# Team2_BANA288_Final_Project.R

andrewgatchalian

2024-03-12

```
# Predictive Analytics Final Project
# BA 288 Winter 2024
# Team 2: Andrew Gatchalian, Davidson Rajasekar,
# Devin Xiang Tian, Kaidi Huang, Yuanhui Yao

setwd("/Users/andrewgatchalian/Documents/UCI MSBA 24/Winter Quarter/BA 288 Predictive An
alytics/Final Project/data")

####################################################
######## EXPLORATORY DATA ANALYSIS ################
###################################################
data <- read.csv("cat_data_cleaned_updated_EDA.csv")

library(dlookr)
```

```
## Registered S3 methods overwritten by 'dlookr':
##   method          from  
##   plot.transform  scales
##   print.transform scales
```

```
## 
## Attaching package: 'dlookr'
```

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

```
library(dplyr)
```

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

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

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

```r
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:dlookr':
##
##     describe
```

```r
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```r
#shape
print(ncol(data))
```

```
## [1] 19
```

```r
print(nrow(data))
```

```
## [1] 28208
```

```r
#sample
data_sample <- data %>% sample_n(5)
print(data_sample)
```

```
##   animal_id    sex Spay.Neuter outcome_age_.years. Cat.Kitten..outcome.
## 1   A727980   Male         Yes          2.00000000                  Cat
## 2   A711038   Male         Yes          3.00000000                  Cat
## 3   A747249 Female          No          0.05753425               Kitten
## 4   A733565 Female         Yes          0.03835616               Kitten
## 5   A705761 Female         Yes          5.00000000                  Cat
##   outcome_weekday outcome_hour              breed1 cfa_breed coat_pattern
## 1        Saturday           16  domestic shorthair     False        tabby
## 2          Friday           15          maine coon      True        tabby
## 3       Wednesday           14             siamese      True        solid
## 4       Wednesday            0  domestic shorthair     False        solid
## 5          Sunday           10 domestic mediumhair     False        tabby
##     coat has_name  is_adopted season is_weekend time_of_day  is_shorthair
## 1 orange      yes     Adopted Summer    Weekend   Afternoon     shorthair
## 2  brown      yes Not Adopted   Fall    Weekday   Afternoon not shorthair
## 3  cream       no Not Adopted Spring    Weekday     Morning not shorthair
## 4   blue       no Not Adopted Summer    Weekday      Closed     shorthair
## 5 orange       no Not Adopted Summer    Weekend      Closed not shorthair
##    is_solid_pattern color
## 1 non solid pattern other
## 2 non solid pattern other
## 3     solid pattern other
## 4     solid pattern other
## 5 non solid pattern other
```

```
# Get column names and data types
col_names <- names(data)
col_names
```

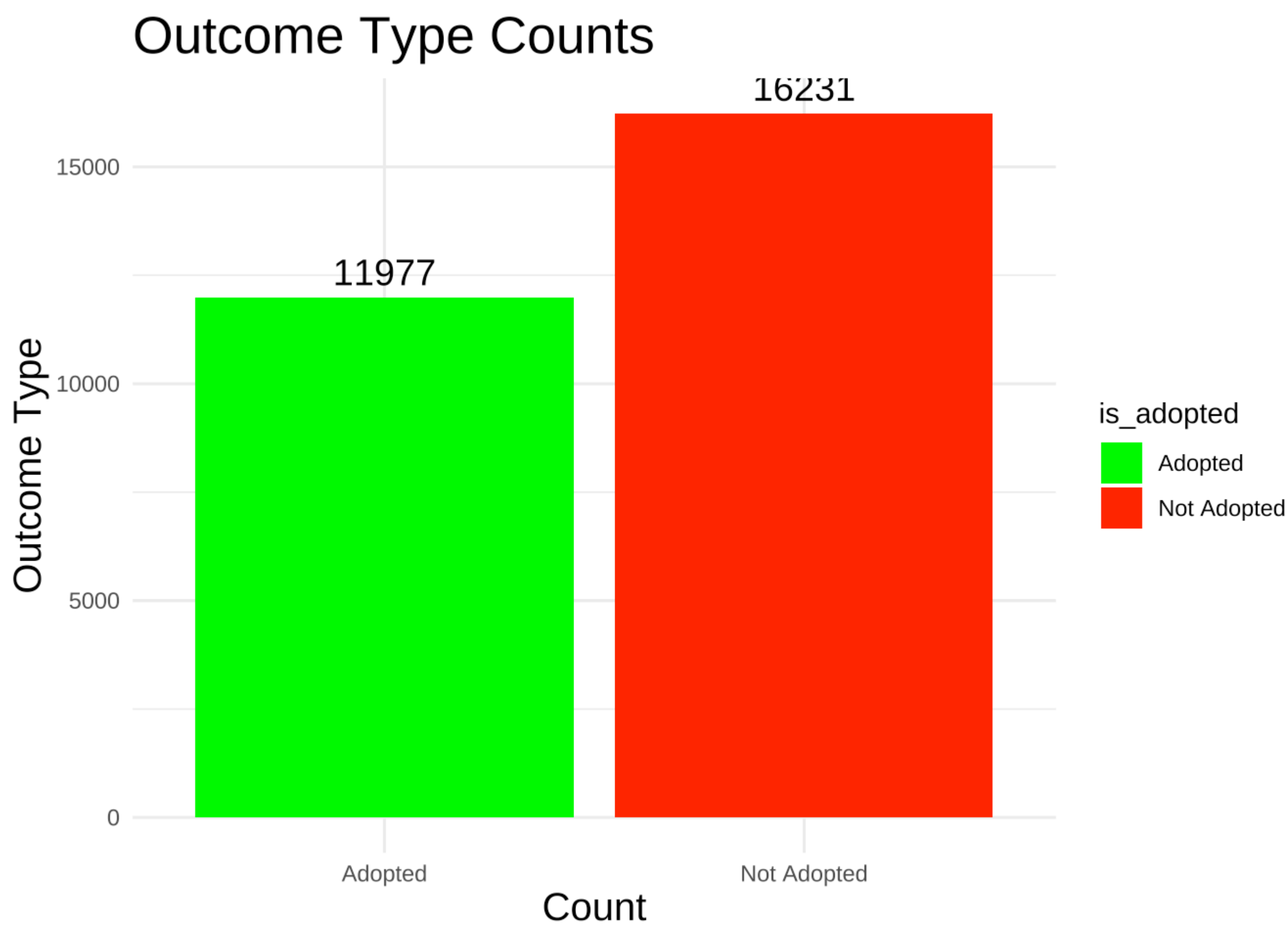
```
##  [1] "animal_id"            "sex"                  "Spay.Neuter"
##  [4] "outcome_age_.years."  "Cat.Kitten..outcome." "outcome_weekday"
##  [7] "outcome_hour"         "breed1"               "cfa_breed"
## [10] "coat_pattern"         "coat"                 "has_name"
## [13] "is_adopted"           "season"               "is_weekend"
## [16] "time_of_day"          "is_shorthair"         "is_solid_pattern"
## [19] "color"
```

```
col_types <- sapply(data, class)

# Get non-null counts for each column
non_null_counts <- sapply(data, function(x) sum(!is.na(x)))

# Create a dataframe to display the summary
df_info <- data.frame(Column_Name = col_names,
                      Data_Type = col_types,
                      Non_Null_Count = non_null_counts)

print(df_info)
```

```
##                                Column_Name Data_Type Non_Null_Count
## animal_id                        animal_id character          28208
## sex                                    sex character          28208
## Spay.Neuter                    Spay.Neuter character          28208
## outcome_age_.years.   outcome_age_.years.   numeric          28208
## Cat.Kitten..outcome. Cat.Kitten..outcome. character          28208
## outcome_weekday            outcome_weekday character          28208
## outcome_hour                  outcome_hour   integer          28208
## breed1                              breed1 character          28208
## cfa_breed                        cfa_breed character          28208
## coat_pattern                  coat_pattern character          28208
## coat                                  coat character          28208
## has_name                          has_name character          28208
## is_adopted                      is_adopted character          28208
## season                              season character          28208
## is_weekend                      is_weekend character          28208
## time_of_day                    time_of_day character          28208
## is_shorthair                  is_shorthair character          28208
## is_solid_pattern          is_solid_pattern character          28208
## color                                color character          28208
```

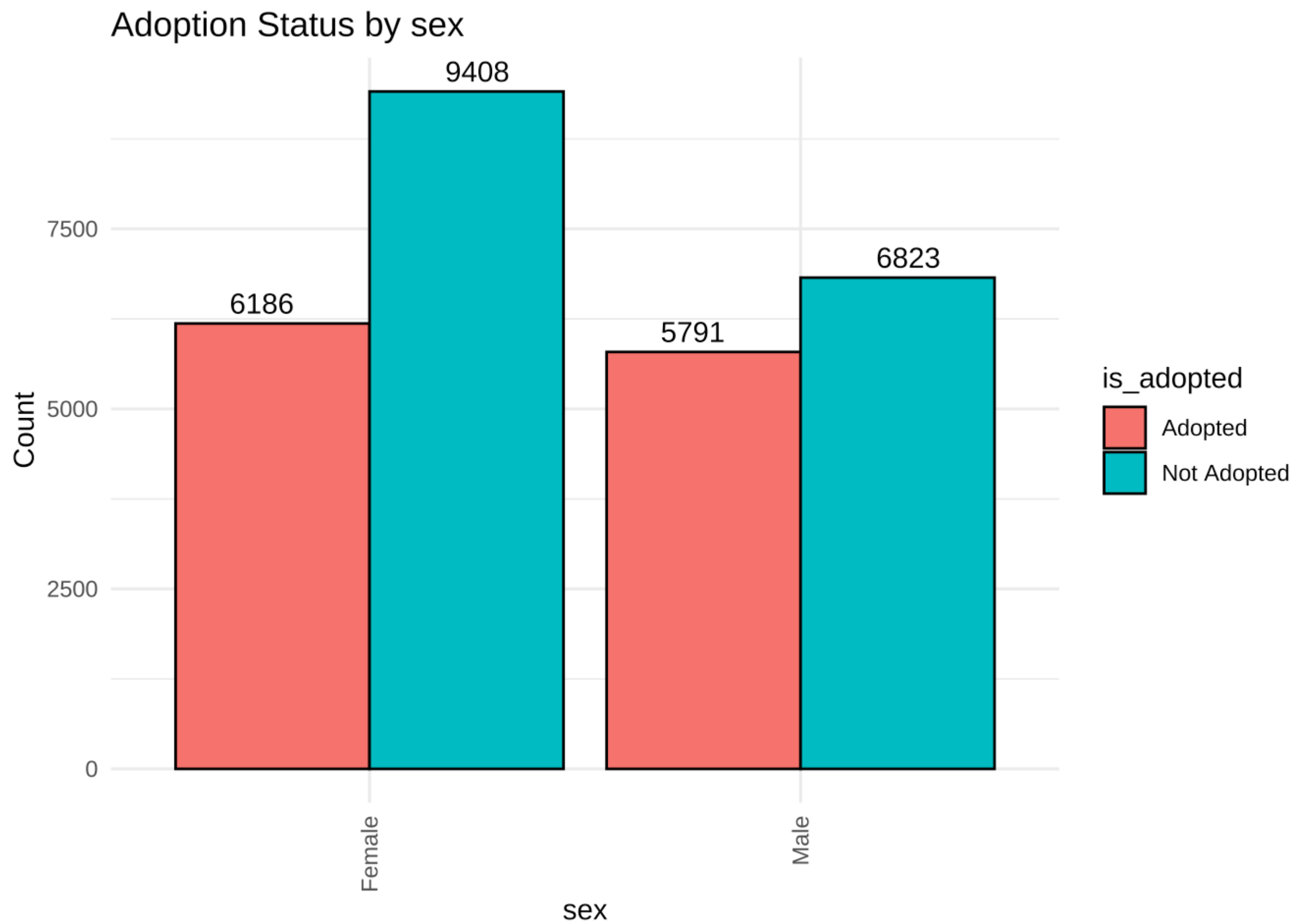
```
#Our independent variable is whether the cat is adopted or not.
#Thus, we will use that as the focus of our visualisations

# Count plot of adopted vs not adopted
plot1 <- ggplot(data = data, aes(x = is_adopted, fill = is_adopted)) +
  geom_bar() +
  geom_text(stat = 'count', aes(label = ..count..), vjust = -0.5, size = 5) +
  scale_fill_manual(values = c("Not Adopted" = "red", "Adopted" = "green")) +
  theme_minimal() +
  ggtitle("Outcome Type Counts") +
  labs(y = "Outcome Type", x = "Count") +
  theme(plot.title = element_text(size = 20),
        axis.title.x = element_text(size = 15),
        axis.title.y = element_text(size = 15))

# Show the plot
print(plot1)
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

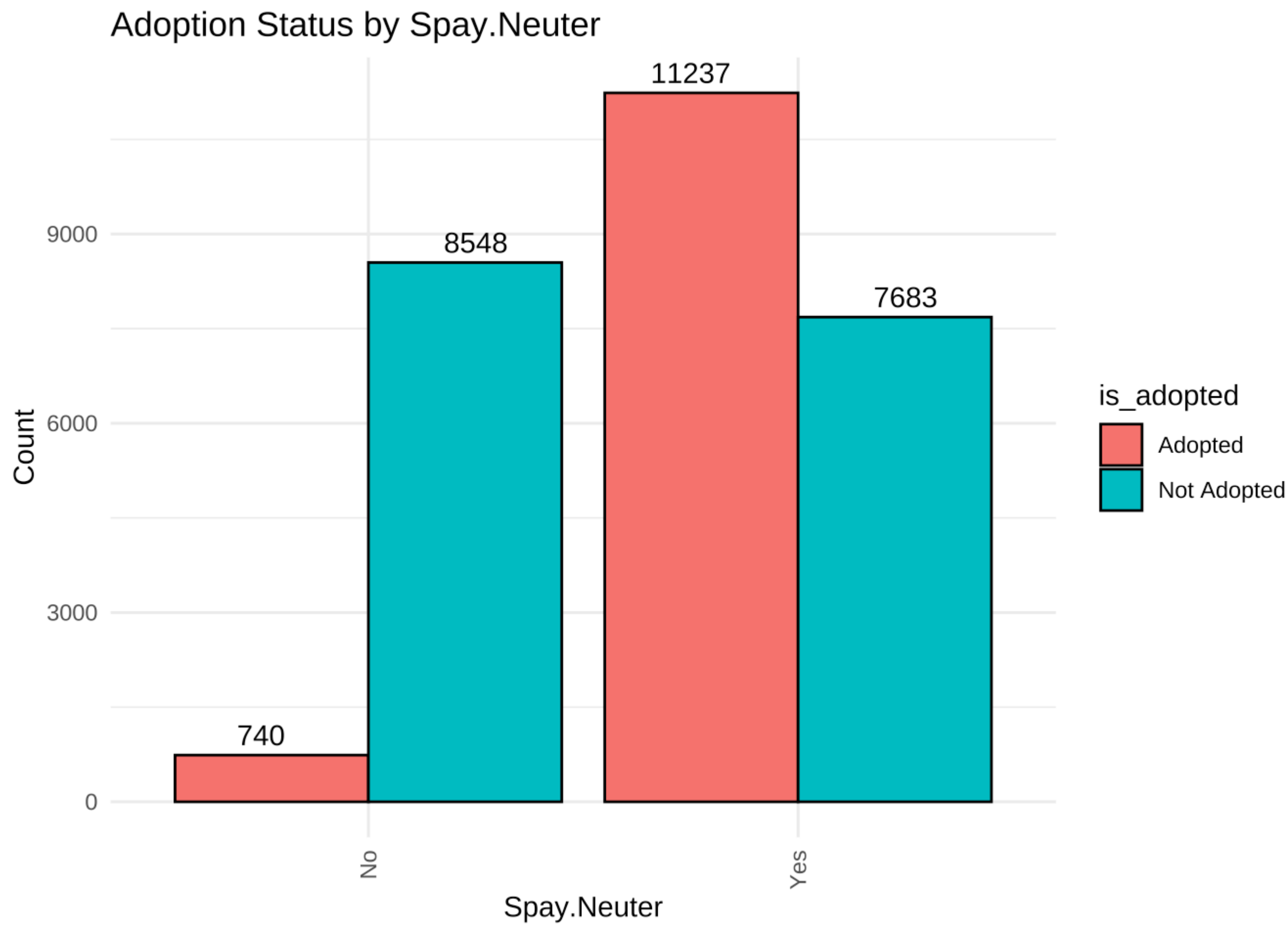

Outcome Type Counts
16231
11977
15000
10000
5000
0
Outcome Type
is_adopted
Adopted
Not Adopted
Adopted
Not Adopted
Count

```
# Function to create plot for each column that isn't time related
create_plot <- function(column) {
  plot_data <- data.frame(column = data[[column]], is_adopted = data$is_adopted)
  if (is.numeric(plot_data$column)) {
    plot <- ggplot(plot_data, aes(x = column, fill = is_adopted)) +
      geom_bar(position = "dodge", color = "black") +
      labs(title = paste("Adoption Status by", column),
           x = column,
           y = "Count") +
      theme_minimal() +
      geom_text(stat = "count", aes(label = ..count..), position = position_dodge(width
= 1), vjust = -0.5) +
      scale_x_continuous(labels = scales::comma) +  # Format numeric labels with comma
      theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
  } else {
    plot <- ggplot(plot_data, aes(x = factor(column), fill = is_adopted)) +
      geom_bar(position = "dodge", color = "black") +
      labs(title = paste("Adoption Status by", column),
           x = column,
           y = "Count") +
      theme_minimal() +
      geom_text(stat = "count", aes(label = ..count..), position = position_dodge(width
= 1), vjust = -0.5) +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
  }
  return(plot)
}

# Create plots for each column
plots <- lapply(c("sex", "Spay.Neuter", "Cat.Kitten..outcome.",
                  "breed1", "cfa_breed",
                  "coat_pattern", "coat", "has_name",
                  "is_shorthair", "is_solid_pattern", "color"), create_plot)

# Print plots
print(plots)
```

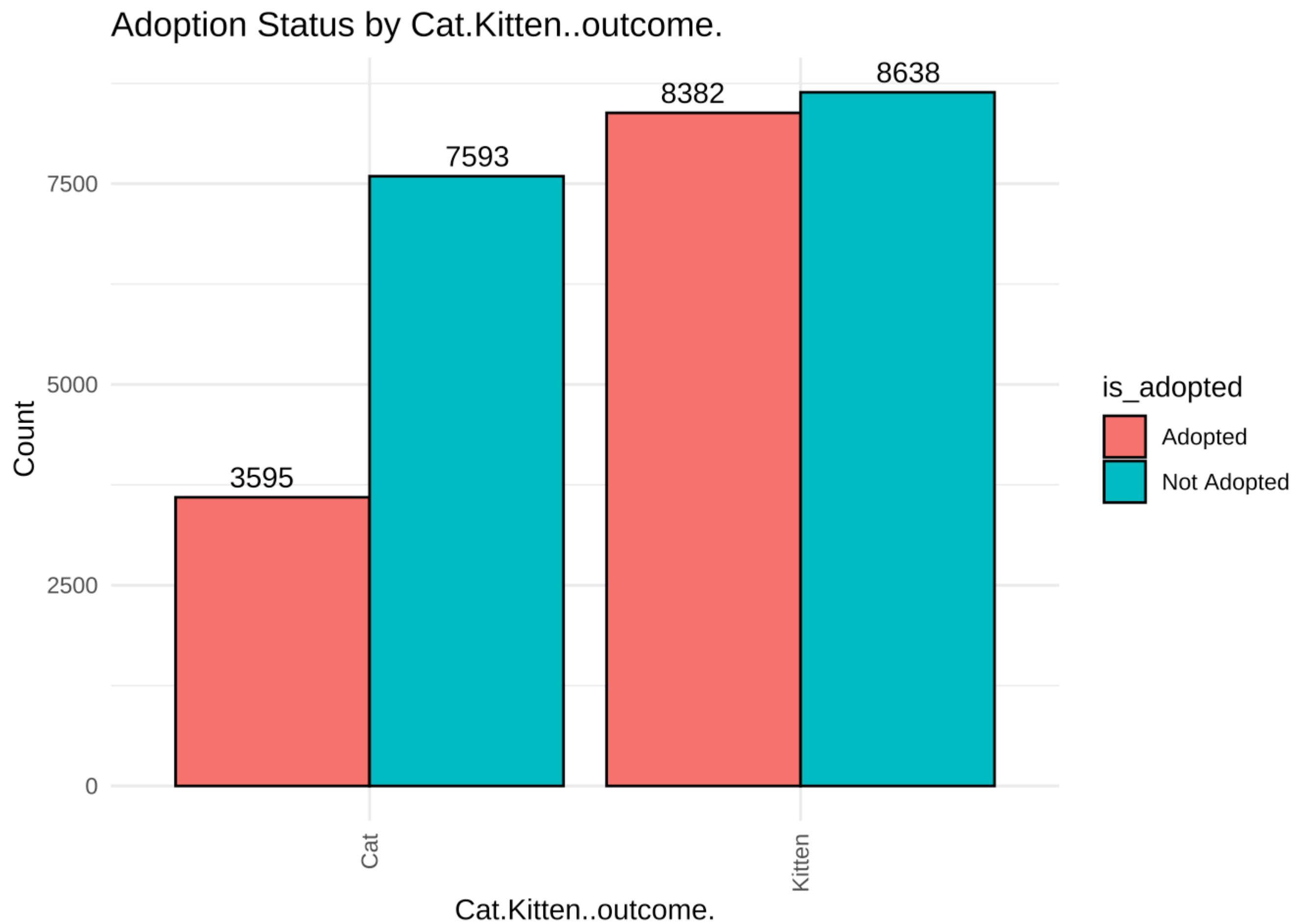
```
## [[1]]
```


Adoption Status by sex
9408
7500
6186
6823
5791
5000
2500
0
Count
is_adopted
Adopted
Not Adopted
Female
Male
sex

```
##
## [[2]]
```

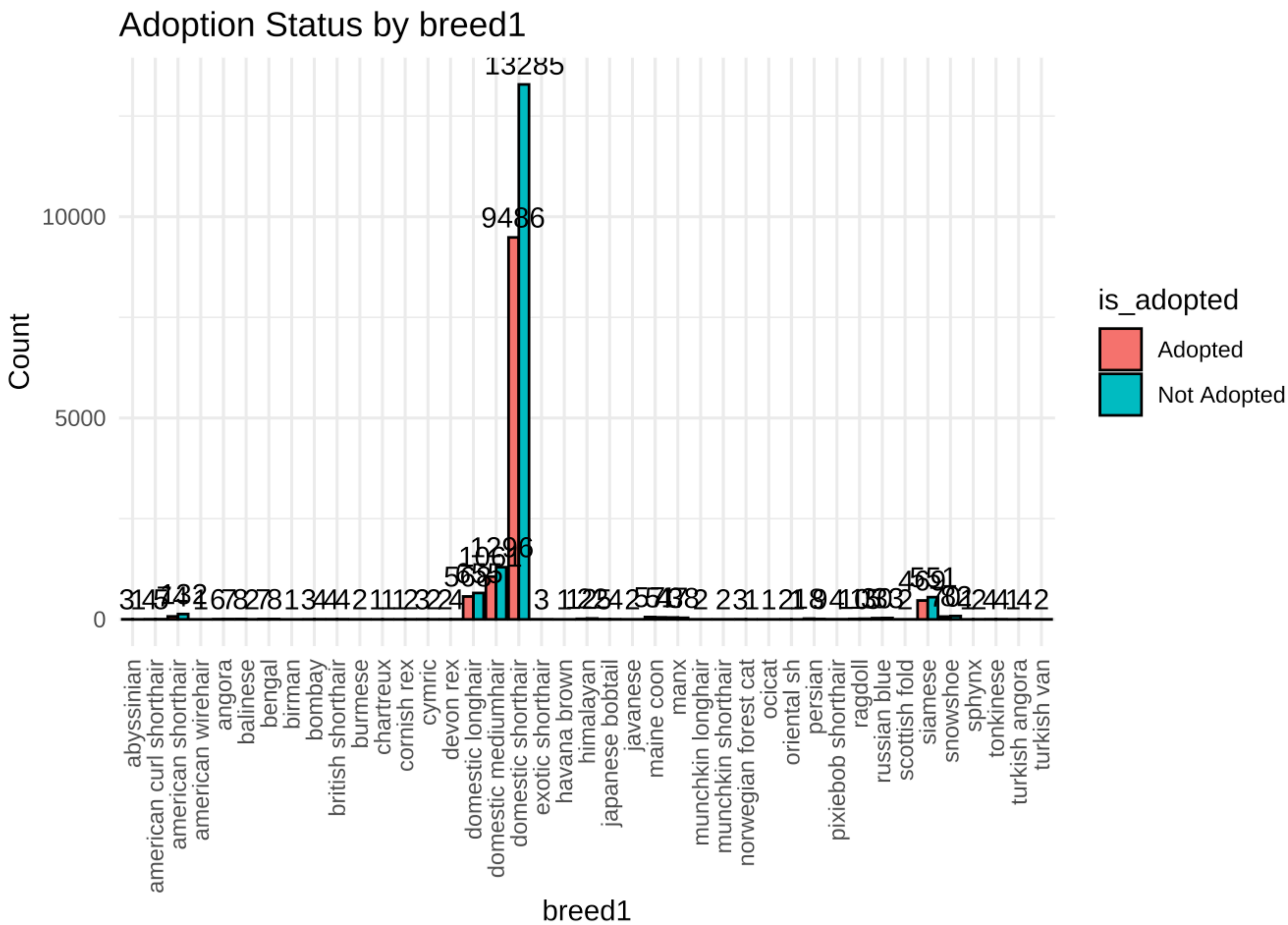

Adoption Status by Spay.Neuter
11237
8548
7683
740
9000
6000
3000
0
Count
No
Yes
Spay.Neuter
is_adopted
Adopted
Not Adopted

