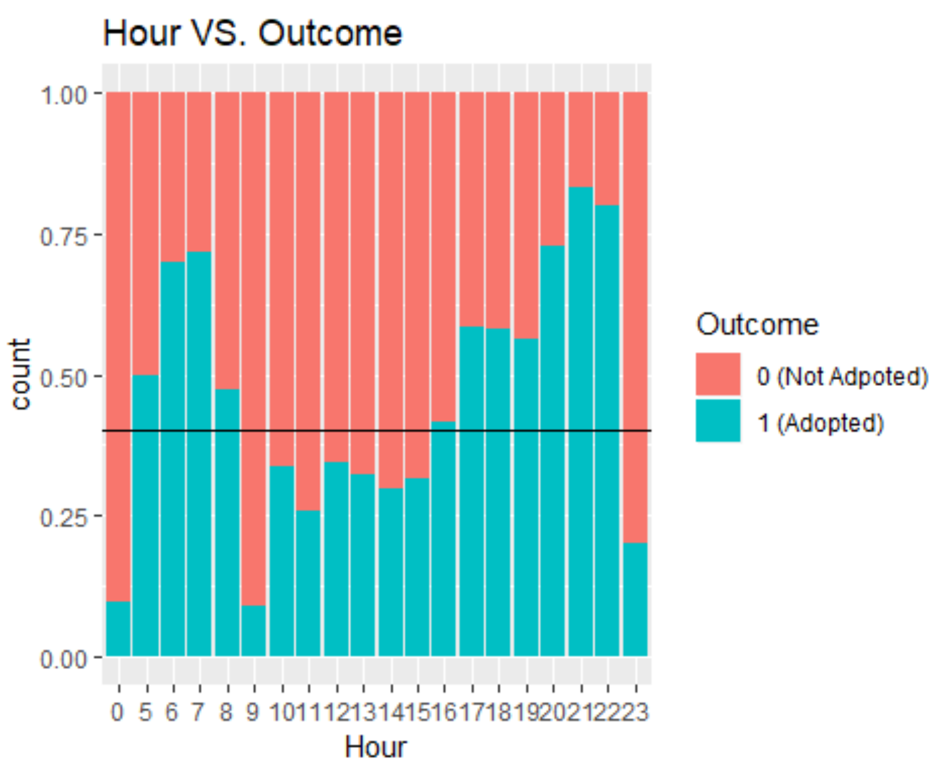



### 2.3.4 Hour & Daytime(Mornings & Evenings or Others)

```r
shelter <- shelter %>%
  mutate(Hour = factor(hour(shelter$DateTime)))

shelter[tv_row,] %>%
  group_by(Hour) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 20 x 3
##    Hour  count   prob
##    <fct> <int>  <dbl>
##  1 0       378 0.0979
##  2 5         2 0.5
##  3 6        10 0.7
##  4 7       110 0.718
##  5 8       302 0.474
##  6 9      1278 0.0892
##  7 10      405 0.336
##  8 11     2042 0.260
##  9 12     2513 0.343
## 10 13     2468 0.323
## 11 14     2800 0.297
```

```
## 12 15      2682 0.314
## 13 16      2690 0.417
## 14 17      4162 0.586
## 15 18      3684 0.581
## 16 19      1083 0.562
## 17 20        77 0.727
## 18 21        18 0.833
## 19 22         5 0.8
## 20 23        20 0.2
```

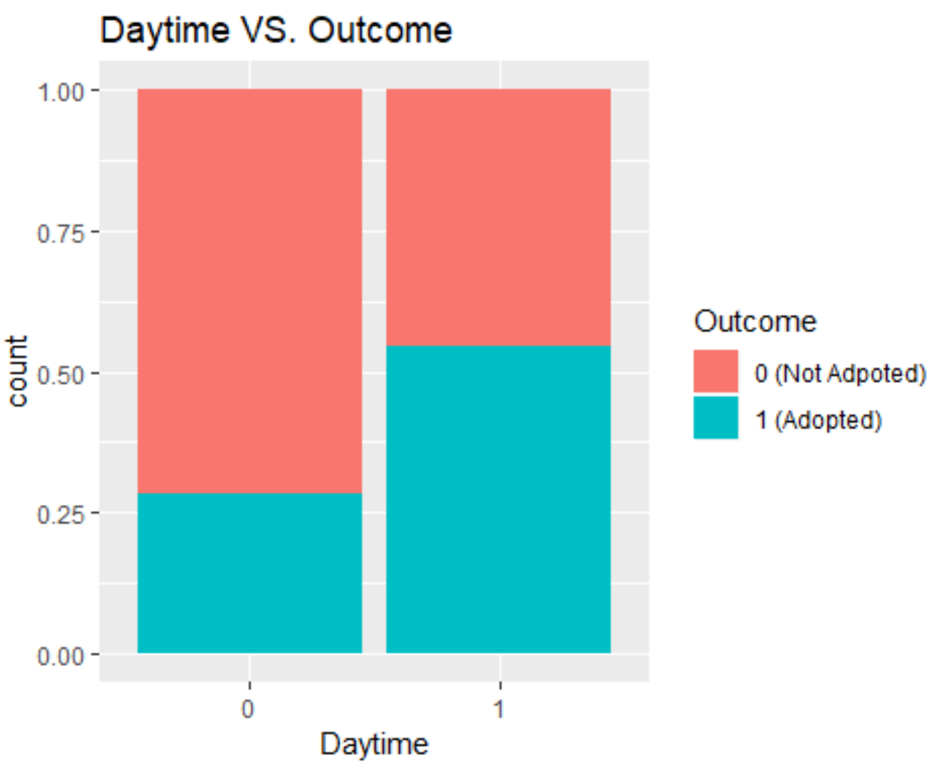
```
shelter[tv_row,] %>%
  ggplot(aes(x = Hour, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Hour VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  geom_hline(aes(yintercept = mean(shelter[tv_row,]$Outcome == 1))) # see the
big pic
```




```
shelter <- shelter %>%
  mutate(Daytime = factor(ifelse(hour(shelter$DateTime) %in% c(5:8, 16:22),
1, 0)))

shelter[tv_row,] %>%
  group_by(Daytime) %>%
  summarize(count = n(), prob = mean(Outcome == 1))
```

```
## # A tibble: 2 x 3
##   Daytime count  prob
##   <fct>   <int> <dbl>
## 1 0       14586 0.285
## 2 1       12143 0.545
```

```r
ggplot(shelter[tv_row,], aes(x = Daytime, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Daytime VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)"))
```

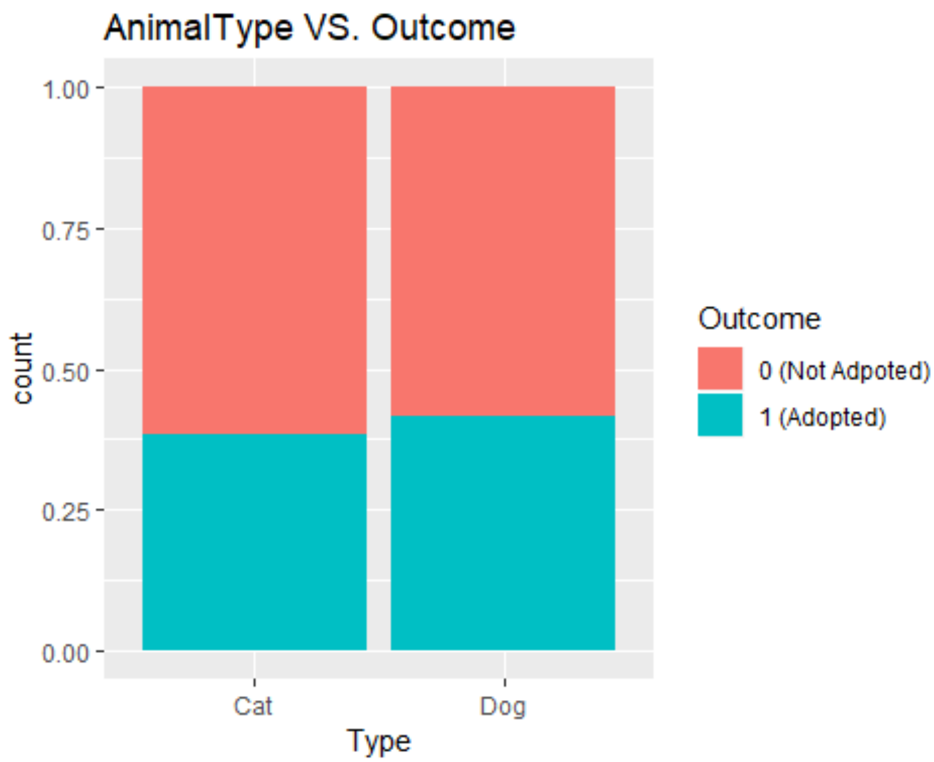



## 2.4 AnimalType

```r
shelter$Type <- factor(shelter$AnimalType)

shelter %$%
  summary(Type)
```

```
##   Cat   Dog
## 15934 22251
```

