Stats For Cats: Classifying Adoption Outcome for Cats at the Austin Animal Shelter

Christian Dueñas

5/17/2022

Contents

Motivation **2**

Cleaning the Data **2**

Refactoring and Removing Redundant Features . . . 2

Defining "Adoption" and Working with Censored Data . . . 4

Exploratory Analysis **5**

Visualizing Cat Characteristics . . . 5

Sex . . . 7

Spaying/Neutering . . . 8

Cat or Kitten . . . 9

Color . . . 10

Coat Pattern . . . 11

Age . . . 13

Wordcloud of Cat Names for Adopted Cats . . . 16

External Time Factors . . . 17

Day of the Week . . . 17

Month of the Year . . . 19

Creating Classifiers **20**

Features for Consideration . . . 20

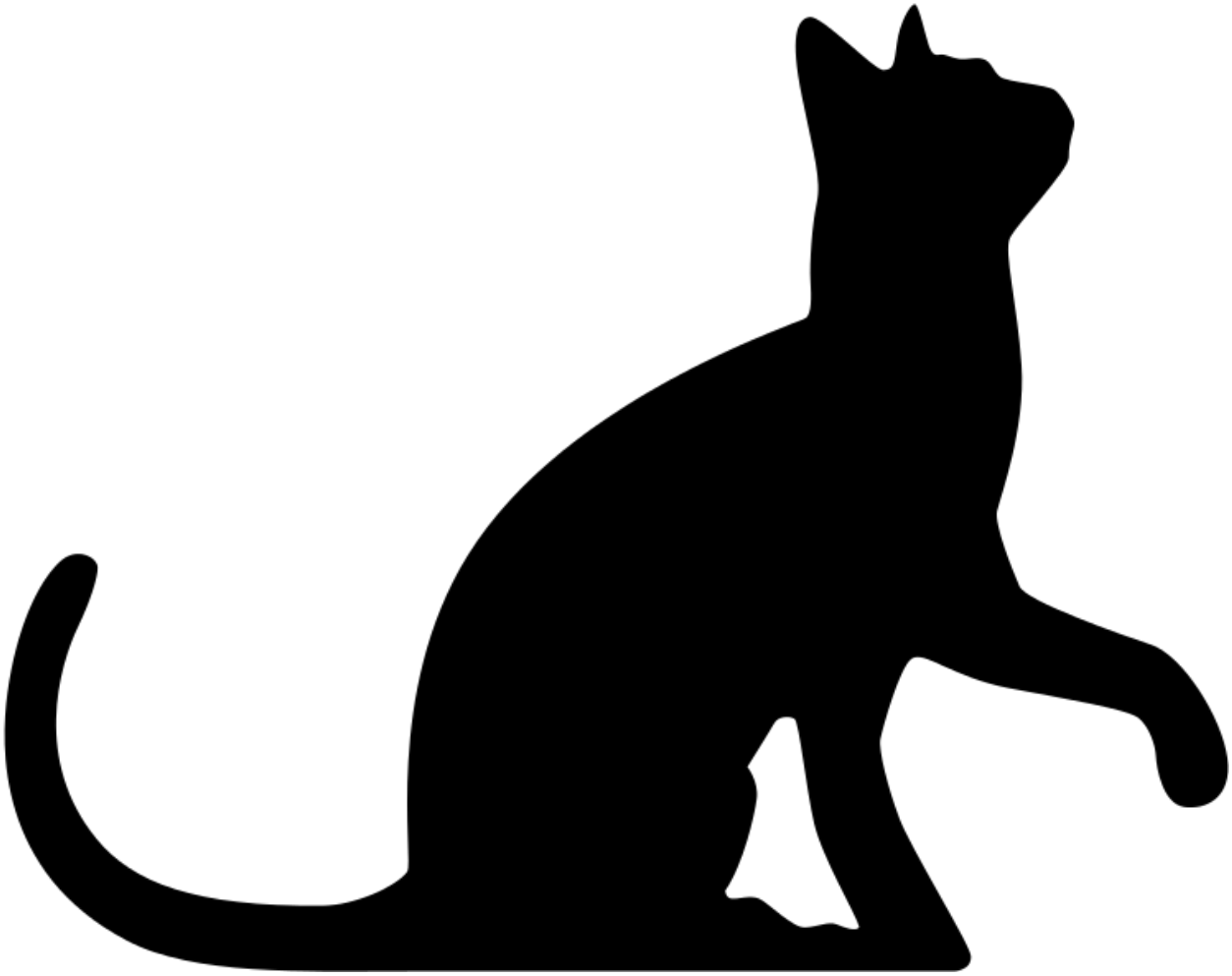
Using Cross-Validation To Prevent Overfitting . . . 21

Logistic Regression . . . 23

Support Vector Machine . . . 26

Decision Tree . . . 27

K-Nearest Neighbors . . . 28



Motivation

Overcrowding of animal shelters is a major problem across the nation. According to ASPCA 6.5 million animals enter U.S shelters annually, about 3.2 million of the animals in shelters are cats. Approximately 920,000 shelter animals are euthanized, even though many of these animals are healthy enough to be adopted.

In the hopes of assisting animal shelters towards finding shelter cats a happy home, we will delve into a data set of over 29000 shelter cats provided by the Austin Animal Center in Texas. The aim is to create a classifier that can predict if a cat will be adopted based on its characteristics. These classifiers may also offer insight into what characteristics play a significant role in adoption of a cat.

The data set can be found at: https://www.kaggle.com/competitions/shelter-animal-outcomes/data

Cleaning the Data

Refactoring and Removing Redundant Features

The `tidyverse` package contains multiple libraries to handle data wrangling. Most of the functions used will come from the `dplyr` library inside tidyverse.

```
# Data wrangling can be done using Tidyverse functions
library(tidyverse)
```

We'll then load the original data set from Kaggle, which is in a csv file.

```
# read in the data from the local machine
cats_orig <- read.csv("cats.csv")
# Look at all the available features in the data set
names(cats_orig)
```

```
##  [1] "age_upon_outcome"     "animal_id"            "animal_type"
##  [4] "breed"                "color"                "date_of_birth"
##  [7] "datetime"             "monthyear"            "name"
## [10] "outcome_subtype"      "outcome_type"         "sex_upon_outcome"
## [13] "count"                "sex"                  "Spay.Neuter"
## [16] "Periods"              "Period.Range"         "outcome_age_.days."
## [19] "outcome_age_.years."  "Cat.Kitten..outcome." "sex_age_outcome"
## [22] "age_group"            "dob_year"             "dob_month"
## [25] "dob_monthyear"        "outcome_month"        "outcome_year"
## [28] "outcome_weekday"      "outcome_hour"         "breed1"
## [31] "breed2"               "cfa_breed"            "domestic_breed"
## [34] "coat_pattern"         "color1"               "color2"
## [37] "coat"
```

With the data loaded, we can begin the cleanup process. Many of the features are encoded as strings and may contain unnecessary characters. We want to re-encode them as `factors` so that they will be compatible with the model building functions. Some features may need to be modified on a case-by-case basis, but we can work towards having a "clean" version of the data set that will for the most part have all of the covariates in a model friendly form.

```
# Refactor variables of interest and store the result in a new variable
# Make feature representation into something more meaningful
cats <- cats_orig %>% rename(age.at.outcome = outcome_age_.years.) %>%
  mutate(sex = as.factor(sex),
         Spay.Neuter = ifelse(Spay.Neuter == "Yes", TRUE, FALSE),
         is.kitten = ifelse(Cat.Kitten..outcome. == "Kitten", TRUE, FALSE),
         breed = as.factor(breed),
         color1 = str_remove(color1, "[ ]"),
         outcome_weekday = as.factor(outcome_weekday),
         outcome_month = as.factor(outcome_month),
         coat_pattern = ifelse(coat_pattern == "", "solid", coat_pattern)
         )
```

Some factors have a lot of unique levels. For example, if we look at the primary color of a cat, we can see that there are 28 unique types for types for the primary color alone.

```
table(cats$color1)
```

```
##
##       apricot         black    blacktiger          blue     bluecream
##             1          7111             1          3619            43
## BreedSpecific         brown    brownmerle    browntiger          buff
```

```
##          4058          6897             2             6            11
##     chocolate         cream          fawn         flame          gray
##            86           804             1           205           382
##         lilac          lynx        orange   orangetiger          pink
##            83           486          3406             1             2
##         sable          seal        silver    silverlynx   silverlynx
##             2           371           100            12             2
##           tan         white        yellow
##             5          1716             8
```

Trying to use all 28 will lead to problems as there are only a few cats under some of the levels. This will especially be a problem when we cross-validate the data, as not all types may show up in training data fit and the model will break during prediction.

We will instead consider only taking some of the more popular colors, and pooling the low frequency or non-descriptive colors into an "other" category. We will do the same thing for the `coat_pattern` variable.

```
# Refactor levels in features with only a few instances of a level
cats$color1 <- ifelse(cats$color1 %in% c("black", "brown", "blue", "orange", "white") , cats$color1, "o
cats$coat_pattern <- ifelse(cats$coat_pattern %in% c("agouti", "brindle", "smoke", "tricolor") , "other
```

Finally we can drop some of the unneeded or redundant features so that the data frame is smaller going forward. This can be done using the `select` function in `dplyr`.

```
# Remove unnecessary and redundant features for classification
cats_selected <- cats %>% select(-age_upon_outcome:-animal_type,
                         -date_of_birth:-name,-sex_upon_outcome,
                         -count, -Periods:-outcome_age_.days.,
                 -Cat.Kitten..outcome.:-sex_age_outcome, -color, -breed1,-breed2)
```

Defining "Adoption" and Working with Censored Data

Looking at the data set, we can see that there are actually more than two outcomes a cat could have experienced. We can show all possible outcomes by using the `unique()` function on the outcome variable

```
unique(cats$outcome_type) # See all possible outcomes
```

```
## [1] "Transfer"        "Adoption"        "Return to Owner" "Died"
## [5] "Euthanasia"      "Missing"         "Disposal"        "Rto-Adopt"
## [9] ""
```

To make the classification problem simpler, we want to simplify the data into only two outcomes: Either the cat was adopted or it was not. In this case, what classifies as "adopted or not adopted" may be a bit subjective.

Defining adopted cats is very simple as there is an "adoption" level. There may be some nuance such as animals that are adopted and then returned, but that data is not presented and would still technically mean that the cat was adopted.

```
# Gather the adopted cats
cats_adopted <- cats_selected %>% filter(outcome_type == "Adoption") %>%
  mutate(adopted = TRUE)
# Ensure that there are no cats with missing information
nrow(cats_adopted) == sum(complete.cases(cats_adopted) )
```

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

"Not Adopted" is the more nuanced case. For example, cats labeled "return to owner" were not technically adopted as they were likely cats who already belonged to somebody and likely got lost.