```
##
## [[3]]
```

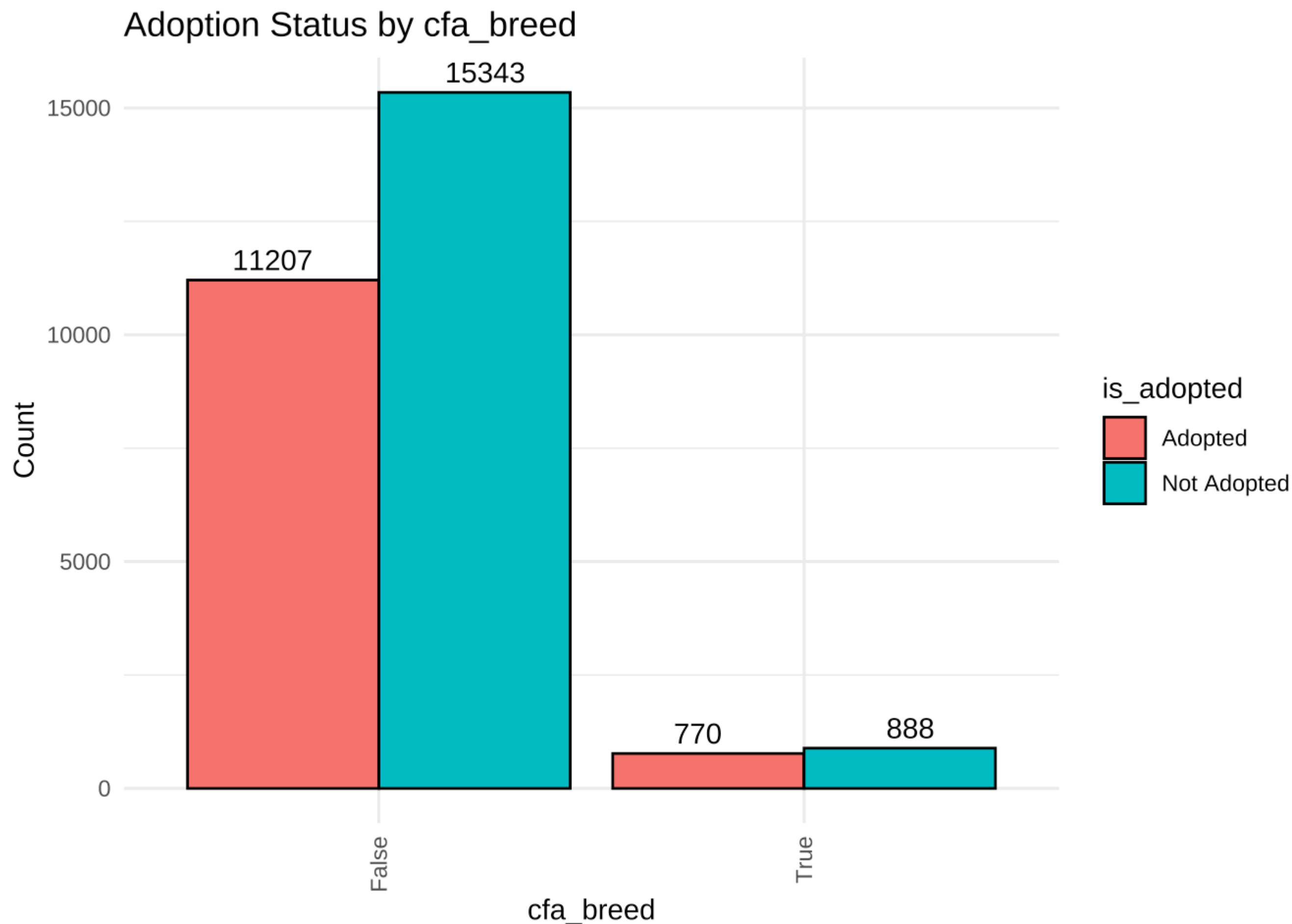


Adoption Status by Cat.Kitten..outcome.
8638
8382
7593
3595
7500
5000
2500
0
Count
is_adopted
Adopted
Not Adopted
Cat
Kitten
Cat.Kitten..outcome.

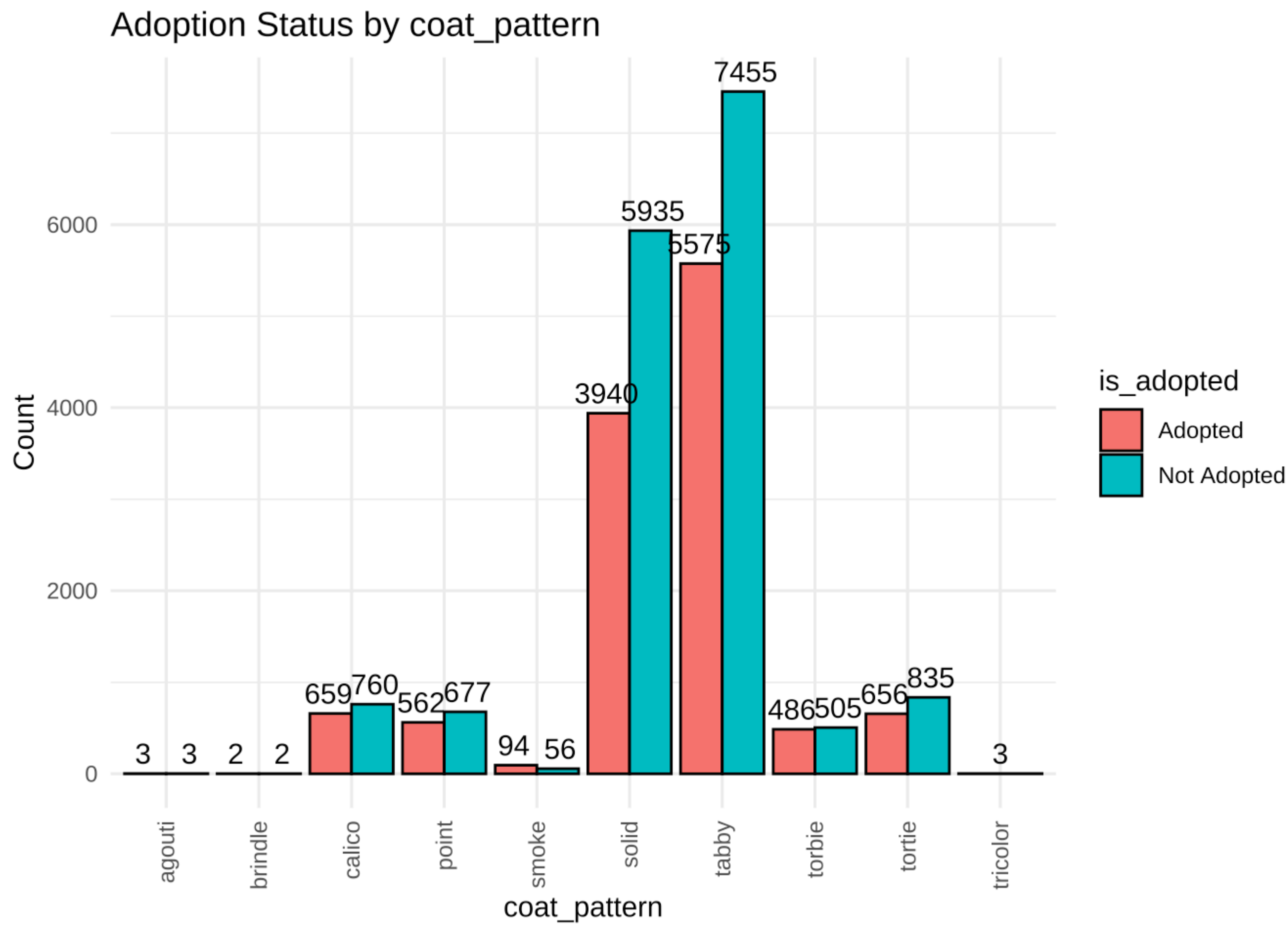

```
##
## [[4]]
```


Adoption Status by breed1
Count
breed1
is_adopted
Adopted
Not Adopted
13285
9486
1296
10000
5000
0
abyssinian
american curl shorthair
american shorthair
american wirehair
angora
balinese
bengal
birman
bombay
british shorthair
burmese
chartreux
cornish rex
cymric
devon rex
domestic longhair
domestic mediumhair
domestic shorthair
exotic shorthair
havana brown
himalayan
japanese bobtail
javanese
maine coon
manx
munchkin longhair
munchkin shorthair
norwegian forest cat
ocicat
oriental sh
persian
pixiebob shorthair
ragdoll
russian blue
scottish fold
siamese
snowshoe
sphynx
tonkinese
turkish angora
turkish van

```
##
## [[5]]
```

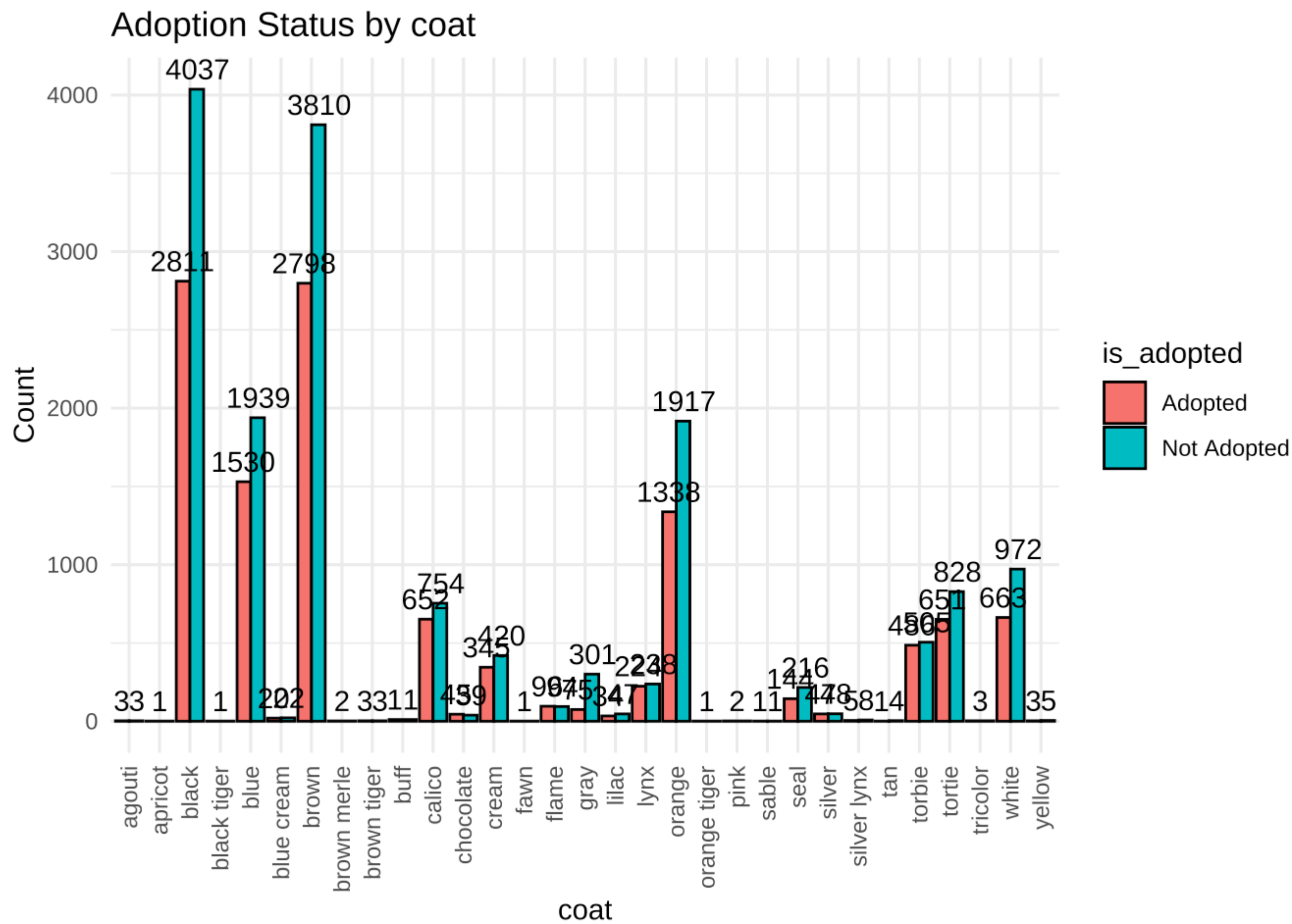

Adoption Status by cfa_breed
15343
11207
770
888
15000
10000
5000
0
Count
False
True
cfa_breed
is_adopted
Adopted
Not Adopted

```
##
## [[6]]
```

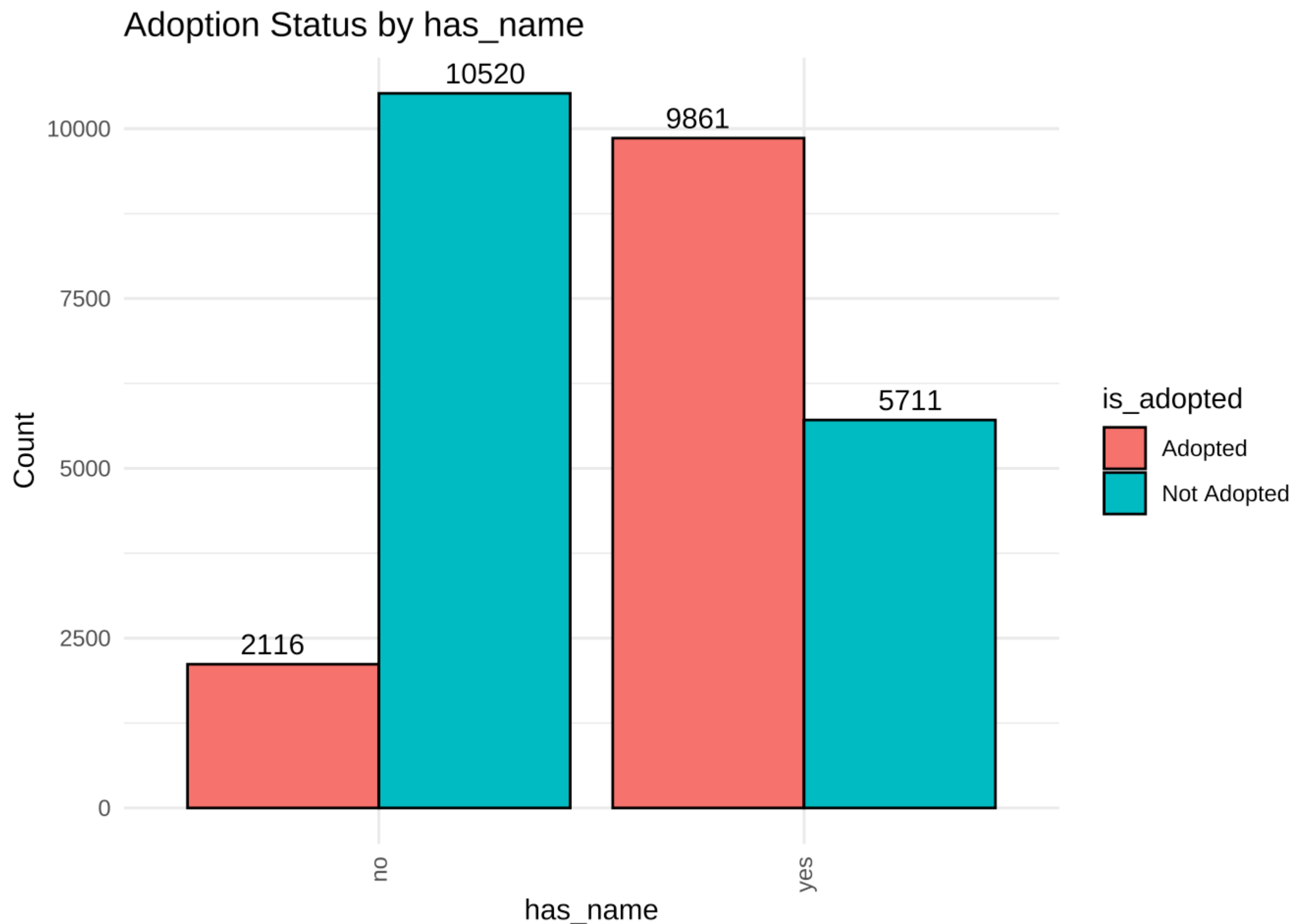


Adoption Status by coat_pattern
Count
6000
4000
2000
0
3
3
2
2
659
760
562
677
94
56
3940
5935
5575
7455
486
505
656
835
3
agouti
brindle
calico
point
smoke
solid
tabby
torbie
tortie
tricolor
coat_pattern
is_adopted
Adopted
Not Adopted

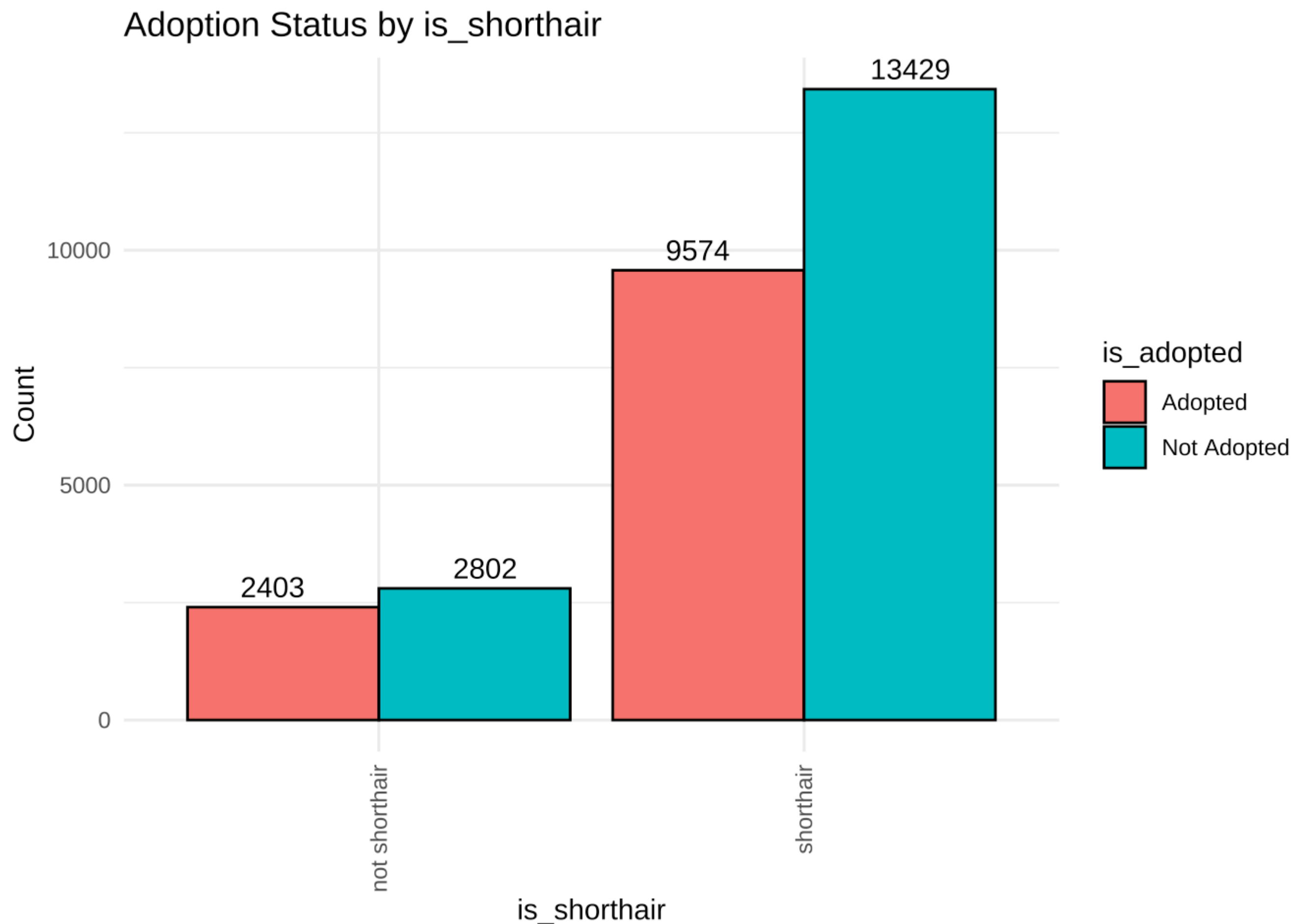

```
##
## [[7]]
```


Adoption Status by coat
4037
3810
2811
2798
1939
1917
1530
1338
972
828
754
663
420
301
216
4000
3000
2000
1000
0
Count
agouti
apricot
black
black tiger
blue
blue cream
brown
brown merle
brown tiger
buff
calico
chocolate
cream
fawn
flame
gray
lilac
lynx
orange
orange tiger
pink
sable
seal
silver
silver lynx
tan
torbie
tortie
tricolor
white
yellow
coat
is_adopted
Adopted
Not Adopted

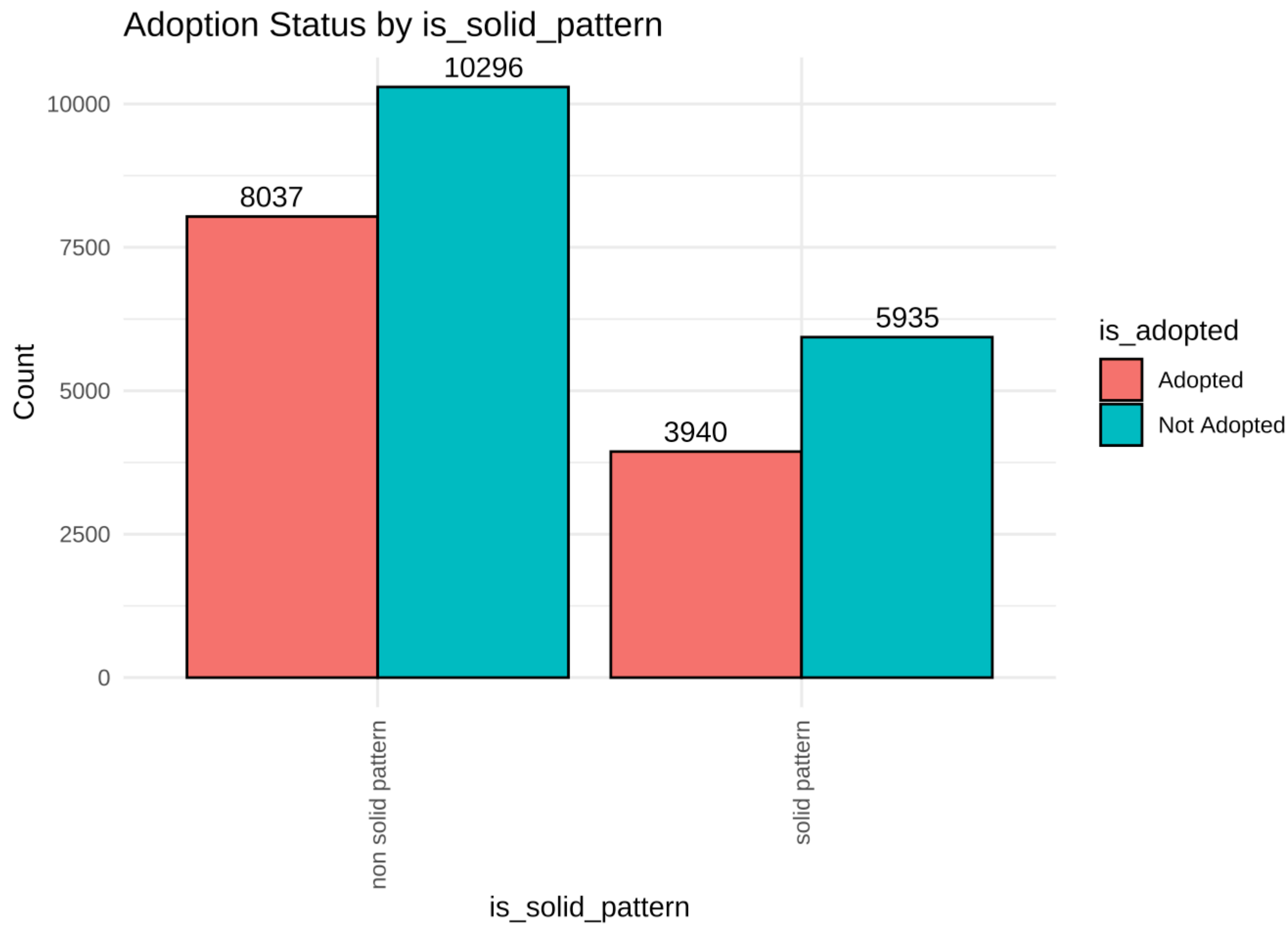
```
##
## [[8]]
```


Adoption Status by has_name
10520
9861
5711
2116
10000
7500
5000
2500
0
Count
no
yes
has_name
is_adopted
Adopted
Not Adopted

