Shelter Animal Outcomes

Gabriel Lapointe

May 13, 2016

Contents

Data Acquisition

Every year, approximately 7.6 million companion animals end up in US shelters. Many animals are given up as unwanted by their owners, while others are picked up after getting lost or taken out of cruelty situations. Many of these animals find forever families to take them home, but just as many are not so lucky. 2.7 million dogs and cats are euthanized in the US every year.

Objective

The objective is to help improving outcomes for shelter animals. Using a dataset of intake information including breed, color, sex, and age from the Austin Animal Center, we have to predict the outcome for each animal.

To support our analysis, we are given the train and test sets in CSV files. From this, we have written a code book to help understanding each feature and their values.

Data Source

The data comes from Austin Animal Center from October 1st, 2013 to March, 2016. Outcomes represent the status of animals as they leave the Animal Center. All animals receive a unique Animal ID during intake. Source of the competition: Shelter Animal Outcomes.

Dataset Questions

Before we start the exploration of the dataset, we need to write a list of questions about this dataset considering the problem we have to solve.

- How big is the dataset? (Described in the Code Book)
- Does the dataset contains `NA` or missing values? Can we replace them by a value? Why?
- Does the data is coherent (date with same format, no out of bound values, no misspelled words, etc.)?
- What does the data look like and what are the relationships between features if they exist?
- What are the measures used?
- Can we solve the problem with this dataset?

Questions on features:

- Are features in the dataset sufficiant to explain each outcome?
- What is the proportion of animals and how many cats and dogs are they for each outcome?
- Are there features that can be split in many other features? If yes, are they improving the score? Why and how?
- For each outcome, what features have the most importance and why?

Evaluation Metrics

Since we have 5 outcome types where we need to calculate a prediction's probability for each class (outcome type), we have to manage 5 classes using a multi-class algorithm. The `Log Loss` function quantifies the accuracy of a classifier by penalising false classifications. Minimising the `Log Loss` function is basically equivalent to maximising the accuracy of the classifier which is what we need.

Thus, Kaggle provides us the evaluation metric we need:

$$LogLoss = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{M} y_{i,j}\log \mathbb{P}_{i,j}$$

where N is the total number of animals, M is the number of outcomes, $y_{i,j}$ is 1 if observation i is in outcome j and 0 otherwise, $\mathbb{P}_{i,j}$ is the predicted probability that observation i belongs to outcome j.

Methodology

When we will evaluate a model, we will train it first with the training set. When our model will be chosen, we will test it with this model. We will also use cross-validation since we are limited with the data given.

Since the output to predict is known and well defined, we will use a supervised algorithm to solve the problem. This is a multi-class problem where we have to give a probability for each class.

In this document, we start by exploring the dataset and build the data story behind it. This will give us important insights which will answer our questions on this dataset. The next step is to do some feature engineering which consists to create, remove or replace features regarding insights we got when exploring the dataset. We will ensure our new dataset is a valid input for our prediction model. After applying our model to the test set, we will visualize the predictions calculated and explain the results.

Data Preparation and Exploratory

In this section, we explore the dataset and we test hypotheses on features. The objective is to visualize and understand the relationships between features in the dataset we have to solve the problem. We will also compare changes we will make to this dataset to validate if they have significant influance on the outcomes or not.

We will also add and remove features from the dataset depending on the importance of insights we will find with data visualization. We will transform each feature in categories represented by a positive integer. The objective is to clean and prepare the dataset for our prediction's model and to answer the questions: `Are there features that can be split in many other features? If yes, are they improving the score? Why and how?`. Those questions will be answered throughout this section and with the data story presented at the conclusion of this document.

We will look at each feature of the dataset and split them if there are more information given than what the feature's name tells. For example, the feature `SexuponOutcome` tells us more than just the sex of the animal regarding the possible values. We can extract the sex and if the animal is sterile or not.

Each features having limited number of unique values (categories) will be numbered by a positive integer starting at 0. The zero-based ID will not hold for new features describing a number of elements (e.g. month or day number). Those ones will be one-base instead.

We first load the test and train datasets, and set the seed. Lets answer the question: `Does the dataset contains NA or missing values? Can we replace them by a value? Why?` All blank, space and unknown values are replaced by the value `0`. After writting the code book, we have seen that unique values in all features except the ID can be interpreted as Categories which can be grouped by another feature. In this way, the unknown, space or blank values can be treated as a Category ID as well. Thus, we decide to set their value to 0.

```r
source("Utilities.R")

strings.toNA <- c("", " ", "Unknown", "0 years")
train <- read.csv("train.csv",
                  header = TRUE,
                  na.strings = strings.toNA,
                  stringsAsFactors = FALSE)
test <- read.csv("test.csv",
                 header = TRUE,
                 na.strings = strings.toNA,
                 stringsAsFactors = FALSE)
```

```r
set.seed(1234)

## Remove scientific notation (e.g. E-005).
options(scipen = 999)

## Remove the AnimalID which are not needed for the analysis.
train$AnimalID <- NULL
test.id <- test$ID
test$ID <- NULL

train[is.na(train)] <- 0
test[is.na(test)] <- 0

## Samples and population sizes.
train.n <- nrow(train)
test.n <- nrow(test)
N <- train.n + test.n
```

Since the number of possible values for each feature is generally greater than 4 or 5, we will use bar charts to visualize our analysis of the dataset.

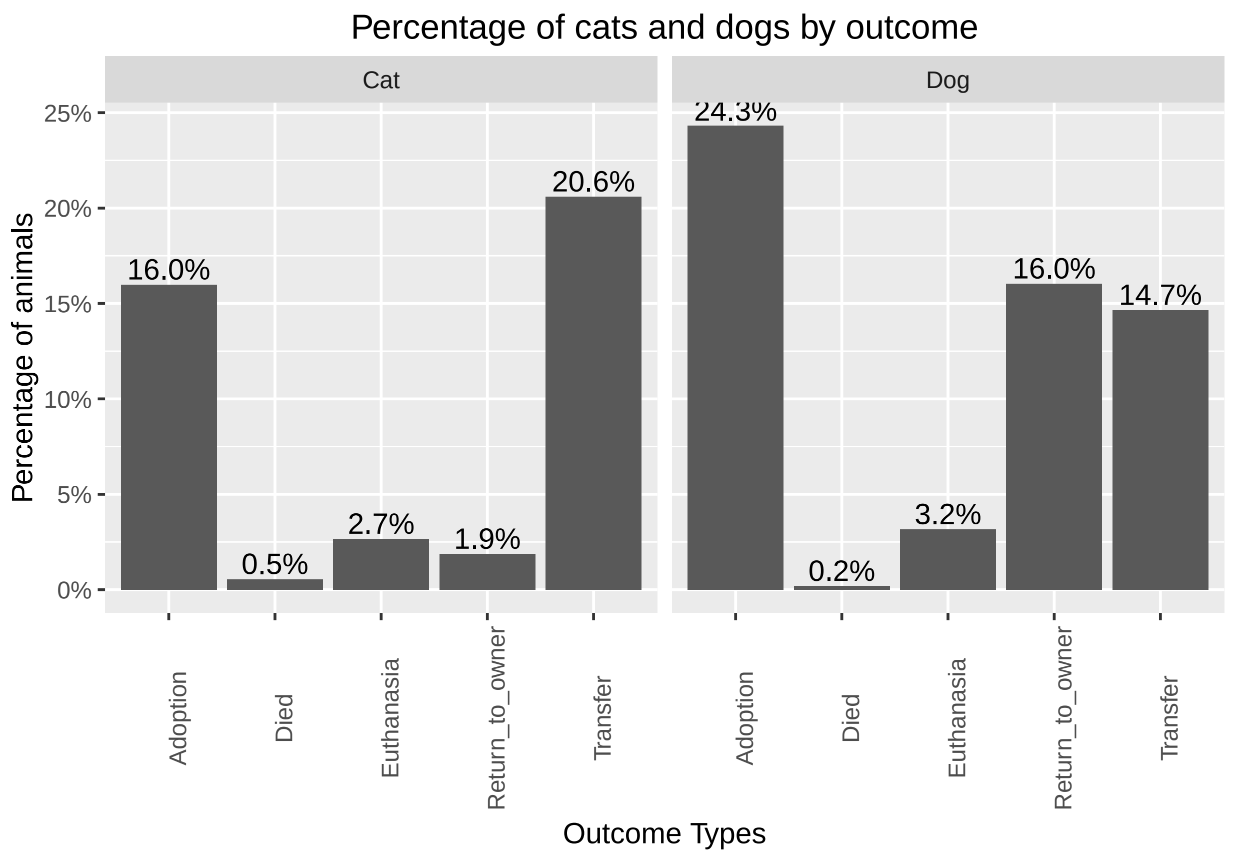
We also know that the population is $N = 38185$ animals. The samples are $n_{test} = 11456$ animals which is 30.0013094 % of the population and $n_{train} = 26729$ animals the other 69.9986906% of the population.