"Transfer," "Euthanasia," "Died" and "Missing" present censored data, as whether the cat was going to be adopted or not is left unknown. An argument could be made that under certain conditions these categories could fall under the not adopted classification. For example, it may be fair to assume that a cat who died in the shelter or transferred after ten years was present long enough to be considered not adopted. This is very different from a cat who was only alive or in the shelter for a few days before reaching an outcome. This is especially a problem with kittens who die in the shelter due to health problems from birth before they can really be put up for adoption.

To handle the censored data, we will use a minimum age to consider if a cat had a reasonable time frame of being adopted. According to the cat website "bechewy", cats at around 8 weeks are considered ready for adoption. We will say that around 16 weeks (or .3 years) will be out cutoff, as that gives kittens about 2 months to be potentially adopted before reaching any other outcome. This is not a perfect solution but is about as good as we can do given the constraints of the data.

```
# Look at cats that weren't adopted and were adoptable for at least 2 months
cats_notadopted <- cats_selected %>%
  filter(outcome_type %in% c("Transfer", "Euthanasia", "Died","Missing") &
           age.at.outcome > .3) %>% mutate(adopted = FALSE)

sum(complete.cases(cats_notadopted)) == nrow(cats_notadopted)
```

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

With the `adoption` outcome recreated, we can combine our separated data into one "clean" data frame that can be used as the baseline going forward.

```
# Baseline clean data set that will work with most models
cats_clean <- rbind(cats_adopted, cats_notadopted)  -> cats_clean
```

Exploratory Analysis

Before we stick the data into a classifier, we can perform some visualization to get a feel for what the data looks like. These visualizations will be done using `ggplot2` and will consist mostly of frequency and proportion visualizations, as the features are mostly categorical.

Visualizing Cat Characteristics

We'll begin by visualizing traits inherit to the cat. We want to see how each of the characteristics interact with adoption. We can do this using two-factor bar plots to assess the proportions of adopted vs non-adopted cats with certain qualities.

Since we will be doing this for multiple factors, it will be useful to create a function that will make the process a bit easier by automating some of the repeated steps. The following helper functions will automate certain tasks going forward. We can then build ontop our plots using the `ggplot` capabilities to add unique attributes such as titles.

```r
# Load in ggplot2 as well as rlang to create the functions
library(ggplot2)
library(scales)
library(rlang)
options(dplyr.summarise.inform = FALSE)

# Function that finds the proportions grouped by two factors
# Assumes first factor is only two levels (Binary yes or no)
get_props_counts <- function(df, factor1, factor2) {
  # Get the total count per outcome of the first factor
  outcomes <- unique(df[[factor1]])
  df_yes <- df %>% filter(!!sym(factor1) == outcomes[1] )
  df_no <- df %>% filter(!!sym(factor1) == outcomes[2] )

  df_yes %>% group_by(!!sym(factor1),!!sym(factor2)) %>%
          summarize(count = n(), prop = n()/ nrow(df_yes) ) -> yes_prop

  df_no %>% group_by(!!sym(factor1),!!sym(factor2)) %>%
        summarize(count = n(), prop = n()/ nrow(df_no) ) -> no_prop

  return( rbind(yes_prop,no_prop))
}


# Function that formats a floating point number into a readable precent
# Used to overlay proportions on top of the bar plots
format_prop <- function(prop) {
  paste0(as.character(round(prop, digits = 3) * 100), '%')
}


# Wrapper for ggplot that will take in data and make a dodged bar plot
# Elements that will be present for all bar plots in this project are
# included here
create_dodge_bars <- function(df, x, fill_var) {
  g1 <- ggplot(data=df, mapping = aes(x = !!sym(x), y = prop)) +
    geom_bar(aes(fill = !!sym(fill_var)),
             stat = "identity", position = "dodge") +
    geom_text(aes(label = format_prop(prop)), size=2.9,
              position = position_dodge2(width = .9), vjust=-.6) +
    scale_y_continuous(labels = scales::percent, limits=c(0,1)) +
    xlab("Was the Cat Adopted?") +ylab("Percentage") + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5)) +
    labs(subtitle = paste("Total Count:", sum(df$count)))
  return(g1)
}
```

We'll also format that data in a ggplot friendly way. This will make generating the axis labels very simple going forward. We will create a new data frame as our `cats_clean` will still be used for model building.

```r
# Create a new data frame
cats_desc <- cats_clean
```

```
# make boolean variables into factors
# order the levels if necessary
cats_desc$adopted <- ifelse(cats_clean$adopted == 1, "Yes", "No")
cats_desc$is.kitten <- ifelse(cats_clean$is.kitten == 1, "Yes", "No")
cats_desc$Spay.Neuter <- ifelse(cats_clean$Spay.Neuter == 1, "Yes", "No")
cats_desc$age_group <- ifelse(
  cats_desc$age_group == "(-0.022, 2.2]", "(0, 2.2]", cats_desc$age_group)
cats_desc$age_group <- factor(cats_desc$age_group, levels =
                            c("(0, 2.2]","(2.2, 4.4]","(4.4, 6.6]",
                              "(6.6, 8.8]","(8.8, 11.0]", "(11.0, 13.2]",
                              "(13.2, 15.4]","(15.4, 17.6]","(17.6, 19.8]",
                              "(19.8, 22.0]"), ordered = T)
```

Using our newly created `get_props_counts` function, we can group our data by the factor of interest and whether it was adopted or not. We can then get the proportions and counts that can be used for the plots.

```
# Gather proportions/counts for the characteristics of interest
props_by_sex <- get_props_counts(cats_desc, "adopted", "sex")
props_by_kitten <- get_props_counts(cats_desc, "adopted", "is.kitten")
props_by_fix <- get_props_counts(cats_desc, "adopted", "Spay.Neuter")
props_by_color <- get_props_counts(cats_desc, "adopted", "color1")
props_by_age_group <- get_props_counts(cats_desc, "adopted", "age_group")
props_by_coat <- get_props_counts(cats_desc, "adopted", "coat_pattern")
```

Sex

The first characteristic that will be visualized is the effect of sex on adopted cats. We can generate the bar plot using our wrapper function and add some unique properties for more descriptive looks.

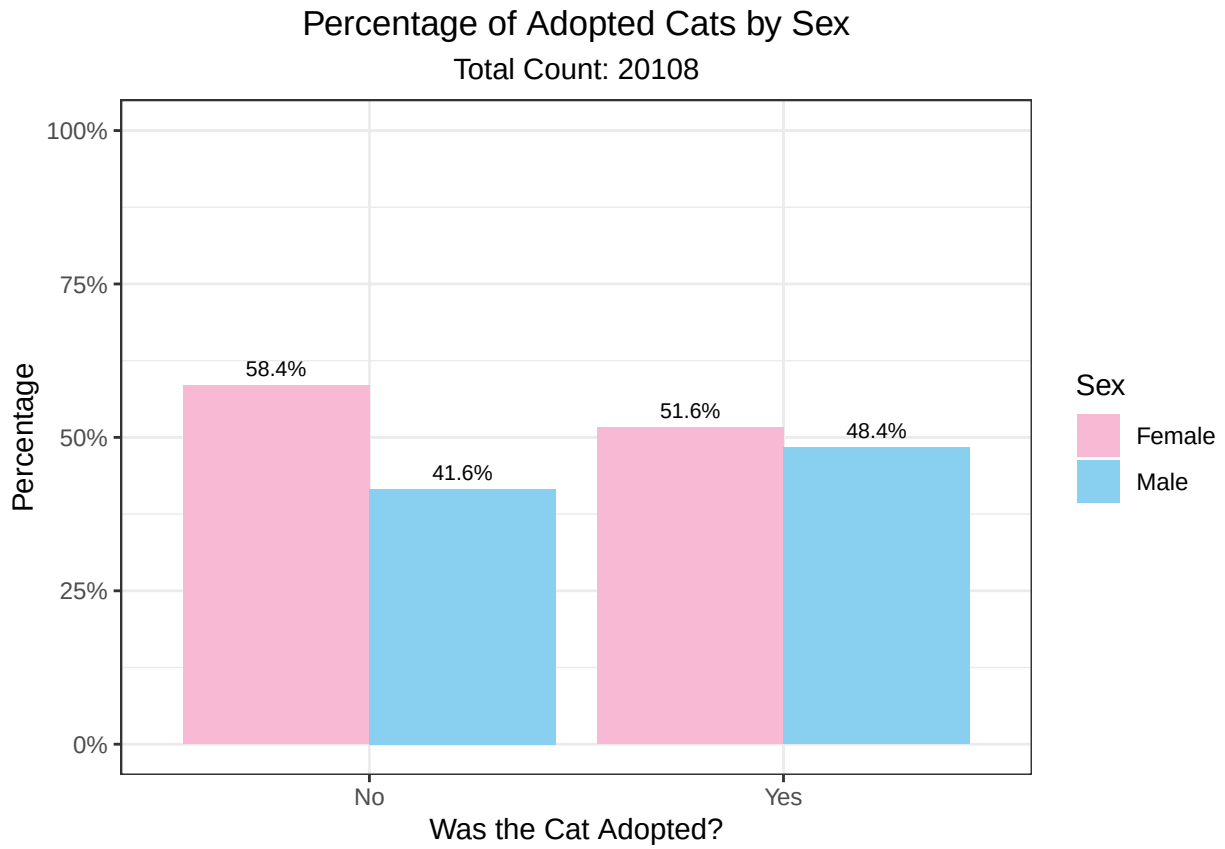
We can print the frequencies and bar plot below

```
library(patchwork) # library that allows us to add to stored plots

# print frequencies
props_by_sex
```

```
## # A tibble: 4 x 4
## # Groups:   adopted [2]
##   adopted sex    count  prop
##   <chr>   <fct>  <int> <dbl>
## 1 Yes     Female  6571 0.516
## 2 Yes     Male    6161 0.484
## 3 No      Female  4311 0.584
## 4 No      Male    3065 0.416
```

```
# Plot the proportions of Adopted Cats by Sex
create_dodge_bars(props_by_sex, "adopted", "sex") +
  labs(fill="Sex") +ggtitle("Percentage of Adopted Cats by Sex") +
  scale_fill_manual(values=c("#f8b9d4", "#89CFF0"))
```



Looking at the plots, we see that the proportions between males and females are relatively close when considering adopted cats; however female cats make up a bit larger proportion of the non-adopted cats compared to males. This small distinguishment may or may not be useful during the classification process.