```
##
## [[9]]
```

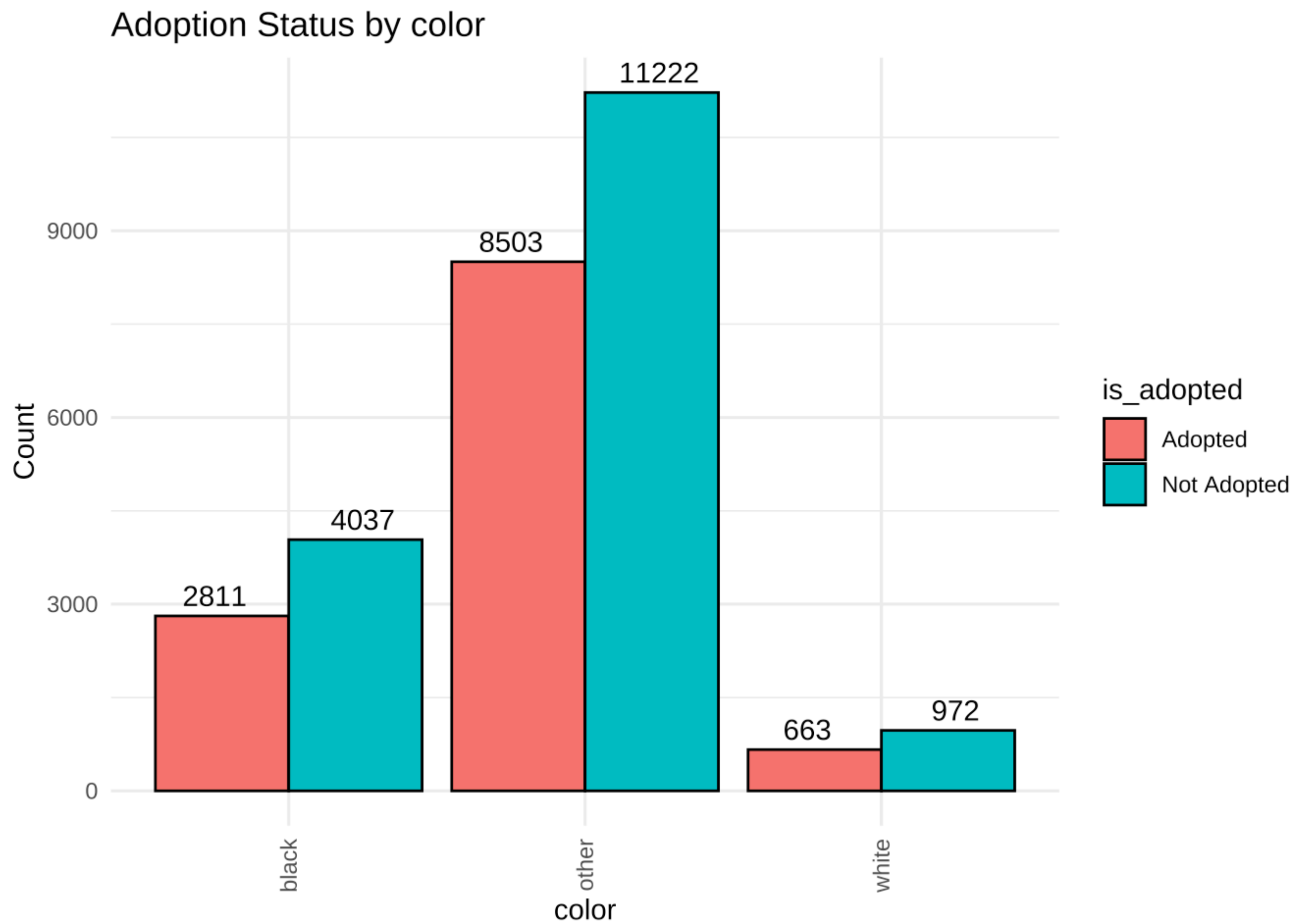

Adoption Status by is_shorthair
13429
9574
2403
2802
10000
5000
0
Count
not shorthair
shorthair
is_shorthair
is_adopted
Adopted
Not Adopted

```
##
## [[10]]
```

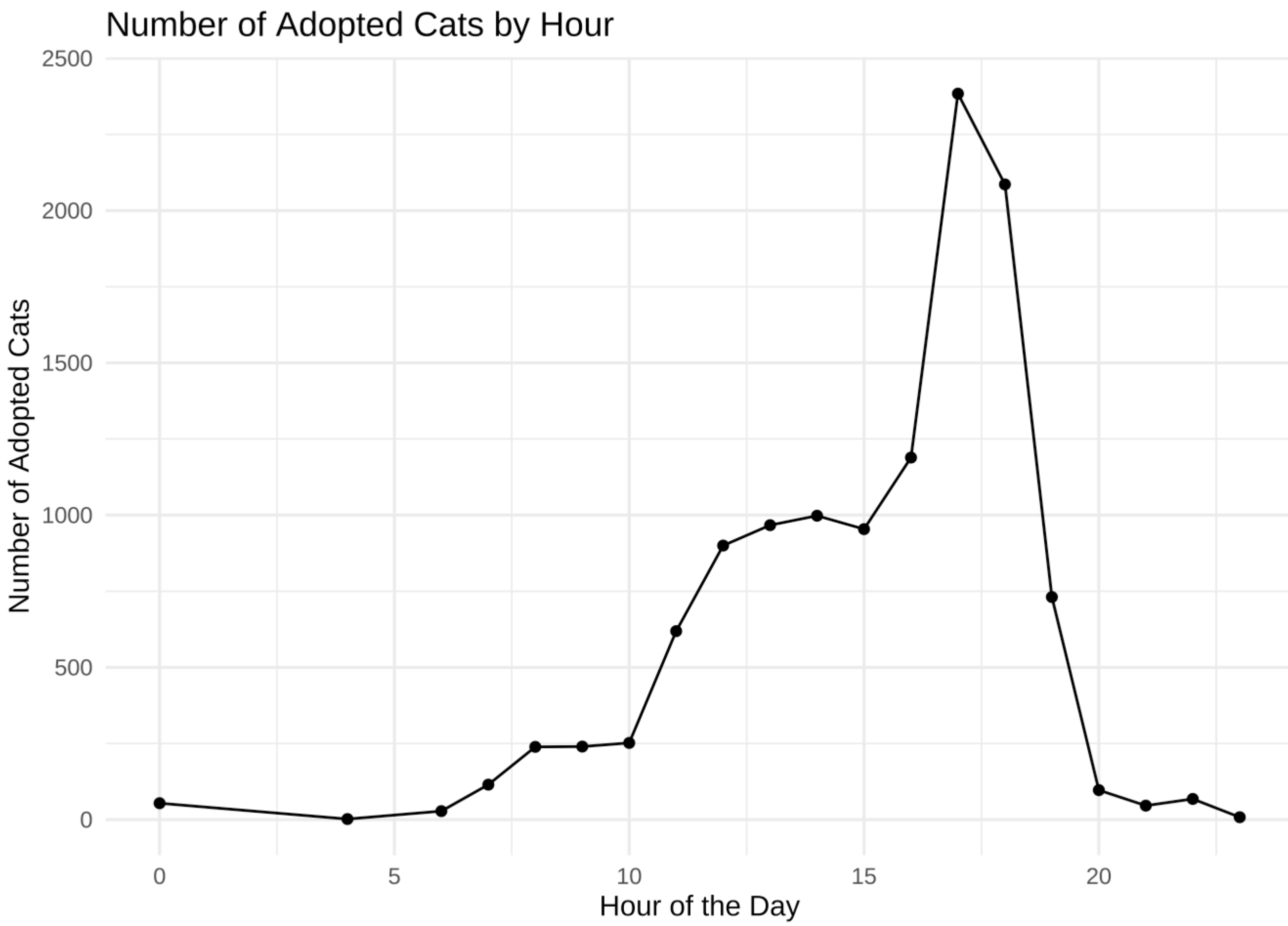


Adoption Status by is_solid_pattern
10296
8037
5935
3940
10000
7500
5000
2500
0
Count
non solid pattern
solid pattern
is_solid_pattern
is_adopted
Adopted
Not Adopted

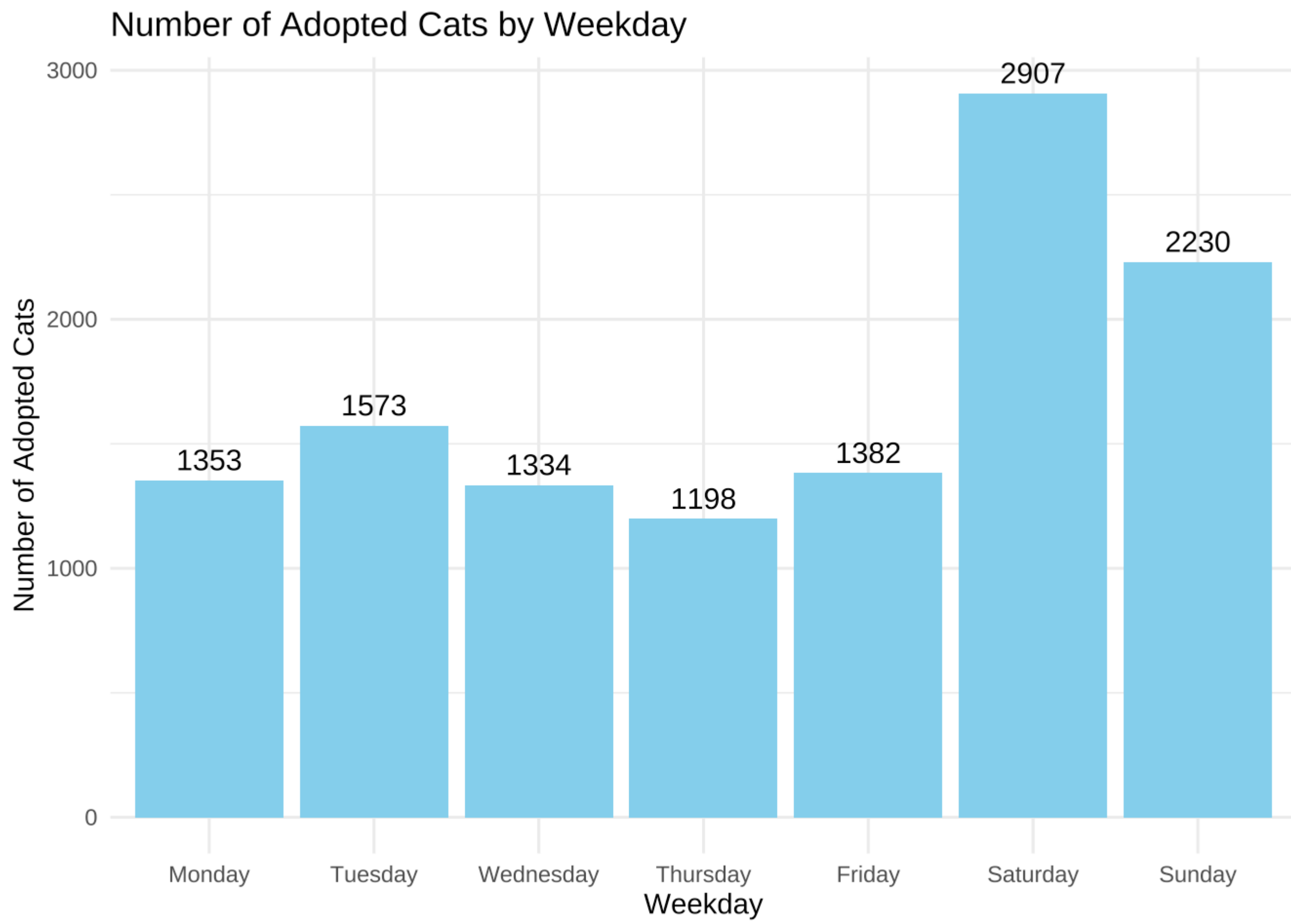

```
##
## [[11]]
```



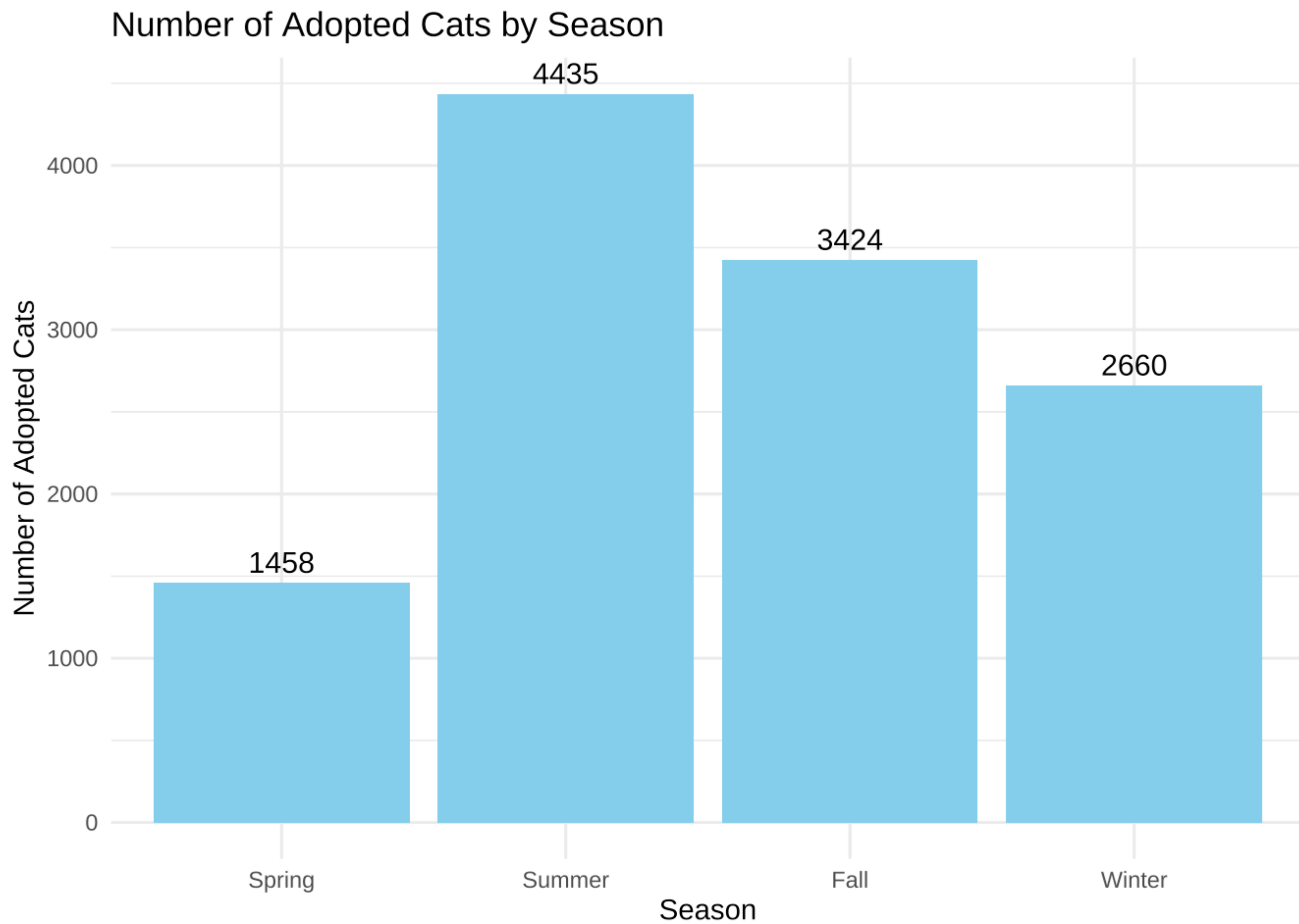

```
#ADOPTION BY HOUR
# Filter data to include only rows where the cat is adopted
adopted_data <- data[data$is_adopted == "Adopted", ]

# Create plot data
plot_data <- data.frame(outcome_hour = adopted_data$outcome_hour)

# Aggregate data by hour and count the number of adopted cats
hour_counts <- aggregate(is_adopted ~ outcome_hour, adopted_data, FUN = length)

# Create line plot
plot_hour <- ggplot(hour_counts, aes(x = outcome_hour, y = is_adopted)) +
  geom_line() +
  geom_point() +
  labs(title = "Number of Adopted Cats by Hour",
       x = "Hour of the Day",
       y = "Number of Adopted Cats") +
  theme_minimal()

# Print plot
print(plot_hour)
```


Number of Adopted Cats by Hour
2500
2000
1500
1000
500
0
Number of Adopted Cats
0
5
10
15
20
Hour of the Day

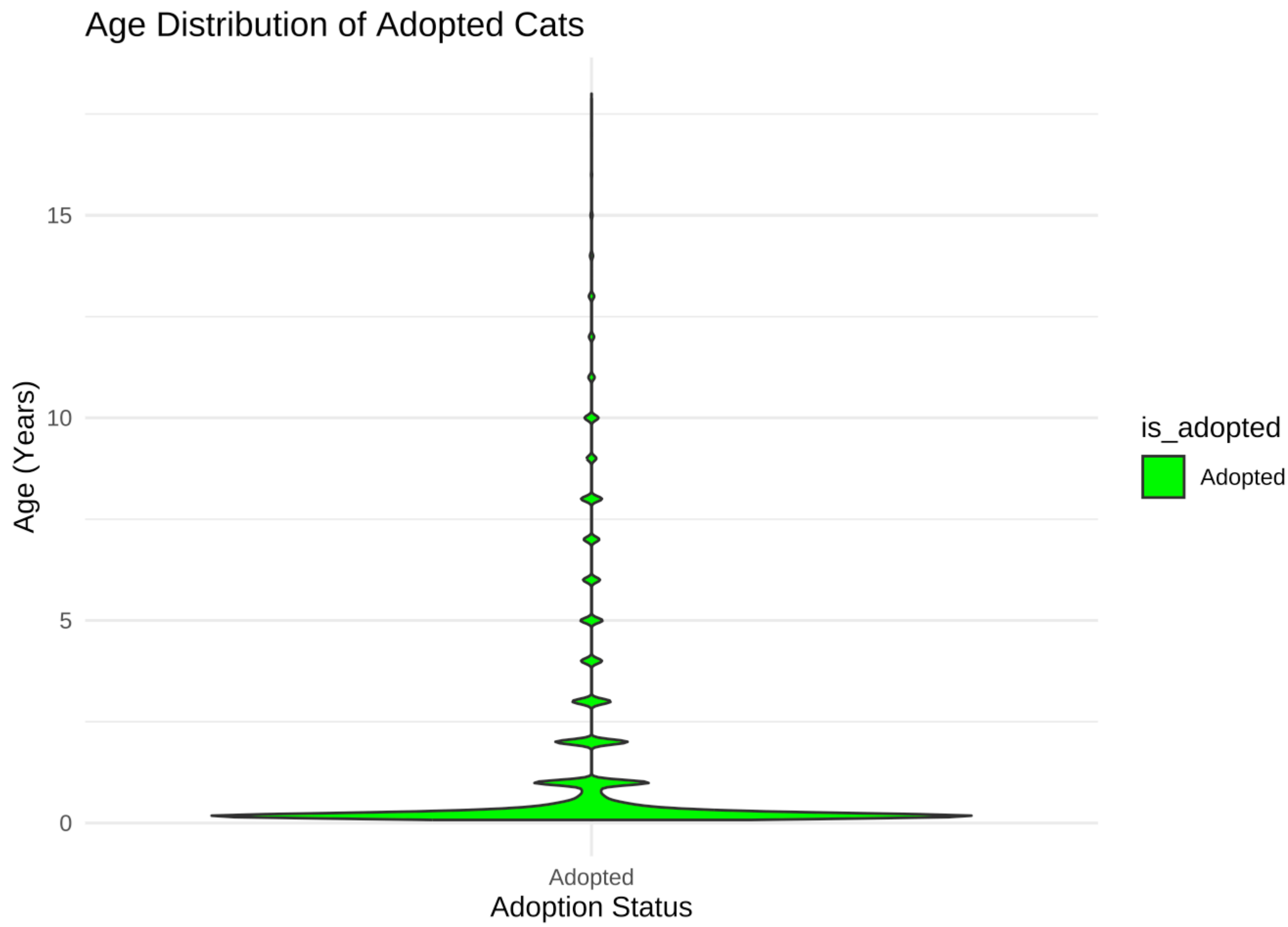
```r
#ADOPTION BY WEEKDAY
library(ggplot2)

# Filter data to include only rows where the cat is adopted
adopted_data <- data[data$is_adopted == "Adopted", ]

# Create plot data
plot_data <- data.frame(outcome_weekday = adopted_data$outcome_weekday)

# Aggregate data by weekday and count the number of adopted cats
weekday_counts <- aggregate(is_adopted ~ outcome_weekday, adopted_data, FUN = length)

# Reorder weekdays to display in the correct order
weekday_counts$outcome_weekday <- factor(weekday_counts$outcome_weekday, levels = c("Mon
day", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"))

# Create bar plot
plot_weekday <- ggplot(weekday_counts, aes(x = outcome_weekday, y = is_adopted)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(aes(label = is_adopted), vjust = -0.5, size = 4, color = "black") +
  labs(title = "Number of Adopted Cats by Weekday",
       x = "Weekday",
       y = "Number of Adopted Cats") +
  theme_minimal()

# Print plot
print(plot_weekday)
```


Number of Adopted Cats by Weekday
3000
2000
1000
0
Number of Adopted Cats
1353
1573
1334
1198
1382
2907
2230
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Weekday

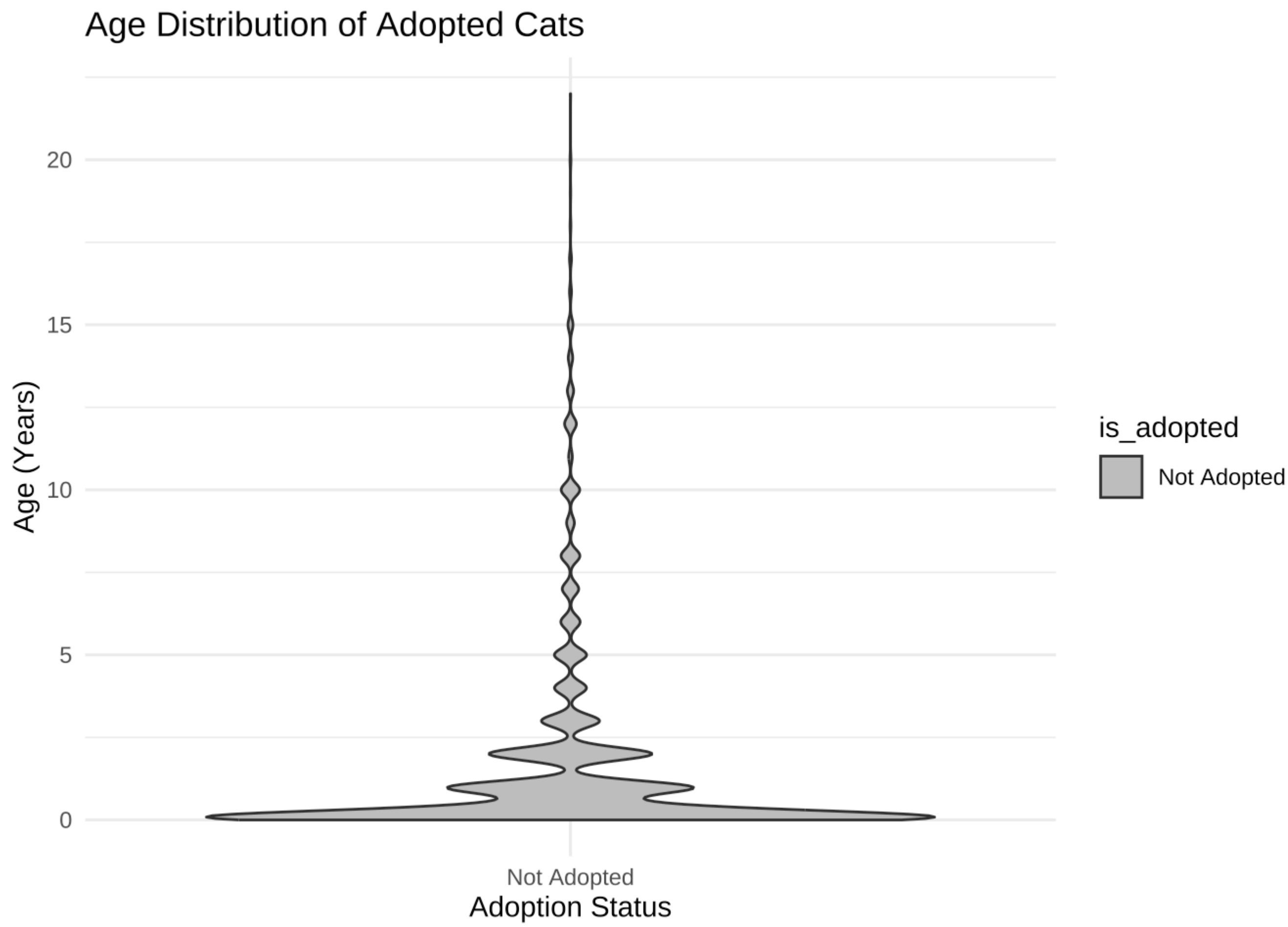
```
#ADOPTION BY SEASON

# Filter data to include only rows where the cat is adopted
adopted_data <- data[data$is_adopted == "Adopted", ]

# Create plot data
plot_data <- data.frame(season = adopted_data$season)

# Aggregate data by season and count the number of adopted cats
season_counts <- aggregate(is_adopted ~ season, adopted_data, FUN = length)

# Reorder seasons to display in the correct order
season_counts$season <- factor(season_counts$season, levels = c("Spring", "Summer", "Fal
l", "Winter"))