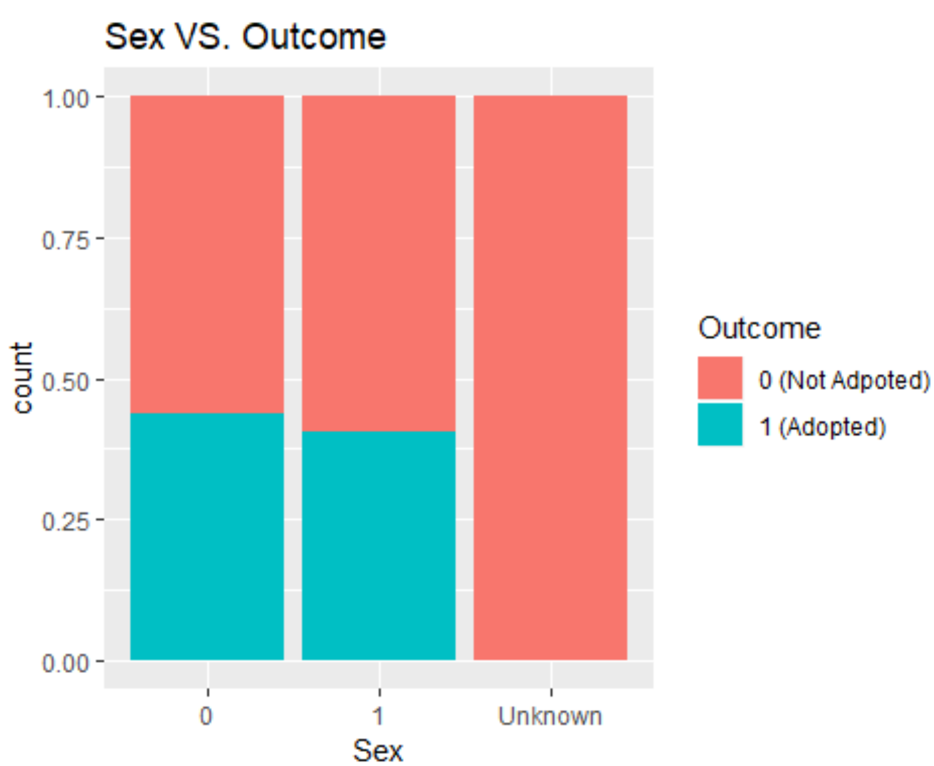
```r
ggplot(shelter[tv_row,], aes(x = Type, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("AnimalType VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)"))
```




## 2.5 SexuponOutcome

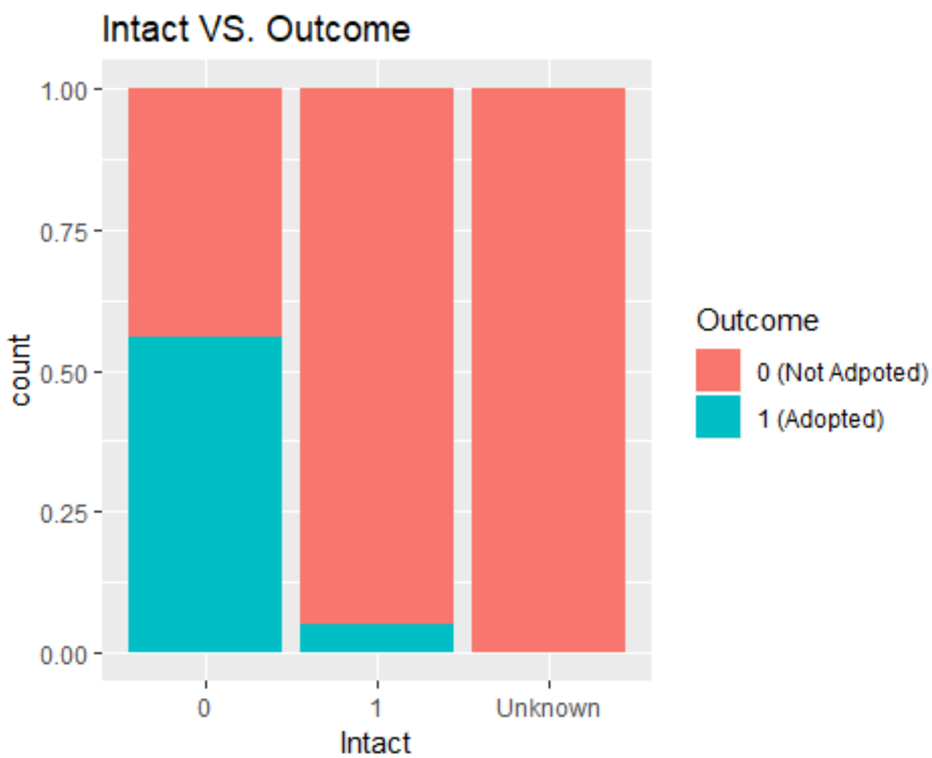
```
shelter$SexuponOutcome <- factor(shelter$SexuponOutcome)

shelter %$%
  summary(SexuponOutcome)

## Intact Female   Intact Male Neutered Male Spayed Female       Unknown 
##          5004          4985         14014         12633          1548 
##          NA's 
##             1

shelter <- shelter %>%
  mutate(
    Sex = ifelse(str_detect(shelter$SexuponOutcome, "Male"), 1, 0),
    Intact = ifelse(str_detect(shelter$SexuponOutcome, "Intact"), 1, 0)
  )

To_NA <- (shelter$SexuponOutcome == "Unknown" |
is.na(shelter$SexuponOutcome))

shelter$Sex <- ifelse(To_NA, NA, shelter$Sex) %>%
  factor() %>%
  fct_explicit_na(na_level = "Unknown")

shelter$Intact <- shelter$Intact <- ifelse(To_NA, NA, shelter$Intact) %>%
```

```
  factor() %>%
  fct_explicit_na(na_level = "Unknown")

shelter[tv_row,] %>%
  group_by(Sex) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 3 x 3
##   Sex     count  prob
##   <fct>   <int> <dbl>
## 1 0       12331 0.437
## 2 1       13304 0.404
## 3 Unknown  1094 0

# If Sex is Unknown, then Outcome is 0
ggplot(shelter[tv_row,], aes(x = Sex, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Sex VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)"))
```




```
shelter[tv_row,] %>%
  group_by(Intact) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 3 x 3
##   Intact  count   prob
##   <fct>   <int>  <dbl>
## 1 0       18599 0.560
```

```
## 2 1         7036 0.0513
## 3 Unknown   1094 0

# If Intact is Unknown, then Outcome is 0
ggplot(shelter[tv_row,], aes(x = Intact, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Intact VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)"))
```



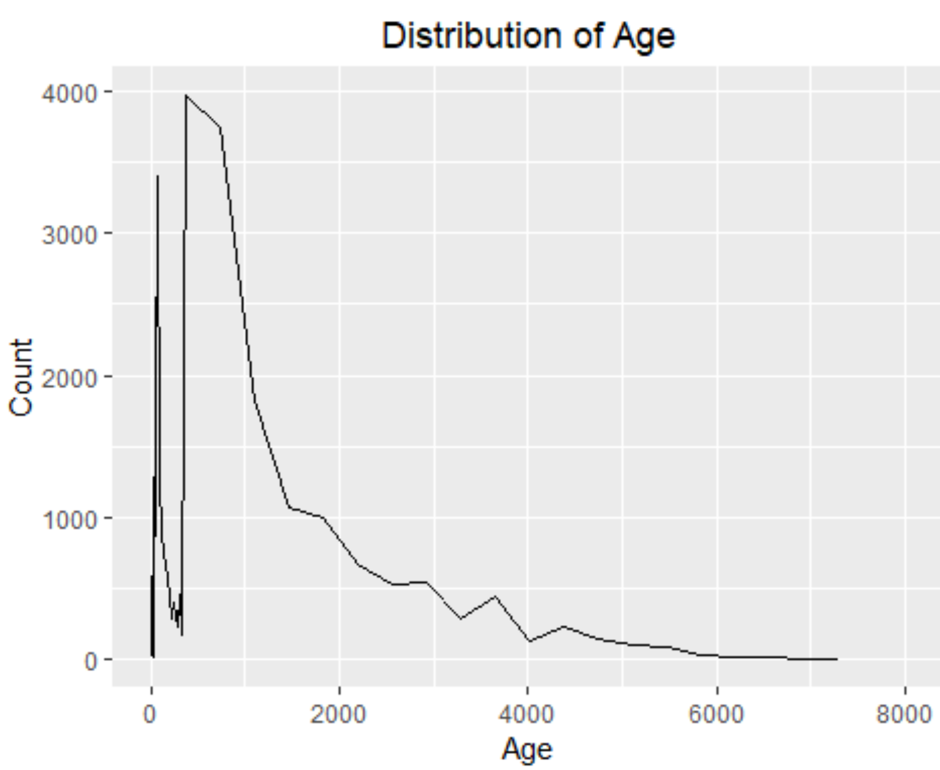

### 2.6 AgeuponOutcome

```
num_part <- shelter %$%
  sapply(AgeuponOutcome, function(x) str_split(x, ' ')[[1]][1]) %>%
  as.vector() %>%
  as.numeric()

unit_part <- shelter %$%
  sapply(AgeuponOutcome, function(x) str_split(x, ' ')[[1]][2]) %>%
  as.vector() %>%
  str_replace("s$", "")

age <- function(x, y){
  if(is.na(x) | is.na(y)){
    return(0)
  } else {
    return(round(as.numeric(duration(x, y)) / (60 * 60 * 24), 2))
  }
}
```