Data Coherence

We answer the question: `Does the data is coherent (date with same format, no out of bound values, no misspelled words, etc.)?` Most of features are coherent, but we found some incoherence in the `AgeuponOutcome` feature. Incoherent values found are `0 years` which is impossible because age cannot be 0. Also, a grammar error on the value `1 weeks` which does not take a ‘s’. Since there is a value `1 week` which is correctly written, we need to find (with `grep`) values containing the string “week”. This will automatically give both, weeks and week.

Outcomes Visualization

The objective is to answer the question: `What is the proportion of animals and how many cats and dogs are they for each outcome?`. Let’s visualize how the outcomes are split in the train set with the following histogram.



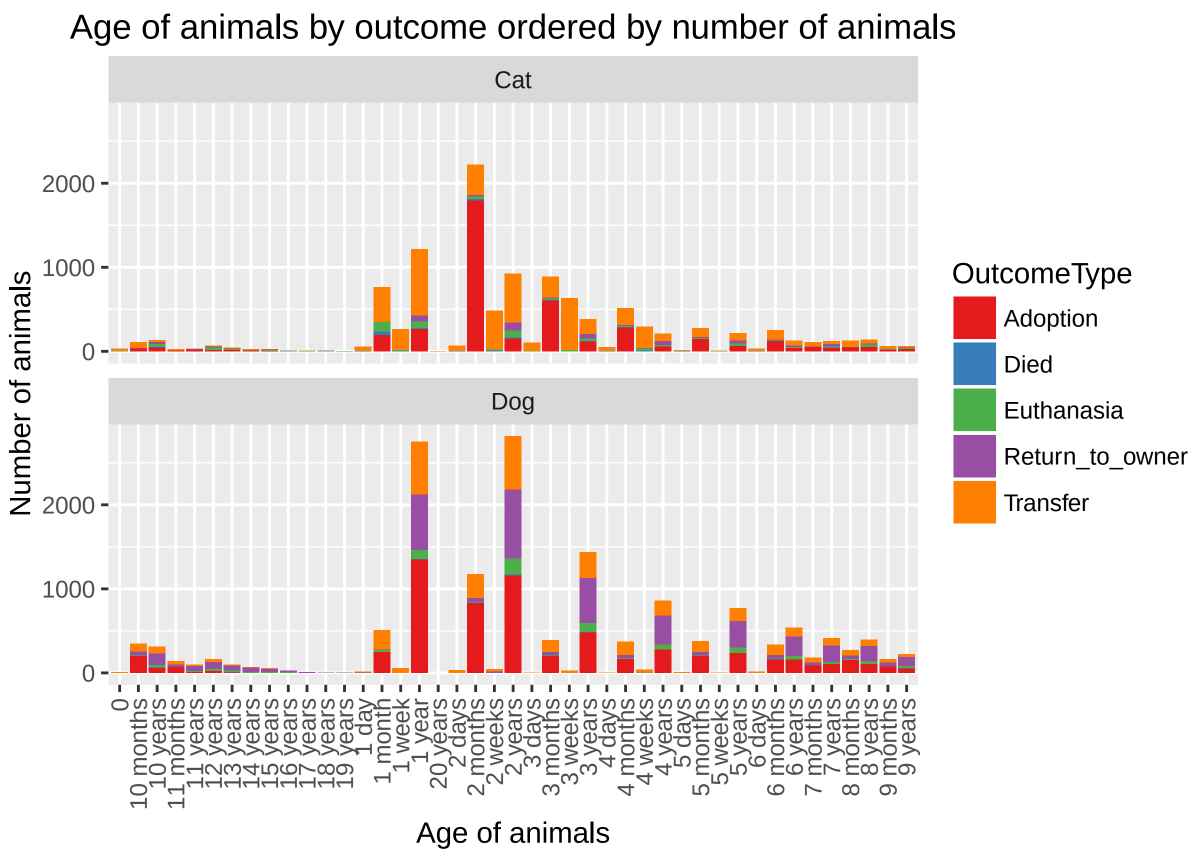
We can see from this bar chart that dogs are returned to their owner almost 8 times (16% for dogs against 1.9% for cats) more than cats. Cats are mostly transferred and dogs are mostly returned to the owner with significant difference. The percentage for the other outcomes are slightly the same.

We replace the possible values of the AnimalType feature by 0 = Dog and 1 = Cat.

```
train.copy <- train

animal.types <- unique(c(train$AnimalType, test$AnimalType))
train$AnimalType <- GetIntegerFeatureFromGroups(train$AnimalType, animal.types) - 1
test$AnimalType <- GetIntegerFeatureFromGroups(test$AnimalType, animal.types) - 1
```

Animal's Age

The age of animals should be an important factor on outcomes. People want to adopt a young animal (between 1 month and 1 year old) because they can raise the animal more easily and keep it longer. Older animals (Over 1 year old) may have more difficulty to respond to their new master. Baby animals may be subject to a transfer for experimentations or for clinical reasons. Adopting a baby animal (less than a month old) needs more attention from the owner. Thus, adoptions should not be frequent for them. We show a bar chart of ages of animals.