# Create bar plot with count labels
plot_season <- ggplot(season_counts, aes(x = season, y = is_adopted)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(aes(label = is_adopted), vjust = -0.5, size = 4, color = "black") +
  labs(title = "Number of Adopted Cats by Season",
       x = "Season",
       y = "Number of Adopted Cats") +
  theme_minimal()

# Print plot
print(plot_season)
```




```
#OUTCOME AGE YEARS

#ADOPTED

# Filter data to include only rows where the cat is adopted
adopted_data <- data[data$is_adopted == "Adopted", ]

# Create violin plot
plot_age_violin <- ggplot(adopted_data, aes(x = is_adopted, y = outcome_age_.years., fil
l = is_adopted)) +
  geom_violin() +
  labs(title = "Age Distribution of Adopted Cats",
       x = "Adoption Status",
       y = "Age (Years)") +
  scale_fill_manual(values = c("Adopted" = "green")) +
  theme_minimal()

# Print plot
print(plot_age_violin)
```




```
#NOT ADOPTED

# Filter data to include only rows where the cat is not adopted
adopted_data <- data[data$is_adopted != "Adopted", ]

# Create violin plot
plot_age_violin <- ggplot(adopted_data, aes(x = is_adopted, y = outcome_age_.years., fil
l = is_adopted)) +
  geom_violin() +
  labs(title = "Age Distribution of Adopted Cats",
       x = "Adoption Status",
       y = "Age (Years)") +
  scale_fill_manual(values = c("Not Adopted" = "grey")) +
  theme_minimal()

# Print plot
print(plot_age_violin)
```




```
# Create an empty list to store chi-squared test results
chi_squared_results <- list()

# List of categorical variables
categorical_variables <- c("sex", "Spay.Neuter", "Cat.Kitten..outcome.", "outcome_weekda
y",
                           "breed1", "cfa_breed", "coat_pattern", "coat", "has_name",
                           "season", "is_weekend", "time_of_day", "is_shorthair",
                           "is_solid_pattern", "color")

# Perform chi-squared test for each categorical variable
for (variable in categorical_variables) {
  # Create a contingency table
  contingency_table <- table(data[[variable]], data$is_adopted)

  # Perform chi-squared test
  chi_squared_results[[variable]] <- chisq.test(contingency_table)
}
```

```
## Warning in chisq.test(contingency_table): Chi-squared approximation may be
## incorrect
```

```
## Warning in chisq.test(contingency_table): Chi-squared approximation may be
## incorrect

## Warning in chisq.test(contingency_table): Chi-squared approximation may be
## incorrect
```

```
# Print the results
for (variable in categorical_variables) {
  print(paste("Chi-squared test for", variable))
  print(chi_squared_results[[variable]])
}
```

```
## [1] "Chi-squared test for sex"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 110.89, df = 1, p-value < 2.2e-16
##
## [1] "Chi-squared test for Spay.Neuter"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 6741.1, df = 1, p-value < 2.2e-16
##
## [1] "Chi-squared test for Cat.Kitten..outcome."
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 808.69, df = 1, p-value < 2.2e-16
##
## [1] "Chi-squared test for outcome_weekday"
##
##  Pearson's Chi-squared test
##
## data:  contingency_table
## X-squared = 1068.9, df = 6, p-value < 2.2e-16
##
## [1] "Chi-squared test for breed1"
##
##  Pearson's Chi-squared test
##
## data:  contingency_table
## X-squared = 88.925, df = 40, p-value = 1.406e-05
##
## [1] "Chi-squared test for cfa_breed"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 11.26, df = 1, p-value = 0.0007921
##
## [1] "Chi-squared test for coat_pattern"
##
##  Pearson's Chi-squared test
##
## data:  contingency_table
## X-squared = 87.077, df = 9, p-value = 6.272e-15
##
## [1] "Chi-squared test for coat"
##
##  Pearson's Chi-squared test
```

```
##
## data:  contingency_table
## X-squared = 160.81, df = 30, p-value < 2.2e-16
##
## [1] "Chi-squared test for has_name"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 6192.8, df = 1, p-value < 2.2e-16
##
## [1] "Chi-squared test for season"
##
##  Pearson's Chi-squared test
##
## data:  contingency_table
## X-squared = 774.86, df = 3, p-value < 2.2e-16
##
## [1] "Chi-squared test for is_weekend"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 1021, df = 1, p-value < 2.2e-16
##
## [1] "Chi-squared test for time_of_day"
##
##  Pearson's Chi-squared test
##
## data:  contingency_table
## X-squared = 1296.8, df = 2, p-value < 2.2e-16
##
## [1] "Chi-squared test for is_shorthair"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 35.726, df = 1, p-value = 2.271e-09
##
## [1] "Chi-squared test for is_solid_pattern"
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 40.623, df = 1, p-value = 1.846e-10
##
## [1] "Chi-squared test for color"
##
##  Pearson's Chi-squared test
##
## data:  contingency_table
## X-squared = 11.412, df = 2, p-value = 0.003326
```

```
# Chi-squared test results summary

# Adoption status showed significant associations with all tested variables,
#including sex, spay/neuter status, cat/kitten outcome, outcome weekday, breed, CFA bree
d,
#coat pattern, coat color, name availability, season, weekend, time of day, shorthair st
atus,
#solid coat pattern, and color (p < 0.05).


################################################
############ ANALYSIS #######################
###############################################

# Predicting Adoption Success in Animal Shelters

set.seed(123)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```
library(ggcorrplot)
library(dplyr)
library(fpp3)
```

```
## ── Attaching packages ──────────────────────────────────────────── fpp3 0.5 ──
```

```
## ✔ tibble      3.2.1     ✔ tsibbledata 0.4.1
## ✔ tidyr       1.3.0     ✔ feasts      0.3.1
## ✔ lubridate   1.9.3     ✔ fable       0.3.3
## ✔ tsibble     1.1.4     ✔ fabletools  0.3.4
```

```
## ── Conflicts ────────────────────────────────────────────────── fpp3_conflicts ──
## ✖ ggplot2::%+%()       masks psych::%+%()
## ✖ ggplot2::alpha()     masks psych::alpha()
## ✖ gridExtra::combine() masks dplyr::combine()
## ✖ lubridate::date()    masks base::date()
## ✖ tidyr::extract()     masks dlookr::extract()
## ✖ dplyr::filter()      masks stats::filter()
## ✖ tsibble::intersect() masks base::intersect()
## ✖ tsibble::interval()  masks lubridate::interval()
## ✖ dplyr::lag()         masks stats::lag()
## ✖ tsibble::setdiff()   masks base::setdiff()
## ✖ tsibble::union()     masks base::union()
```

```
dat <- read.csv("cat_data_cleaned_updated.csv")

# PRE-PROCESSING

#remove animal_id column
dat$animal_id <- NULL
names(dat)
```

```
##  [1] "has_name"                "is_adopted"              "sex_male"
##  [4] "spay_neuter"             "cfa_approved"            "is_kitten"
##  [7] "season_Fall"             "season_Spring"           "season_Summer"
## [10] "season_Winter"           "is_weekend"              "time_of_day_Afternoon"
## [13] "time_of_day_Closed"      "time_of_day_Morning"     "is_shorthair"
## [16] "is_solid_pattern"        "color_black"             "color_other"
## [19] "color_white"
```

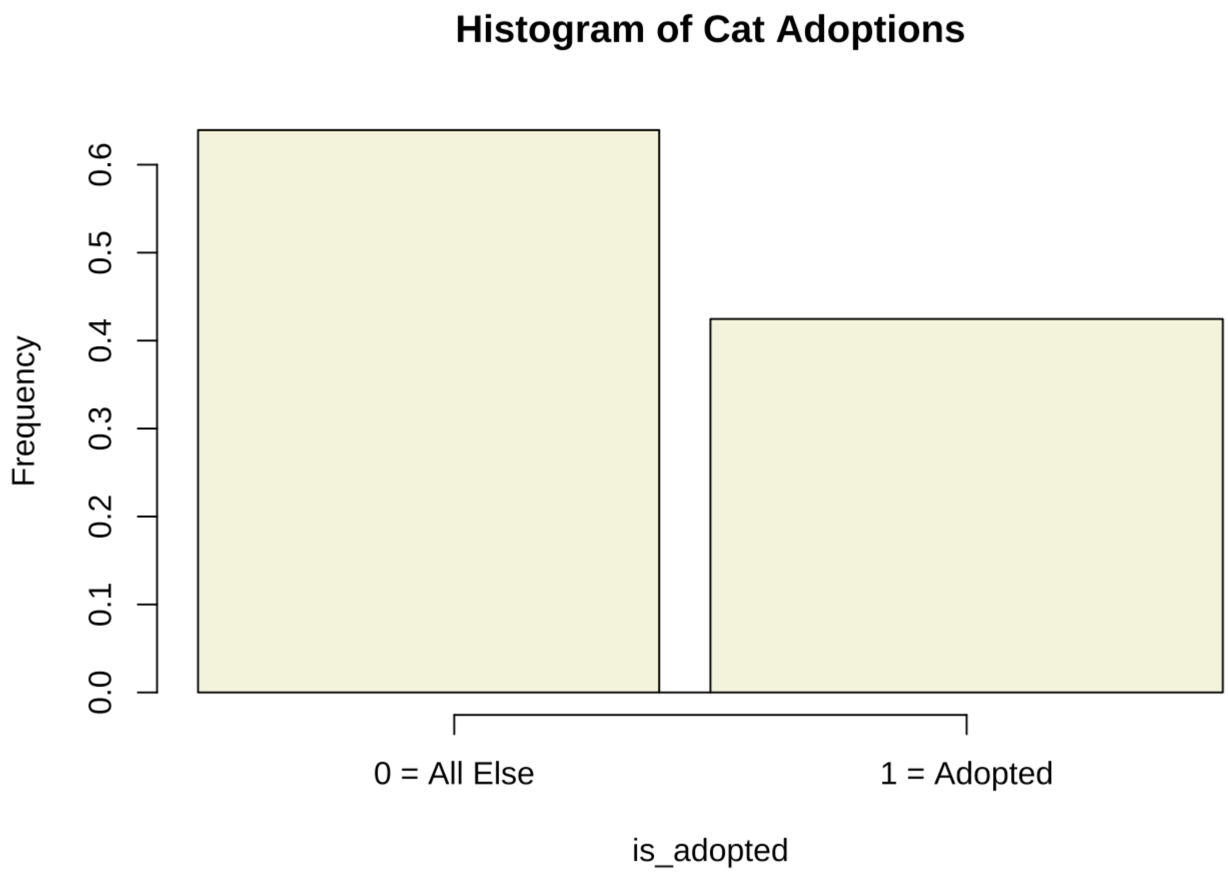
```
# Categorical variables
# "season"
# "time of day"
# "color"

# remove one variable for each categorical variable
dat$season_Fall <- NULL
dat$time_of_day_Closed <- NULL
dat$color_other <- NULL

# summary of all variables in data
summary(dat)
```

```
##     has_name       is_adopted        sex_male        spay_neuter    
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :1.000   Median :0.0000   Median :0.0000   Median :1.0000  
##  Mean   :0.552   Mean   :0.4246   Mean   :0.4472   Mean   :0.6707  
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
##   cfa_approved       is_kitten      season_Spring    season_Summer   
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.00000   Median :1.0000   Median :0.0000   Median :0.0000  
##  Mean   :0.05878   Mean   :0.6034   Mean   :0.1924   Mean   :0.3403  
##  3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:1.0000  
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
##  season_Winter      is_weekend     time_of_day_Afternoon time_of_day_Morning
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000        Min.   :0.0000     
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000        1st Qu.:0.0000     
##  Median :0.0000   Median :0.0000   Median :0.0000        Median :0.0000     
##  Mean   :0.1825   Mean   :0.3251   Mean   :0.4319        Mean   :0.3416     
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.0000        3rd Qu.:1.0000     
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000        Max.   :1.0000     
##   is_shorthair    is_solid_pattern  color_black      color_white     
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000  
##  1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000  
##  Median :1.0000   Median :0.0000   Median :0.0000   Median :0.00000  
##  Mean   :0.8155   Mean   :0.3501   Mean   :0.2428   Mean   :0.05796  
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000  
```

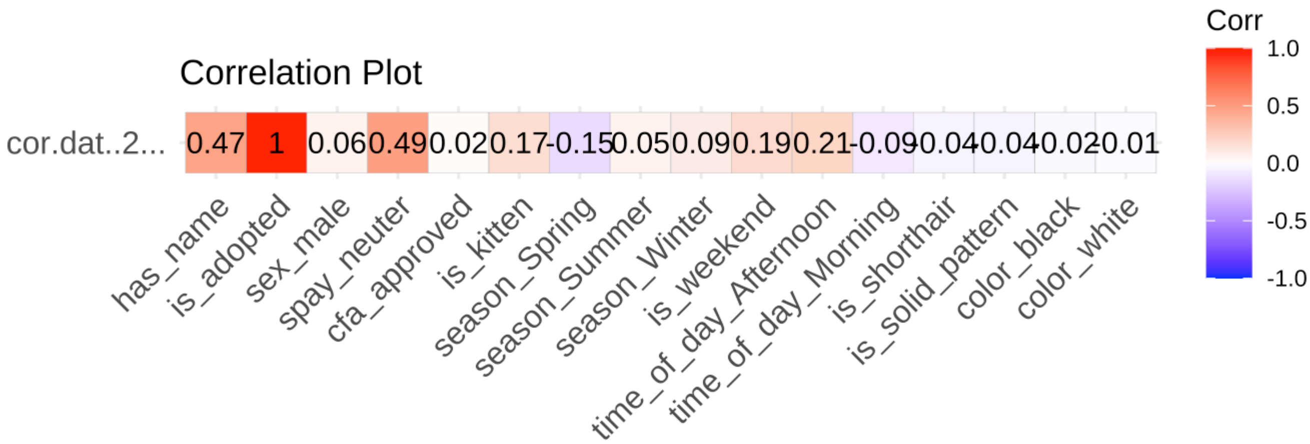
```
# dependent variable: is_adopted
table(dat$is_adopted)
```

```
## 
##     0     1 
## 16231 11977
```

```
hist(dat$is_adopted, breaks=c(-0.5, 0.5, 1.5, 0.4),
     col="beige", main="Histogram of Cat Adoptions",
     xlab="is_adopted", ylab="Frequency", xaxt='n')
axis(1, at=c(0, 1), labels=c("0 = All Else", "1 = Adopted"))
```




```
#  Look at the correlations of all variables with the
#    Adoption (is_adoption) variable 1 = cat is adopted;
#    0 = cat is not adopted (either euthanized or transferred)
#
cor_matrix <- data.frame(cor(dat)[2,])
ggcorrplot(cor_matrix, type = "full", lab = TRUE) +
  ggtitle("Correlation Plot")
```




```
# highly correlated variables:
# has_name, spay_neuter

# general logistic regression model (no training)
lr.all <- glm(is_adopted ~ . , data = dat,
                    family = "binomial")
sum.lr.all <- summary(lr.all)
sum.lr.all
```

```
##
## Call:
## glm(formula = is_adopted ~ ., family = "binomial", data = dat)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -6.00551    0.09923 -60.518  < 2e-16 ***
## has_name              2.43582    0.04244  57.390  < 2e-16 ***
## sex_male              0.27970    0.03459   8.086 6.16e-16 ***
## spay_neuter           3.28058    0.05193  63.176  < 2e-16 ***
## cfa_approved          0.15704    0.07571   2.074 0.038071 *
## is_kitten             2.14310    0.04308  49.746  < 2e-16 ***
## season_Spring        -0.38167    0.05039  -7.574 3.62e-14 ***
## season_Summer         0.15948    0.04348   3.668 0.000244 ***
## season_Winter         0.40123    0.05010   8.009 1.16e-15 ***
## is_weekend            0.89545    0.03634  24.640  < 2e-16 ***
## time_of_day_Afternoon  0.63544    0.04731  13.431  < 2e-16 ***
## time_of_day_Morning  -0.28754    0.04912  -5.854 4.79e-09 ***
## is_shorthair         -0.19776    0.04760  -4.154 3.26e-05 ***
## is_solid_pattern     -0.19451    0.05424  -3.586 0.000336 ***
## color_black          -0.11733    0.06022  -1.949 0.051352 .
## color_white          -0.03725    0.07896  -0.472 0.637065
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 38461  on 28207  degrees of freedom
## Residual deviance: 21854  on 28192  degrees of freedom
## AIC: 21886
##
## Number of Fisher Scoring iterations: 6
```

```
### TRAINING AND TESTING
#  Create a 50/50 training set
table(dat$is_adopted)
```

```
##
##     0     1
## 16231 11977
```

```
dat.A <- dat[dat$is_adopted == 1,]
dat.notA <- dat[dat$is_adopted == 0,]
train.A <- sample(nrow(dat.A),1000)
train.notA <- sample(nrow(dat.notA),1000)
dat.train <- rbind(dat.A[train.A,],
                   dat.notA[train.notA,])
table(dat.train$is_adopted)
```

```
## 
##    0    1 
## 1000 1000
```

```
# Create a test data set similar to the training set
dat.notA.notsel <- dat.notA[-train.notA,]
dat.A.notsel <- dat.A[-train.A,]
test.notA <- sample(nrow(dat.notA.notsel),nrow(dat.A.notsel))
dat.test <- rbind(dat.A[-train.A,],
                  dat.notA.notsel[test.notA,])
# Check distribution
mean(dat.test$is_adopted)
```

```
## [1] 0.5
```

```
mean(dat$is_adopted)
```

```
## [1] 0.4245959
```

```
table(dat.test$is_adopted)/nrow(dat.test)
```

```
## 
##   0   1 
## 0.5 0.5
```

```
table(dat$is_adopted)/nrow(dat)
```

```
## 
##         0         1 
## 0.5754041 0.4245959
```

```
#  Remove unneeded objects
rm(dat.A, dat.notA, dat.notA.notsel)
rm(test.notA, train.A, train.notA)
#######

# Logistic Regression TRAINING Model
lr.all.train <- glm(is_adopted ~ . , data = dat.train,
                    family = "binomial")
sum.lr.all.train <- summary(lr.all.train)
sum.lr.all
```

```
##
## Call:
## glm(formula = is_adopted ~ ., family = "binomial", data = dat)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -6.00551    0.09923 -60.518  < 2e-16 ***
## has_name               2.43582    0.04244  57.390  < 2e-16 ***
## sex_male               0.27970    0.03459   8.086 6.16e-16 ***
## spay_neuter            3.28058    0.05193  63.176  < 2e-16 ***
## cfa_approved           0.15704    0.07571   2.074 0.038071 *
## is_kitten              2.14310    0.04308  49.746  < 2e-16 ***
## season_Spring         -0.38167    0.05039  -7.574 3.62e-14 ***
## season_Summer          0.15948    0.04348   3.668 0.000244 ***
## season_Winter          0.40123    0.05010   8.009 1.16e-15 ***
## is_weekend             0.89545    0.03634  24.640  < 2e-16 ***
## time_of_day_Afternoon  0.63544    0.04731  13.431  < 2e-16 ***
## time_of_day_Morning   -0.28754    0.04912  -5.854 4.79e-09 ***
## is_shorthair          -0.19776    0.04760  -4.154 3.26e-05 ***
## is_solid_pattern      -0.19451    0.05424  -3.586 0.000336 ***
## color_black           -0.11733    0.06022  -1.949 0.051352 .
## color_white           -0.03725    0.07896  -0.472 0.637065
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 38461  on 28207  degrees of freedom
## Residual deviance: 21854  on 28192  degrees of freedom
## AIC: 21886
##
## Number of Fisher Scoring iterations: 6
```

```
#  Compute a percentage reduction in deviance using
#    the null and residual deviance

#  Null deviance = Deviance (value of -2 Log Likelihood)
#    when the "naive" model is fit, that is, the model
#    with just the intercept and no predictors
lr.naive.train <- glm(is_adopted ~ 1 , data = dat.train,
                      family = "binomial")
lr.naive.train$deviance
```

```
## [1] 2772.589
```

```
lr.all.train$null.deviance
```

```
## [1] 2772.589
```

```
summary(lr.naive.train)
```

```
##
## Call:
## glm(formula = is_adopted ~ 1, family = "binomial", data = dat.train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 5.210e-19  4.472e-02       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2772.6  on 1999  degrees of freedom
## Residual deviance: 2772.6  on 1999  degrees of freedom
## AIC: 2774.6
##
## Number of Fisher Scoring iterations: 2
```

```
#  Residual deviance = Deviance with the current model
#    including all the predictors
DevRed <- 1 - (lr.all.train$deviance /
                 lr.all.train$null.deviance)
DevRed
```

```
## [1] 0.4394017
```

```
#  According this measure, the model reduced the
#    null deviance by about 43.7%.  Remember this
#    is not an R-squared measure-so it
#    can be not be interpreted as such

#  Try a simpler version of the logistic regression
#    using most of the highly significant variables
#    from the "all in" model.
sum.lr.all
```

```
##
## Call:
## glm(formula = is_adopted ~ ., family = "binomial", data = dat)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -6.00551    0.09923 -60.518  < 2e-16 ***
## has_name               2.43582    0.04244  57.390  < 2e-16 ***
## sex_male               0.27970    0.03459   8.086 6.16e-16 ***
## spay_neuter            3.28058    0.05193  63.176  < 2e-16 ***
## cfa_approved           0.15704    0.07571   2.074 0.038071 *
## is_kitten              2.14310    0.04308  49.746  < 2e-16 ***
## season_Spring         -0.38167    0.05039  -7.574 3.62e-14 ***
## season_Summer          0.15948    0.04348   3.668 0.000244 ***
## season_Winter          0.40123    0.05010   8.009 1.16e-15 ***
## is_weekend             0.89545    0.03634  24.640  < 2e-16 ***
## time_of_day_Afternoon  0.63544    0.04731  13.431  < 2e-16 ***
## time_of_day_Morning   -0.28754    0.04912  -5.854 4.79e-09 ***
## is_shorthair          -0.19776    0.04760  -4.154 3.26e-05 ***
## is_solid_pattern      -0.19451    0.05424  -3.586 0.000336 ***
## color_black           -0.11733    0.06022  -1.949 0.051352 .
## color_white           -0.03725    0.07896  -0.472 0.637065
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 38461  on 28207  degrees of freedom
## Residual deviance: 21854  on 28192  degrees of freedom
## AIC: 21886
##
## Number of Fisher Scoring iterations: 6
```

```
sum.lr.all.train
```

```
##
## Call:
## glm(formula = is_adopted ~ ., family = "binomial", data = dat.train)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -5.74367    0.37562 -15.291  < 2e-16 ***
## has_name              2.43226    0.16368  14.860  < 2e-16 ***
## sex_male              0.27024    0.13069   2.068  0.03866 *
## spay_neuter           3.57606    0.20808  17.186  < 2e-16 ***
## cfa_approved          0.28148    0.29581   0.952  0.34133
## is_kitten             2.17771    0.16684  13.053  < 2e-16 ***
## season_Spring        -0.54640    0.18600  -2.938  0.00331 **
## season_Summer        -0.01031    0.16134  -0.064  0.94906
## season_Winter         0.48530    0.18957   2.560  0.01047 *
## is_weekend            0.81872    0.13721   5.967 2.42e-09 ***
## time_of_day_Afternoon  0.53005    0.17453   3.037  0.00239 **
## time_of_day_Morning  -0.47038    0.18295  -2.571  0.01014 *
## is_shorthair         -0.23645    0.17751  -1.332  0.18284
## is_solid_pattern     -0.56200    0.20707  -2.714  0.00665 **
## color_black           0.36318    0.23260   1.561  0.11842
## color_white           0.09321    0.30246   0.308  0.75796
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2772.6  on 1999  degrees of freedom
## Residual deviance: 1554.3  on 1984  degrees of freedom
## AIC: 1586.3
##
## Number of Fisher Scoring iterations: 6
```

```
# Logistic Regression MODEL 2 (simplified)
lr.2.train <- glm(is_adopted ~ . - cfa_approved - season_Summer
                  - is_shorthair - color_black - color_white,
                  data = dat.train,
                  family = "binomial")
sum.lr.2.train <- summary(lr.2.train)
sum.lr.2.train
```

```
##
## Call:
## glm(formula = is_adopted ~ . - cfa_approved - season_Summer -
##     is_shorthair - color_black - color_white, family = "binomial",
##     data = dat.train)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -5.8889     0.3331 -17.680  < 2e-16 ***
## has_name                2.4488     0.1631  15.018  < 2e-16 ***
## sex_male                0.2725     0.1301   2.095 0.036213 *
## spay_neuter             3.5572     0.2069  17.190  < 2e-16 ***
## is_kitten               2.1560     0.1654  13.038  < 2e-16 ***
## season_Spring          -0.5826     0.1655  -3.520 0.000431 ***
## season_Winter           0.5061     0.1722   2.940 0.003285 **
## is_weekend              0.8247     0.1369   6.022 1.72e-09 ***
## time_of_day_Afternoon   0.5283     0.1737   3.041 0.002358 **
## time_of_day_Morning    -0.4585     0.1817  -2.523 0.011638 *
## is_solid_pattern       -0.3472     0.1323  -2.624 0.008690 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2772.6  on 1999  degrees of freedom
## Residual deviance: 1561.2  on 1989  degrees of freedom
## AIC: 1583.2
##
## Number of Fisher Scoring iterations: 6
```