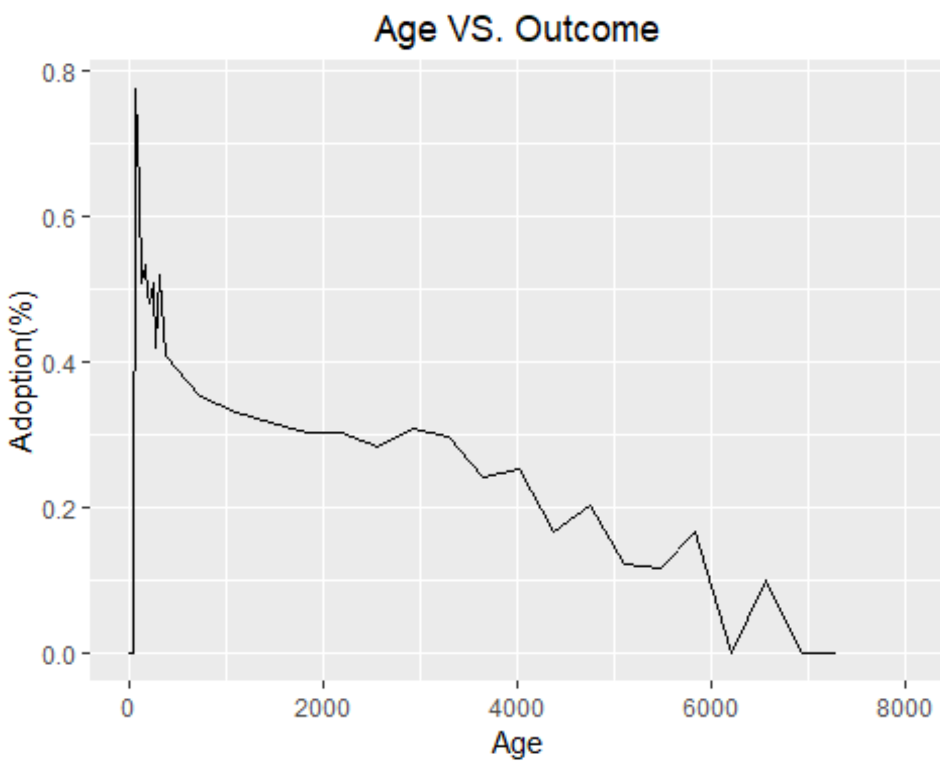
```r
temp <-  NULL
for (i in 1:nrow(shelter)){
  temp[i] <- age(num_part[i], unit_part[i])
}

shelter <- shelter %>%
  mutate(Age = round(temp, 1))

shelter %$%
  summary(Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    60.8   365.0   788.6  1095.0  8030.0
```

```r
# Anomaly Detection
shelter[tv_row,] %>%
  group_by(Age) %>%
  summarize(count = n(), mean = mean(Outcome == 1)) %>%
  ggplot(aes(Age, count)) +
  geom_line() +
  xlim(0, 8030) +
  ggtitle("Distribution of Age") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Age") +
  ylab("Count")
```



```r
# Age VS. Outcome
shelter[tv_row,] %>%
  group_by(Age) %>%
  summarize(count = n(), mean = mean(Outcome == 1)) %>%
  ggplot(aes(Age, mean)) +
  geom_line() +
  xlim(0, 8030) +
  ggtitle("Age VS. Outcome") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Age") +
  ylab("Adoption(%)")
```




### 2.7 Breed

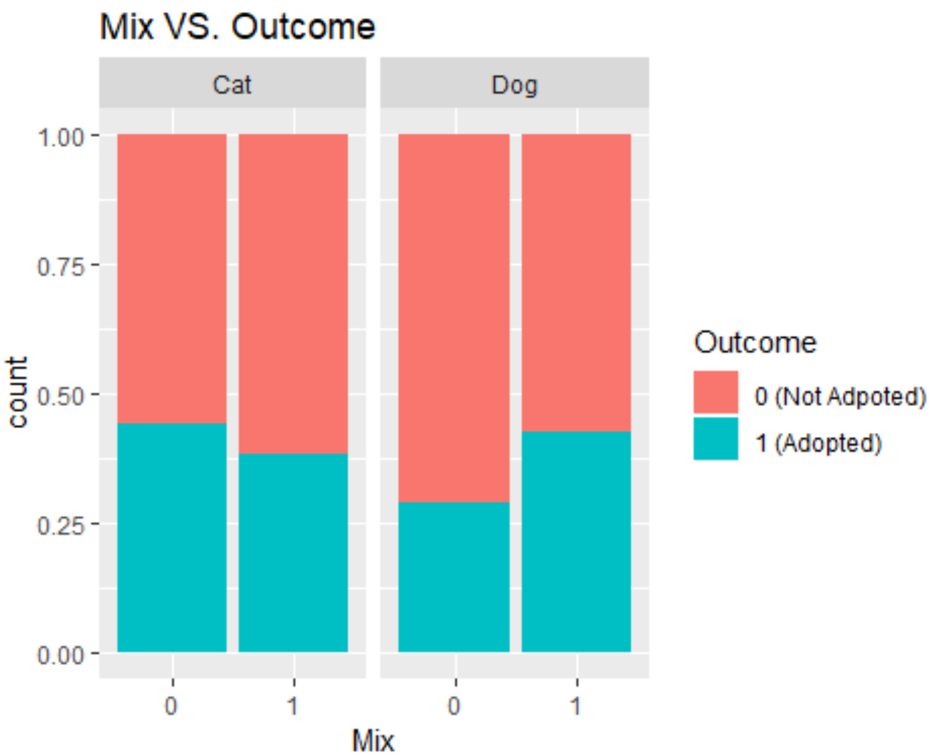
```r
length(unique(shelter$Breed)) # That's too many!

## [1] 1678
```

#### 2.7.1 Mix

```r
shelter <- shelter %>%
  mutate(Mix = factor(ifelse((str_detect(shelter$Breed, "(Mix|/)")),
                             1, 0)))

shelter[tv_row,] %>%
  group_by(Mix) %>%
  summarize(count = n(), prob = mean(Outcome == 1)) # too unbalanced
```

```
## # A tibble: 2 x 3
##   Mix   count  prob
##   <fct> <int> <dbl>
## 1 0      1391 0.318
## 2 1     25338 0.408
```

```
ggplot(shelter[tv_row,], aes(x = Mix, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Mix VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  facet_wrap(~Type)
```




### 2.7.2 Shorthair

```
shelter <- shelter %>%
  mutate(Shorthair = factor(ifelse(str_detect(shelter$Breed, "Shorthair"), 1,
0)))

shelter[tv_row,] %>%
  group_by(Shorthair) %>%
  summarize(count = n(), prob = mean(Outcome == 1)) # little help
```

```
## # A tibble: 2 x 3
##   Shorthair count  prob
##   <fct>     <int> <dbl>
## 1 0         15317 0.412
## 2 1         11412 0.390
```

```r
ggplot(shelter[tv_row,], aes(x = Shorthair, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Shorthair VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  facet_wrap(~Type)
```


### 2.7.3 Take Type into consideration

```r
shelter[tv_row,] %>%
  group_by(Type, Mix) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 4 x 4
## # Groups:   Type [2]
##   Type  Mix   count  prob
##   <fct> <fct> <int> <dbl>
## 1 Cat   0       253 0.443
## 2 Cat   1     10881 0.382
## 3 Dog   0      1138 0.291
## 4 Dog   1     14457 0.427

shelter[tv_row,] %>%
  group_by(Type, Shorthair) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 4 x 4
## # Groups:   Type [2]
##   Type  Shorthair count  prob
```

```
##   <fct> <fct>     <int> <dbl>
## 1 Cat   0          2145 0.435
## 2 Cat   1          8989 0.371
## 3 Dog   0         13172 0.409
## 4 Dog   1          2423 0.459
```

```
shelter[tv_row,] %>%
  group_by(Type, Mix, Shorthair) %>%
  summarize(count = n(), prob = mean(Outcome == 1))
```

```
## # A tibble: 8 x 5
## # Groups:   Type, Mix [4]
##   Type  Mix   Shorthair count  prob
##   <fct> <fct> <fct>     <int> <dbl>
## 1 Cat   0     0           109 0.523
## 2 Cat   0     1           144 0.382
## 3 Cat   1     0          2036 0.430
## 4 Cat   1     1          8845 0.371
## 5 Dog   0     0          1049 0.286
## 6 Dog   0     1            89 0.348
## 7 Dog   1     0         12123 0.419
## 8 Dog   1     1          2334 0.463
```

## 2.8 Color

```
length(unique(shelter$Color)) # That's a lot!
```

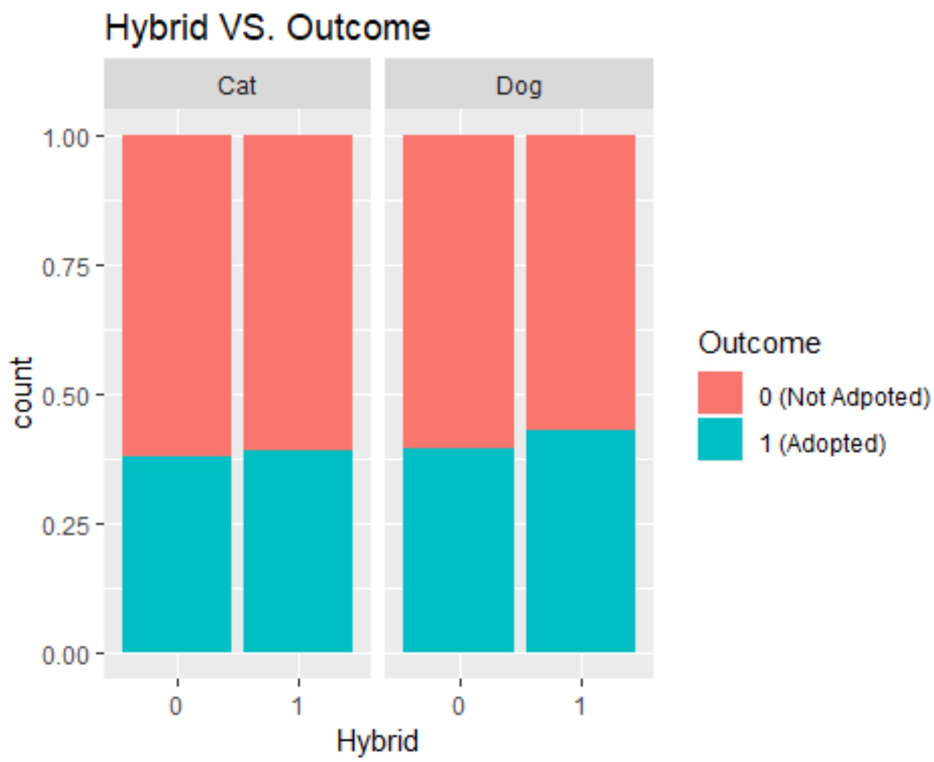
```
## [1] 411
```

### 2.8.1 Hybrid

```
shelter <- shelter %>%
  mutate(Hybrid = factor(ifelse(str_detect(shelter$Color, "/"), 1, 0)))

shelter[tv_row,] %>%
  group_by(Hybrid) %>%
  summarize(count = n(), prob = mean(Outcome == 1)) # little help
```