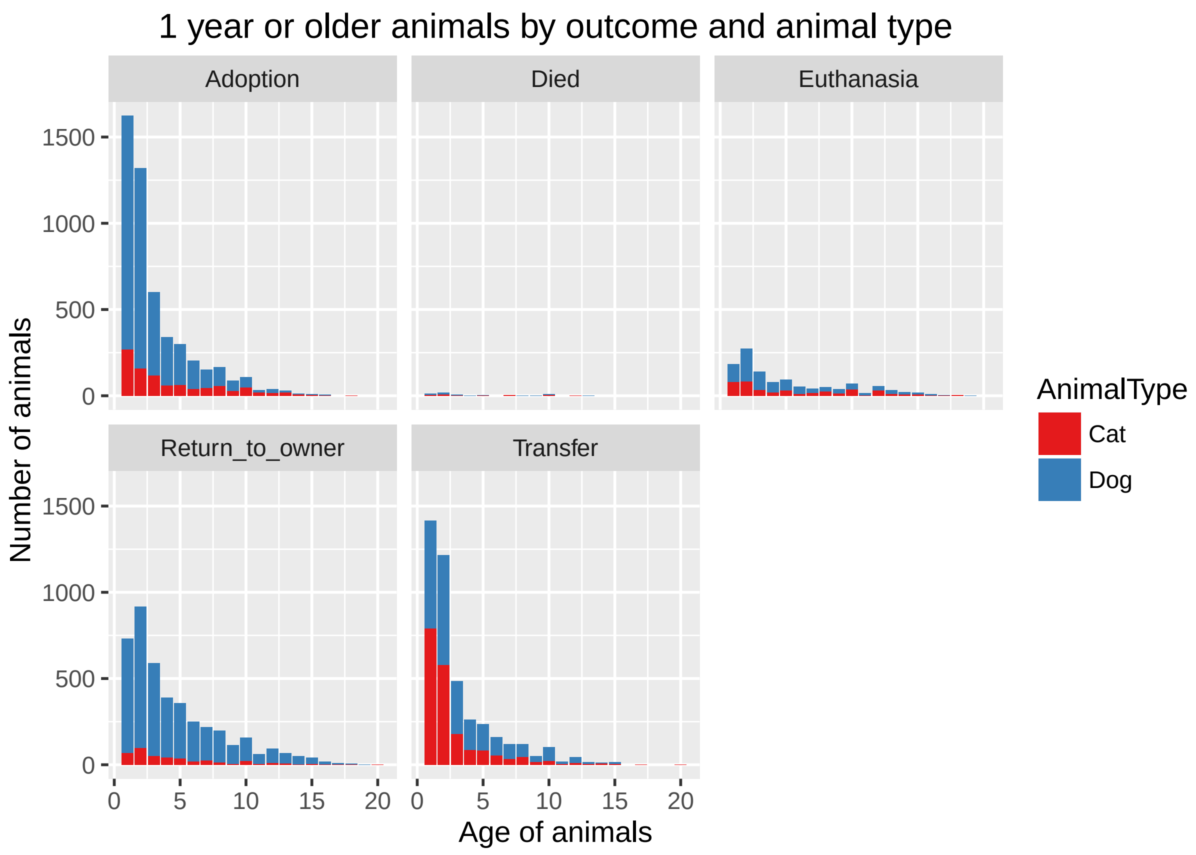
We can see that between 1 day and 4 weeks old, animals are mostly transferred including unknowns identified by 0. Animals are mostly adopted between 1 month and 1 years old. Between 1 year and 10 years old, animal adoptions decrease while age increases. Finally, animals are euthanasied or returned to their owner specially between 11 years and 20 years old.

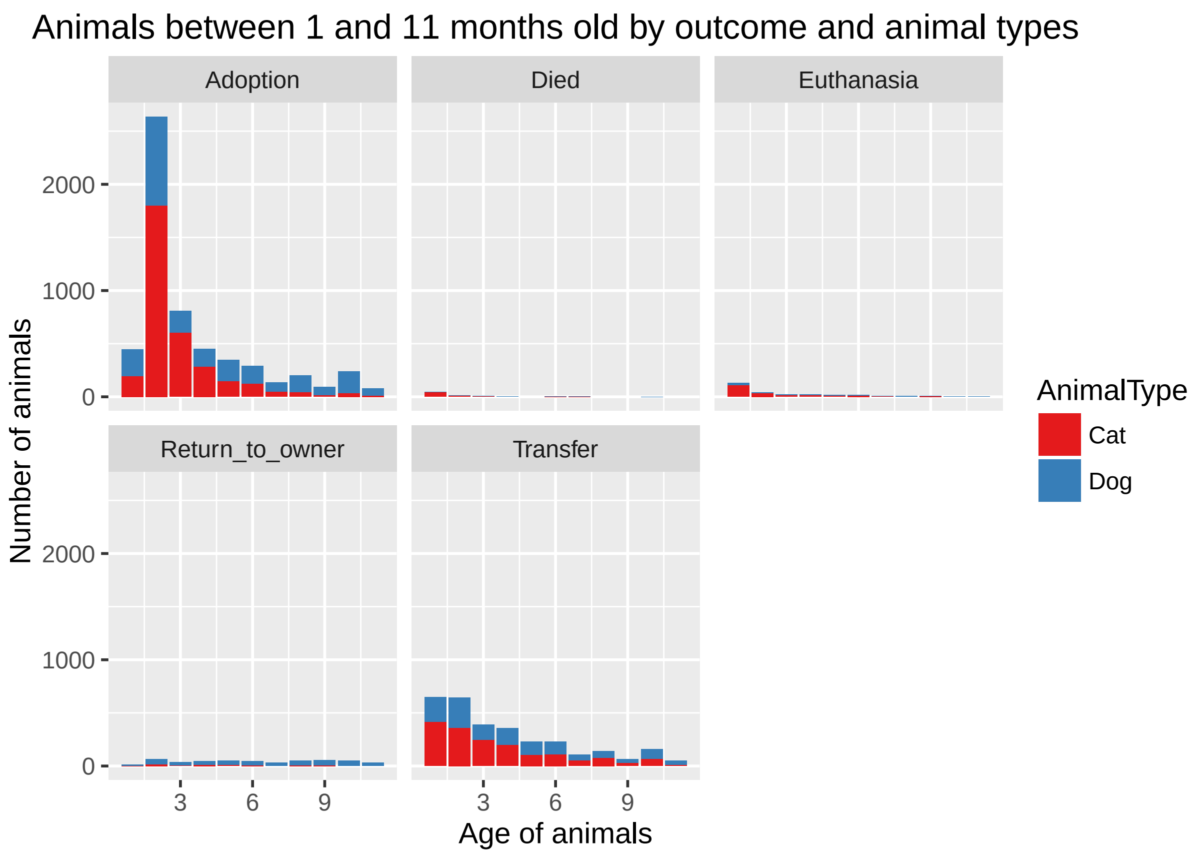
We transform the feature `AgeuponOutcome` to integer values. For example, the age should be counted in days. Thus, `2 years` is replaced by the value 2 * 365 = 730. For months, the formula is `age * 30`. For weeks, the formula is `age * 7`. For years, the formula is `age * 365`.