```
lr.all.train$deviance
```

```
## [1] 1554.308
```

```
lr.2.train$deviance
```

```
## [1] 1561.164
```

```
# slightly higher deviance in model 2

lr.all.train$df.null
```

```
## [1] 1999
```

```
lr.all.train$df.residual
```

```
## [1] 1984
```

```
lr.2.train$df.residual
```

```
## [1] 1989
```

```
# slightly higher residiual in model 2


#  The Likelihood Ratio Test
#     Ho:  Models fit equally well
#     Ha:  Model 2 is just as good as Model 1
anova(lr.2.train, lr.all.train, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: is_adopted ~ (has_name + sex_male + spay_neuter + cfa_approved +
##     is_kitten + season_Spring + season_Summer + season_Winter +
##     is_weekend + time_of_day_Afternoon + time_of_day_Morning +
##     is_shorthair + is_solid_pattern + color_black + color_white) -
##     cfa_approved - season_Summer - is_shorthair - color_black -
##     color_white
## Model 2: is_adopted ~ has_name + sex_male + spay_neuter + cfa_approved +
##     is_kitten + season_Spring + season_Summer + season_Winter +
##     is_weekend + time_of_day_Afternoon + time_of_day_Morning +
##     is_shorthair + is_solid_pattern + color_black + color_white
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      1989     1561.2
## 2      1984     1554.3  5    6.856   0.2316
```

```
# p-value: 0.2316
# FAIL to reject the null hypothesis
# Simple model is just as good as all in model


#  Check performance of these two models on training data
yhat.all.train <- predict(lr.all.train, dat.train,
                          type = "response")
yhat.all.train.cl <- ifelse(yhat.all.train > 0.5, 1, 0)
tab.all.train <- table(dat.train$is_adopted,
                       yhat.all.train.cl,
                       dnn = c("Actual","Predicted"))
tab.all.train
```

```
##       Predicted
## Actual   0   1
##      0 773 227
##      1 162 838
```

```
train.all.err <- mean(yhat.all.train.cl !=
                        dat.train$is_adopted)
train.all.err # 0.1945, 19.45%
```

```
## [1] 0.1945
```

```
# Model 1 All-in training accuracy: 80.55%

#  Create confusion matrix and error on the training
#    data for model 2
yhat.2.train <- predict(lr.2.train, dat.train,
                        type = "response")
yhat.2.train.cl <- ifelse(yhat.2.train > 0.5, 1, 0)
tab.2.train <- table(dat.train$is_adopted, yhat.2.train.cl,
                     dnn = c("Actual","Predicted"))
tab.2.train
```

```
##       Predicted
## Actual   0   1
##      0 776 224
##      1 163 837
```

```
train.2.err <- mean(dat.train$is_adopted != yhat.2.train.cl)
train.2.err # 0.1935, 19.35%
```

```
## [1] 0.1935
```

```
# Model 2 Training Accuracy: 80.65%

#  Now, create confusion matrix and compute the error
#    on the TEST data.  First use the "all-in" model:
yhat.all.test <- predict(lr.all.train, dat.test,
                         type = "response")
yhat.all.test.cl <- ifelse(yhat.all.test > 0.5, 1, 0)
tab.all.test <- table(dat.test$is_adopted, yhat.all.test.cl,
                      dnn = c("Actual","Predicted"))
tab.all.test
```

```
##       Predicted
## Actual    0    1
##      0 8626 2351
##      1 1790 9187
```

```
test.all.err <- mean(dat.test$is_adopted !=
                       yhat.all.test.cl)
test.all.err # 0.1886217, 18.86%
```

```
## [1] 0.1886217
```

```
# Model 1 All-in Testing Accuracy: 81.14%

#  Compute confusion matrix and error for test data using Model 2
yhat.2.test <- predict(lr.2.train, dat.test,
                       type = "response")
yhat.2.test.cl <- ifelse(yhat.2.test > 0.5, 1, 0)
tab.2.test <- table(dat.test$is_adopted, yhat.2.test.cl,
                    dnn = c("Actual","Predicted"))
tab.2.test
```

```
##       Predicted
## Actual    0    1
##      0 8698 2279
##      1 1807 9170
```

```
test.2.err <- mean(dat.test$is_adopted != yhat.2.test.cl)
test.2.err # 0.1861164, 18.61%
```

```
## [1] 0.1861164
```

```
# Model 2 Testing Accuracy: 81.39%

# All-in model shows training and test errors at 19.45% and 18.86% respectively.
# Model 2 shows training and test errors at 19.35% and 18.61% respectively.
# Model 2 (simplified) is BETTER

##### RIDGE REGRESSION and LASSO
library(glmnet)
```

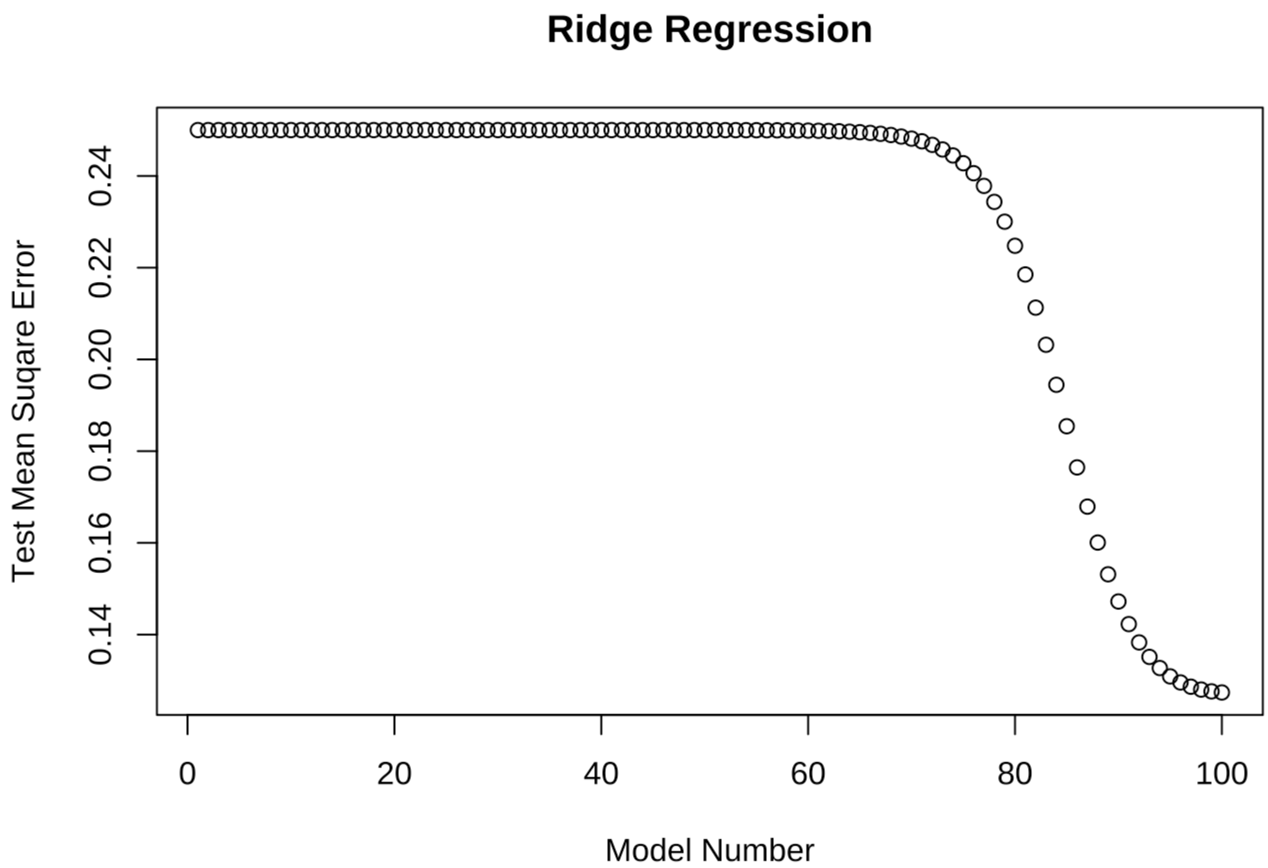
```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-8
```

```
set.seed(123)

X.train <- model.matrix(is_adopted~., dat.train)[,-1]
y.train <- dat.train$is_adopted
X.test <- model.matrix(is_adopted~., dat.test)[,-1]
y.test <- dat.test$is_adopted

#### for plotting ridge regression
y <- dat$is_adopted
X <- model.matrix(is_adopted~., dat)[,-1]

grid <- 10^seq(10,-2,length=100)
ridge.mod <- glmnet(X.train, y.train, alpha = 0, family = "binomial",
                    lambda = grid, thresh = 1e-12)

ridge.coeff <- matrix(0, nrow = ncol(X), ncol = 100)
ridge.pred <- matrix(0,nrow = length(y.test), ncol = 100)
testerr <- matrix(0, nrow = 100, ncol = 1)

for (j in 1:100) {
  ridge.coeff[,j] <- ridge.mod$beta[,j]
  ridge.pred[,j] <- predict(ridge.mod, s = grid[j], type = "response",
                            newx = X.test)
  testerr[j] <- mean((ridge.pred[,j] - y.test)^2)
}

plot(testerr, xlab = "Model Number",
     ylab = "Test Mean Suqare Error",
     main = "Ridge Regression",
     )
```




```
which.min(testerr)
```

```
## [1] 100
```

```
# model 100 has the lowest MSE
ridge.mod$lambda[100]
```

```
## [1] 0.01
```

```
# Lambda = 0.01

##################

# Let's use cross-validation to determine the best lambda to use in ridge regression
cv.out <- cv.glmnet(X.train, y.train, alpha = 0, family = "binomial")
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 0.02544792
```

```
# best lambda = 0.02544792

# fit the ridge regression model
ridge.mod.cv <- glmnet(X.train, y.train, alpha = 0, family = "binomial",
                    lambda = bestlam)

# predict on the testing data
ridge.pred.cv <- predict(ridge.mod.cv, s=bestlam, newx = X.test, type = "response")

MSE.R.CV <- mean((ridge.pred.cv-y.test)^2)
RMSE.R.CV <- MSE.R.CV^0.5
RMSE.R.CV
```

```
## [1] 0.359061
```

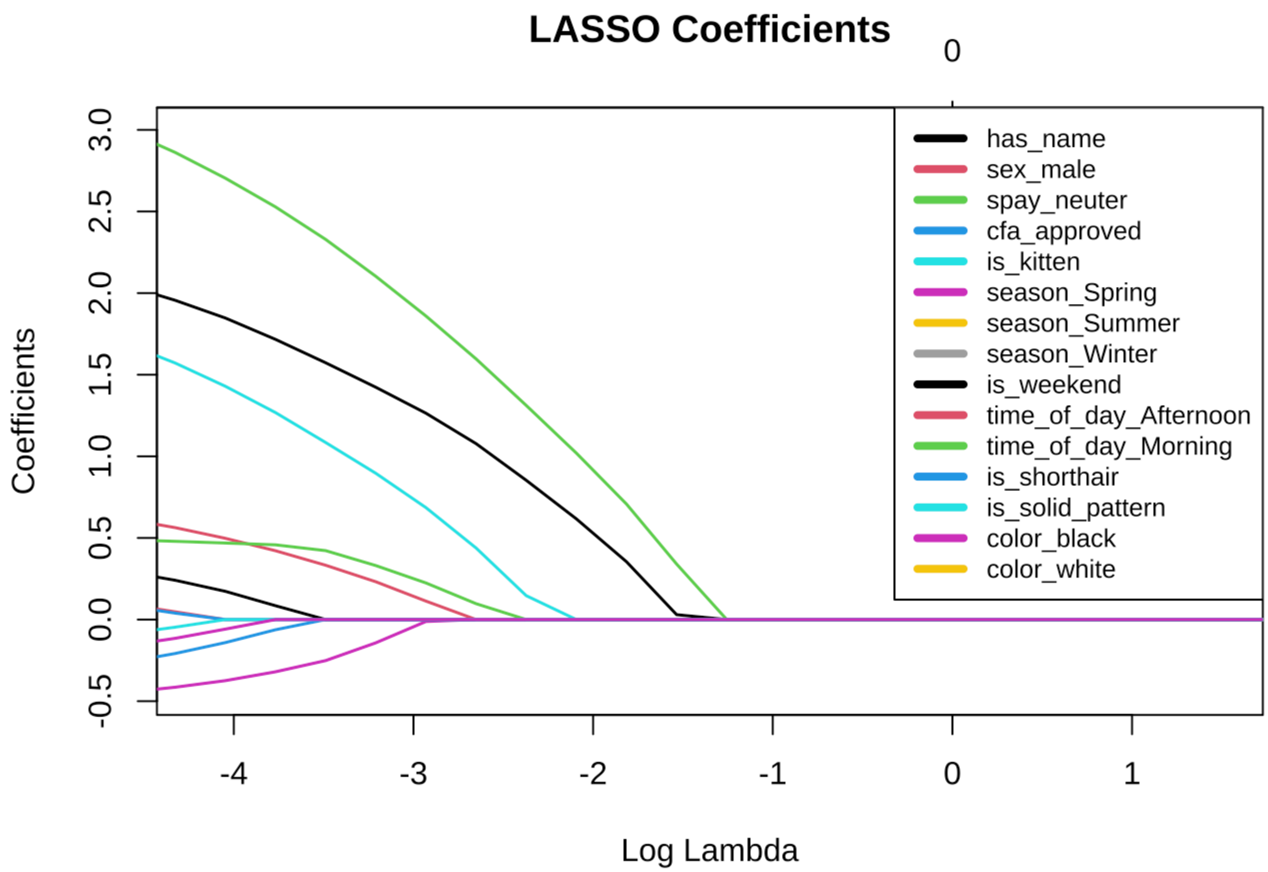
```
# RMSE.R.CV = 0.359061

#  Compute confusion matrix and error for ridge regression using best lambda
yhat.ridge.cl <- ifelse(ridge.pred.cv > 0.5, 1, 0)
tab.4.test <- table(dat.test$is_adopted, yhat.ridge.cl,
                    dnn = c("Actual","Predicted"))
tab.4.test
```

```
##       Predicted
## Actual    0    1
##      0 8623 2354
##      1 1791 9186
```

```
test.4.err <- mean(dat.test$is_adopted != yhat.ridge.cl)
test.4.err # 0.1888039, 18.88%
```

```
## [1] 0.1888039
```

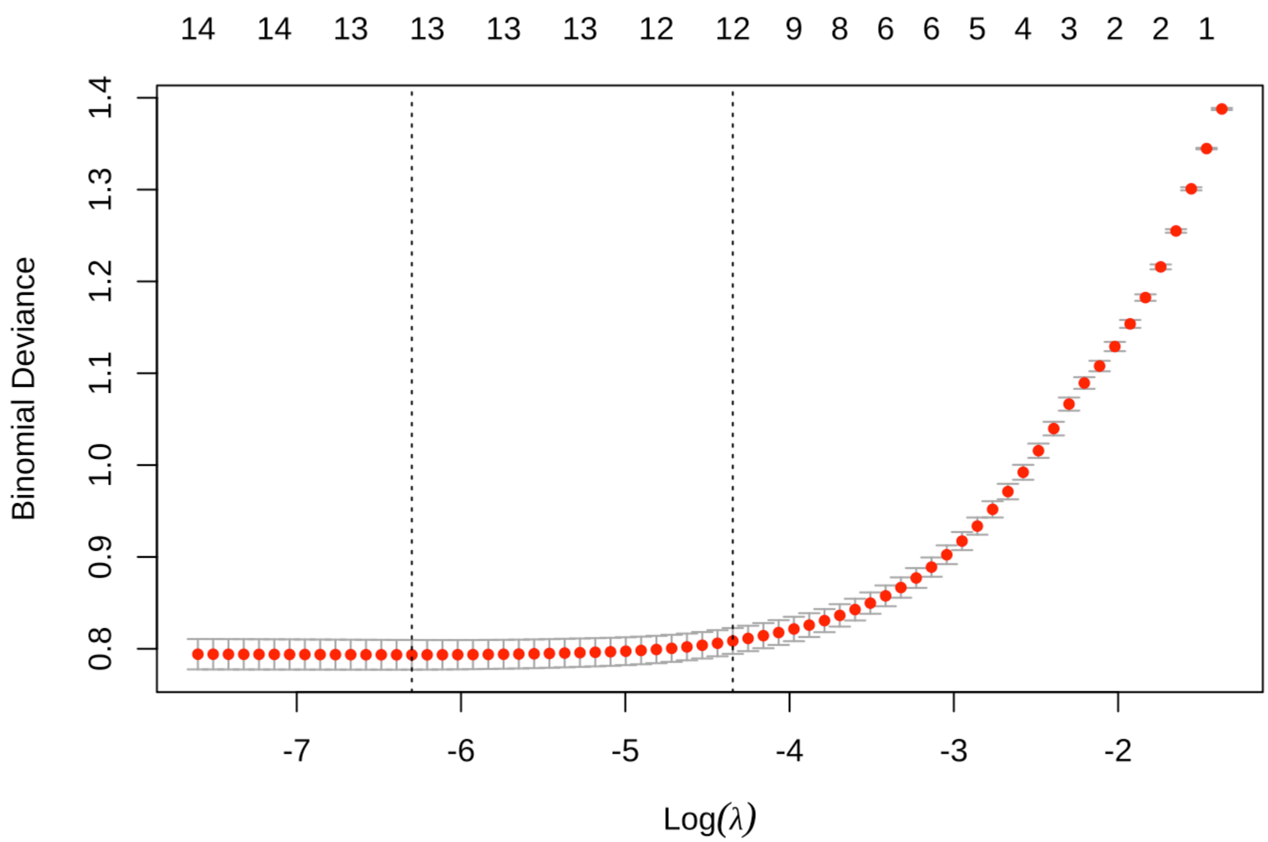
```
# Ridge Regression (best lambda) Testing Accuracy: 81.12%
# Logistic Regression Model 2 Testing Accuracy: 81.39%

### LASSO regression
# plotting LASSO coefficients that are being driven to 0
lasso.mod <- glmnet(X.train, y.train, alpha=1,  family = "binomial",
                    lambda=grid, thresh = 1e-12)
plot(lasso.mod, xvar="lambda", label = FALSE, lwd = 1.5,
     xlim=c(-4.2, 1.5), main = "LASSO Coefficients")

var_names <- colnames(X.train)
legend("topright", legend = var_names, col = 1:ncol(coef(lasso.mod)), lty = 1,
       cex = 0.8, lwd = 4)
```




```
# Let's use cross-validation to determine the best lambda for LASSO
cv.out1 <- cv.glmnet(X.train, y.train,  family = "binomial", alpha = 1)
coef(cv.out1) # for reference on which variables were driven to 0 by LASSO
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)           -4.60454729
## has_name               1.96220971
## sex_male               0.05104639
## spay_neuter            2.87126205
## cfa_approved           0.04291348
## is_kitten              1.58030600
## season_Spring         -0.41701319
## season_Summer          .
## season_Winter          0.24486343
## is_weekend             0.56729046
## time_of_day_Afternoon  0.48044590
## time_of_day_Morning   -0.21166674
## is_shorthair          -0.04980740
## is_solid_pattern      -0.11835015
## color_black            .
## color_white            .
```

```
# plot LASSO
plot(cv.out1, main = "")
```




```
# find best lambda for LASSO
bestlam1 <- cv.out1$lambda.min
bestlam1
```

```
## [1] 0.001837546
```

```
# predict on testing data
lasso.pred.cv <- predict(lasso.mod, s=bestlam1, type = "response",
                        newx = X.test)
MSE.L.CV <- mean((lasso.pred.cv-y.test)^2)
RMSE.L.CV <- MSE.L.CV^0.5
RMSE.L.CV
```

```
## [1] 0.3576907
```

```
# RMSE.L.CV (lasso) = 0.3576907
RMSE.R.CV
```

```
## [1] 0.359061
```

```
# RMSE.R.CV (ridge) = 0.359061

# RMSE is around the same for LASSO and Ridge. LASSO slightly better.

#  Compute confusion matrix and error for LASSO using best lambda
yhat.lasso.cl <- ifelse(lasso.pred.cv > 0.5, 1, 0)
tab.5.test <- table(dat.test$is_adopted, yhat.lasso.cl,
                    dnn = c("Actual","Predicted"))
tab.5.test
```

```
##       Predicted
## Actual    0    1
##      0 8561 2416
##      1 1761 9216
```

```
test.5.err <- mean(dat.test$is_adopted != yhat.lasso.cl)
test.5.err # 0.1902615, 19.03%
```

```
## [1] 0.1902615
```

```
# LASSO Regression (best lambda) Testing Accuracy: 80.97%
# Ridge Regression (best lambda) Testing Accuracy: 81.12%
# Logistic Regression Model 2 Testing Accuracy: 81.39%

# Logistic Regression Model 1 (All-in)
test.all.err
```

```
## [1] 0.1886217
```

```
# Logistic Regression Model 2 (simplified model)
test.2.err
```

```
## [1] 0.1861164
```

```
# Ridge Regression
test.4.err
```

```
## [1] 0.1888039
```

```
# LASSO Regression
test.5.err
```

```
## [1] 0.1902615
```

```
# Both models perform the same:
# These are the coefficients being driven to 0 by LASSO
coef(cv.out1)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)          -4.60454729
## has_name              1.96220971
## sex_male              0.05104639
## spay_neuter           2.87126205
## cfa_approved          0.04291348
## is_kitten             1.58030600
## season_Spring        -0.41701319
## season_Summer         .
## season_Winter         0.24486343
## is_weekend            0.56729046
## time_of_day_Afternoon  0.48044590
## time_of_day_Morning   -0.21166674
## is_shorthair         -0.04980740
## is_solid_pattern     -0.11835015
## color_black           .
## color_white           .
```

```
# season_Summer, color_black, color_white

#### CONCLUSION
# According to the results, the simplified logistic regression gives us the best accurac
y
# and trade off for model complexity and interpretability.

sum.lr.2.train
```

```
##
## Call:
## glm(formula = is_adopted ~ . - cfa_approved - season_Summer -
##     is_shorthair - color_black - color_white, family = "binomial",
##     data = dat.train)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -5.8889     0.3331 -17.680  < 2e-16 ***
## has_name                2.4488     0.1631  15.018  < 2e-16 ***
## sex_male                0.2725     0.1301   2.095 0.036213 *
## spay_neuter             3.5572     0.2069  17.190  < 2e-16 ***
## is_kitten               2.1560     0.1654  13.038  < 2e-16 ***
## season_Spring          -0.5826     0.1655  -3.520 0.000431 ***
## season_Winter           0.5061     0.1722   2.940 0.003285 **
## is_weekend              0.8247     0.1369   6.022 1.72e-09 ***
## time_of_day_Afternoon   0.5283     0.1737   3.041 0.002358 **
## time_of_day_Morning    -0.4585     0.1817  -2.523 0.011638 *
## is_solid_pattern       -0.3472     0.1323  -2.624 0.008690 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2772.6  on 1999  degrees of freedom
## Residual deviance: 1561.2  on 1989  degrees of freedom
## AIC: 1583.2
##
## Number of Fisher Scoring iterations: 6
```

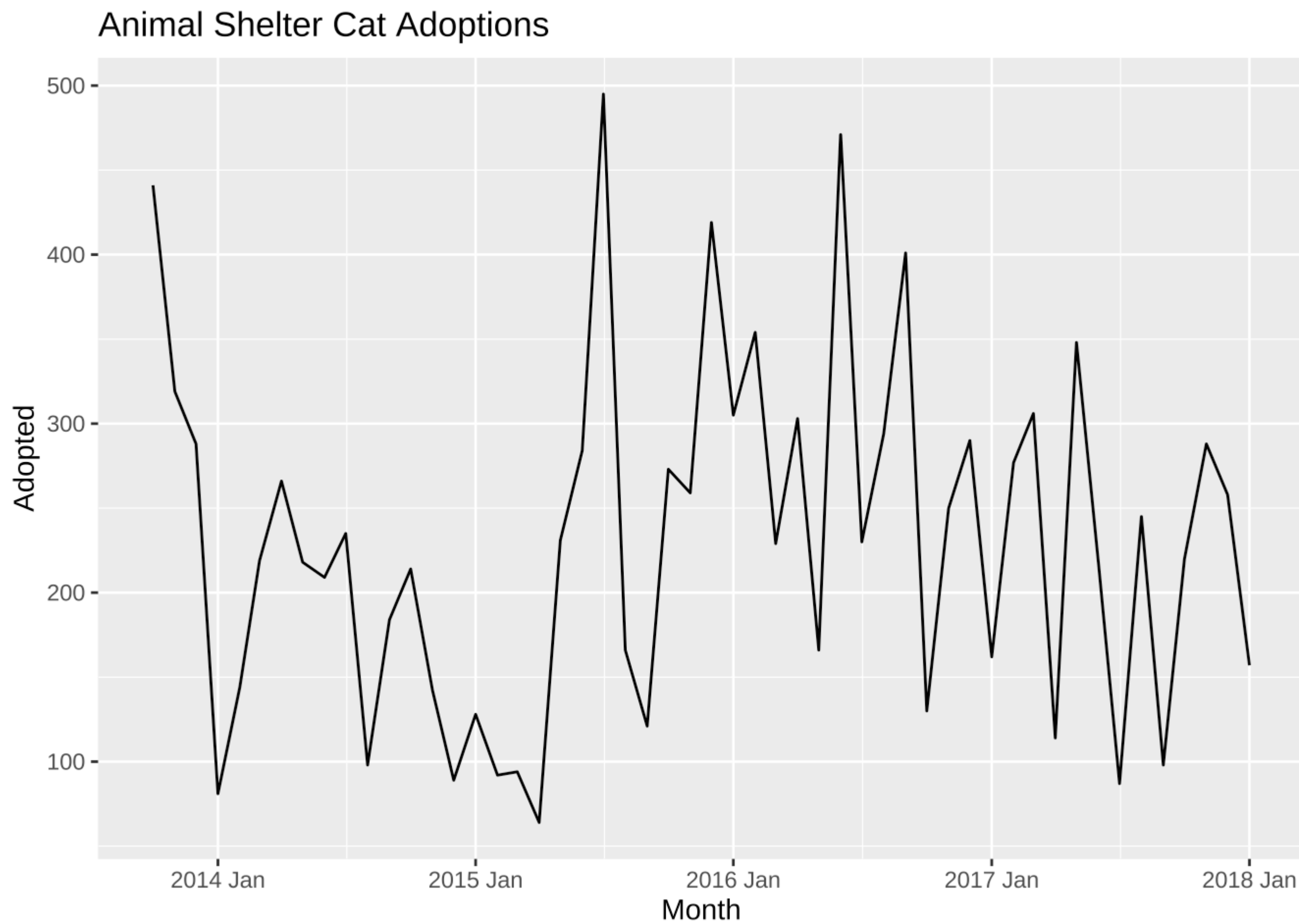
```
# BEST MODEL!!!
# Logistic Regression Model 2 Simplified (10 predictors)
coef(lr.2.train)
```