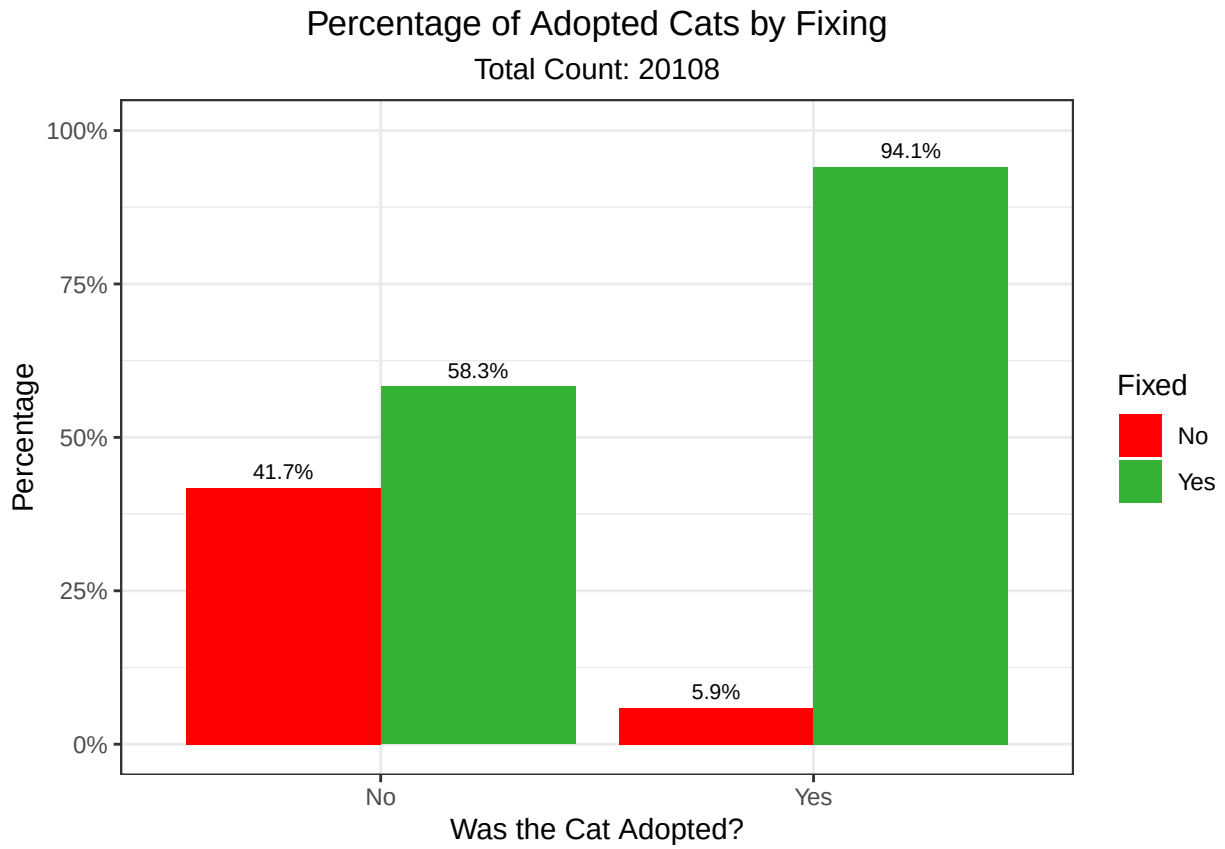
Spaying/Neutering

One of the features included in the data was whether or not the cat was fixed before its outcome.

```
props_by_fix
```

```
## # A tibble: 4 x 4
## # Groups:   adopted [2]
##   adopted Spay.Neuter count   prop
##   <chr>   <chr>       <int>  <dbl>
## 1 Yes     No            755 0.0593
## 2 Yes     Yes         11977 0.941
## 3 No      No           3079 0.417
## 4 No      Yes          4297 0.583
```

```
create_dodge_bars(props_by_fix, "adopted", "Spay.Neuter") +
  labs(fill="Fixed") +ggtitle("Percentage of Adopted Cats by Fixing") +
  scale_fill_manual(values=c("red", "#34b233"))
```



Looking at the plots, we see that nearly all of the adopted cats were fixed. Although it may be seen as potential significant feature, we should be careful as such drastic proportions may represent something happening behind the scenes.

Looking at the Austin Animal Shelter, it is stated that animals are required to be fixed before adoption. This raises a few questions. For example, why were some of the adopted cats not fixed? And is this variable going to cause an overfit? This is especially a concern if the model is used on cats at a different shelter with different guidelines.

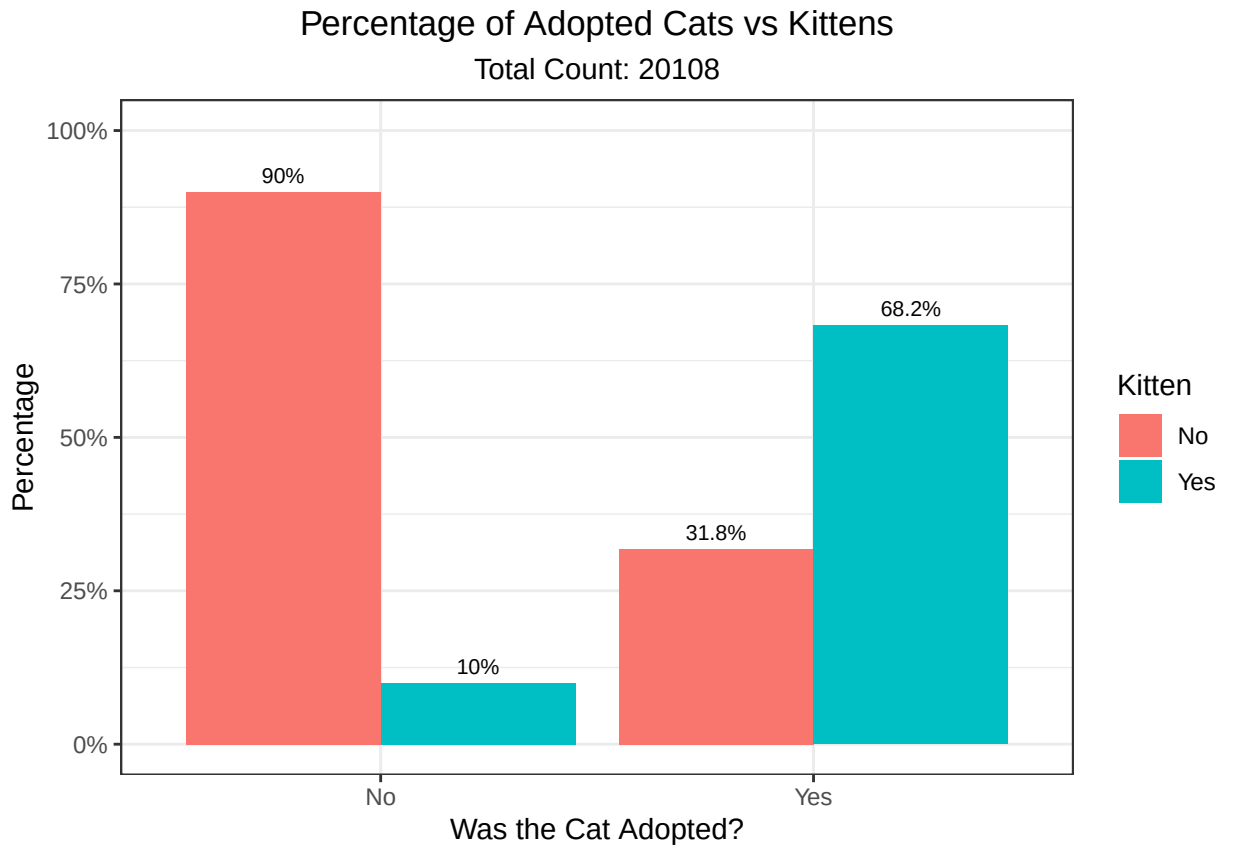
Cat or Kitten

According to the data set, a cat under one year old is defined as kitten. Is this distinguishment notable enough to include in our model?

```
# print frequencies
props_by_kitten
```

```
## # A tibble: 4 x 4
## # Groups:   adopted [2]
##   adopted is.kitten count  prop
##   <chr>   <chr>     <int> <dbl>
## 1 Yes     No         4048 0.318
## 2 Yes     Yes        8684 0.682
## 3 No      No         6638 0.900
## 4 No      Yes         738 0.100
```

```
# Plot the proportions of Adopted Cats by whether it is a kitten or not
create_dodge_bars(props_by_kitten, "adopted", "is.kitten") +
  labs(fill="Kitten") +ggtitle("Percentage of Adopted Cats vs Kittens")
```



Looking at the proportions, we see that kittens make up a small fraction of the cats who were not adopted. This plot seems to suggest that the cat being classified as a kitten will be a good predictor that the cat will be adopted.

Color

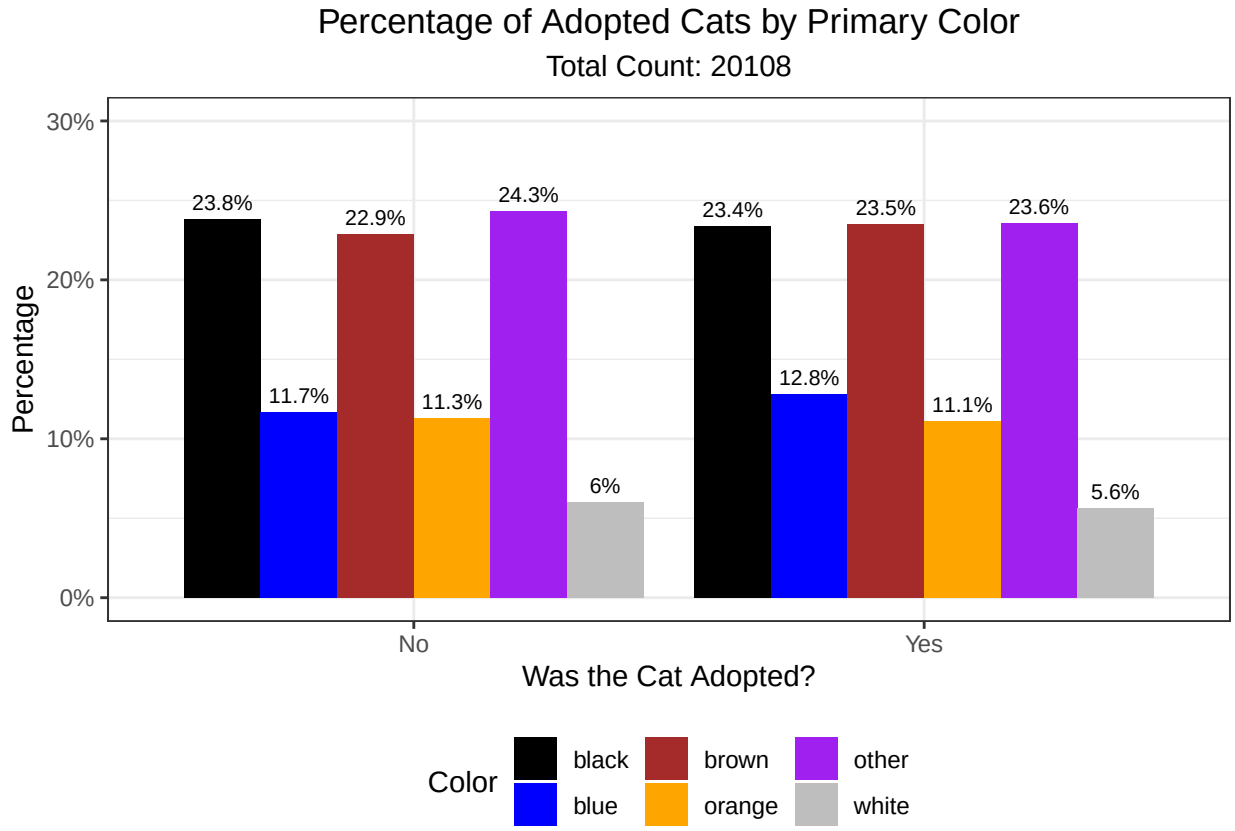
Next we'll be looking at the primary color of the cat. There are 6 levels that we will consider

```
# Print frequencies
props_by_color
```

```
## # A tibble: 12 x 4
## # Groups:   adopted [2]
##    adopted color1 count   prop
##    <chr>   <chr>  <int>  <dbl>
##  1 Yes     black   2976 0.234
##  2 Yes     blue    1630 0.128
##  3 Yes     brown   2990 0.235
##  4 Yes     orange  1416 0.111
##  5 Yes     other   3003 0.236
```

```
##  6 Yes     white    717 0.0563
##  7 No      black   1755 0.238
##  8 No      blue     862 0.117
##  9 No      brown   1687 0.229
## 10 No      orange   835 0.113
## 11 No      other   1792 0.243
## 12 No      white    445 0.0603
```

```
# Create bar charts based on the color
create_dodge_bars(props_by_color, "adopted", "color1") +
  scale_fill_manual(values=c("black", "blue", "brown",
                             "orange", "purple", "gray")) +
  scale_y_continuous(labels = scales::percent, limits=c(0,.3)) +
  labs(fill="Color") + ggtitle("Percentage of Adopted Cats by Primary Color") +
  theme(legend.position = "bottom")
```