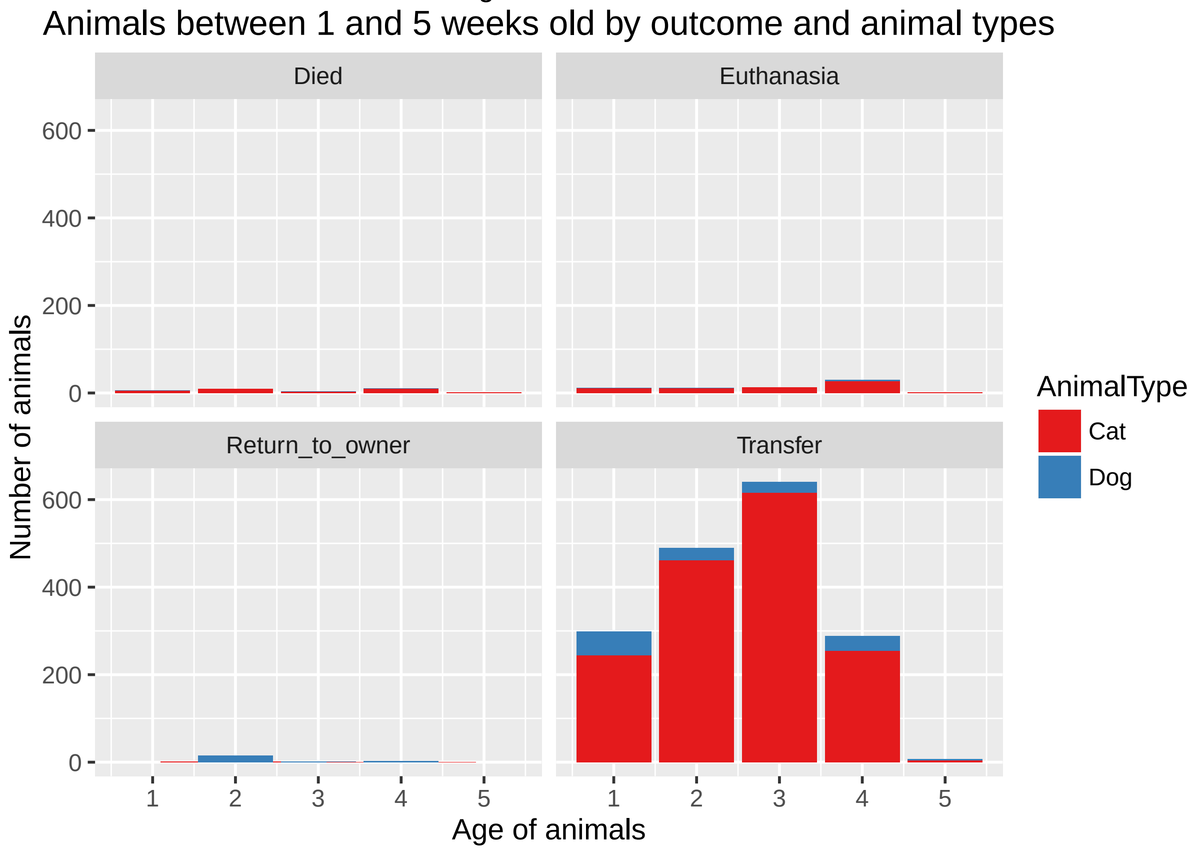
```
## Get the list of integers extracted from the feature AgeuponOutcome.
train.age <- as.integer(regmatches(train$AgeuponOutcome,
                                   regexpr("[[:digit:]]+", train$AgeuponOutcome)))
test.age <- as.integer(regmatches(test$AgeuponOutcome,
                                  regexpr("[[:digit:]]+", test$AgeuponOutcome)))

## Get row index list where AgeuponOutcome contains "year", "month", "week" or "day".
## Get the correspondant integer from each row index and
## apply the formula to get all ages in days.
train.year.list <- grep("year", train$AgeuponOutcome)
train$AgeuponOutcome[train.year.list] <- train.age[train.year.list] * 365
train.month.list <- grep("month", train$AgeuponOutcome)
train$AgeuponOutcome[train.month.list] <- train.age[train.month.list] * 30
train.week.list <- grep("week", train$AgeuponOutcome)
train$AgeuponOutcome[train.week.list] <- train.age[train.week.list] * 7
train.day.list <- grep("day", train$AgeuponOutcome)
train$AgeuponOutcome[train.day.list] <- train.age[train.day.list]

test.year.list <- grep("year", test$AgeuponOutcome)
```

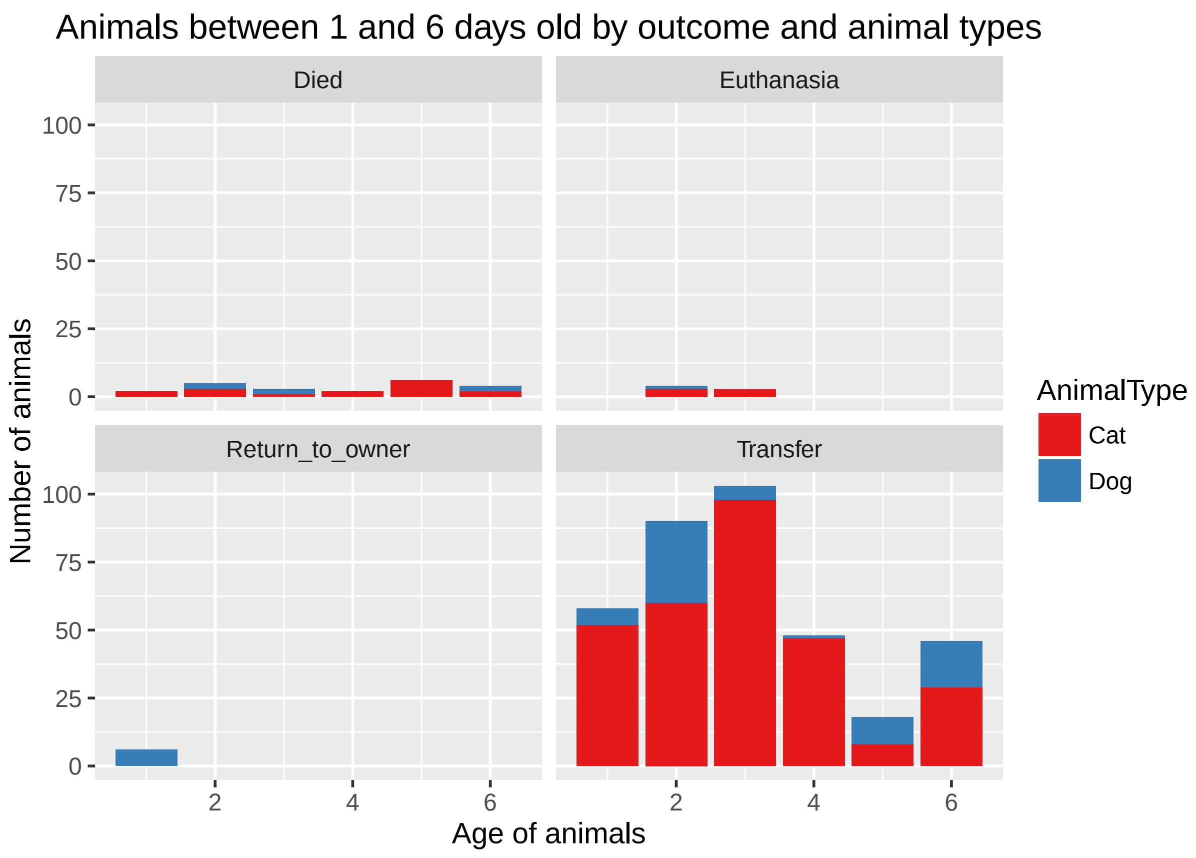
```r
test$AgeuponOutcome[test.year.list] <- test.age[test.year.list] * 365
test.month.list <- grep("month", test$AgeuponOutcome)
test$AgeuponOutcome[test.month.list] <- test.age[test.month.list] * 30
test.week.list <- grep("week", test$AgeuponOutcome)
test$AgeuponOutcome[test.week.list] <- test.age[test.week.list] * 7
test.day.list <- grep("day", test$AgeuponOutcome)
test$AgeuponOutcome[test.day.list] <- test.age[test.day.list]

train$AgeuponOutcome <- as.integer(train$AgeuponOutcome)
test$AgeuponOutcome <- as.integer(test$AgeuponOutcome)
```