```
##           (Intercept)              has_name              sex_male
##            -5.8889201             2.4488180             0.2724767
##           spay_neuter             is_kitten         season_Spring
##             3.5571981             2.1560202            -0.5826473
##         season_Winter            is_weekend time_of_day_Afternoon
##             0.5060945             0.8246565             0.5283066
##   time_of_day_Morning      is_solid_pattern
##            -0.4585087            -0.3471697
```

```
####### FORECASTING ########
## ETS
setwd("/Users/andrewgatchalian/Documents/UCI MSBA 24/Winter Quarter/BA 288 Predictive An
alytics/Final Project/data")

dat.f <- read.csv("cat_data_cleaned_updated_forecast.csv")
dat.f <- dat.f[-53, ] # last month is outlier, # of observations ends
dat.f$Date <- as.Date(dat.f$Date, format = "%m/%d/%Y")
ADPts <- ts(dat.f[,2], start = c(2013, 10), frequency = 12)
ADPtts <- as_tsibble(ADPts)
names(ADPtts)[2] <- "Adopted"
str(ADPtts)
```

```
## tbl_ts [52 × 2] (S3: tbl_ts/tbl_df/tbl/data.frame)
##  $ index  : mth [1:52] 2013 Oct, 2013 Nov, 2013 Dec, 2014 Jan, 2014 Feb, 2014 Mar,...
##  $ Adopted: num [1:52] 441 319 288 81 144 219 266 218 209 235 ...
##  - attr(*, "key")= tibble [1 × 1] (S3: tbl_df/tbl/data.frame)
##   ..$ .rows: list<int> [1:1]
##   .. ..$ : int [1:52] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..@ ptype: int(0)
##  - attr(*, "index")= chr "index"
##   ..- attr(*, "ordered")= logi TRUE
##  - attr(*, "index2")= chr "index"
##  - attr(*, "interval")= interval [1:1] 1M
##   ..@ .regular: logi TRUE
```

```
autoplot(ADPtts, Adopted) +
  labs(y = "Adopted", title = "Animal Shelter Cat Adoptions",
       x = "Month")
```




```
# best ETS model
fit_ETS <- model(ADPtts, ETS(Adopted))
fit_ETS
```

```
## # A mable: 1 x 1
##   `ETS(Adopted)`
##          <model>
## 1   <ETS(A,N,N)>
```

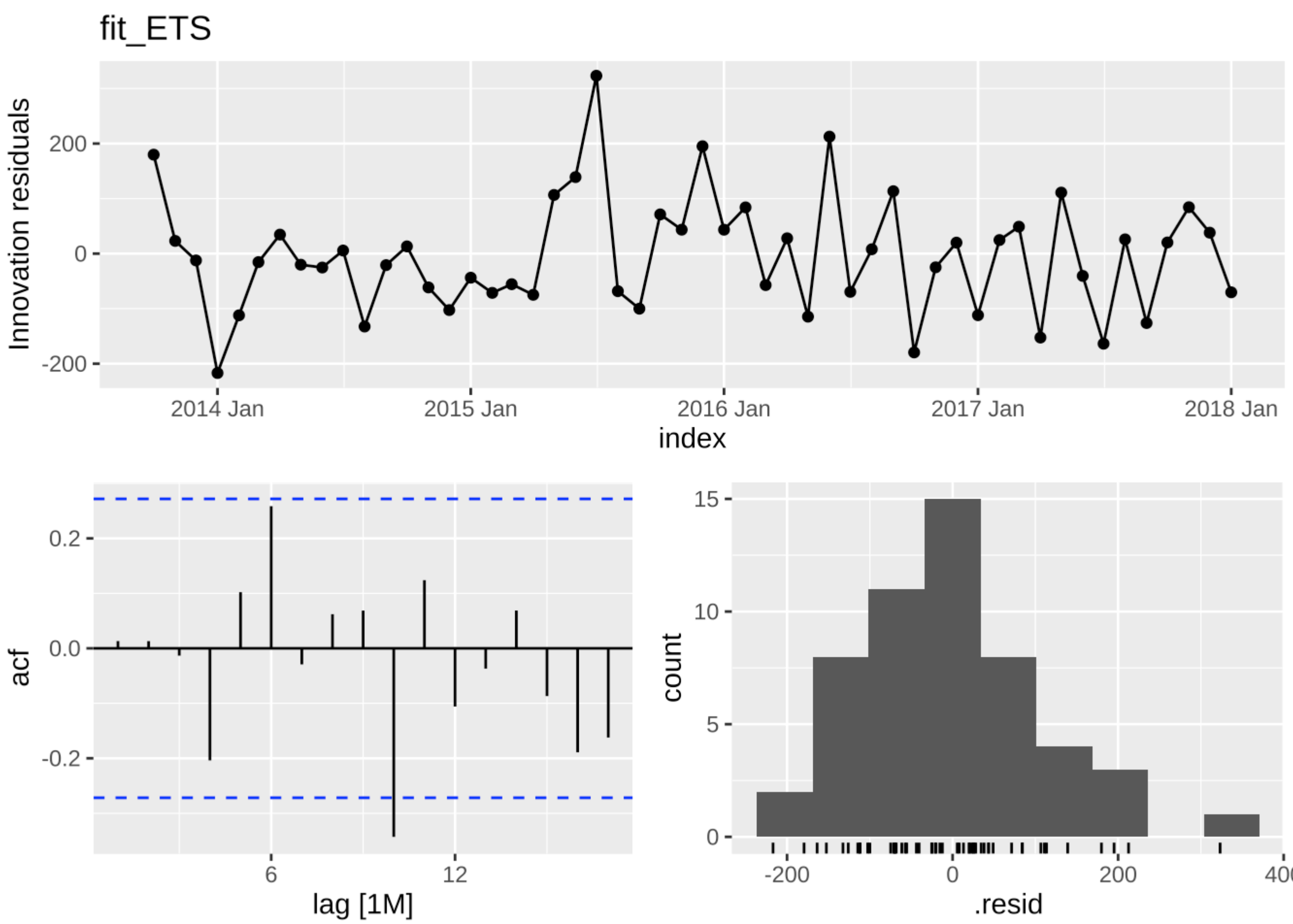
```
accuracy(fit_ETS) # 105
```

```
## # A tibble: 1 × 10
##   .model       .type       ME  RMSE   MAE   MPE  MAPE  MASE RMSSE   ACF1
##   <chr>        <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 ETS(Adopted) Training -4.72  105.  81.6 -26.5  48.5 0.581 0.632 0.0132
```

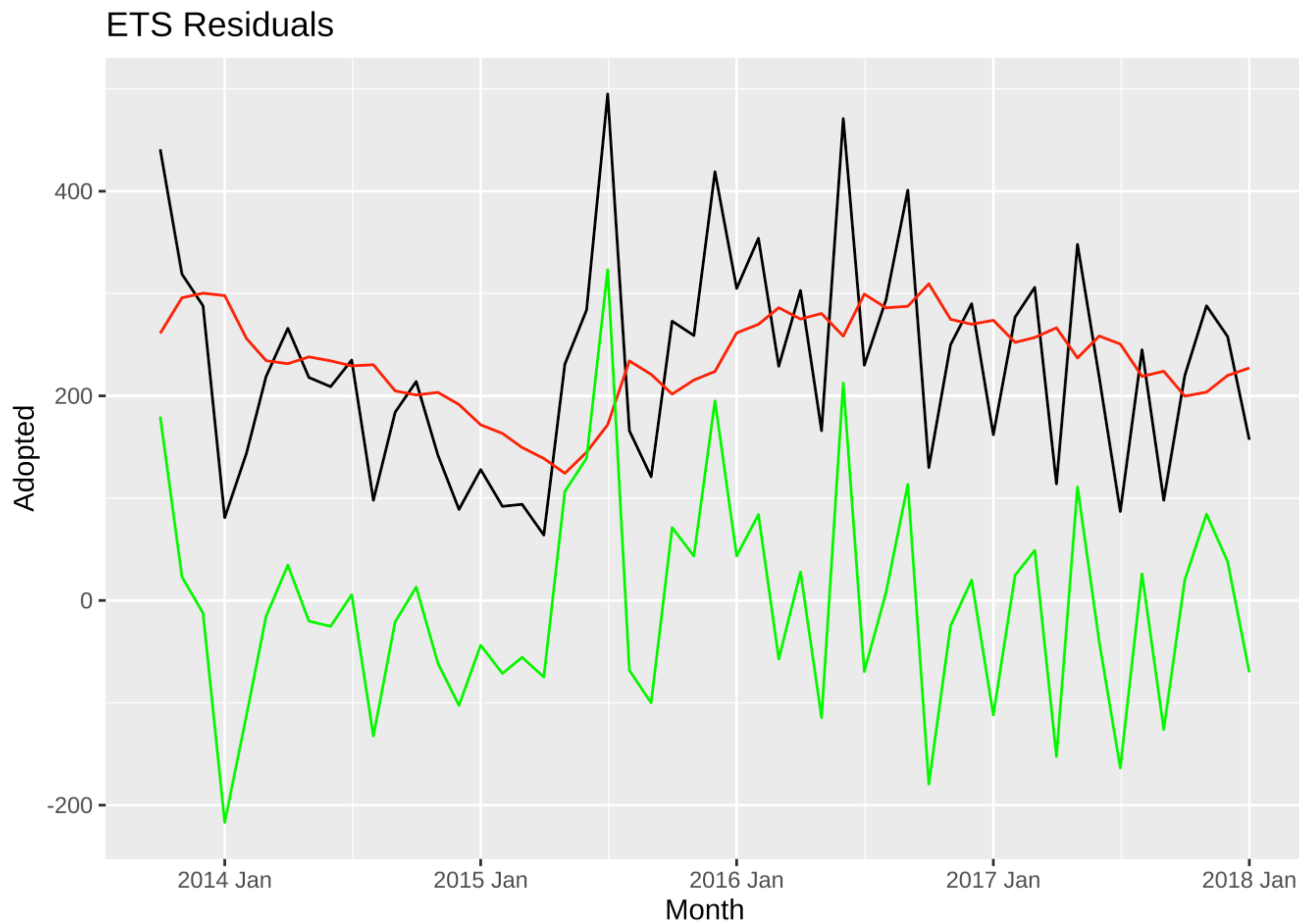
```
# parameters
report(fit_ETS)
```

```
## Series: Adopted
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.1930786
##
##   Initial states:
##      l[0]
##  261.1431
##
##   sigma^2:  11408.59
##
##      AIC     AICc      BIC
## 695.2155 695.7155 701.0693
```

```
gg_tsresiduals(fit_ETS) +
  ggtitle("fit_ETS")
```




```
# Display the model components (chart)
aug_ETS <- augment(fit_ETS)
autoplot(aug_ETS, Adopted) +
  autolayer(aug_ETS,.fitted, colour = "Red") +
  autolayer(aug_ETS,.resid, colour = "Green") +
  labs(y = "Adopted", title = "ETS Residuals",
       x = "Month")
```




```
# forecast next year
forc_ETS <- forecast(fit_ETS, h = 12)
forc_ETS
```

```
## # A fable: 12 x 4 [1M]
## # Key:     .model [1]
##    .model          index       Adopted .mean
##    <chr>           <mth>        <dist> <dbl>
##  1 ETS(Adopted) 2018 Feb N(214, 11409)  214.
##  2 ETS(Adopted) 2018 Mar N(214, 11834)  214.
##  3 ETS(Adopted) 2018 Apr N(214, 12259)  214.
##  4 ETS(Adopted) 2018 May N(214, 12685)  214.
##  5 ETS(Adopted) 2018 Jun N(214, 13110)  214.
##  6 ETS(Adopted) 2018 Jul N(214, 13535)  214.
##  7 ETS(Adopted) 2018 Aug N(214, 13960)  214.
##  8 ETS(Adopted) 2018 Sep N(214, 14386)  214.
##  9 ETS(Adopted) 2018 Oct N(214, 14811)  214.
## 10 ETS(Adopted) 2018 Nov N(214, 15236)  214.
## 11 ETS(Adopted) 2018 Dec N(214, 15662)  214.
## 12 ETS(Adopted) 2019 Jan N(214, 16087)  214.
```

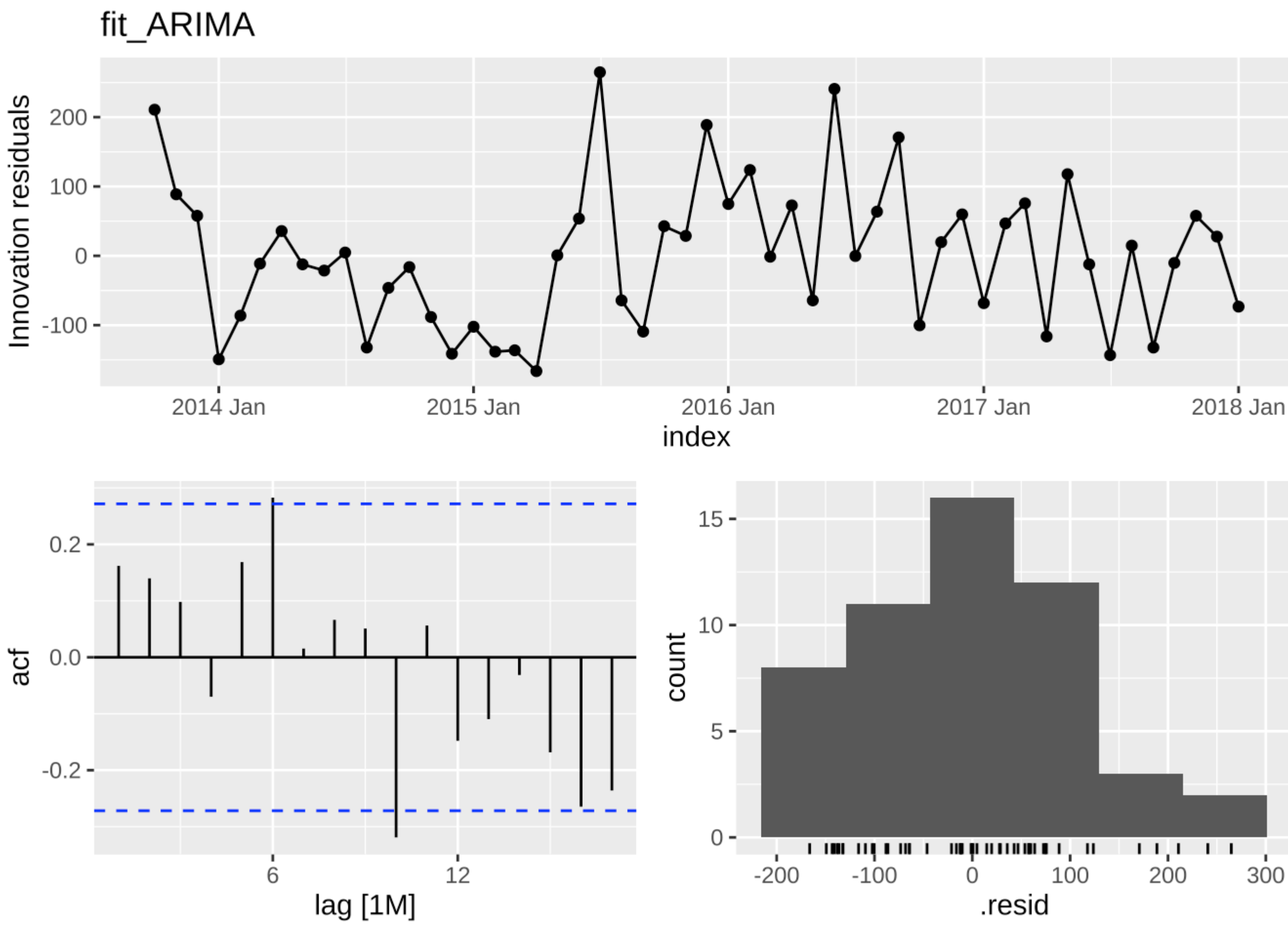
```
######## ARIMA
library(fpp3)

fit_ARIMA <- model(ADPtts, ARIMA(Adopted))
accuracy(fit_ARIMA) # RMSE 104
```

```
## # A tibble: 1 × 10
##   .model         .type          ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>          <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(Adopted) Training 3.83e-15  104.  82.4 -27.7  50.6 0.587 0.627 0.162
```

```
report(fit_ARIMA)
```

```
## Series: Adopted
## Model: ARIMA(0,0,0) w/ mean
##
## Coefficients:
##       constant
##       230.2692
## s.e.   14.4107
##
## sigma^2 estimated as 11010:  log likelihood=-315.25
## AIC=634.5   AICc=634.75   BIC=638.41
```

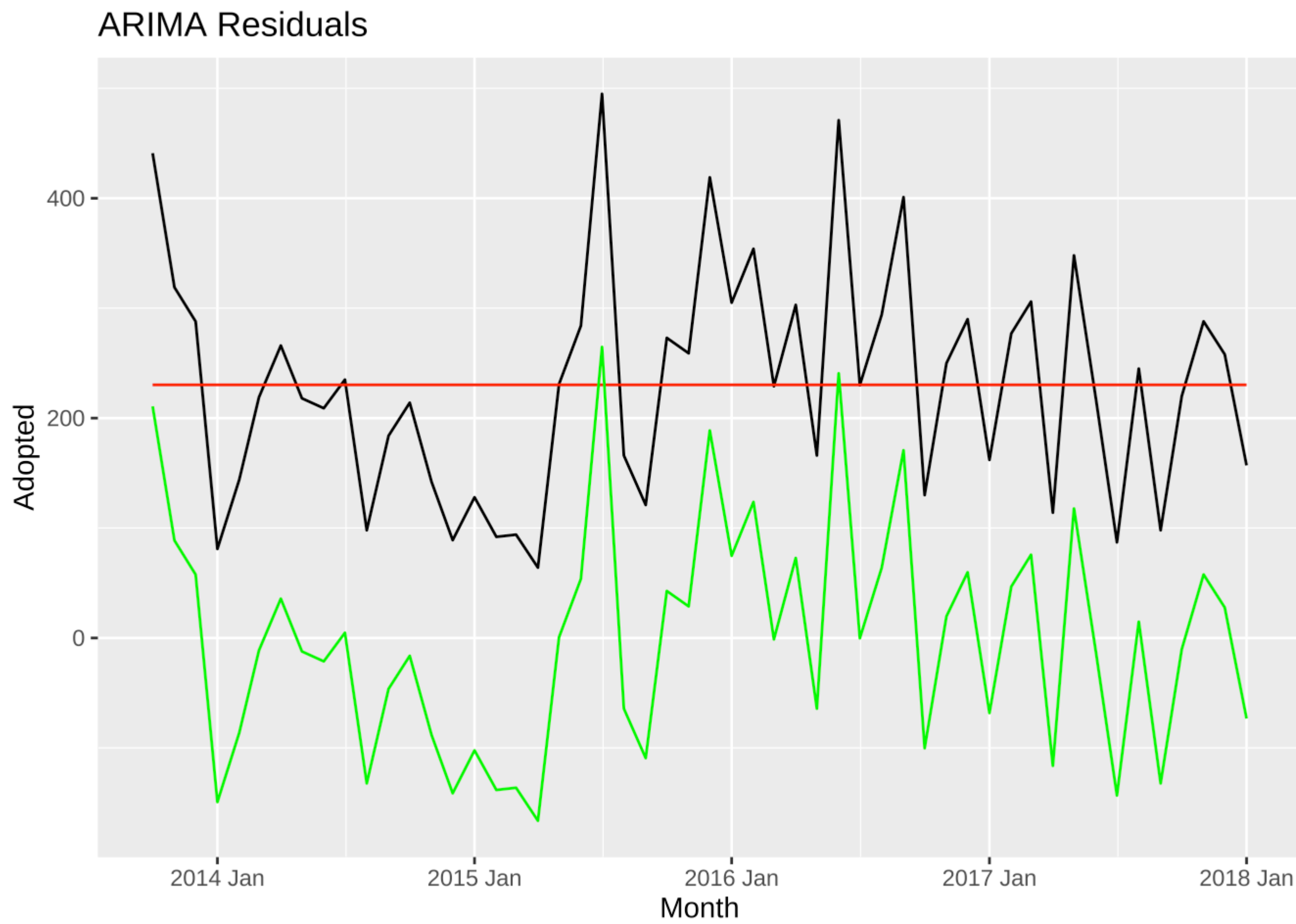
```
gg_tsresiduals(fit_ARIMA)+
  ggtitle("fit_ARIMA")
```



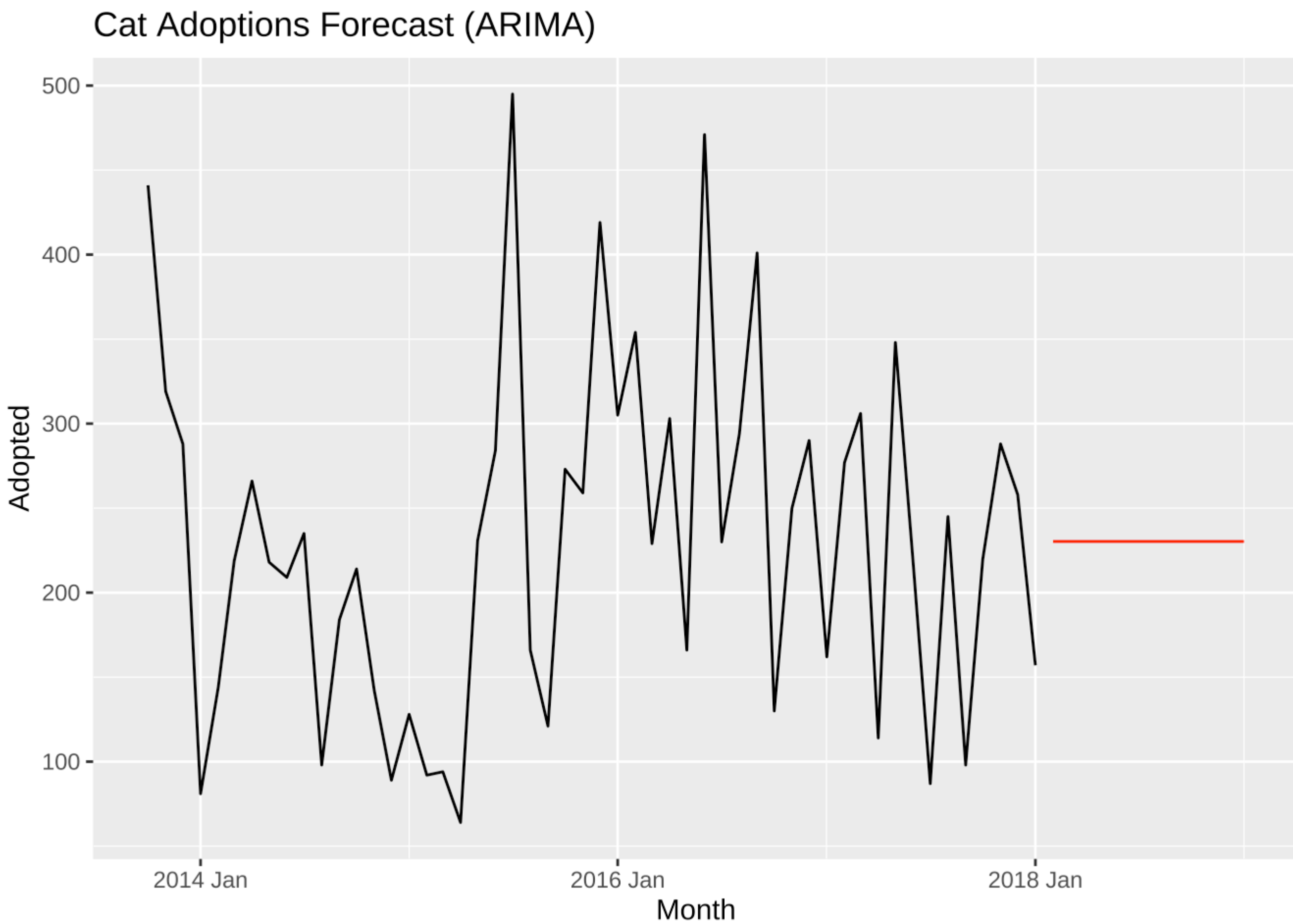