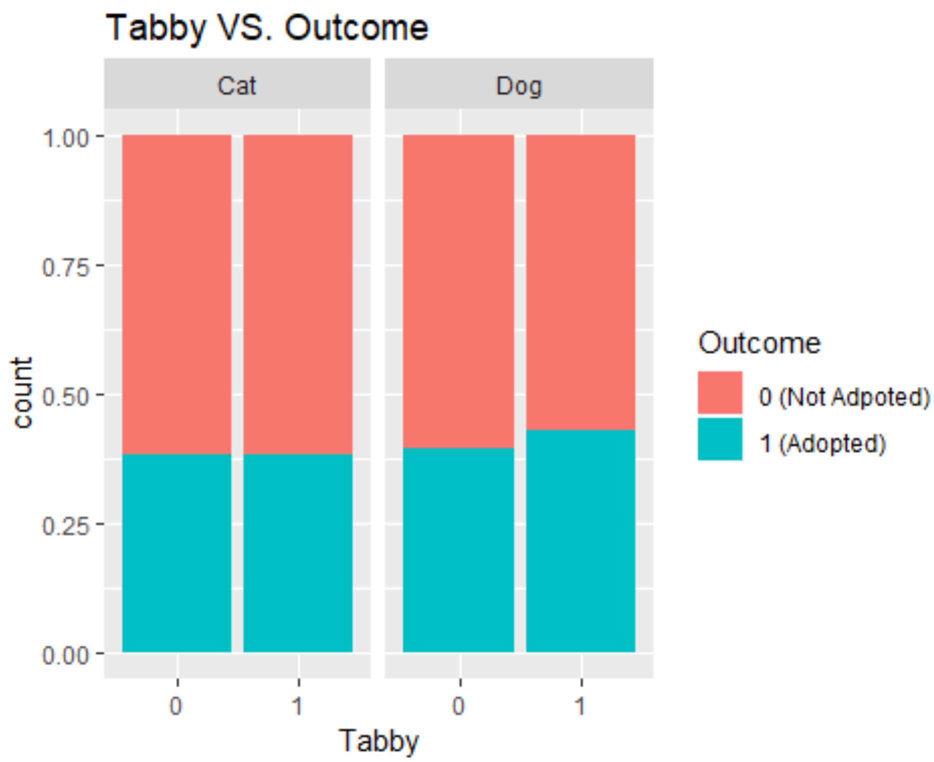
```
## # A tibble: 2 x 3
##   Hybrid count  prob
##   <fct>  <int> <dbl>
## 1 0      12805 0.387
## 2 1      13924 0.418
```

```
ggplot(shelter[tv_row,], aes(x = Hybrid, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Hybrid VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  facet_wrap(~Type)
```




### 2.8.2 Tabby

```
shelter <- shelter %>%
  mutate(Tabby = factor(ifelse(str_detect(shelter$Color, "(Tabby|/)"), 1,
0)))

shelter[tv_row,] %>%
  group_by(Tabby) %>%
  summarize(count = n(), prob = mean(Outcome == 1)) # little help

## # A tibble: 2 x 3
##   Tabby count  prob
##   <fct> <int> <dbl>
## 1 0      9593 0.390
## 2 1     17136 0.410

ggplot(shelter[tv_row,], aes(x = Tabby, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Tabby VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  facet_wrap(~Type)
```




### 2.8.3 Take Type into consideration

```
shelter[tv_row,] %>%
  group_by(Type, Hybrid) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 4 x 4
## # Groups:   Type [2]
##   Type  Hybrid count  prob
##   <fct> <fct>  <int> <dbl>
## 1 Cat   0       7207 0.380
## 2 Cat   1       3927 0.390
## 3 Dog   0       5598 0.395
## 4 Dog   1       9997 0.429

shelter[tv_row,] %>%
  group_by(Type, Tabby) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 4 x 4
## # Groups:   Type [2]
##   Type  Tabby count  prob
##   <fct> <fct> <int> <dbl>
## 1 Cat   0      3995 0.383
## 2 Cat   1      7139 0.384
## 3 Dog   0      5598 0.395
## 4 Dog   1      9997 0.429
```

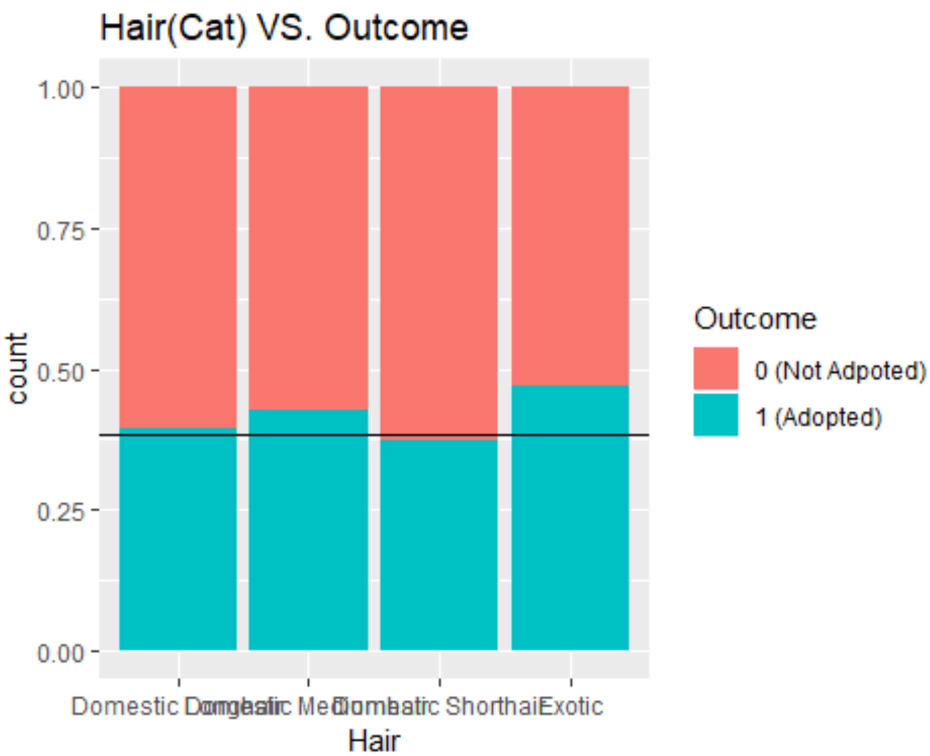
## 2.9 DO some magic

### 2.9.0 Preperation

```
Cat <- shelter %>%
  filter(Type == "Cat")
Not_NA_cat <- which(!is.na(Cat$Outcome))

Dog <- shelter %>%
  filter(Type == "Dog")
Not_NA_dog <- which(!is.na(Dog$Outcome))
```

### 2.9.1 Breed-Cat

```
Cat <- Cat %>%
  mutate(Breed = str_remove(Cat$Breed, "Mix")) %>%
  mutate(
    Hair = ifelse(str_detect(Cat$Breed, "/") | !str_detect(Cat$Breed,
"Domestic"), "Exotic",
                     ifelse(str_detect(Cat$Breed, "Shorthair"), "Domestic
Shorthair",
                            ifelse(str_detect(Cat$Breed, "Longhair"),
"Domestic Longhair",
                                            "Domestic Mediumhair")))
    )

Cat$Hair <- as.factor(Cat$Hair)

Cat[Not_NA_cat,] %>%
  group_by(Hair) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 4 x 3
##   Hair                count  prob
##   <fct>               <int> <dbl>
## 1 Domestic Longhair     543 0.396
## 2 Domestic Mediumhair   881 0.426
## 3 Domestic Shorthair   8953 0.372
## 4 Exotic                757 0.469

Cat[Not_NA_cat,] %>%
  ggplot(aes(x = Hair, y = ..count.., fill = Outcome)) +
    geom_bar(stat = "count", position = "fill") +
    ggtitle("Hair(Cat) VS. Outcome") +
    scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
    geom_hline(aes(yintercept = mean(Cat[Not_NA_cat,]$Outcome == 1)))
```