The frequencies for adopted cats looks very similar to the frequencies of non-adopted cats. The plot suggests that classifying whether a cat was adopted using its color may prove a bit difficult.

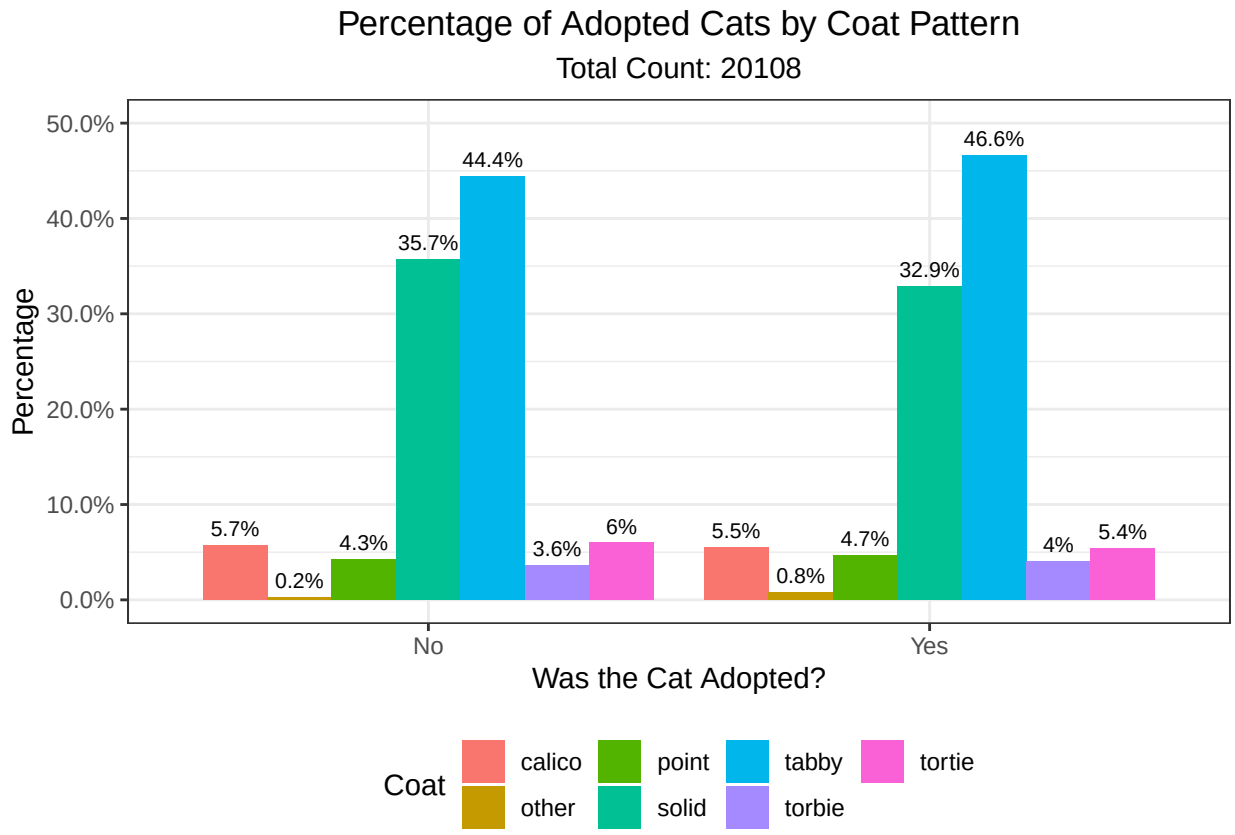
Coat Pattern

Coat pattern is another visual characteristic we can look at.

```
props_by_coat
```

```
## # A tibble: 14 x 4
## # Groups:   adopted [2]
##    adopted coat_pattern count    prop
##    <chr>   <chr>        <int>   <dbl>
##  1 Yes     calico         706 0.0555
##  2 Yes     other          103 0.00809
##  3 Yes     point          597 0.0469
##  4 Yes     solid         4190 0.329
##  5 Yes     tabby         5933 0.466
##  6 Yes     torbie         512 0.0402
##  7 Yes     tortie         691 0.0543
##  8 No      calico         423 0.0573
##  9 No      other           18 0.00244
## 10 No      point          314 0.0426
## 11 No      solid         2635 0.357
## 12 No      tabby         3276 0.444
## 13 No      torbie         268 0.0363
## 14 No      tortie         442 0.0599
```

```
create_dodge_bars(props_by_coat, "adopted", "coat_pattern") +
  scale_y_continuous(labels = scales::percent, limits=c(0,.5)) +
  labs(fill="Coat")+
  theme(legend.position = "bottom") +
  ggtitle("Percentage of Adopted Cats by Coat Pattern")
```



Coat pattern exhibits similar trends that the color variable did. Looking forward, it will be interesting to

see if visual characteristics of a cat provide any valuable insight as to whether the cat will be adopted or not.

Age

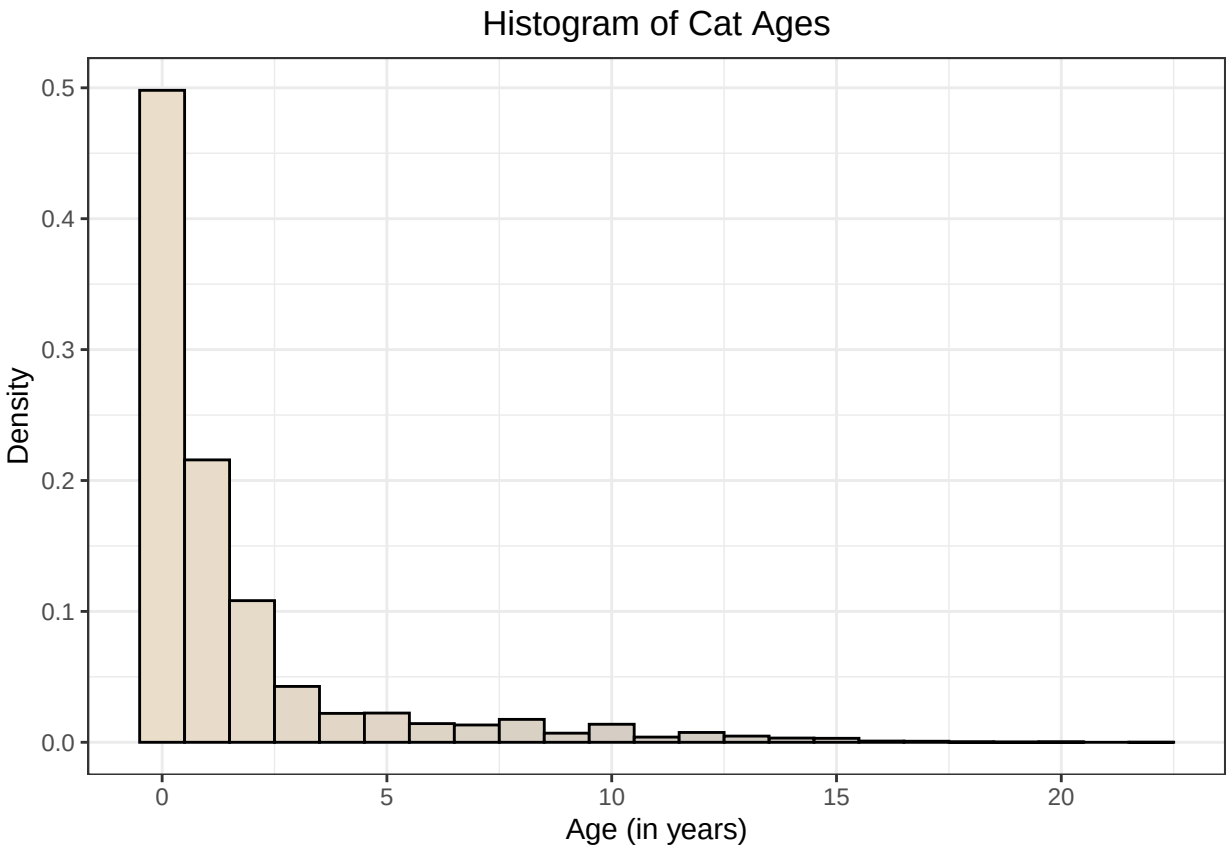
Next we will look at the age of the cat. First, we can generate a histogram of the overall dataset to get a feel for the distribtion of cat ages across the dataset.

```
# Create a histogram for the cat ages
# Use a pre-build to get the number of bins for a color gradient
age_hist <- ggplot_build(
    ggplot(data = cats_desc, mapping = aes(x = age.at.outcome)) +
    geom_histogram(aes(y=..density.., ),binwidth=1) +
    theme_bw() )
# Get the number of bins
num_bins <- dim(age_hist$data[[1]])[1]

# Print summary statistics of cat ages
summary(cats_desc$age.at.outcome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1644  0.5753  1.7106  2.0000 22.0000
```

```
# generate the plot with the color gradient
ggplot(data = cats_desc, mapping = aes(x = age.at.outcome)) +
    geom_histogram(aes(y=..density.., ),binwidth=1, color="black",
        fill=colorRampPalette(c("#EADDCA", "gray"), bias=2)(num_bins)) +
    theme_bw() + xlab("Age (in years)") + ylab("Density") +
    ggtitle("Histogram of Cat Ages") +
    theme(plot.title = element_text(hjust = 0.5))
```



Looking at the histogram, we can see that a majority of the cats are less than a year old, with the median value being around half a year old. There are also some cats who lived to be incredibly old, with the max being at 22. We also see 0 as the minimum age. This seems to imply that a kitten found a home immediately after it was born, perhaps as a package deal with its previously-pregnant mother as kittens typically need to be with their mothers during the first 8 weeks.