```
aug_ARIMA <- augment(fit_ARIMA)
aug_ARIMA
```

```
## # A tsibble: 52 x 6 [1M]
## # Key:       .model [1]
##    .model            index Adopted .fitted  .resid  .innov
##    <chr>             <mth>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 ARIMA(Adopted) 2013 Oct     441    230.  211.    211.
##  2 ARIMA(Adopted) 2013 Nov     319    230.   88.7    88.7
##  3 ARIMA(Adopted) 2013 Dec     288    230.   57.7    57.7
##  4 ARIMA(Adopted) 2014 Jan      81    230. -149.   -149.
##  5 ARIMA(Adopted) 2014 Feb     144    230.  -86.3   -86.3
##  6 ARIMA(Adopted) 2014 Mar     219    230.  -11.3   -11.3
##  7 ARIMA(Adopted) 2014 Apr     266    230.   35.7    35.7
##  8 ARIMA(Adopted) 2014 May     218    230.  -12.3   -12.3
##  9 ARIMA(Adopted) 2014 Jun     209    230.  -21.3   -21.3
## 10 ARIMA(Adopted) 2014 Jul     235    230.    4.73    4.73
## # ℹ 42 more rows
```

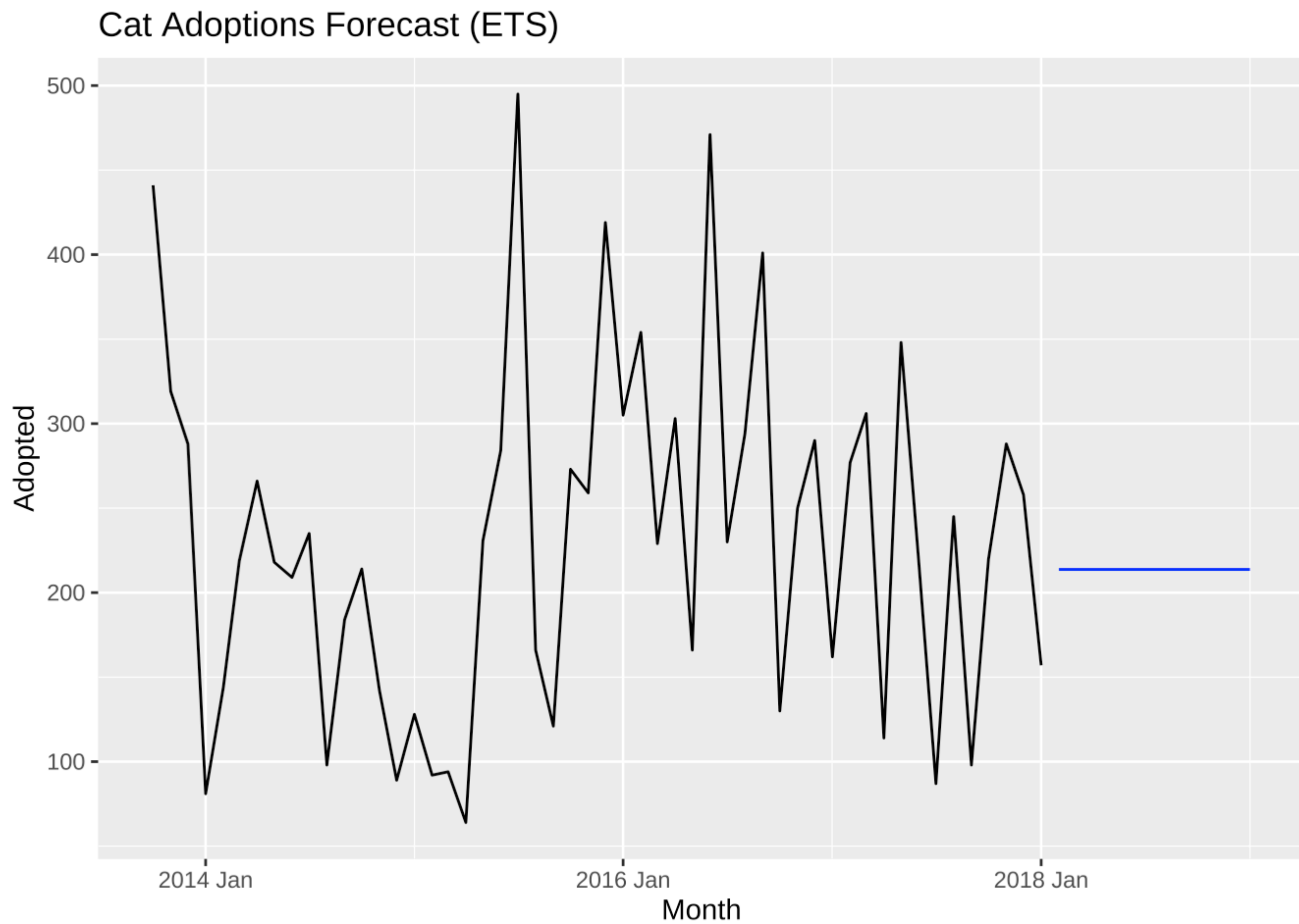
```
autoplot(aug_ARIMA, Adopted) +
  autolayer(aug_ARIMA,.fitted, colour = "Red") +
  autolayer(aug_ARIMA,.resid, colour = "Green") +
  labs(y = "Adopted", title = "ARIMA Residuals",
       x = "Month")
```



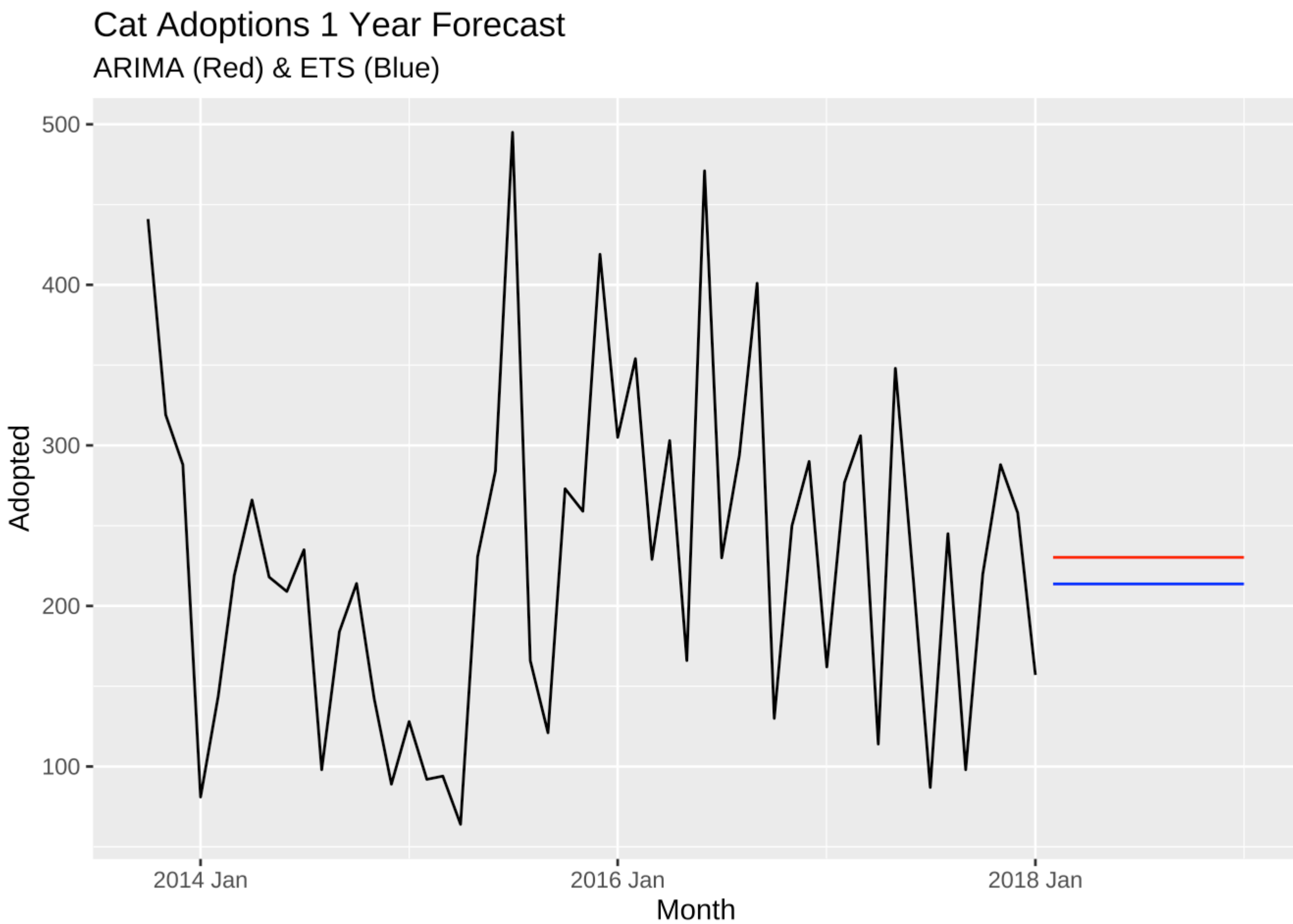

```
forc_ARIMA <- forecast(fit_ARIMA, h = 12)

# plot ARIMA forecast
autoplot(forc_ARIMA, ADPtts, level = NULL, colour = "Red") +
  labs(y = "Adopted", title = "Cat Adoptions Forecast (ARIMA)",
       x = "Month")
```




```
# plot ETS forecast
autoplot(forc_ETS, ADPtts, level = NULL, colour = "Blue") +
  labs(y = "Adopted", title = "Cat Adoptions Forecast (ETS)",
       x = "Month")
```




```
# plot both forecast
autoplot(forc_ETS, ADPtts, level = NULL, colour = "Blue") +
  autolayer(forc_ARIMA, ADPtts, level = NULL, colour = "Red") +
  labs(y = "Adopted", title = "Cat Adoptions 1 Year Forecast",
       x = "Month", subtitle = "ARIMA (Red) & ETS (Blue)")
```




```
forc_ARIMA # 230 adoptions
```

```
## # A fable: 12 x 4 [1M]
## # Key:     .model [1]
##    .model            index       Adopted .mean
##    <chr>             <mth>        <dist> <dbl>
##  1 ARIMA(Adopted) 2018 Feb N(230, 11010)  230.
##  2 ARIMA(Adopted) 2018 Mar N(230, 11010)  230.
##  3 ARIMA(Adopted) 2018 Apr N(230, 11010)  230.
##  4 ARIMA(Adopted) 2018 May N(230, 11010)  230.
##  5 ARIMA(Adopted) 2018 Jun N(230, 11010)  230.
##  6 ARIMA(Adopted) 2018 Jul N(230, 11010)  230.
##  7 ARIMA(Adopted) 2018 Aug N(230, 11010)  230.
##  8 ARIMA(Adopted) 2018 Sep N(230, 11010)  230.
##  9 ARIMA(Adopted) 2018 Oct N(230, 11010)  230.
## 10 ARIMA(Adopted) 2018 Nov N(230, 11010)  230.
## 11 ARIMA(Adopted) 2018 Dec N(230, 11010)  230.
## 12 ARIMA(Adopted) 2019 Jan N(230, 11010)  230.
```

```
forc_ETS # 214 adoptions
```

```
## # A fable: 12 x 4 [1M]
## # Key:     .model [1]
##    .model          index       Adopted .mean
##    <chr>           <mth>        <dist> <dbl>
##  1 ETS(Adopted) 2018 Feb N(214, 11409)  214.
##  2 ETS(Adopted) 2018 Mar N(214, 11834)  214.
##  3 ETS(Adopted) 2018 Apr N(214, 12259)  214.
##  4 ETS(Adopted) 2018 May N(214, 12685)  214.
##  5 ETS(Adopted) 2018 Jun N(214, 13110)  214.
##  6 ETS(Adopted) 2018 Jul N(214, 13535)  214.
##  7 ETS(Adopted) 2018 Aug N(214, 13960)  214.
##  8 ETS(Adopted) 2018 Sep N(214, 14386)  214.
##  9 ETS(Adopted) 2018 Oct N(214, 14811)  214.
## 10 ETS(Adopted) 2018 Nov N(214, 15236)  214.
## 11 ETS(Adopted) 2018 Dec N(214, 15662)  214.
## 12 ETS(Adopted) 2019 Jan N(214, 16087)  214.
```

```
fit_ARIMA # <ARIMA(0,0,0) w/ mean>
```

```
## # A mable: 1 x 1
##         `ARIMA(Adopted)`
##                  <model>
## 1 <ARIMA(0,0,0) w/ mean>
```

```
fit_ETS # <ETS(A,N,N)>
```

```
## # A mable: 1 x 1
##   `ETS(Adopted)`
##          <model>
## 1   <ETS(A,N,N)>
```

```
accuracy(fit_ARIMA)
```

```
## # A tibble: 1 × 10
##   .model         .type          ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>          <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(Adopted) Training 3.83e-15  104.  82.4 -27.7  50.6 0.587 0.627 0.162
```

```
accuracy(fit_ETS)
```

```
## # A tibble: 1 × 10
##   .model       .type       ME  RMSE   MAE   MPE  MAPE  MASE RMSSE   ACF1
##   <chr>        <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 ETS(Adopted) Training -4.72  105.  81.6 -26.5  48.5 0.581 0.632 0.0132
```

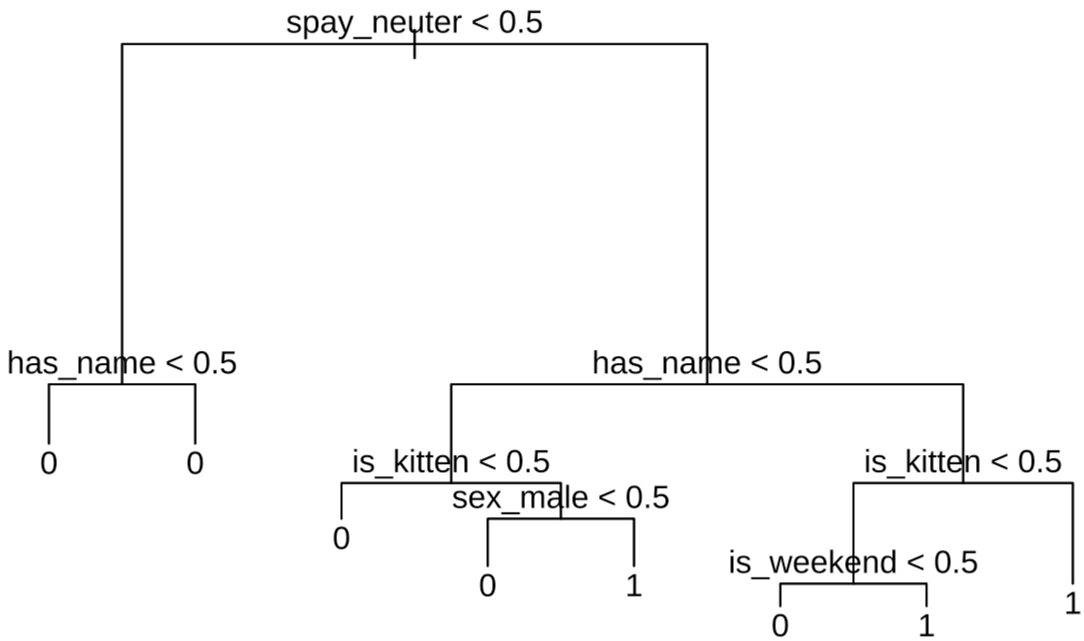
```r
#################
#### Tree
#tree model without any feature selection and parameter tunning
library(tree)
dat.train[,2] <- as.factor(dat.train[,2])
dat.test[,2] <- as.factor(dat.test[,2])
str(dat.train)
```

```
## 'data.frame':    2000 obs. of  16 variables:
##  $ has_name             : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ is_adopted           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ sex_male             : int  0 1 1 1 1 0 0 0 0 1 ...
##  $ spay_neuter          : int  1 1 1 0 1 1 1 1 1 1 ...
##  $ cfa_approved         : int  0 0 1 0 0 0 0 1 0 1 ...
##  $ is_kitten            : int  1 0 0 1 0 0 0 0 0 1 ...
##  $ season_Spring        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ season_Summer        : int  0 0 0 1 0 1 0 0 0 0 ...
##  $ season_Winter        : int  1 1 0 0 0 0 1 1 1 0 ...
##  $ is_weekend           : int  0 0 1 1 1 1 1 0 0 1 ...
##  $ time_of_day_Afternoon: int  0 1 1 0 1 1 0 0 0 1 ...
##  $ time_of_day_Morning  : int  1 0 0 1 0 0 0 1 1 0 ...
##  $ is_shorthair         : int  1 1 0 1 0 1 1 0 1 0 ...
##  $ is_solid_pattern     : int  1 0 0 0 0 1 0 0 0 0 ...
##  $ color_black          : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ color_white          : int  1 0 0 0 0 0 0 0 0 0 ...
```

```r
tree1 <- tree(is_adopted~., data = dat.train)
summary(tree1)
```

```
##
## Classification tree:
## tree(formula = is_adopted ~ ., data = dat.train)
## Variables actually used in tree construction:
## [1] "spay_neuter" "has_name"    "is_kitten"   "sex_male"    "is_weekend"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7857 = 1565 / 1992
## Misclassification error rate: 0.1955 = 391 / 2000
```

```r
#number of terminal nodes is 8
#residual mean deviance is around 0.7857
#error rate is 0.1955
plot(tree1)
text(tree1, pretty = 0)
```




```
#spay_neuter is the variable has the most predicting power
#the ranking of the importance of features are
# 1. spay_neuter
# 2. has_name
# 3. is_kitten
# 4. sex_male
# 5. is_weekend

tree1.pred.tr <- predict(tree1, dat.train, type = "class")
table(dat.train$is_adopted, tree1.pred.tr,
      dnn = c("Actual", "Predicted"))
```

```
##       Predicted
## Actual   0   1
##      0 915  85
##      1 306 694
```

```
err.tree.tr <- mean(dat.train$is_adopted != tree1.pred.tr) #0.2
err.tree.tr # 0.1955, 19.55%
```

```
## [1] 0.1955
```

```r
# Tree TRAINING Accuracy 80.45%

tree.pred.tst <- predict(tree1, dat.test, type = "class")
table(dat.test$is_adopted, tree.pred.tst,
      dnn = c("Actual", "Predicted"))
```

```
##        Predicted
## Actual     0     1
##      0 10031   946
##      1  3297  7680
```

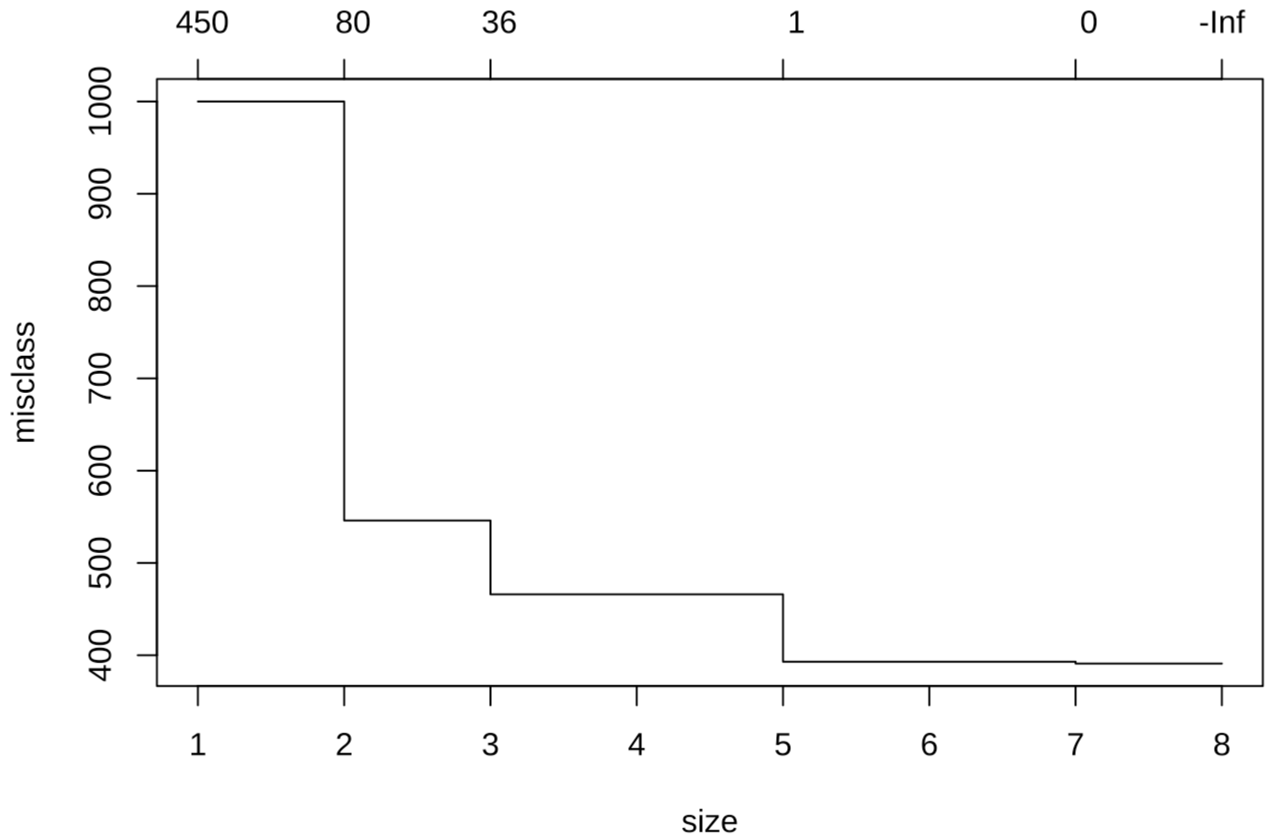
```r
err.tree.tst <- mean(dat.test$is_adopted != tree.pred.tst) #0.198
err.tree.tst #0.1932677, 19.33%
```

```
## [1] 0.1932677
```

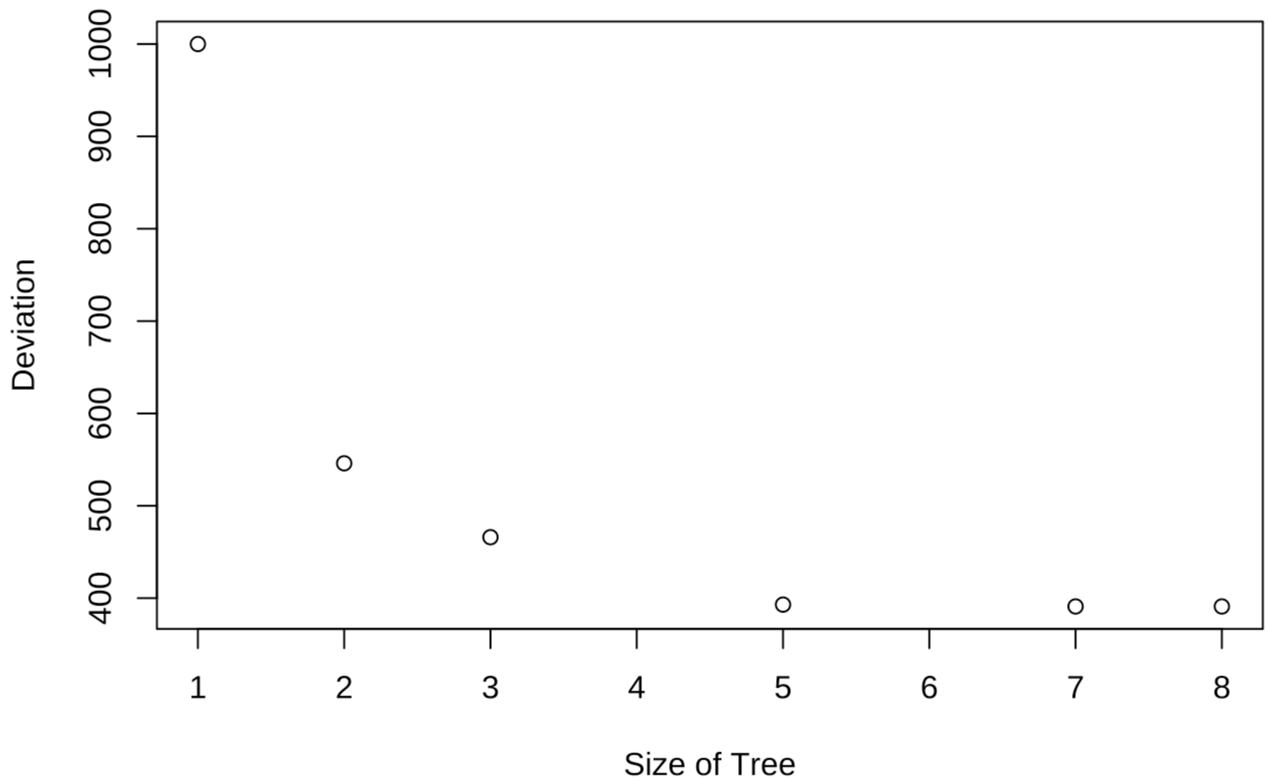
```r
# Tree TESTING Accuracy 80.67%

# Lets see if we can improve tree by pruning
prune1 <- prune.misclass(tree1)
names(prune1)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```r
plot(prune1)
```