Lets see now how the ages are distributed in function of the number of animals and outcomes.




Animals between 1 and 11 months old by outcome and animal types
Adoption
Died
Euthanasia
Return_to_owner
Transfer
Number of animals
0
1000
2000
3
6
9
Age of animals
AnimalType
Cat
Dog


Animals between 1 and 5 weeks old by outcome and animal types
Died
Euthanasia
Return_to_owner
Transfer
Number of animals
0
200
400
600
1
2
3
4
5
Age of animals
AnimalType
Cat
Dog

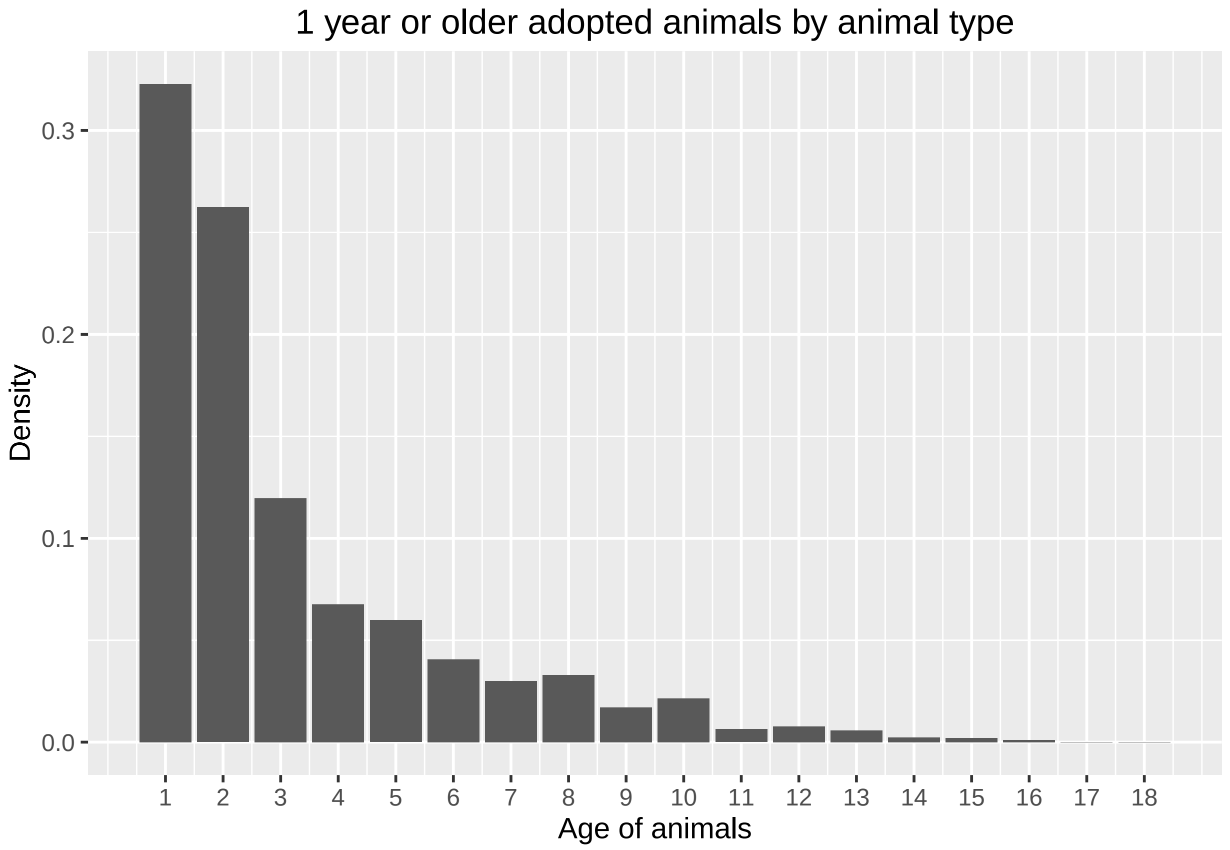


We can see 3 distinct age groups: days and weeks, months and years. The days and weeks group contains no adoptions and mostly transfers are done. We can state the following hypotheses from this group.

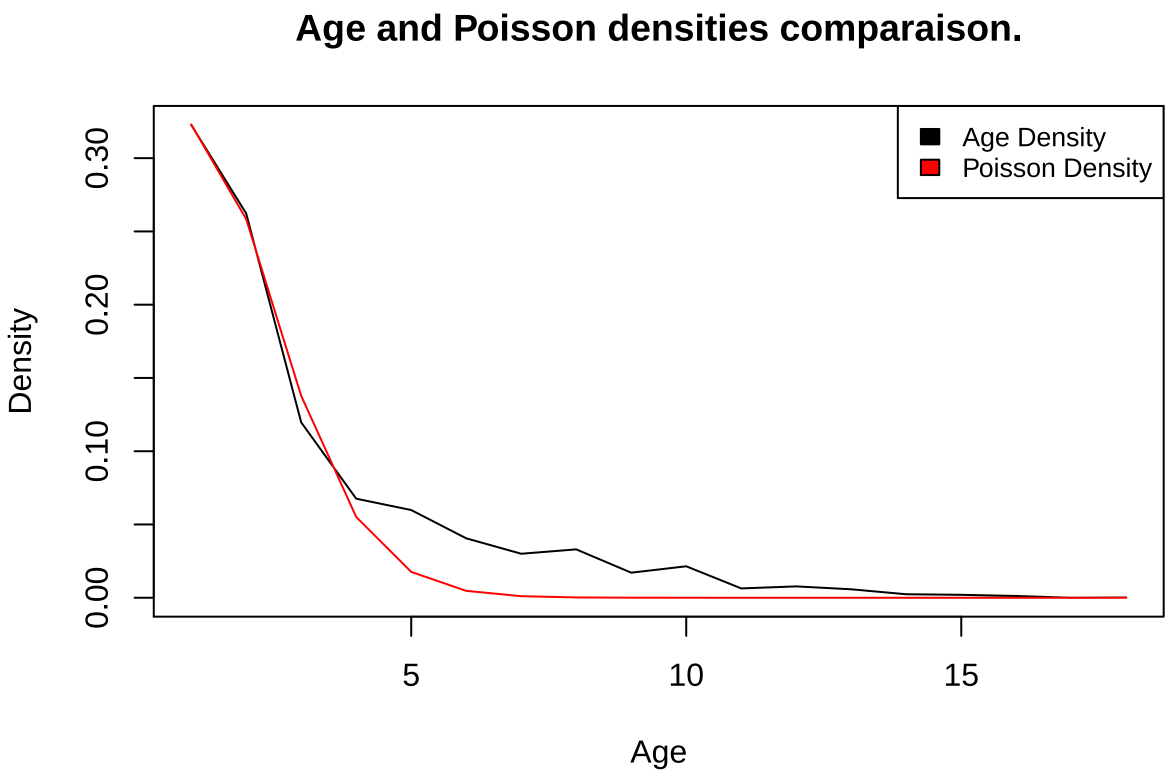
1. The Animal Center makes the adoption possible only when an animal is 1 month or older.
2. People do not want to adopt baby animals because they need to learn to their animal to be clean.
3. Animals get neutered or spayed starting at 1 month old.