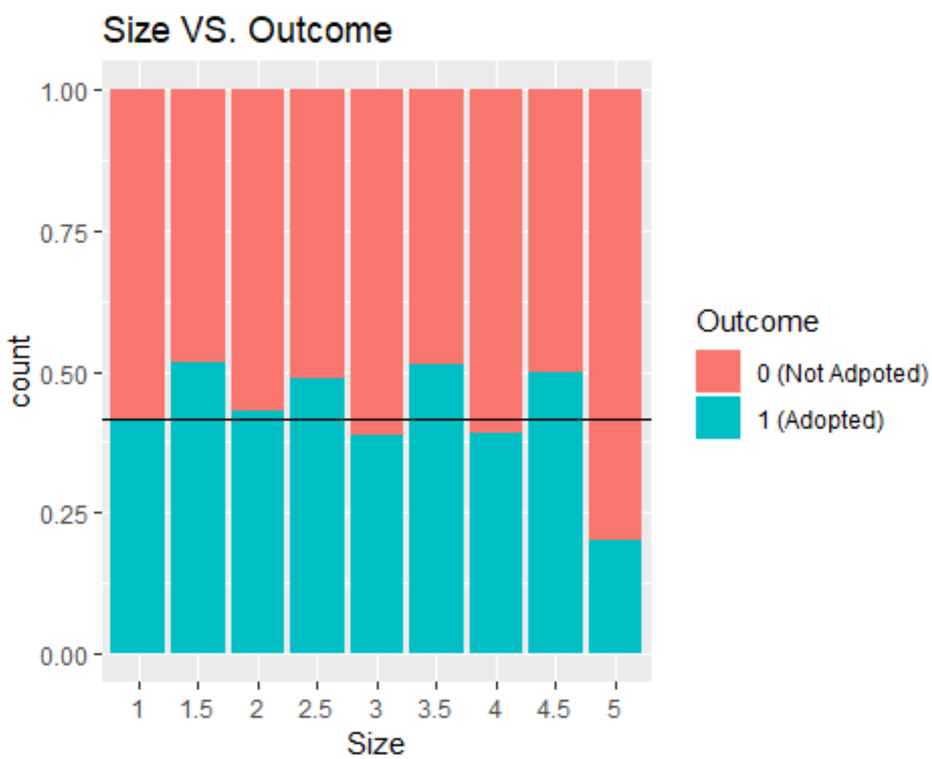



### 2.9.2 Breed-Dog

```
dog_breed <- dog_breed %>%
  mutate(Mark1 = ifelse(dog_breed$Point1 == 0, median(dog_breed$Point),
dog_breed$Point1),
         Mark2 = ifelse(dog_breed$Point2 == 0, median(dog_breed$Point),
dog_breed$Point2),
         Mark = (Mark1 + Mark2)/2)

Dog <- Dog %>%
  mutate(Size = as.factor(dog_breed$Mark))

Dog[Not_NA_dog,] %>%
  group_by(Size) %>%
  summarize(count = n(), prob = mean(Outcome == 1))

## # A tibble: 9 x 3
##   Size  count  prob
##   <fct> <int> <dbl>
## 1 1      2811 0.417
## 2 1.5     616 0.516
## 3 2      4268 0.431
## 4 2.5     862 0.487
## 5 3      6494 0.388
## 6 3.5     168 0.512
## 7 4       337 0.392
```

```
## 8 4.5       4 0.5
## 9 5        35 0.2
```

```
Dog[Not_NA_dog,] %>%
  ggplot(aes(x = Size, y = ..count.., fill = Outcome)) +
  geom_bar(stat = "count", position = "fill") +
  ggtitle("Size VS. Outcome") +
  scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
  geom_hline(aes(yintercept = mean(Dog[Not_NA_dog,]$Outcome == 1)))# see the
big pic
```



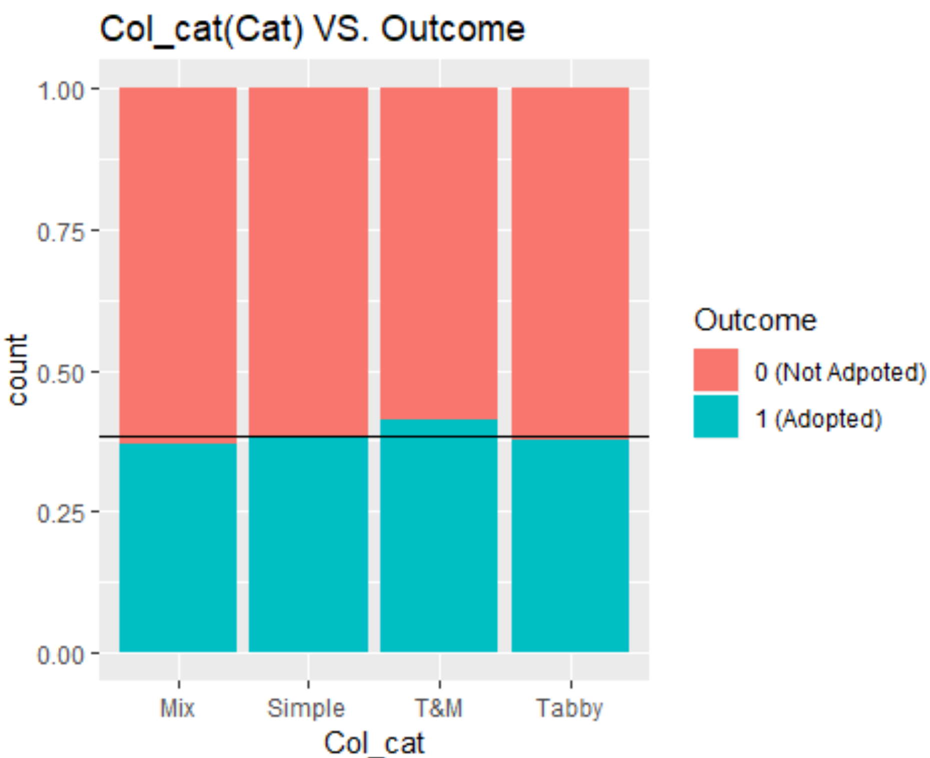

```
# coord_cartesian(ylim=c(5,15))
```

### 2.9.3 Color-Cat

```
Cat <- Cat %>%
  mutate(Col_cat = ifelse(str_detect(Cat$Color, "(Tabby)(/)"), "T&M",
                     ifelse(str_detect(Cat$Color, "/"), "Mix",
                            ifelse(str_detect(Cat$Color, "Tabby"), "Tabby",
                                   "Simple")))
         )

Cat$Col_cat <- as.factor(Cat$Col_cat)

Cat[Not_NA_cat,] %>%
  group_by(Col_cat) %>%
  summarize(count = n(), prob = mean(Outcome == 1))
```

```
## # A tibble: 4 x 3
##   Col_cat count  prob
##   <fct>   <int> <dbl>
## 1 Mix      2140 0.371
## 2 Simple   3995 0.383
## 3 T&M      1787 0.413
## 4 Tabby    3212 0.377
```

```
Cat[Not_NA_cat,] %>%
  ggplot(aes(x =Col_cat, y = ..count.., fill = Outcome)) +
    geom_bar(stat = "count", position = "fill") +
    ggtitle("Col_cat(Cat) VS. Outcome") +
    scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
    geom_hline(aes(yintercept = mean(Cat[Not_NA_cat,]$Outcome == 1)))
```




### 2.9.4 Color-Dog

```
temp_col <- c(as.vector(unlist(str_split(Dog[Dog$Hybrid==1,]$Color, "/"))),
                    as.vector(Dog[Dog$Hybrid==0,]$Color))
table(temp_col)
```

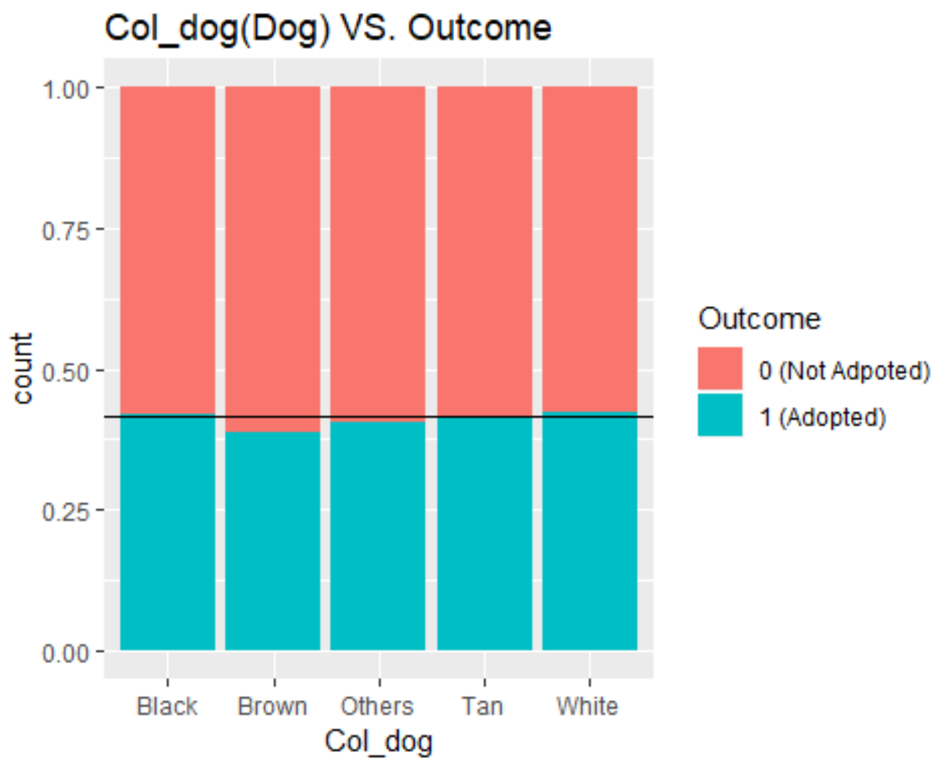
```
## temp_col
##        Apricot          Black  Black Brindle    Black Smoke    Black Tiger
##             45           7265            164              9              1
##           Blue     Blue Cream     Blue Merle     Blue Smoke      Blue Tick
##            845             10            273              2             72
##     Blue Tiger          Brown  Brown Brindle    Brown Merle    Brown Tabby
##             14           4276           1266            109              1
```

```
##    Brown Tiger           Buff      Chocolate          Cream           Fawn
##              3            431            687            387            294
##           Gold           Gray     Gray Tiger          Liver     Liver Tick
##            113            418              1             40              4
##         Orange           Pink            Red      Red Merle       Red Tick
##             29              3           1240             76             80
##          Ruddy          Sable         Silver            Tan       Tricolor
##              1            459            113           4370           1279
##          White         Yellow Yellow Brindle
##          11730            357             57
```

```
Dog <- Dog %>%
  mutate(
    Col_dog = ifelse(str_detect(Dog$Color, "White"), "White",
                       ifelse(str_detect(Dog$Color, "Black"), "Black",
                              ifelse(str_detect(Dog$Color, "Tan"), "Tan",
                                     ifelse(str_detect(Dog$Color, "Brown"),
"Brown",
                                            "Others"))))
    )

Dog$Col_dog <- as.factor(Dog$Col_dog)

Dog[Not_NA_dog,] %>%
  group_by(Col_dog) %>%
  summarize(count = n(), prob = mean(Outcome == 1))
```

```
## # A tibble: 5 x 3
##   Col_dog count  prob
##   <fct>   <int> <dbl>
## 1 Black    2868 0.420
## 2 Brown    1018 0.389
## 3 Others   2473 0.405
## 4 Tan      1006 0.413
## 5 White    8230 0.423
```

```
Dog[Not_NA_dog,] %>%
  ggplot(aes(x = Col_dog, y = ..count.., fill = Outcome)) +
    geom_bar(stat = "count", position = "fill") +
    ggtitle("Col_dog(Dog) VS. Outcome") +
    scale_fill_discrete(labels=c("0 (Not Adpoted)", "1 (Adopted)")) +
    geom_hline(aes(yintercept = mean(Dog[Not_NA_dog,]$Outcome == 1)))
```