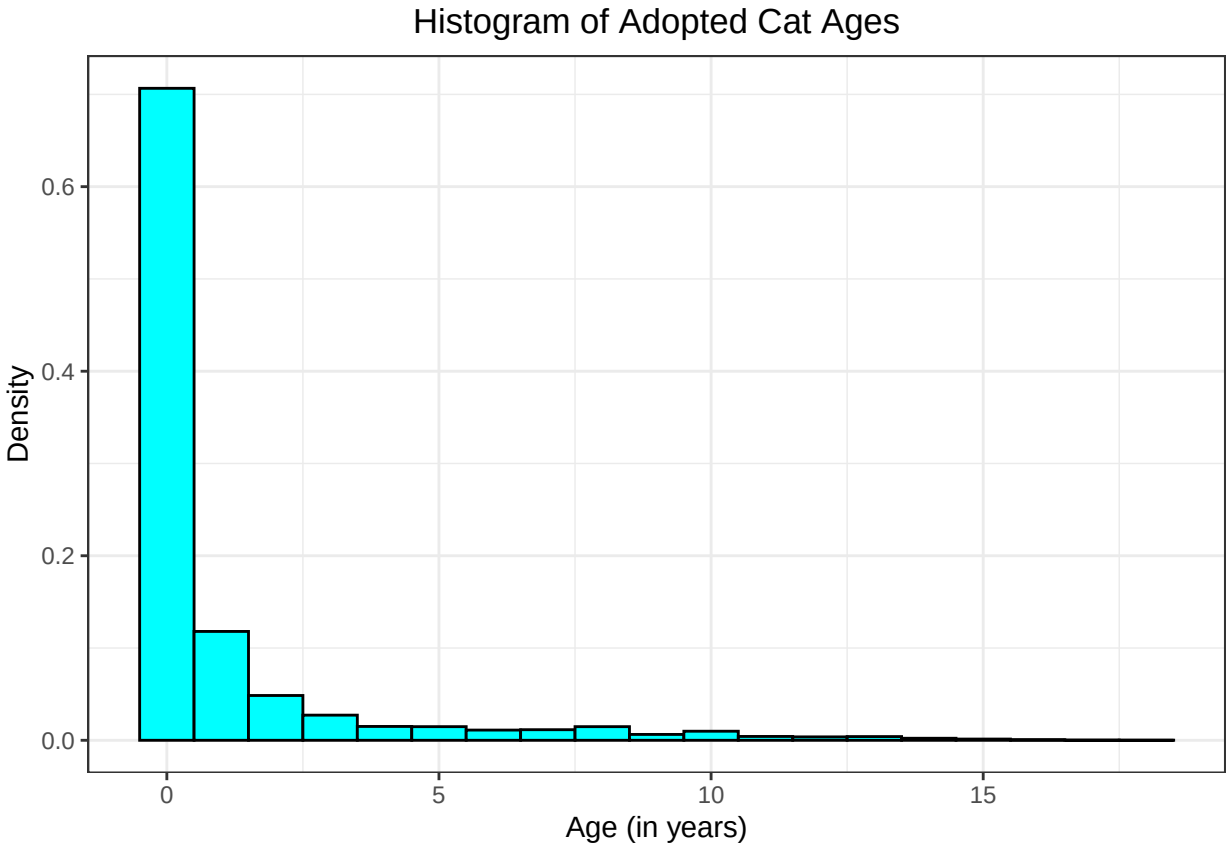
We can create more histograms, this time looking at the distribution of adopted cats and non adopted cats

```
# Subset to cats who got adopted
cats_desc %>% filter(adopted =="Yes") -> cats_desc_adopt

# Print summary stats for the age of adopted cats
summary(cats_desc_adopt$age.at.outcome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1644  0.2466  1.1991  1.0000 18.0000
```

```
ggplot(data = cats_desc_adopt, mapping = aes(x = age.at.outcome)) +
    geom_histogram(aes(y=..density.., ),binwidth=1, color="black",
        fill="#00FFFF") +
    theme_bw() + xlab("Age (in years)") + ylab("Density") +
    ggtitle("Histogram of Adopted Cat Ages") +
    theme(plot.title = element_text(hjust = 0.5))
```



Looking at the distribution, we see a heavy right skew. This implies that a majority of the cats being adopted are generally younger in age. Looking at the summary output, the median adopted cat was 0.2466 years, which is around 3 months.

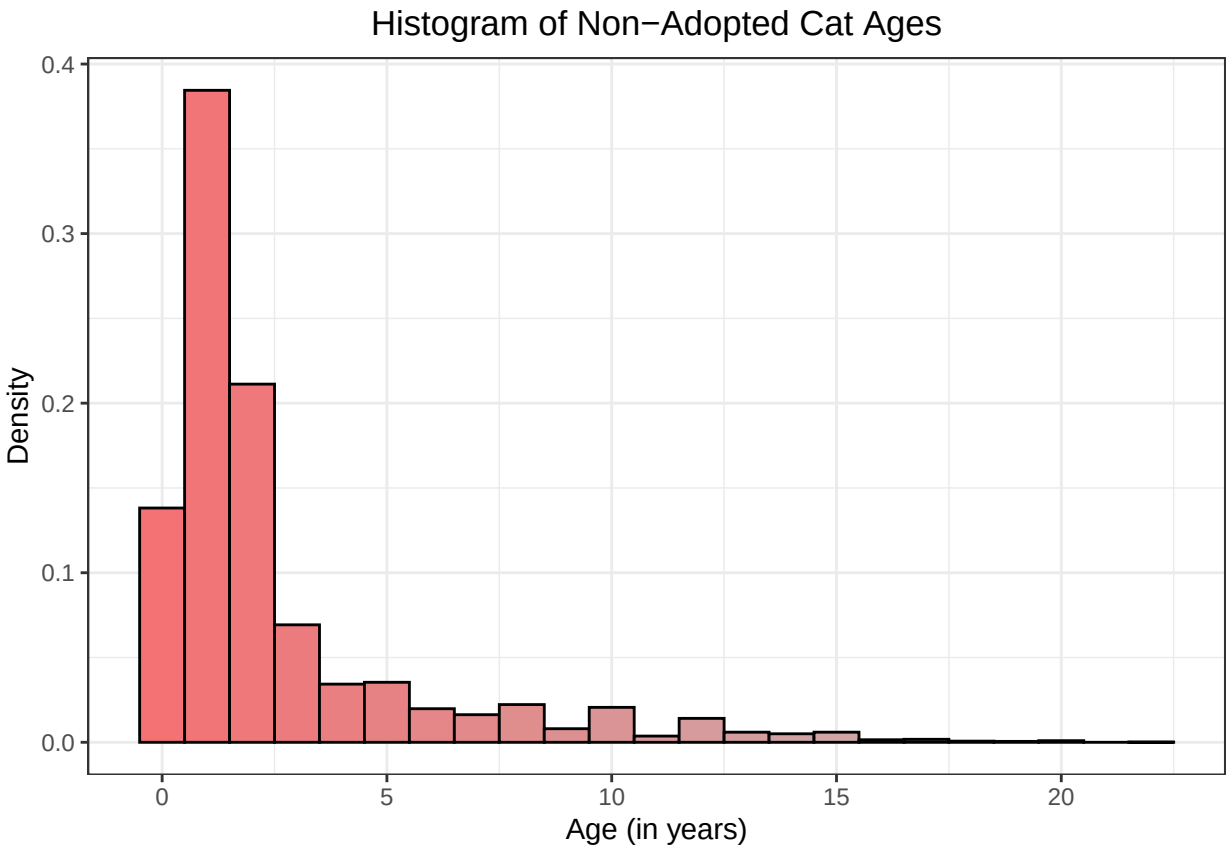
We can do the same for the non-adopted cats

```
# Look at the distribution of cats who were not adopted
cats_desc %>% filter(adopted =="No") -> cats_desc_nadopt

# Print summary stats for the age of non-adopted cats
summary(cats_desc_nadopt$age.at.outcome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3288  1.0000  1.0000  2.5935  3.0000 22.0000
```

```
# generate the plot with the color gradient
ggplot(data = cats_desc_nadopt, mapping = aes(x = age.at.outcome)) +
    geom_histogram(aes(y=..density.., ),binwidth=1, color="black",
        fill=colorRampPalette(c("#f47174", "gray"), bias=2)(23)) +
    theme_bw() + xlab("Age (in years)") + ylab("Density") +
    ggtitle("Histogram of Non-Adopted Cat Ages") +
    theme(plot.title = element_text(hjust = 0.5))
```



The distribution here is shifted more towards the right, implying the non-adopted cats tended to be older. The median age of a non-adopted cat was about a year old, which is the cutoff for defining it as a kitten.

Wordcloud of Cat Names for Adopted Cats

Cat names will not be usable in the classifiers due to how unique they are, but as a fun exercise, we can generate a word cloud containing the most frequent names among adopted cats. This can be done using the `wordcloud` package. We will also transform the names to lowercase and remove any astrichs in the names to normalize the strings for counting.

```
library(wordcloud)
library(stringr)

# Remove cats without names and consider only cats who were adopted
cat_names <- cats_orig %>% filter(name != "" & outcome_type== "Adoption") %>%
  pull(name)
# Lower all names to lowercase and remove astrichs to normalize the text
cat_names <- tolower( str_replace(cat_names, "[*]", "") )


wordcloud(cat_names, min.freq = 20, max.words = 40, scale=c(3.5,0.25))
```

Charlie, Luna, Bella and Kitty seem to a few of the most popular names among the cats who were adopted.

External Time Factors

Do more cats get adopted on a particular day? Or month of the year? We can use the data set to see the time periods that effect cat adoption. We will restrict our data to only adopted cats in this case, as it is more meaningful in terms of interpretation.