The months group representing animals from 1 month to 11 months is clearly showing that 2 months old animals are mostly adopted.

The years group representing animals having 1 year old or more is described by the following density bar chart.



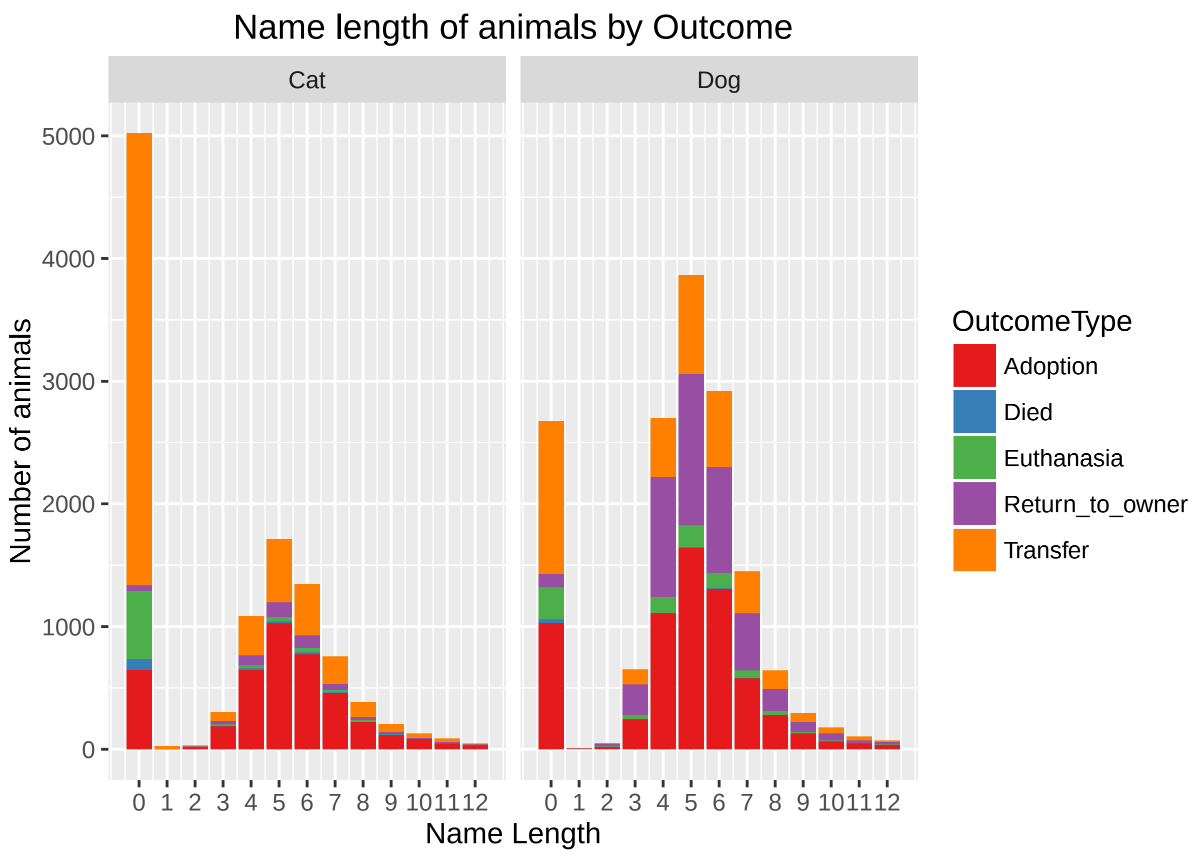
```
##    year_density       poisson_density
## 1  0.3227281766 0.3230344287914486689
## 2  0.2622787831 0.2584275430331589574
## 3  0.1197057069 0.1378280229510180643
## 4  0.0676078743 0.0551312091804072521
## 5  0.0598528534 0.0176419869377303189
## 6  0.0405647246 0.0047045298500614204
## 7  0.0300258501 0.0010753211085854665
## 8  0.0330085504 0.0002150642217170937
## 9  0.0171008153 0.0000382336394163721
## 10 0.0214754424 0.0000061173823066195
## 11 0.0063630941 0.0000008898010627810
## 12 0.0077550209 0.0000001186401417041
## 13 0.0057665540 0.0000000146018635944
## 14 0.0023861603 0.0000000016687844108
## 15 0.0019884669 0.0000000001780036705
## 16 0.0011930801 0.0000000000178003670
## 17 0.0000000000 0.0000000000016753287
## 18 0.0001988467 0.0000000000001489181
```



The discrete random variable Age is greater than 0 and ages are independant between each other. This means that age X cannot influance Age Y. Regarding the bar chart, it makes sense that the age approximately follows the Poisson distribution with lambda (mean) 1.6.