### 2.10 Put them together

```
adoption <- shelter %>%
  dplyr::select(11:27, 10)
glimpse(adoption)

## Observations: 38,185
## Variables: 18
## $ Has_Name  <fct> 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0...
## $ Len_Name  <dbl> 7, 5, 6, 0, 0, 4, 5, 0, 4, 0, 0, 0, 6, 7, 7, 6, 0, 0...
## $ Year      <fct> 2014, 2013, 2015, 2014, 2013, 2014, 2015, 2015, 2014...
## $ Month     <fct> 2, 10, 1, 7, 11, 4, 3, 4, 2, 5, 12, 11, 2, 6, 11, 7,...
## $ Season    <fct> Winter, Autumn, Winter, Summer, Autumn, Spring, Wint...
## $ Day       <fct> 4, 1, 7, 6, 6, 6, 7, 5, 3, 7, 5, 2, 4, 2, 4, 7, 7, 7...
## $ Wday      <fct> 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0...
## $ Hour      <fct> 18, 12, 12, 19, 12, 13, 13, 17, 17, 7, 15, 14, 11, 1...
## $ Daytime   <fct> 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0...
## $ Type      <fct> Dog, Cat, Dog, Cat, Dog, Dog, Cat, Cat, Dog, Dog, Ca...
## $ Sex       <fct> 1, 0, 1, 1, 1, 0, 1, Unknown, 0, 0, Unknown, 0, 1, 1...
## $ Intact    <fct> 0, 0, 0, 1, 0, 1, 1, Unknown, 0, 0, Unknown, 0, 0, 0...
## $ Age       <dbl> 365.0, 365.0, 730.0, 21.0, 730.0, 30.4, 21.0, 21.0, ...
## $ Mix       <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1...
## $ Shorthair <fct> 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0...
## $ Hybrid    <fct> 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1...
## $ Tabby     <fct> 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1...
## $ Outcome   <fct> 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0...
```

```
cat_adoption <- Cat %>%
  dplyr::select(11:29, 10)
glimpse(cat_adoption)

## Observations: 15,934
## Variables: 20
## $ Has_Name  <fct> 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1...
## $ Len_Name  <dbl> 5, 0, 5, 0, 0, 6, 0, 7, 8, 5, 7, 3, 0, 0, 0, 4, 0, 5...
## $ Year      <fct> 2013, 2014, 2015, 2015, 2013, 2014, 2014, 2014, 2015...
## $ Month     <fct> 10, 7, 3, 4, 12, 7, 5, 5, 9, 10, 11, 7, 1, 7, 7, 8, ...
## $ Season    <fct> Autumn, Summer, Winter, Spring, Autumn, Summer, Spri...
## $ Day       <fct> 1, 6, 7, 5, 5, 7, 7, 7, 6, 7, 7, 5, 6, 5, 4, 3, 3, 7...
## $ Wday      <fct> 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0...
## $ Hour      <fct> 12, 19, 13, 17, 15, 12, 16, 11, 15, 15, 13, 14, 13, ...
## $ Daytime   <fct> 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1...
## $ Type      <fct> Cat, Cat, Cat, Cat, Cat, Cat, Cat, Cat, Cat, Cat, Ca...
## $ Sex       <fct> 0, 1, 1, Unknown, Unknown, 1, 1, 0, 0, 0, 0, 1, 0, 0...
## $ Intact    <fct> 0, 1, 1, Unknown, Unknown, 0, 1, 1, 0, 1, 0, 0, 0, 0...
## $ Age       <dbl> 365.0, 21.0, 21.0, 21.0, 730.0, 91.2, 21.0, 730.0, 3...
## $ Mix       <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ Shorthair <fct> 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1...
## $ Hybrid    <fct> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0...
## $ Tabby     <fct> 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1...
## $ Hair      <fct> Domestic Shorthair, Domestic Shorthair, Domestic Sho...
## $ Col_cat   <fct> Tabby, Simple, Tabby, Tabby, Simple, T&M, Tabby, Sim...
## $ Outcome   <fct> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1...

dog_adoption <- Dog %>%
  dplyr::select(11:29, 10)
glimpse(dog_adoption)

## Observations: 22,251
## Variables: 20
## $ Has_Name  <fct> 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1...
## $ Len_Name  <dbl> 7, 6, 0, 4, 4, 0, 0, 6, 7, 7, 0, 4, 6, 7, 7, 7, 3, 5...
## $ Year      <fct> 2014, 2015, 2013, 2014, 2014, 2014, 2013, 2016, 2015...
## $ Month     <fct> 2, 1, 11, 4, 2, 5, 11, 2, 6, 11, 6, 7, 1, 8, 10, 4, ...
## $ Season    <fct> Winter, Winter, Autumn, Spring, Winter, Spring, Autu...
## $ Day       <fct> 4, 7, 6, 6, 3, 7, 2, 4, 2, 4, 7, 4, 1, 7, 2, 4, 6, 4...
## $ Wday      <fct> 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1...
## $ Hour      <fct> 18, 12, 12, 13, 17, 7, 14, 11, 16, 15, 12, 17, 15, 1...
## $ Daytime   <fct> 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1...
## $ Type      <fct> Dog, Dog, Dog, Dog, Dog, Dog, Dog, Dog, Dog, Dog, Do...
## $ Sex       <fct> 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0...
## $ Intact    <fct> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0...
## $ Age       <dbl> 365.0, 730.0, 730.0, 30.4, 152.1, 365.0, 730.0, 1460...
## $ Mix       <fct> 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ Shorthair <fct> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1...
## $ Hybrid    <fct> 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1...
## $ Tabby     <fct> 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1...
```

```
## $ Size      <fct> 3, 3, 2, 1, 3, 1, 2, 3, 1, 4, 3, 2, 2, 2.5, 3, 3, 3,...
## $ Col_dog   <fct> White, White, Tan, Black, White, White, Others, Brow...
## $ Outcome   <fct> 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1...
```

## 3. Missing Data

```
adoption %>%
  mutate(Age = ifelse(adoption$Age == 0, NA, adoption$Age),
         Sex = ifelse(adoption$Sex == "Unknown", NA, adoption$Sex),
         Intact = ifelse(adoption$Intact == "Unknown", NA, adoption$Intact))
%>%
  miss_var_summary()
```

```
## # A tibble: 18 x 3
##    variable  n_miss pct_miss
##    <chr>      <int>    <dbl>
##  1 Outcome    11456   30.0
##  2 Sex         1549    4.06
##  3 Intact      1549    4.06
##  4 Age           59    0.155
##  5 Has_Name       0    0
##  6 Len_Name       0    0
##  7 Year           0    0
##  8 Month          0    0
##  9 Season         0    0
## 10 Day            0    0
## 11 Wday           0    0
## 12 Hour           0    0
## 13 Daytime        0    0
## 14 Type           0    0
## 15 Mix            0    0
## 16 Shorthair      0    0
## 17 Hybrid         0    0
## 18 Tabby          0    0
```

## 4. Feature Selection

```
lapply(adoption[tv_row,], function(x) IV(as.factor(x),
adoption$Outcome[tv_row]))
```

```
## $Has_Name
## [1] 0.262105
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Len_Name
## [1] 0.2679963
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Year
## [1] 0.007429457
```

```
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Month
## [1] 0.043693
## attr(,"howgood")
## [1] "Somewhat Predictive"
##
## $Season
## [1] 0.02821162
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Day
## [1] 0.1190329
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Wday
## [1] 0.1169087
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Hour
## [1] 0.4324247
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Daytime
## [1] 0.293723
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Type
## [1] 0.004562768
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Sex
## [1] 0.01175315
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Intact
## [1] 1.218538
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Age
## [1] 0.412179
```

```
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Mix
## [1] 0.007060048
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Shorthair
## [1] 0.002134874
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Hybrid
## [1] 0.004119418
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Tabby
## [1] 0.001567777
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Outcome
## [1] 0
## attr(,"howgood")
## [1] "Not Predictive"
```

```
lapply(cat_adoption[Not_NA_cat,], function(x) IV(as.factor(x),
cat_adoption$Outcome[Not_NA_cat]))
```

```
## $Has_Name
## [1] 1.112424
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Len_Name
## [1] 1.126505
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Year
## [1] 0.03523207
## attr(,"howgood")
## [1] "Somewhat Predictive"
##
## $Month
## [1] 0.1703459
## attr(,"howgood")
## [1] "Highly Predictive"
```

```
##
## $Season
## [1] 0.09328181
## attr(,"howgood")
## [1] "Somewhat Predictive"
##
## $Day
## [1] 0.1580237
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Wday
## [1] 0.1539276
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Hour
## [1] 0.7520755
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Daytime
## [1] 0.4276546
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Type
## [1] 0
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Sex
## [1] 0.0138755
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Intact
## [1] 1.727592
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Age
## [1] 0.7568352
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Mix
## [1] 0.001414264
## attr(,"howgood")
## [1] "Not Predictive"
```

```
##
## $Shorthair
## [1] 0.01104346
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Hybrid
## [1] 0.0004023263
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Tabby
## [1] 5.057958e-06
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Hair
## [1] 0.01326034
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Col_cat
## [1] 0.003240824
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Outcome
## [1] 0
## attr(,"howgood")
## [1] "Not Predictive"
```

```
lapply(dog_adoption[Not_NA_dog,], function(x) IV(as.factor(x),
dog_adoption$Outcome[Not_NA_dog]))
```

```
## $Has_Name
## [1] 0.003261239
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Len_Name
## [1] 0.01147657
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Year
## [1] 0.002203604
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Month
```

```
## [1] 0.0195553
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Season
## [1] 0.01111056
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Day
## [1] 0.09674868
## attr(,"howgood")
## [1] "Somewhat Predictive"
##
## $Wday
## [1] 0.09408025
## attr(,"howgood")
## [1] "Somewhat Predictive"
##
## $Hour
## [1] 0.3143306
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Daytime
## [1] 0.2104621
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Type
## [1] 0
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Sex
## [1] 0.01204246
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Intact
## [1] 0.9462186
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Age
## [1] 0.3282416
## attr(,"howgood")
## [1] "Highly Predictive"
##
## $Mix
```

```
## [1] 0.02246697
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Shorthair
## [1] 0.005525534
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Hybrid
## [1] 0.004312068
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Tabby
## [1] 0.004312068
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Size
## [1] 0.02167032
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Col_dog
## [1] 0.001594248
## attr(,"howgood")
## [1] "Not Predictive"
##
## $Outcome
## [1] 0
## attr(,"howgood")
## [1] "Not Predictive"
```