Day of the Week

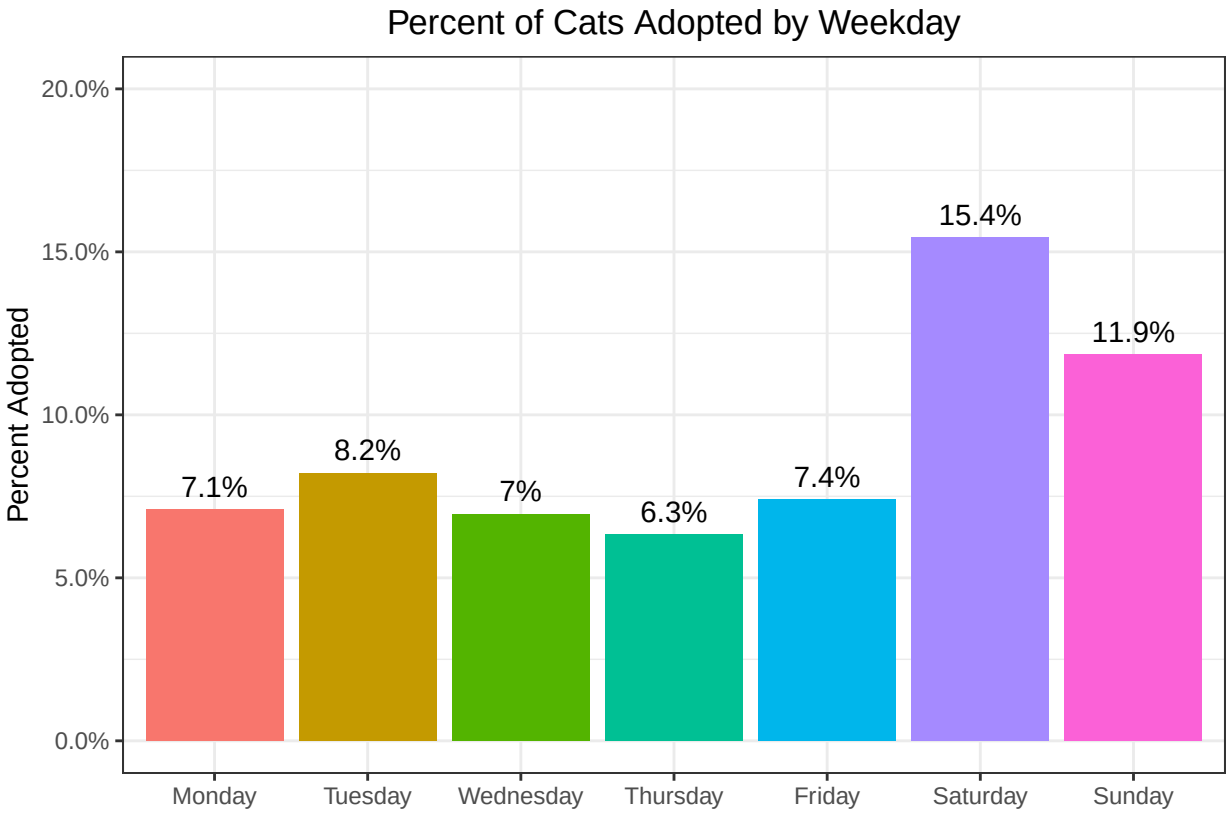
Knowing which day of the week most cats are adopted on can help shelters when organizing events and adoption fairs. We can create a bar chart to see the proportions of which weekday the cats were adopted on.

```
cats_desc %>% filter(adopted == "Yes") %>% group_by(outcome_weekday) %>%
  summarise(count = n(), prop= n() / nrow(cats_desc)) -> cats_by_day

cats_by_day$outcome_weekday <- factor(cats_by_day$outcome_weekday, levels =
  c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"))


ggplot(data=cats_by_day, mapping=aes(x = outcome_weekday, y = prop)) +
  geom_bar(aes(fill = outcome_weekday), stat= "identity") +
  geom_text(aes(label = format_prop(prop)),
              position = position_dodge2(width = .9), vjust=-.6) + theme_bw() +
  scale_y_continuous(labels = scales::percent, limits=c(0,.2)) +
```

```
  theme(legend.position = "None", plot.title = element_text(hjust = 0.5)) +
  xlab("") + ylab("Percent Adopted") +
  ggtitle("Percent of Cats Adopted by Weekday")
```



Looking at the plot, the 'proportions seem roughly similar until the weekend, where we see an increase. We can formally test if this increase is significant using a χ^2 Goodness-Of-Fit test. We will test the following hypothesis

H_0 : The proportion of adopted cats is the same across all weekdays

H_A : At least one weekday has a different adoption rate

```
# Perform a goodness of fit test
chisq.test(cats_by_day$count)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  cats_by_day$count
## X-squared = 1502.4, df = 6, p-value < 2.2e-16
```

The p-value is near 0, so we reject the null hypothesis. There is significant evidence to suggest that the day of the week are not equal in their adoption rates. It seems that weekends have higher adoption rates.

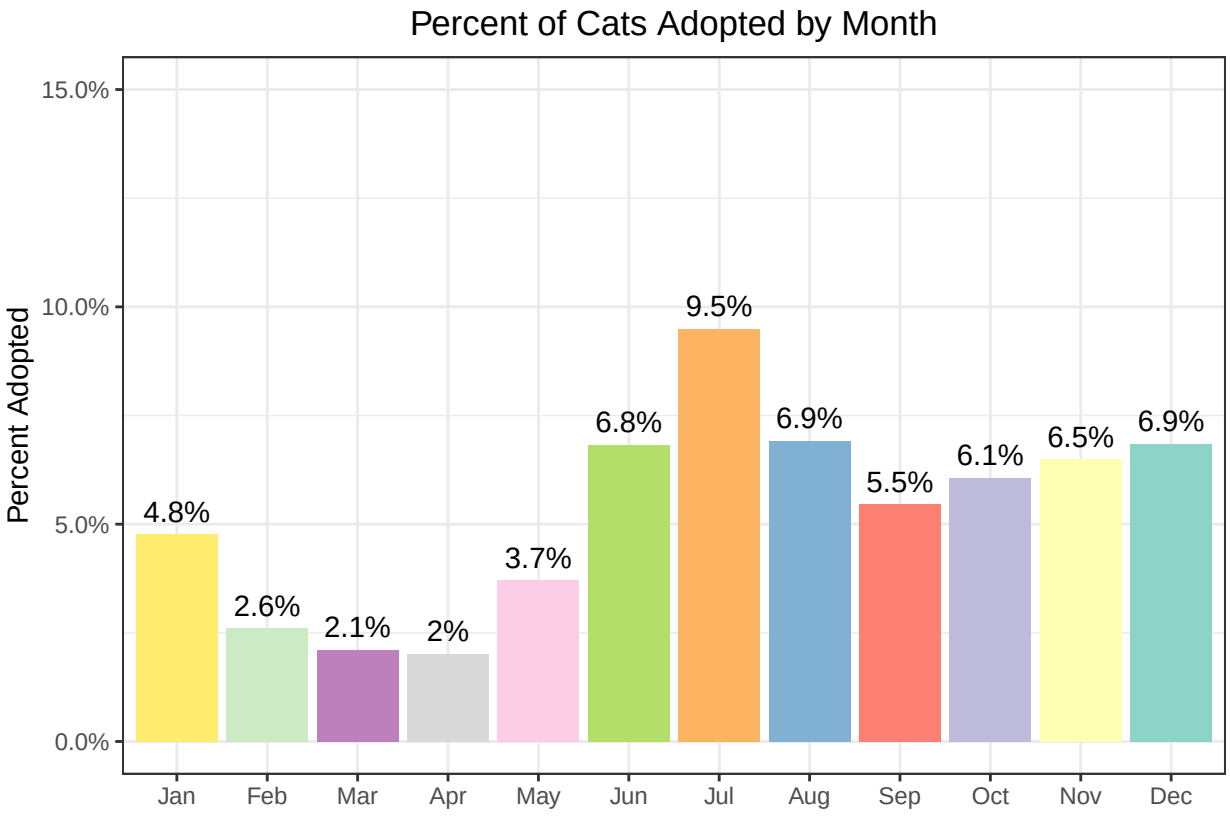
Month of the Year

Does the month of the year effect the adoption rate? Perhaps we can see trends around holidays, or seasonality. We can plot the proportions and look for trends.

```
# Get the proportion of cats adopted by month
cats_desc %>% filter(adopted == "Yes") %>% group_by(outcome_month) %>%
  summarise(count = n(), prop= n() / nrow(cats_desc)) -> cats_by_month

# Change the numeric months to their abbreviations
cats_by_month$outcome_month <- factor(month.abb[cats_by_month$outcome_month],
                                      levels = month.abb)

# Create a bar chart for the month
ggplot(data=cats_by_month, mapping=aes(x = outcome_month, y = prop)) +
  geom_bar(aes(fill = outcome_month), stat= "identity") +
  geom_text(aes(label = format_prop(prop)),
              position = position_dodge2(width = .9), vjust=-.6) + theme_bw() +
  scale_y_continuous(labels = scales::percent, limits=c(0,.15)) +
  theme(legend.position = "None", plot.title = element_text(hjust = 0.5)) +
  xlab("") + ylab("Percent Adopted") +
  ggtitle("Percent of Cats Adopted by Month")  +
  scale_fill_manual(values=RColorBrewer::brewer.pal(12,"Set3")[12:1])
```



Looking at the plot, we see some interesting results. July seems to be the month with the most cat adoptions. Excluding the jump in July, June through December have a roughly consistent adoption rate. What is

interesting is that we do see a drop off starting in January and continuing throughout the Winter and Spring until the start of Summer.

We can formally test this as well

H_0 : The proportion of adopted cats is the same across all 12 months

H_A : At least one month has a different adoption rate

```
# Perform a goodness of fit test
chisq.test(cats_by_month$count)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  cats_by_month$count
## X-squared = 2212, df = 11, p-value < 2.2e-16
```

The p-value is near 0, so we reject the null hypothesis. There is significant evidence to suggest that the months are not equal in their adoption rates.

Creating Classifiers

Now that we've explored the different features of the data, we can begin creating our classifiers.

Features for Consideration

We'll use the following variables in our classification models

- Sex of the cat
- Age of the cat in years
- Whether the cat was fixed or not
- If the cat is a CFA breed (Cat Fanciers Association - A registry of pedigreed cats)
- If the cat is a domestic breed
- The hour the outcome occurred
- The day of the week the outcome occurred
- The month the outcome occurred
- The primary color of the cat
- The coat pattern of the cat
- Whether or not the cat is considered a kitten

We can generate a string of our model in an R friendly format and continually pass it through our models.

```r
# Create the model formula we will be using for the classifiers
cat.formula <- paste0("adopted  ~ sex + Spay.Neuter + age.at.outcome",
" + cfa_breed + domestic_breed + outcome_weekday + outcome_month",
" + color1 + coat_pattern + outcome_hour + is.kitten")
```

Using Cross-Validation To Prevent Overfitting

To avoid potential data leakage, we will use a 5-fold cross validation procedure on each of our classifiers. We will compute metrics using each partition as the test data and pool together the predictive accuracy. This will help us assess the predictive power of the model outside the data set.

As a fun coding challenge, we will also create a cross validation function from scratch and use it on some of our classifiers as opposed to the black box implementations from other packages.