```
plot(prune1$size, prune1$dev, xlab = "Size of Tree",
     ylab = "Deviation")
```



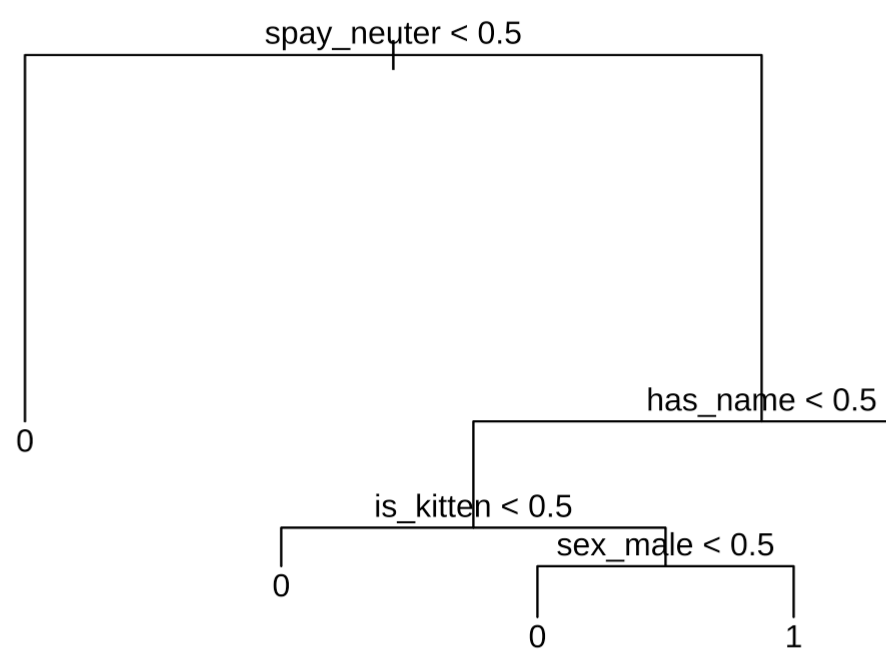

```
prune.tree1 <- prune.misclass(tree1, best=5 )
summary(prune.tree1)
```

```
##
## Classification tree:
## snip.tree(tree = tree1, nodes = c(2L, 7L))
## Variables actually used in tree construction:
## [1] "spay_neuter" "has_name"    "is_kitten"   "sex_male"
## Number of terminal nodes:  5
## Residual mean deviance:  0.9412 = 1878 / 1995
## Misclassification error rate: 0.1965 = 393 / 2000
```

```
prune.tree1
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2000 2773.00 0 ( 0.50000 0.50000 )
##    2) spay_neuter < 0.5 548  320.70 0 ( 0.91423 0.08577 ) *
##    3) spay_neuter > 0.5 1452 1869.00 1 ( 0.34366 0.65634 )
##      6) has_name < 0.5 448  606.70 0 ( 0.58929 0.41071 )
##       12) is_kitten < 0.5 139  118.00 0 ( 0.84892 0.15108 ) *
##       13) is_kitten > 0.5 309  427.40 1 ( 0.47249 0.52751 )
##         26) sex_male < 0.5 218  287.70 0 ( 0.62844 0.37156 ) *
##         27) sex_male > 0.5 91   58.72 1 ( 0.09890 0.90110 ) *
##      7) has_name > 0.5 1004 1093.00 1 ( 0.23406 0.76594 ) *
```

```
plot(prune.tree1)
text(prune.tree1, pretty = 0)
```




```
tree2.pred.tr <- predict(tree1, dat.train, type = "class")
table(dat.train$is_adopted, tree2.pred.tr,
      dnn = c("Actual", "Predicted"))
```

```
##       Predicted
## Actual   0   1
##      0 915  85
##      1 306 694
```

```
err.ptree.tr <- mean(dat.train$is_adopted != tree2.pred.tr)
err.ptree.tr #0.1955, 19.55%
```

```
## [1] 0.1955
```

```
# Pruned Tree TRAINING Accuracy 80.45% (same as un-pruned tree)

tree2.pred.tst <- predict(tree1, dat.test, type = "class")
table(dat.test$is_adopted, tree.pred.tst,
      dnn = c("Actual", "Predicted"))
```

```
##       Predicted
## Actual     0     1
##      0 10031   946
##      1  3297  7680
```

```
err.ptree.tst <- mean(dat.test$is_adopted != tree2.pred.tst)
err.ptree.tst # 0.1932677, 19.33%
```

```
## [1] 0.1932677
```

```
# Pruned Tree TESTING Accuracy 80.67% (same as un-pruned tree)

# BOTH Trees have the same accuracy.

############## RANDOM FOREST
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

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

```
## The following object is masked from 'package:psych':
##
##     outlier
```

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

```
library(caret)
```

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

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

```
## The following objects are masked from 'package:fabletools':
##
##     MAE, RMSE
```

```
library(future)
```

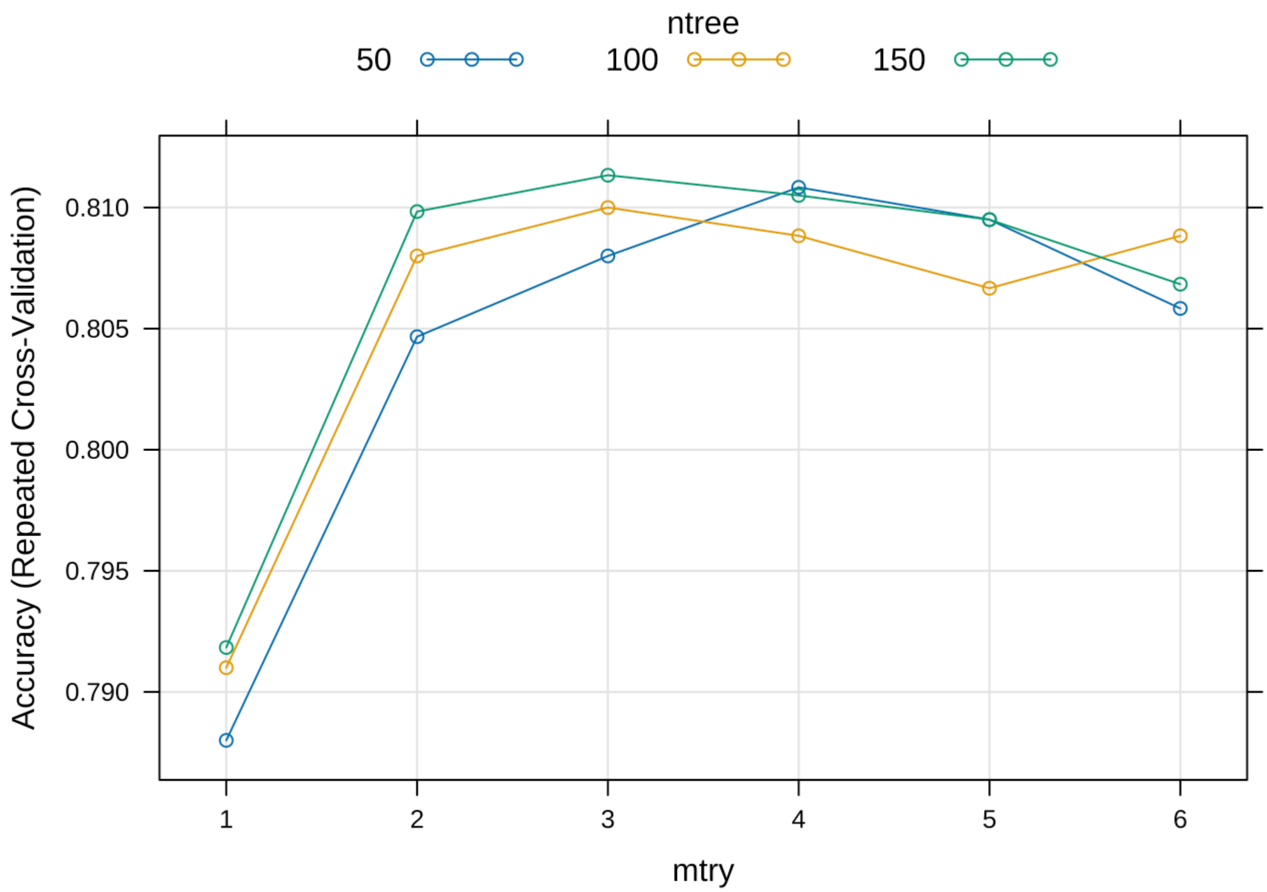
```
##
## Attaching package: 'future'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
set.seed(123)

metric = 'Accuracy'
# Create a custom RF to optimize mtry and ntree
customRF <- list(type = "Classification", library = "randomForest", loop = NULL)
customRF$parameters <- data.frame(parameter = c("mtry", "ntree"), class = rep("numeric",
2), label = c("mtry", "ntree"))
customRF$grid <- function(x, y, len = NULL, search = "grid") {}
customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {
  randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
}
customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
  predict(modelFit, newdata)
customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
  predict(modelFit, newdata, type = "prob")
customRF$sort <- function(x) x[order(x[,1]),]
customRF$levels <- function(x) x$classes

# Train model
control <- trainControl(method="repeatedcv", number=10, repeats=3)
tunegrid <- expand.grid(.mtry=c(1:6), .ntree=c(50,100,150))
custom <- train(is_adopted~., data=dat.train, method=customRF, metric=metric, tuneGrid=t
unegrid, trControl=control)
summary(custom)
```

```
##                  Length Class      Mode
## call                5   -none-     call
## type                1   -none-     character
## predicted        2000   factor     numeric
## err.rate          450   -none-     numeric
## confusion           6   -none-     numeric
## votes            4000   matrix     numeric
## oob.times        2000   -none-     numeric
## classes             2   -none-     character
## importance         15   -none-     numeric
## importanceSD        0   -none-     NULL
## localImportance     0   -none-     NULL
## proximity           0   -none-     NULL
## ntree               1   -none-     numeric
## mtry                1   -none-     numeric
## forest             14   -none-     list
## y                2000   factor     numeric
## test                0   -none-     NULL
## inbag               0   -none-     NULL
## xNames             15   -none-     character
## problemType         1   -none-     character
## tuneValue           2   data.frame list
## obsLevels           2   -none-     character
## param               0   -none-     list
```

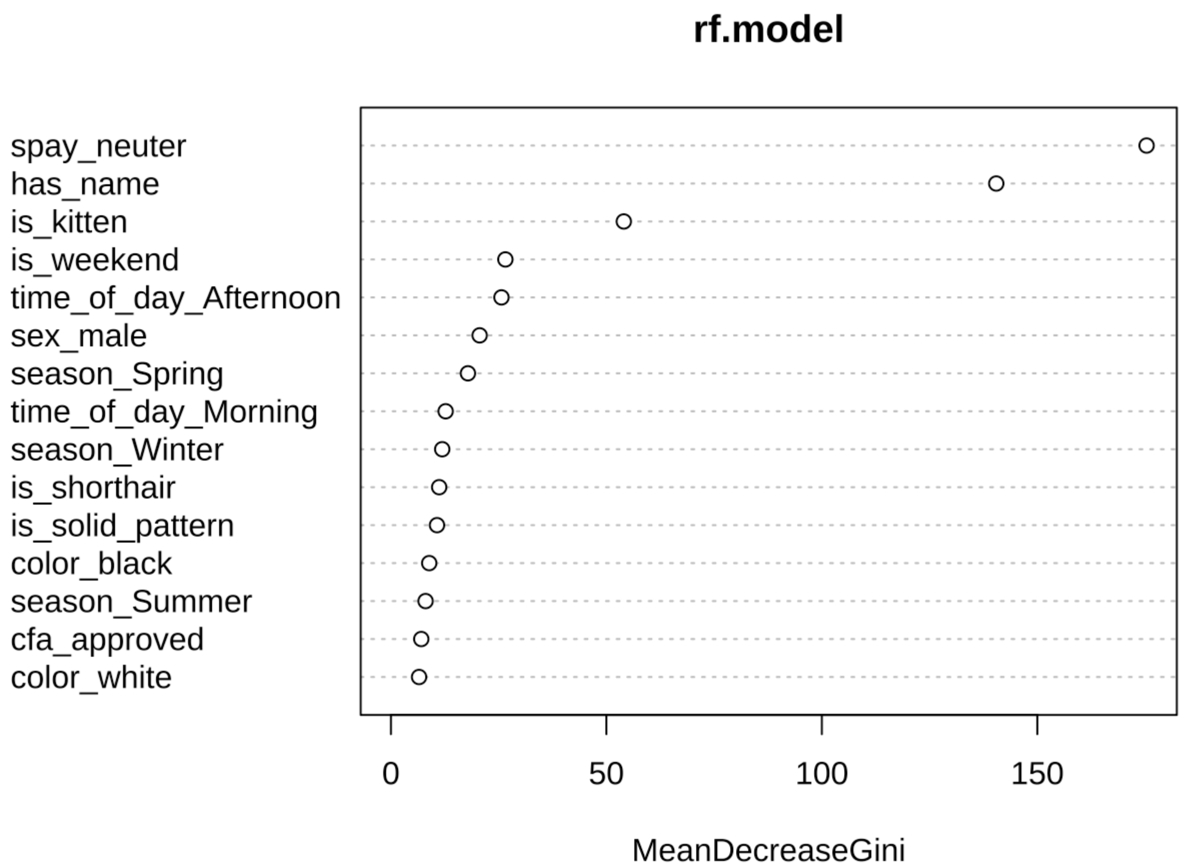
```
plot(custom) # best mtry = 3, ntree = 50
```




```
# RF model using best parameters
rf.model <- randomForest(is_adopted ~ ., data = dat.train, ntree = 50, mtry = 3, nodesiz
e = 5)
rf.model
```

```
##
## Call:
##  randomForest(formula = is_adopted ~ ., data = dat.train, ntree = 50,      mtry = 3,
nodesize = 5)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 19.4%
## Confusion matrix:
##     0   1 class.error
## 0 798 202       0.202
## 1 186 814       0.186
```

```
# Feature importance plot
varImpPlot(rf.model)
```




```
# Predict on testing data
rf.pred.tst <- predict(rf.model, newdata = dat.test)
table(dat.test$is_adopted, rf.pred.tst,
      dnn = c("Actual", "Predicted"))
```

```
##        Predicted
## Actual    0    1
##      0 8966 2011
##      1 1842 9135
```

```
err.rf.tst <- mean(dat.test$is_adopted != rf.pred.tst)
err.rf.tst #0.1755033, 17.55%
```

```
## [1] 0.1755033
```

```
# Random Forest TESTING Accuracy 82.45%

##### Support Vector Machines SVM
# Build model
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:fabletools':
##
##     interpolate
```

```
## The following objects are masked from 'package:dlookr':
##
##     kurtosis, skewness
```

```
svm.model <- svm(is_adopted ~ ., data = dat.train, kernel = 'linear', cost = 10)
# Prediction on training data
svm.pred.tr <- predict(svm.model, dat.train)
table(dat.train$is_adopted, svm.pred.tr,
      dnn = c("Actual", "Predicted"))
```

```
##       Predicted
## Actual   0   1
##      0 774 226
##      1 154 846
```

```
err.svm.tr <- mean(dat.train$is_adopted != svm.pred.tr)
err.svm.tr #0.19, 19%
```

```
## [1] 0.19
```

```
# SVM TRAINING Accuracy 81%

# Prediction on testing data
svm.pred.tst <- predict(svm.model, dat.test)
table(dat.test$is_adopted, svm.pred.tst,
      dnn = c("Actual", "Predicted"))
```

```
##       Predicted
## Actual    0    1
##      0 8588 2389
##      1 1728 9249
```

```
err.svm.tst <- mean(dat.test$is_adopted != svm.pred.tst)
err.svm.tst #0.1875285, 18.75%
```

```
## [1] 0.1875285
```

```r
# SVM TESTING Accuracy 81.25%

err.svm.tst # SVM TESTING Accuracy 81.25%
```

```
## [1] 0.1875285
```

```r
########
#### we have explored the accuracy of all of our models
#### the models are all similar in performance
#### we have selected our favorite models which we feel have a good
#### trade off for fit and interpretability:
#### Random Forest and Decision Tree

# with this in mind, lets try slightly changing our data to fit our model
# and use a CONTINOUS variable for AGE instead of binary


#### TREE using continous variable
dat <- read.csv("cat_data_cleaned_updated_continous_age.csv")
#remove animal_id
dat$animal_id <- NULL
# remove one variable for each categorical variable
dat$season_Fall <- NULL
dat$time_of_day_Closed <- NULL
dat$color_other <- NULL
set.seed(123)
#build the tree with all dataset and features
library(tree)
#set up dependent variable as factor
dat$is_adopted <- as.factor(dat$is_adopted)
tree.all <- tree(is_adopted~., dat)
summary(tree.all)
```

```
##
## Classification tree:
## tree(formula = is_adopted ~ ., data = dat)
## Variables actually used in tree construction:
## [1] "spay_neuter"         "has_name"            "outcome_age_.years."
## [4] "is_weekend"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7016 = 19790 / 28200
## Misclassification error rate: 0.1558 = 4395 / 28208
```

```r
#splitting training and testing dataset
table(dat$is_adopted)
```

```
##
##     0     1
## 16231 11977
```

```
dat.A <- dat[dat$is_adopted == 1,]
dat.notA <- dat[dat$is_adopted == 0,]
train.A <- sample(nrow(dat.A),1000)
train.notA <- sample(nrow(dat.notA),1000)
dat.train <- rbind(dat.A[train.A,],
                   dat.notA[train.notA,])
table(dat.train$is_adopted)
```

```
##
##    0    1
## 1000 1000
```

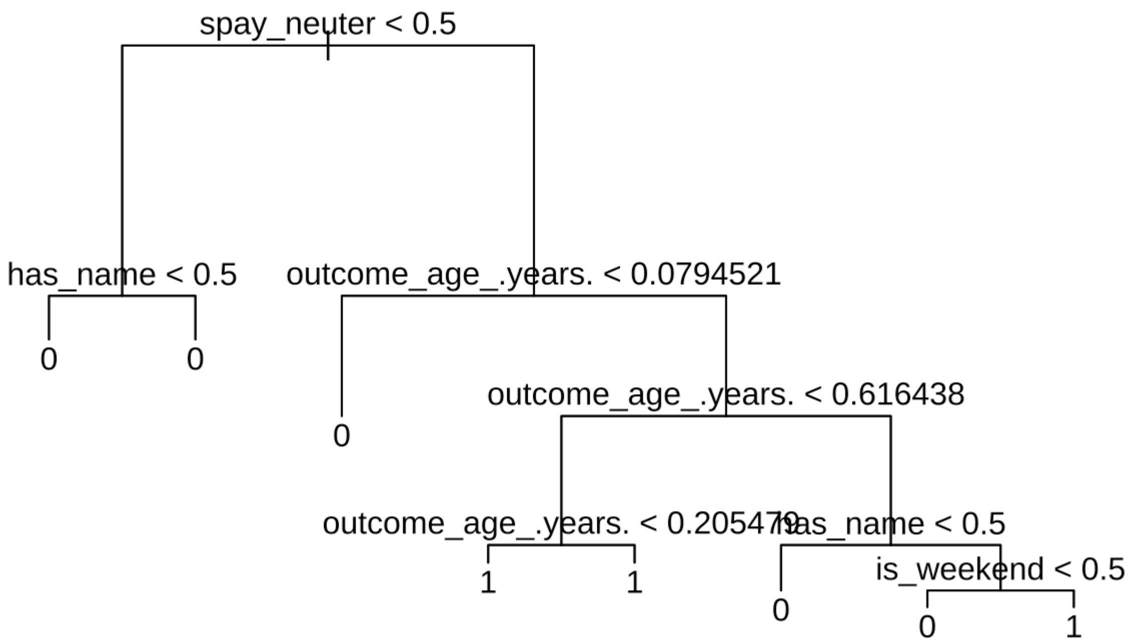
```
# Create a test data set similar to the training set
dat.notA.notsel <- dat.notA[-train.notA,]
dat.A.notsel <- dat.A[-train.A,]
test.notA <- sample(nrow(dat.notA.notsel),nrow(dat.A.notsel))
dat.test <- rbind(dat.A[-train.A,],
                  dat.notA.notsel[test.notA,])
table(dat.test$is_adopted)
```

```
##
##     0     1
## 10977 10977
```

```
#  Remove unneeded objects
rm(dat.A, dat.notA, dat.notA.notsel)
rm(test.notA, train.A, train.notA)

#build a tree on training dataset
tree.train.1 <-tree(is_adopted~., dat.train)
summary(tree.train.1)
```

```
##
## Classification tree:
## tree(formula = is_adopted ~ ., data = dat.train)
## Variables actually used in tree construction:
## [1] "spay_neuter"         "has_name"            "outcome_age_.years."
## [4] "is_weekend"
## Number of terminal nodes:  8
## Residual mean deviance:  0.6632 = 1321 / 1992
## Misclassification error rate: 0.1555 = 311 / 2000
```

```
plot(tree.train.1)
text(tree.train.1, pretty = 0)
```




```
sum.tree.train.1 <- summary(tree.train.1)
sum.tree.train.1$misclass
```

```
## [1]  311 2000
```

```
err.tree.train.1 <- sum.tree.train.1$misclass[1]/
  sum.tree.train.1$misclass[2]
err.tree.train.1 #0.1555
```

```
## [1] 0.1555
```

```
#tree splits at the spay_neuter first, if the cat not neutered and does not have a name,
#the cat is 100% would not be adopted according to this tree model.

#at the age node, it seems like if a cat is yonger than half years, then there is a big
#chance of this cat being adopted.

#model's performance on training dataset
yhat.tree.train.1 <- predict(tree.train.1, dat.train)
head(yhat.tree.train.1)
```

```
##                0         1
## 6563  0.03571429 0.9642857
## 6717  0.52397260 0.4760274
## 24738 0.23888889 0.7611111
## 20965 0.72891566 0.2710843
## 7903  0.23888889 0.7611111
## 5019  0.23888889 0.7611111
```

```
yhat.tree.train.1.cl <-
  ifelse(yhat.tree.train.1[,2] > 0.5, 1, 0)
tab.tree.train.1 <- table(dat.train$is_adopted, yhat.tree.train.1.cl,
                          dnn = c("Actual", "Predicted"))
tab.tree.train.1
```

```
##       Predicted
## Actual   0   1
##      0 887 113
##      1 198 802
```

```
mean(dat.train$is_adopted != yhat.tree.train.1.cl) #0.1555
```

```
## [1] 0.1555
```

```
#model's performance on testing dataset
yhat.tree.test.1 <- predict(tree.train.1, dat.test)
yhat.tree.test.1.cl <-
  ifelse(yhat.tree.test.1[,2] > 0.5, 1, 0)
tab.tree.test.1 <- table(dat.test$is_adopted, yhat.tree.test.1.cl,
                         dnn = c("Actual", "Predicted"))
tab.tree.test.1
```

```
##       Predicted
## Actual    0    1
##      0 9713 1264
##      1 2275 8702
```

```
mean(dat.test$is_adopted != yhat.tree.test.1.cl) #0.1612007
```

```
## [1] 0.1612007
```

```
# TREE with continous variable Testing Accuracy: 83.88%


### RANDOM FOREST using continous variable
#Random Forest Models
#number of features selected in each tree
#sqrt of 15 , which is around 4
rf.train.1 <- randomForest(is_adopted ~ ., data = dat.train,
                           mtry = 4, ntree = 100,
                           importance = TRUE)
rf.train.1
```

```
##
## Call:
##  randomForest(formula = is_adopted ~ ., data = dat.train, mtry = 4,      ntree = 100,
importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 14.85%
## Confusion matrix:
##     0   1 class.error
## 0 826 174       0.174
## 1 123 877       0.123
```

```
yhat.rf.1 <- predict(rf.train.1, dat.test)
tab.rf.1 <- table(dat.test$is_adopted, yhat.rf.1)
tab.rf.1
```

```
##    yhat.rf.1
##        0    1
##   0 9024 1953
##   1 1339 9638
```

```
err.rf.1 <- mean(dat.test$is_adopted != yhat.rf.1)
err.rf.1 #0.1499499
```

```
## [1] 0.1499499
```

```
rf.train.2 <- randomForest(is_adopted ~ ., data = dat.train,
                           mtry = 4, ntree = 1000,
                           importance = TRUE)
rf.train.2
```

```
##
## Call:
##  randomForest(formula = is_adopted ~ ., data = dat.train, mtry = 4,      ntree = 100
0, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 14.85%
## Confusion matrix:
##     0   1 class.error
## 0 825 175       0.175
## 1 122 878       0.122
```

```
yhat.rf.2 <- predict(rf.train.2, dat.test)
tab.rf.2 <- table(dat.test$is_adopted, yhat.rf.2)
tab.rf.2
```

```
##    yhat.rf.2
##        0    1
##   0 9007 1970
##   1 1317 9660
```

```
err.rf.2 <- mean(dat.test$is_adopted != yhat.rf.2)
err.rf.2 #0.1497221
```

```
## [1] 0.1497221
```

```
# RANDOM FOREST with continous variable Testing Accuracy: 85%

sort(importance(rf.train.2)[,3], decreasing = TRUE)[1:5]
```

```
##           spay_neuter   outcome_age_.years.              has_name
##             133.39166             106.43753              74.72879
## time_of_day_Afternoon            is_weekend
##              38.62248              32.46150
```

```
#spay_neuter, outcome_age_.years, has_name, is_weekend, time_of_day_afternoon

#Gini importance
sort(importance(rf.train.2)[,4], decreasing = TRUE)[1:5]
```

```
##   outcome_age_.years.           spay_neuter              has_name
##             278.41532             183.21745             113.79087
## time_of_day_Afternoon            is_weekend
##              33.22559              29.85856
```

```
#outcome_age_.years, spay_neuter, has_name,time_of_day_afternoon,is_weekend



######
## As we can see adding a continous variable improves the accuracy of our models
## Whether these models are "better" can be up to choice, to determine the best
## trade-off for fit and interpretability.
## Overall, we prefer the Random Forest model with a continous age variable.
```