Result-based

```
adopt <- adoption %>%
  dplyr::select(c(1,5,7,9,10,12,13,18))
```

# 5. Modeling (Building & CV & Hyper-tuning)

## 5.1 Fit Control for Hyper-Tuning

```
fitControl <- trainControl(method = "repeatedcv",
                           number = 3,
                           repeats = 5,
                           search = "random")
```

## 5.2 rpart Missing Imputation

```
Age_NA <- which(adopt$Age == 0)
```

```r
training <- adopt[-Age_NA, -8]
testing <- adopt[Age_NA, -8]

tic()
mod_imp <- train(Age ~., training,
                 method = "rpart",
                 trControl = fitControl,
                 tuneLength = 5)
toc()

## 7.9 sec elapsed

adopt$Age[Age_NA] <- predict(mod_imp, testing)
```

### 5.3 Dummy Variables

```r
temp <- adopt[tv_row, ]

dmy <- dummyVars(~ ., data = temp[,-ncol(temp)])
dmy_data <- data.frame(predict(dmy, newdata = temp[,-ncol(temp)]))
dmy_data <- bind_cols(dmy_data, temp[,ncol(temp)])

levels(dmy_data$Outcome) <- c("Not_Adopted", "Adopted")
```

### 5.4 Stratified Sampling

```r
index <- createDataPartition(dmy_data$Outcome, p = .70, list = FALSE)
```

### 5.5 Advanced Modeling: Ensemble/Stacking

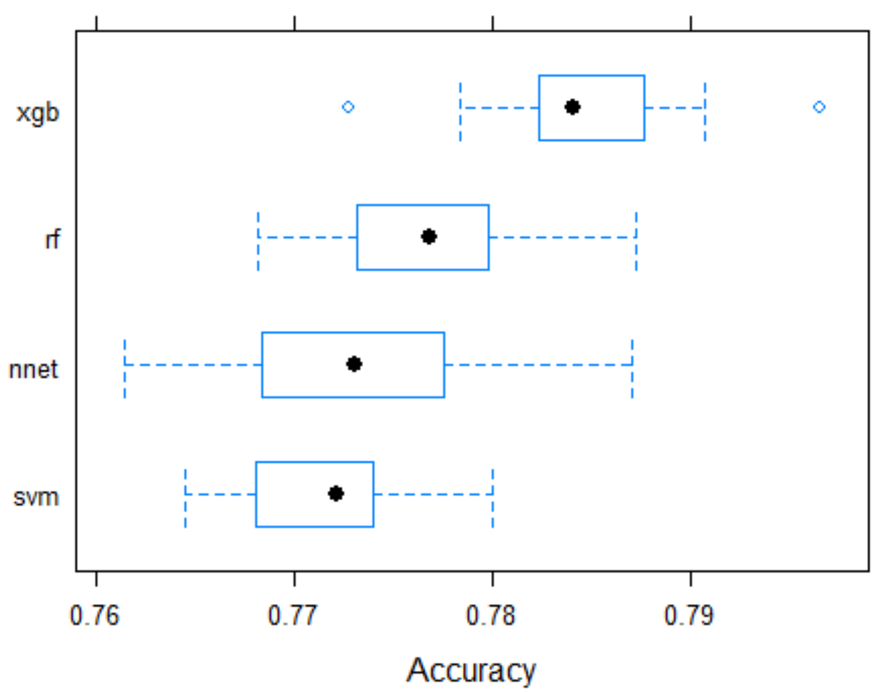
```r
stackControl <- trainControl(
  method = "boot",
  number = 25,
  savePredictions = "final",
  classProbs = TRUE,
  index = createResample(dmy_data[index,]$Outcome, 25)
)

# Need about 7000s to run
tic()
model_list <- caretList(
  Outcome ~ ., data = dmy_data[index,],
  trControl = stackControl,
  methodList = c("xgbTree", "rf", "svmRadial", "nnet")
)

toc()

## 8044.86 sec elapsed

tic()
glm_ensemble <- caretStack(
  model_list,
  method = "glm",
```

```
  trControl = trainControl(
    method = "boot",
    number = 10,
    savePredictions = "final",
    classProbs = TRUE
  )
)
toc()

## 15.19 sec elapsed

pred <- predict(glm_ensemble, dmy_data[-index,], type = "raw")
caret::confusionMatrix(pred, dmy_data[-index,]$Outcome, mode = "prec_recall")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Not_Adopted Adopted
##   Not_Adopted        4106     987
##   Adopted             682    2243
##
##                Accuracy : 0.7918
##                  95% CI : (0.7828, 0.8007)
##     No Information Rate : 0.5972
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5606
##
##  Mcnemar's Test P-Value : 9.974e-14
##
##               Precision : 0.8062
##                  Recall : 0.8576
##                      F1 : 0.8311
##              Prevalence : 0.5972
##          Detection Rate : 0.5121
##    Detection Prevalence : 0.6352
##       Balanced Accuracy : 0.7760
##
##        'Positive' Class : Not_Adopted
##

model_preds <-
  lapply(model_list, predict, newdata = dmy_data[-index,], type = "raw")
conf_matrix <-
  lapply(model_preds, caret::confusionMatrix, dmy_data[-index,]$Outcome, mode
= "prec_recall")
```

### 5.6 Model Comparison

```
mothod_list <- list(
  xgb = model_list$xgbTree,
  rf = model_list$rf,
```

```r
  svm = model_list$svmRadial,
  nnet = model_list$nnet
)

# Together
resamps <- resamples(mothod_list)
summary(resamps, metric = "Accuracy")

##
## Call:
## summary.resamples(object = resamps, metric = "Accuracy")
##
## Models: xgb, rf, svm, nnet
## Number of resamples: 25
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## xgb  0.7727601 0.7823947 0.7840860 0.7845170 0.7876055 0.7965532    0
## rf   0.7681937 0.7731387 0.7768166 0.7765909 0.7798416 0.7872061    0
## svm  0.7644816 0.7680397 0.7721850 0.7717739 0.7739419 0.7799650    0
## nnet 0.7614198 0.7684134 0.7730681 0.7724099 0.7775850 0.7870600    0

bwplot(resamps, metric = "Accuracy")
```




```r
# From the plot, it's reasonable to say that XGBoost performs best.
```

Sense-based

```r
adopt_Cat <- cat_adoption %>%
  dplyr::select(c(1,5,7,9,12,13,18,19,20))

adopt_Dog <- dog_adoption %>%
  dplyr::select(c(2,5,7,9,12,13,18,19,20))
```

# 6. Modeling (Building & CV & Hyper-tuning)

## 6.1 rpart Missing Imputation

### For cat

```r
Age_NA_c <- which(adopt_Cat$Age == 0)

training_c <- adopt_Cat[-Age_NA_c, -9]
testing_c <- adopt_Cat[Age_NA_c, -9]

tic()
mod_imp_c <- train(Age ~., training_c,
                 method = "rpart",
                 trControl = fitControl,
                 tuneLength = 5)
toc()

## 3.83 sec elapsed

adopt_Cat$Age[Age_NA_c] <- predict(mod_imp_c, testing_c)
```

### For dog

```r
Age_NA_d <- which(adopt_Dog$Age == 0)

training_d <- adopt_Dog[-Age_NA_d, -9]
testing_d <- adopt_Dog[Age_NA_d, -9]

tic()
mod_imp_d <- train(Age ~., training_d,
                   method = "rpart",
                   trControl = fitControl,
                   tuneLength = 5)
toc()

## 7.53 sec elapsed

adopt_Dog$Age[Age_NA_d] <- predict(mod_imp_d, testing_d)
```

## 6.2 Dummy Variables

```r
# For cat
temp_c <- cat_adoption[Not_NA_cat,]
```

```r
dmy_c <- dummyVars(~ ., data = temp_c[,-ncol(temp_c)])
dmy_data_c <- data.frame(predict(dmy_c, newdata = temp_c[,-ncol(temp_c)]))
dmy_data_c <- bind_cols(dmy_data_c, temp_c[,ncol(temp_c)])

levels(dmy_data_c$Outcome) <- c("Not_Adopted", "Adopted")

# For dog
temp_d <- dog_adoption[Not_NA_dog,]

dmy_d <- dummyVars(~ ., data = temp_d[,-ncol(temp)])
dmy_data_d <- data.frame(predict(dmy, newdata = temp_d[,-ncol(temp_d)]))
dmy_data_d <- bind_cols(dmy_data_d, temp_d[,ncol(temp_d)])

levels(dmy_data_d$Outcome) <- c("Not_Adopted", "Adopted")
```

### 6.3 Stratified Sampling

```r
index_c <- createDataPartition(dmy_data_c$Outcome, p = .70, list = FALSE)
index_d <- createDataPartition(dmy_data_d$Outcome, p = .70, list = FALSE)
```