The following is the custom Cross Validation function along with some helper functions. These were kept separate for modularization purposes. It is designed to work with most models in R, although some implementations may require more care.

```r
# Helper Function that assigns fold ids to a dataframe
assign_fold_id <- function(k, n) {
  foldid <- sample(rep(1:k, length = n))
  return(foldid)
}

# Helper Function that computes accuracy given a confusion matrix
compute_accuracy <- function(conf_matr) {
  errors <- conf_matr[2,1] + conf_matr[1,2]
  tot <- conf_matr[2,1] + conf_matr[1,2] + conf_matr[1,1] + conf_matr[2,2]
  return( 1- errors/tot)
}

# Helper function that computes the optimal cutoff value using the ROC curve
compute_cutoff <- function(data, formula, labels) {
    all_fit <- glm(formula = formula, data=data,
                   family = "binomial")
      # Use the training data to get the optimal cutoff for accuracy
    pred <- predict(all_fit, type="response")
    pred <- prediction(pred, data[[labels]])

    perf <- ROCR::performance(prediction.obj =pred, measure = "acc")
    # Get the optimal cutoff value
    max_ind <- which.max(slot(perf, "y.values")[[1]] )
    cutoff <- slot(perf, "x.values")[[1]][max_ind]
    return(cutoff)
}

# Helper function that fits a logistic regression model
fit_logistic_model <- function(data, formula) {
  glm(formula = formula, data=data,
                   family = "binomial")
}

# gets predictions from different classifiers
```

```r
get_predictions <- function(fit,test=NULL, LogReg =FALSE, cutoff = 0) {
  # Classify for Logistic Regression using cutoff
  if(LogReg) {
    return(LogReg_predicitons(fit,test, cutoff ))
  }
  # Handle most other classifiers using predict
  else {
    # If test data is provided, use it
    if (!is.null(test)) {
      return(predict(fit,test, type="class"))
    }
    return(predict(fit, type="class"))
  }
}

# Special case that handles logistic regression prediction with ROC curve
LogReg_predicitons <- function(fit,test, cutoff ) {
  # If test data is supplied, test that
  if(!is.null(test)) {
    test_pred <- ifelse(predict.glm(fit, test, type="response") > cutoff, 1, 0)
    return(test_pred)
  }
  # Otherwise, just check the training data
  train_pred <- predict(fit, type="response")
  train_pred <- ifelse(train_pred > cutoff, 1, 0)
  return(train_pred)

}

# Function that performs k-fold Cross Validation for a logistic regression fit
Cross_Validate <- function(data, formula, K, labels, fit_model, LogReg=F) {
  # shuffle the data
  data <- data[sample(1:nrow(data)),]
  # Assign fold IDs to the data
  folds <- assign_fold_id(K, nrow(data))
  formula <- as.formula(formula)

  # Create vectors to store proportions for each iteration
  train_accuracy_vec <- double(K)
  test_accuracy_vec <- double(K)
  FPR_vec <- double(K)
  FNR_vec <- double(K)

  # Get the cutoff from the ROC curve using all the data if doing a LogReg fit
  cutoff <- 0
  if (LogReg) {
    cutoff <- compute_cutoff(data = data, formula = formula,labels=labels)
  }

  # Fit the model and test for each partition
  for (i in 1:K) {
    # Subset the data into training and testing
    train_data <- subset(data, folds != i)
```

```
    test_data <- subset(data, folds == i)

    # Fit the model using the training data
    train_fit <- fit_model(data=train_data, formula=formula)

    # Check the accuracy of the training data
    train_pred <- get_predictions(train_fit, LogReg = LogReg,cutoff=cutoff)
    # Create the Confusion Matrix with these predictions
    confusion.matrix.train <- table(train_pred,train_data[[labels]])
    train_accuracy_vec[i] <- compute_accuracy(confusion.matrix.train)

    # Predict the test data points using this cutoff
    test_pred <- get_predictions(fit=train_fit, test=test_data,
                                 LogReg = LogReg,cutoff =cutoff )

    # Create the Confusion Matrix with these predictions
    confusion.matrix <- table(test_pred,test_data[[labels]])
    #store test accuracy, FPR and FNR for the iteration
    test_accuracy_vec[i] <- compute_accuracy(confusion.matrix)
    FPR_vec<-confusion.matrix[1,2]/(confusion.matrix[1,1]+confusion.matrix[1,2])
    FNR_vec<-confusion.matrix[2,1]/(confusion.matrix[2,1]+confusion.matrix[2,2])

  }
  # Average accuracy, FPR, and FNR over all folds
  print(paste0(K,"-Fold Cross Validation Metrics" ))
  print(paste("Pooled Training Data Accuracy:",mean(train_accuracy_vec)))
  print(paste("Pooled Test Data Accuracy:", mean(test_accuracy_vec)))
  print(paste("Pooled False Positve Rate:", mean(FPR_vec)))
  print(paste("Pooled False Negative Rate:", mean(FNR_vec)))

}
```

Logistic Regression

The first classifier will be a logistic regression model. Before testing the model, we need to consider what threshold we want to use for classifying using the estimated proportions. We can use functions from the `ROCR` package to tune an optimal cutoff. False Positive or False Negative rates may be under individual consideration in some contexts, but here we will tune for overall accuracy. We will get this cutoff from all of the training data and use it as the cutoff in the Cross Validation models.

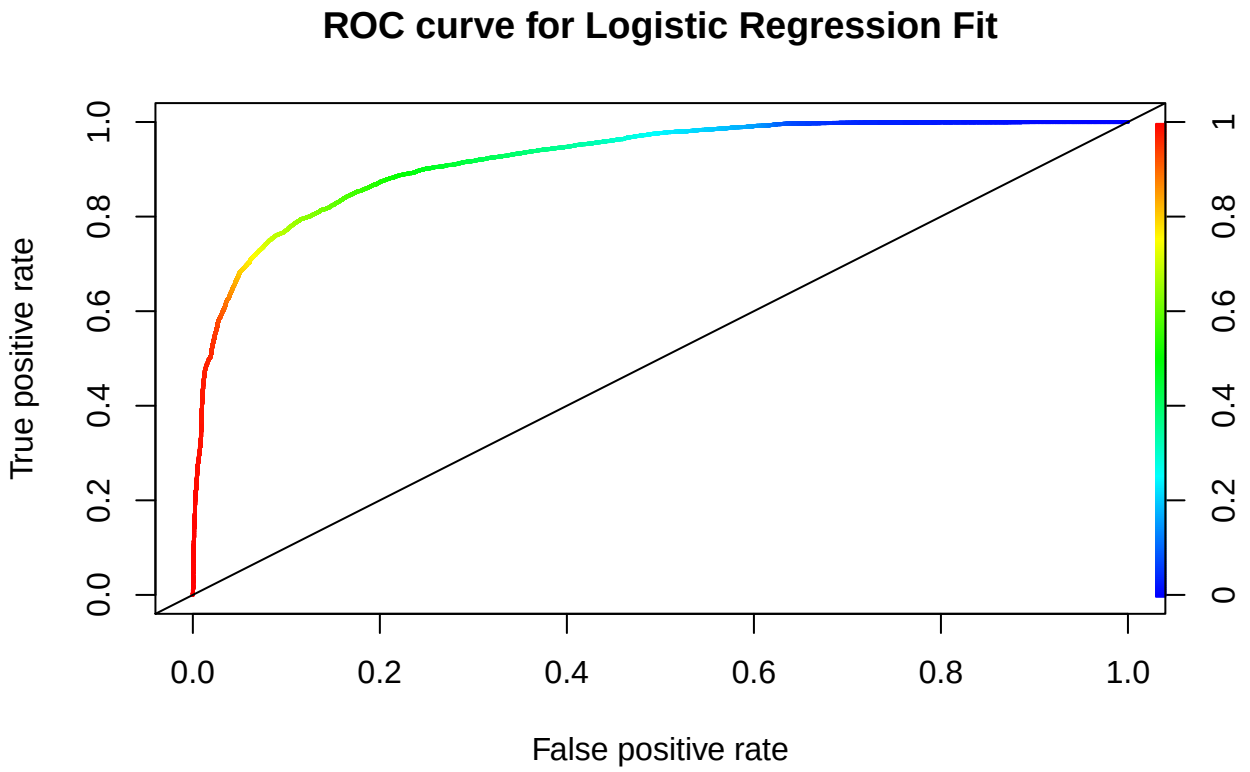
```
library(ROCR)
# Fit all of the data into a model to see the training data ROC curve
log_fit <- glm(formula = cat.formula, data= cats_clean, family = "binomial")
pred <- predict(log_fit, type="response")
pred <- prediction(pred, cats_clean$adopted)

# Look at the training fit to see the variable effects
summary(log_fit)
```

```
##
## Call:
## glm(formula = cat.formula, family = "binomial", data = cats_clean)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4643  -0.3177   0.1551   0.4588   3.2412
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -5.779408   0.522370 -11.064  < 2e-16 ***
## sexMale                     0.044515   0.047403   0.939  0.34769
## Spay.NeuterTRUE             3.299312   0.072229  45.678  < 2e-16 ***
## age.at.outcome             -0.007309   0.007003  -1.044  0.29662
## cfa_breedTRUE               0.134848   0.471570   0.286  0.77491
## domestic_breedTRUE         -0.336519   0.480219  -0.701  0.48345
## outcome_weekdayMonday      -0.076491   0.080229  -0.953  0.34039
## outcome_weekdaySaturday     1.254682   0.079250  15.832  < 2e-16 ***
## outcome_weekdaySunday       0.967936   0.078866  12.273  < 2e-16 ***
## outcome_weekdayThursday    -0.015886   0.083867  -0.189  0.84976
## outcome_weekdayTuesday     -0.004893   0.078063  -0.063  0.95002
## outcome_weekdayWednesday   -0.022835   0.081213  -0.281  0.77858
## outcome_month2              0.087380   0.109944   0.795  0.42675
## outcome_month3             -0.301715   0.111363  -2.709  0.00674 **
## outcome_month4             -0.273399   0.114608  -2.386  0.01706 *
## outcome_month5             -0.132815   0.104618  -1.270  0.20425
## outcome_month6              0.157137   0.101272   1.552  0.12075
## outcome_month7              0.183566   0.102350   1.794  0.07289 .
## outcome_month8              0.007723   0.101760   0.076  0.93950
## outcome_month9             -0.106638   0.102037  -1.045  0.29598
## outcome_month10            -0.204847   0.097039  -2.111  0.03477 *
## outcome_month11            -0.180403   0.100066  -1.803  0.07141 .
## outcome_month12             0.031891   0.095447   0.334  0.73828
## color1blue                  0.137674   0.084889   1.622  0.10484
## color1brown                 0.008385   0.107346   0.078  0.93774
## color1orange               -0.109375   0.115499  -0.947  0.34365
## color1other                -0.046844   0.124454  -0.376  0.70662
## color1white                 0.009800   0.100563   0.097  0.92237
## coat_patternother           0.438100   0.361518   1.212  0.22558
## coat_patternpoint          -0.324638   0.150927  -2.151  0.03148 *
## coat_patternsolid          -0.530075   0.146962  -3.607  0.00031 ***
## coat_patterntabby          -0.364088   0.133582  -2.726  0.00642 **
## coat_patterntorbie         -0.096188   0.142167  -0.677  0.49867
## coat_patterntortie         -0.224553   0.125190  -1.794  0.07286 .
## outcome_hour                0.205475   0.006388  32.167  < 2e-16 ***
## is.kittenTRUE               3.577106   0.067449  53.034  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26432  on 20107  degrees of freedom
## Residual deviance: 13532  on 20072  degrees of freedom
## AIC: 13604
##
## Number of Fisher Scoring iterations: 6
```

```
# Plot the ROC curve for the training data
roc <- performance(pred, "tpr", "fpr")
plot(roc, colorize=T, lwd=2, main = "ROC curve for Logistic Regression Fit")
abline(a=0,b=1)
```




```
# Get the optimal cutoff value and see what the accuracy is for the training set
perf <- ROCR::performance(prediction.obj =pred, measure = "acc")
max_ind <- which.max(slot(perf, "y.values")[[1]] )
(acc <- slot(perf, "y.values")[[1]][max_ind])
```

```
## [1] 0.8477223
```

```
(cutoff <- slot(perf, "x.values")[[1]][max_ind])
```

```
##      9919
## 0.5042054
```

According to the Logistic Regression model, some of the significant predictors include if the cat was fixed, if the outcome occurred on a weekend, the month of the outcome (for certain months), the coat pattern, the hour the outcome occurred, and whether or not the cat was a kitten.