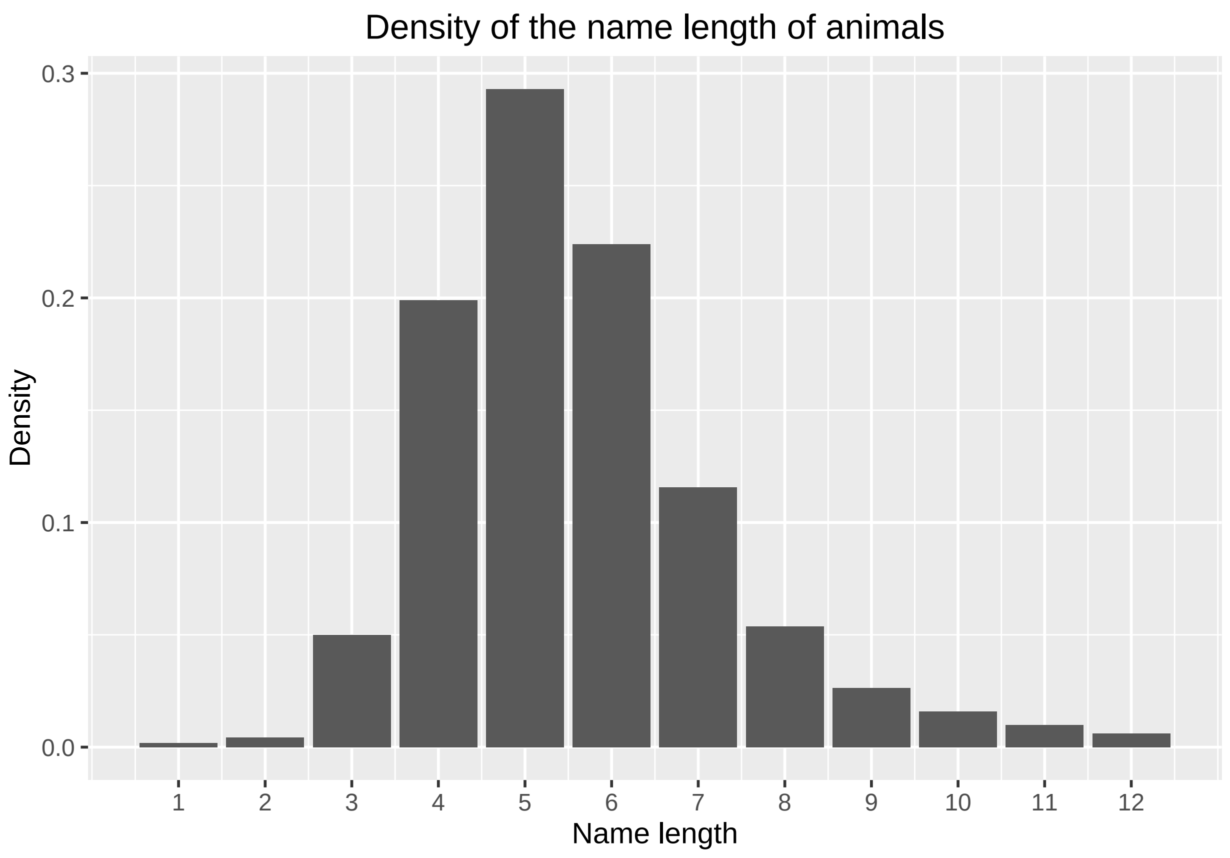
Animal's Name

The feature `Name` is transformed to a boolean value where the value is 0 when the animal has no name and 1 otherwise. Logically, the name of an animal should not have any impact on the outcomes. But knowing that an animal has no name versus has a name may influance the outcomes.

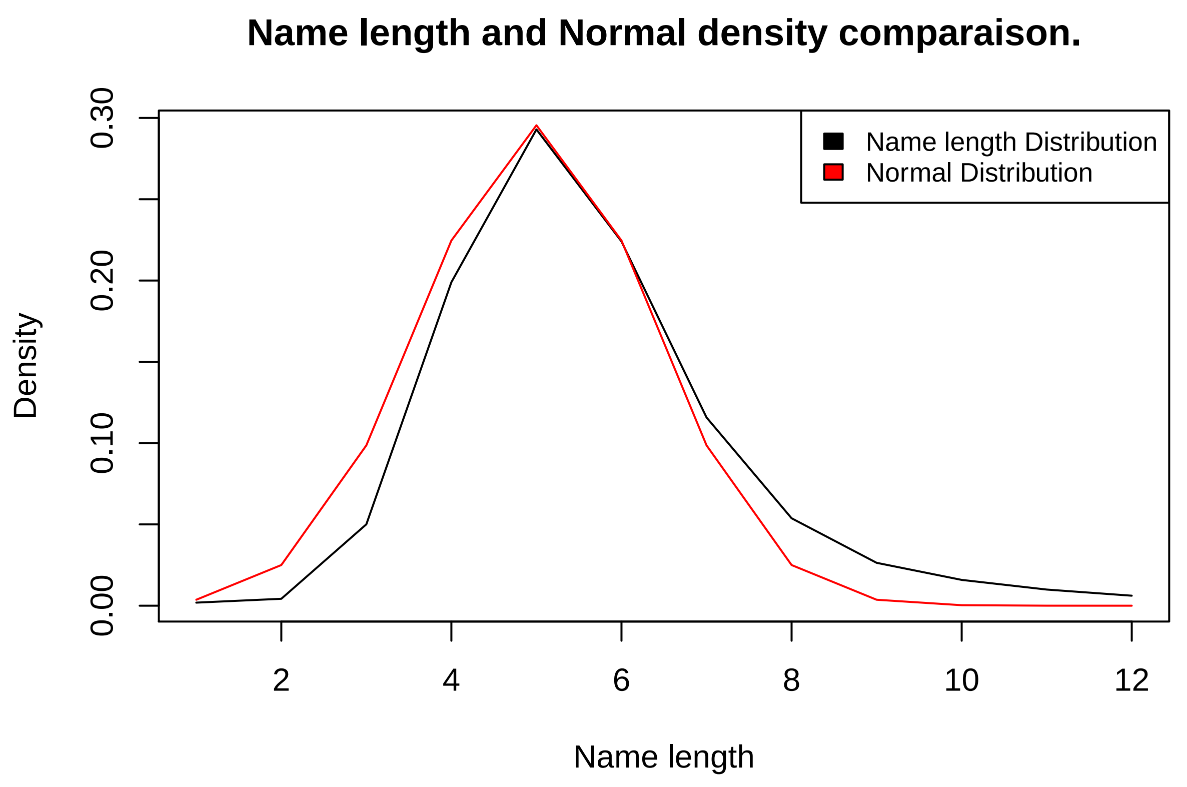


```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.000   5.000   5.585   6.000  12.000
```

From the bar chart, we can see that the name length follow the normal distribution if we exclude length of 0. In this context, animals having no name is possible, so it is not considered as an anomaly.



```
##    length_density  normal_density
## 1     0.001890955 0.0036661328728
## 2     0.004254649 0.0250175192854
## 3     0.050057779 0.0986243771037
## 4     0.198970480 0.2246095880476
## 5     0.292940435 0.2955128002974
## 6     0.224025633 0.2246095880476
## 7     0.115768463 0.0986243771037
## 8     0.053787163 0.0250175192854
## 9     0.026368316 0.0036661328728
## 10    0.015863011 0.0003103674350
## 11    0.009927513 0.0000151791720
## 12    0.006145604 0.0000004288684
```