### 6.4 Advanced Modeling: Ensemble/Stacking

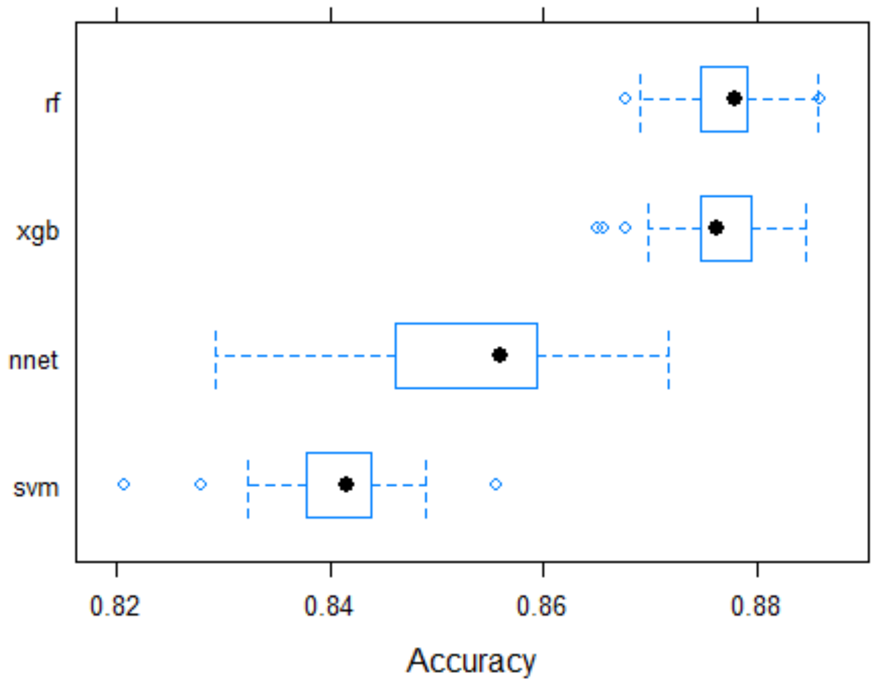
```r
stackControl_c <- trainControl(
  method = "boot",
  number = 25,
  savePredictions = "final",
  classProbs = TRUE,
  index = createResample(dmy_data_c[index_c,]$Outcome, 25)
)

stackControl_d <- trainControl(
  method = "boot",
  number = 25,
  savePredictions = "final",
  classProbs = TRUE,
  index = createResample(dmy_data_d[index_d,]$Outcome, 25)
)
```

## For cat

```r
# Need about 7000s to run
tic()
model_list_c <- caretList(
  Outcome ~ ., data = dmy_data_c[index_c,],
  trControl = stackControl_c,
  methodList = c("xgbTree", "rf", "svmRadial", "nnet")
)

toc()
```

```
## 3938.5 sec elapsed

tic()
glm_ensemble_c <- caretStack(
  model_list_c,
  method = "glm",
  trControl = trainControl(
    method = "boot",
    number = 10,
    savePredictions = "final",
    classProbs = TRUE
  )
)
toc()

## 6.71 sec elapsed

pred_c <- predict(glm_ensemble_c, dmy_data_c[-index_c,], type = "raw")
caret::confusionMatrix(pred_c, dmy_data_c[-index_c,]$Outcome, mode =
"prec_recall")

## Confusion Matrix and Statistics
##
##                Reference
## Prediction     Not_Adopted Adopted
##   Not_Adopted        1886     191
##   Adopted             172    1090
##
##                Accuracy : 0.8913
##                  95% CI : (0.8802, 0.9016)
##     No Information Rate : 0.6164
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7695
##
##  Mcnemar's Test P-Value : 0.3448
##
##               Precision : 0.9080
##                  Recall : 0.9164
##                      F1 : 0.9122
##              Prevalence : 0.6164
##          Detection Rate : 0.5648
##    Detection Prevalence : 0.6220
##       Balanced Accuracy : 0.8837
##
##        'Positive' Class : Not_Adopted
##

model_preds_c <-
  lapply(model_list_c, predict, newdata = dmy_data_c[-index_c,], type =
"raw")
```

```r
conf_matrix_c <-
  lapply(model_preds_c, caret::confusionMatrix, dmy_data_c[-
index_c,]$Outcome, mode = "prec_recall")

mothod_list_c <- list(
  xgb = model_list_c$xgbTree,
  rf = model_list_c$rf,
  svm = model_list_c$svmRadial,
  nnet = model_list_c$nnet
)

# Together
resamps_c <- resamples(mothod_list_c)
summary(resamps_c, metric = "Accuracy")

##
## Call:
## summary.resamples(object = resamps_c, metric = "Accuracy")
##
## Models: xgb, rf, svm, nnet
## Number of resamples: 25
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## xgb  0.8649405 0.8746920 0.8762530 0.8759676 0.8793651 0.8844677    0
## rf   0.8676522 0.8746489 0.8778866 0.8774731 0.8791402 0.8859369    0
## svm  0.8206186 0.8377707 0.8414763 0.8403476 0.8437390 0.8554974    0
## nnet 0.8292174 0.8460457 0.8558496 0.8540332 0.8593422 0.8716049    0

bwplot(resamps_c, metric = "Accuracy")
```




## For dog

```r
# Need about 4000s to run
tic()
model_list_d <- caretList(
  Outcome ~ ., data = dmy_data_d[index_d,],
  trControl = stackControl_d,
  methodList = c("xgbTree", "rf", "svmRadial", "nnet")
)

toc()

## 3336.42 sec elapsed

tic()
glm_ensemble_d <- caretStack(
  model_list_d,
  method = "glm",
  trControl = trainControl(
    method = "boot",
    number = 10,
    savePredictions = "final",
    classProbs = TRUE
  )
)
toc()
```

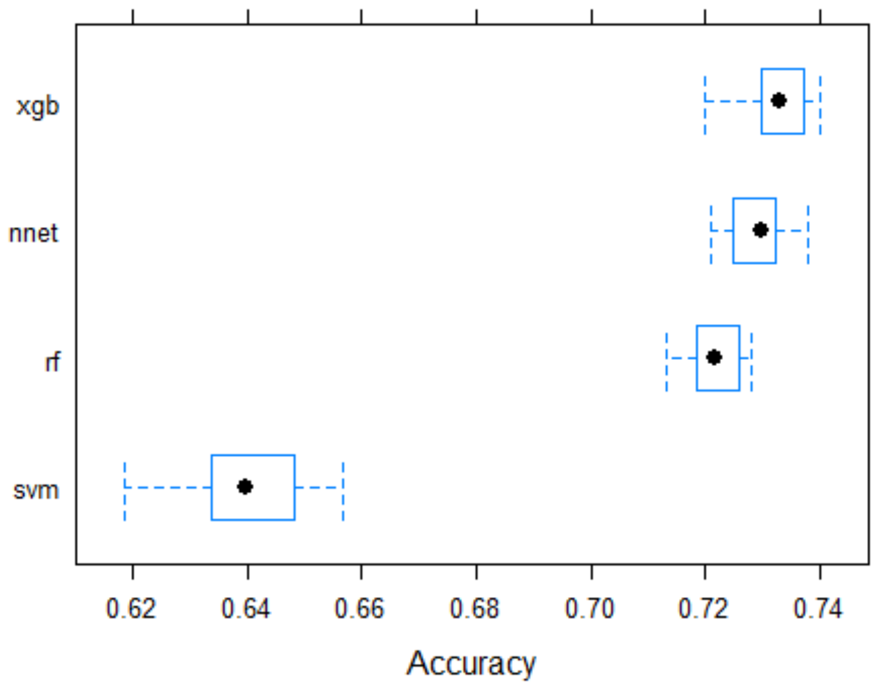
```
## 9 sec elapsed
```

```r
pred_d <- predict(glm_ensemble_d, dmy_data_d[-index_d,], type = "raw")
caret::confusionMatrix(pred_d, dmy_data_d[-index_d,]$Outcome, mode =
"prec_recall")
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction    Not_Adopted Adopted
##   Not_Adopted         708    1338
##   Adopted            2021     611
##
##                Accuracy : 0.282
##                  95% CI : (0.2691, 0.2951)
##     No Information Rate : 0.5834
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.4067
##
##  Mcnemar's Test P-Value : <2e-16
##
##               Precision : 0.3460
##                  Recall : 0.2594
##                      F1 : 0.2965
##              Prevalence : 0.5834
##          Detection Rate : 0.1513
##    Detection Prevalence : 0.4374
##       Balanced Accuracy : 0.2865
##
##        'Positive' Class : Not_Adopted
##
```

```r
model_preds_d <-
  lapply(model_list_d, predict, newdata = dmy_data_d[-index_d,], type =
"raw")
conf_matrix_d <-
  lapply(model_preds_d, caret::confusionMatrix, dmy_data_d[-
index_d,]$Outcome, mode = "prec_recall")

mothod_list_d <- list(
  xgb = model_list_d$xgbTree,
  rf = model_list_d$rf,
  svm = model_list_d$svmRadial,
  nnet = model_list_d$nnet
)

# Together
resamps_d <- resamples(mothod_list_d)
summary(resamps_d, metric = "Accuracy")
```

```
##
## Call:
## summary.resamples(object = resamps_d, metric = "Accuracy")
##
## Models: xgb, rf, svm, nnet
## Number of resamples: 25
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## xgb  0.7198599 0.7297297 0.7331174 0.7329162 0.7372354 0.7401615    0
## rf   0.7133550 0.7185583 0.7217933 0.7221589 0.7257742 0.7281481    0
## svm  0.6184840 0.6337253 0.6397407 0.6401194 0.6481203 0.6565934    0
## nnet 0.7208665 0.7249489 0.7297634 0.7288340 0.7322540 0.7378446    0

bwplot(resamps_d, metric = "Accuracy")
```




## 6. Conclusion

According to the results, it's better to use "Cat Model"(Stacking Model, which performs best) to predict cats' adoption, and use "Full Model"(XGBoost, which is fast but also generates good results) to predict dogs' adoption.