We can perform the Cross Validation Procedure to see how well the model predicts new data

```
# Perform CV to see how well the model predicts new data
Cross_Validate(data = cats_clean, formula = cat.formula, K=5, labels = "adopted",
            fit_model = fit_logistic_model, LogReg = T)
```

```
## [1] "5-Fold Cross Validation Metrics"
## [1] "Pooled Training Data Accuracy: 0.847349307999743"
## [1] "Pooled Test Data Accuracy: 0.8460809615753"
## [1] "Pooled False Positve Rate: 0.207239176721079"
## [1] "Pooled False Negative Rate: 0.129019908116386"
```

The accuracy rate is between 84% and 85% accuracy for the training and test set. We see that the False Positive Rate is higher than the False Negative rate.

Support Vector Machine

Next we will consider a support vector classifier. We will use a linear kernel. This model requires a "cost" parameter that needs to be tuned. This can be done using Cross Validation.

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.5
```

```
# Reformat the data to work well with SVM
cats_clean_svm <- cats_clean

# Redefine the adoption outcome so that it is 1 or -1
cats_clean_svm$adopted <- as.factor(cats_clean_svm$adopted)

# Tune for the best cost value using Cross Validation
tuned_svm <- tune(svm, train.x=as.formula(cat.formula), data = cats_clean_svm,kernel="linear",
                  scale=F,ranges=list(cost=seq(.1,.10,.1)),
                  tunecontrol=tune.control(sampling="cross",cross=2))


# Save the best tuned cost
best_cost_svm <- tuned_svm$best.parameters$cost
# Compute a confusion matrix for the training data
CMatrix <- table(predict(tuned_svm$best.model), cats_clean_svm$adopted)
colnames(CMatrix) <- c("Not Adopted", "Adopted")
rownames(CMatrix) <- c("Not Adopted", "Adopted")
CMatrix # print the confusion Matrix for the Training Data
```

```
##
##                Not Adopted Adopted
##   Not Adopted         5801    1620
##   Adopted             1575   11112
```

With the optimal tune parameter, we can perform cross validation on our model to see its predictive capabilities.

```
fit_SVM <- function(data, formula) {
  return(
    svm(formula = as.formula(formula), data = data, kernel="linear",
        cost = best_cost_svm)
  )
}

Cross_Validate(data = cats_clean_svm, formula = cat.formula, K=5, labels = "adopted",
            fit_model = fit_SVM, LogReg = F)
```

```
## [1] "5-Fold Cross Validation Metrics"
## [1] "Pooled Training Data Accuracy: 0.840250122304864"
## [1] "Pooled Test Data Accuracy: 0.838670723109444"
## [1] "Pooled False Positve Rate: 0.223684210526316"
## [1] "Pooled False Negative Rate: 0.123950419832067"
```

Looking at the predictive accuracy, we are still getting around 84% for the training and testing sets.

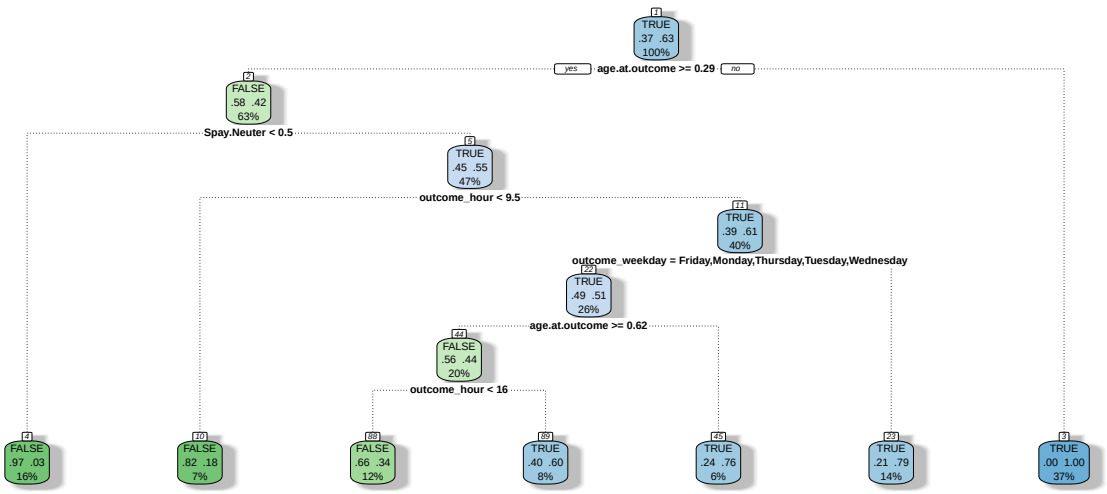
Decision Tree

Using the `rpart` library, we can create a decision tree classifier. We can draw a tree using all the training data and visualize it to get a sense of what variables result in the most information gain.

```
library(rpart)
library(rpart.plot)
library(rattle)

fit_tree <- function(data, formula) {
  return(
    rpart(formula = as.formula(formula), data=data,
                  method = "class")
  )
}

fancyRpartPlot( fit_tree(cats_clean, cat.formula),
                main = "Decision Cat Tree", sub="" )
```



Looking at the decision tree, we can see that age, fixing and time (hour, day, month) are considered the most influential observations. It also suggests that young kittens are especially likely to get adopted, which is not too suprising of a result.

We can then use cross validation to assess the predictive power of the decision tree for classifying adoption. Note that the trees compute local solutions, so not all trees might be the same when fit with different training sets.

```
Cross_Validate(data = cats_clean,formula = cat.formula, K=5, labels = "adopted",
            fit_model = fit_tree, LogReg = F)
```

```
## [1] "5-Fold Cross Validation Metrics"
## [1] "Pooled Training Data Accuracy: 0.865190494360667"
## [1] "Pooled Test Data Accuracy: 0.864929767650714"
## [1] "Pooled False Positve Rate: 0.169418521177315"
## [1] "Pooled False Negative Rate: 0.114155251141553"
```

The decision tree has a predictive power of around 86% accuracy, which so far is the best performance.

K-Nearest Neighbors

The last classifier we will consider is the K-Nearest Neighbors. This classifier requires a bit more caution as it will not be able to work well with some of the categorical variables. This is because KNN computes euclidean distance, which is not possible on a feature like color. For some features such as the day of the week, we can re-factor them as ordinal variables and use their numerical values inside the distance.

```r
# Remove non ordinal variables
cats_clean %>% select(-outcome_subtype,-outcome_type,-age_group:-dob_monthyear,
                      -color2,-coat, -breed, -outcome_year, -coat_pattern, -color1) %>%
  mutate(sex = ifelse(sex=="Male", 1,0),
         outcome_month= as.numeric(outcome_month),
         adopted = as.factor(adopted),
         outcome_weekday = as.integer(outcome_weekday)) -> cats_knn
```

Which K should we use for KNN? We can tune for this using the K that has the best predictive accuracy. The `caret` function has a handy framework that will allow us to do this.

```r
library(caret)
# Set how we will train the model
trControl <- caret::trainControl(method  = "cv",
                                 number  = 5)
# Fit the KNN classifier and tune for the best k
fit <- caret::train(adopted ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:20),
             trControl  = trControl,
             metric     = "Accuracy",
             data       = cats_knn)
# See which model performed the best
fit
```

```
## k-Nearest Neighbors
##
## 20108 samples
##     9 predictor
##     2 classes: 'FALSE', 'TRUE'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 16086, 16087, 16087, 16086, 16086
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.8422023  0.6599686
##    2  0.8421528  0.6598172
##    3  0.8496622  0.6754117
##    4  0.8473748  0.6702097
##    5  0.8505574  0.6767586
##    6  0.8497619  0.6748922
##    7  0.8505079  0.6764850
##    8  0.8486680  0.6721378
##    9  0.8495632  0.6743792
##   10  0.8497124  0.6742831
##   11  0.8487675  0.6721709
##   12  0.8476235  0.6695920
##   13  0.8485187  0.6715203
##   14  0.8472257  0.6684945
##   15  0.8476235  0.6692007
##   16  0.8467781  0.6673918
```

```
##   17  0.8472256  0.6684328
##   18  0.8464796  0.6669577
##   19  0.8449380  0.6635061
##   20  0.8451866  0.6640201
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

Based on predictive accuracy, k = 5 was considered the best. This returned a predictive accuracy of 0.8505574.

Key Takeaways

The classifiers were able to predict correctly around 83%-86% of the time. Some of the most important features mentioned throughout were

- The Age of the Cat
- Time, Month and Day of the outcome
- Whether or not the cat was considered a kitten
- Whether or not the cat was fixed

Some of these variables were not terribly suprising. It is no secret that kittens generally have an easier time being adopted compared to older cats as they can get accustomed to living with humans better. Fixing cats may or may not be required at shelters, which may be biasing the predictive results a bit. Nonetheless, fixing a cat can be a bit pricey, so it is not unreasonable to say that adopting a fixed cat is a huge plus monetarily.

Physical characteristics such as sex and color didn't seem to have a huge impact in the classification process. This was also hinted at in the exploratory analysis. Perhaps the aesthetics of a cat is something that people may not care much about when deciding to adopt one.

The date effect was an intersting find. It seems as though adoptions were especially likely during the weekends. This could be due to outside influences such as adoption events being more prevalent during these times. A follow up analysis could be to track if any large scale adoption events occurred on weekends. If not, the shelters should definitely emphasize any events on weekends as people are generally more free anyways.

Cat adoptions seemed to be popular during the Summer and Fall, and then fall off in post-Christmas into Spring. Perhaps shelters could incorporate some sort of marketing campaign to make up for these downturns.

One suggestion I would make to the shelter would be to consider including the length of time that the animal was in the shelter. This was somewhat included, but not very precisely and left to some interpretation. If this was included, a survival analysis could be done on the time until outcome for the cats. This would provide valuable insight while also being able to account for the censoring data.