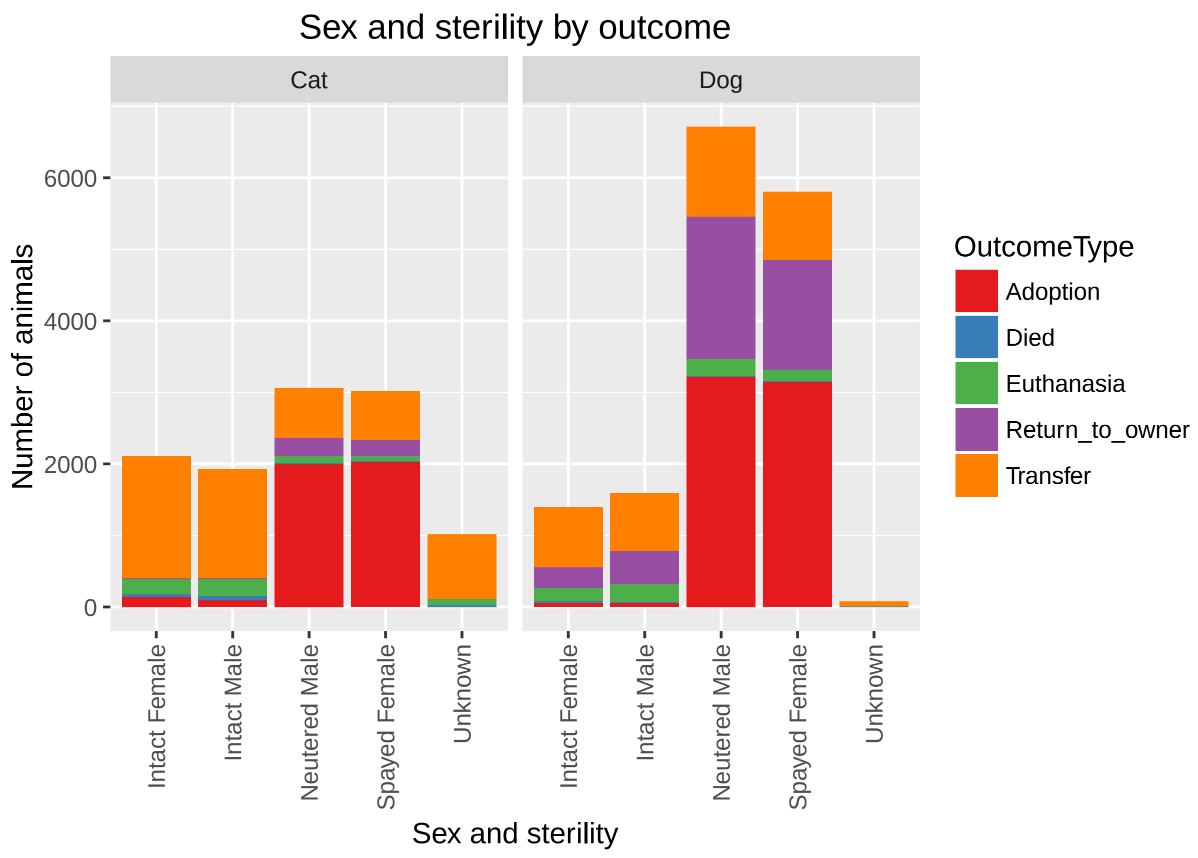
The density of the name length distribution is approximated by the Normal distribution. Mathematically, we define the random variable L as the name length of an animal such that $L \sim N(5, 1.35^2)$.

Animals having no name are mostly transfered but for cats, this is very significant. Transferred cats having no name represent 13.7790415 % of the train set and 35.2501029 % of all the transferred animals of the train set. This is clearly an insight to consider. Therefore, we transform string values of the `Name` feature in boolean values telling if the animal has a name = 1 or has no name = 0.

```
train$Name[train$Name != 0] <- 1
test$Name[test$Name != 0] <- 1
```

Animal's Sterility and Sex

We want to see if extracting information that check if the animal is sterile or not will have influance on the outcomes. Generally, people who want to adopt a dog or a cat want to know if the animal is sterile or intact. Let's see if this is true with our dataset.



We can see that sterile animals are mostly adopted which confirms our hypothesis. Unknown or intact animals are mostly transferred. This makes sense with unknown ones since they may need to be transferred to the clinic to identify clearly their sex and if they are sterile or not. Note that animals that we do not have information on their sterility and sex are not adopted. Thus, knowing if the animal is sterile or not has influance on the outcomes.

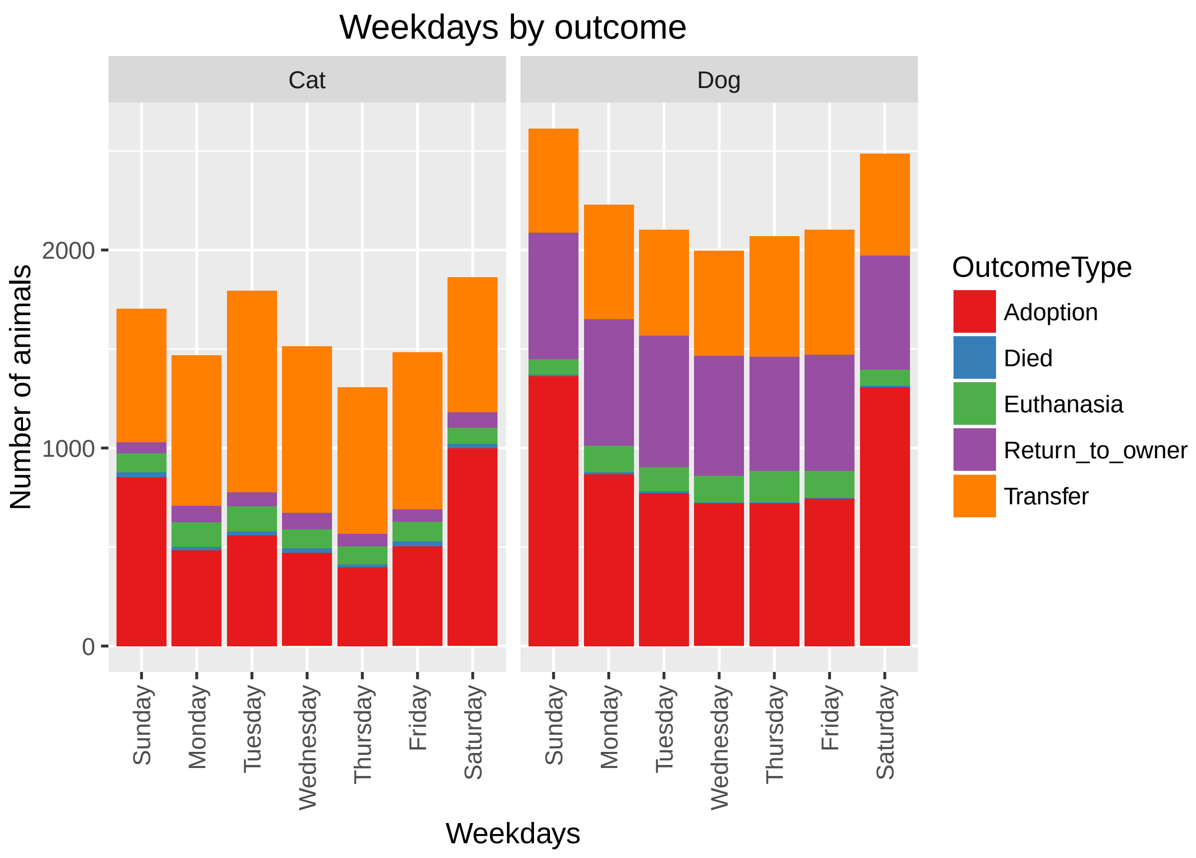
From this bar chart, we distinguish 3 groups: Unknown, Intact (not neutered or not spayed), Sterile (neutered or spayed).

We replace the feature `SexuponOutcome` by `Sterility` where Unknown = 0, Intact = 1 and Sterile = 2. Since there is no significant difference between sexes, we won't create a feature Sex.

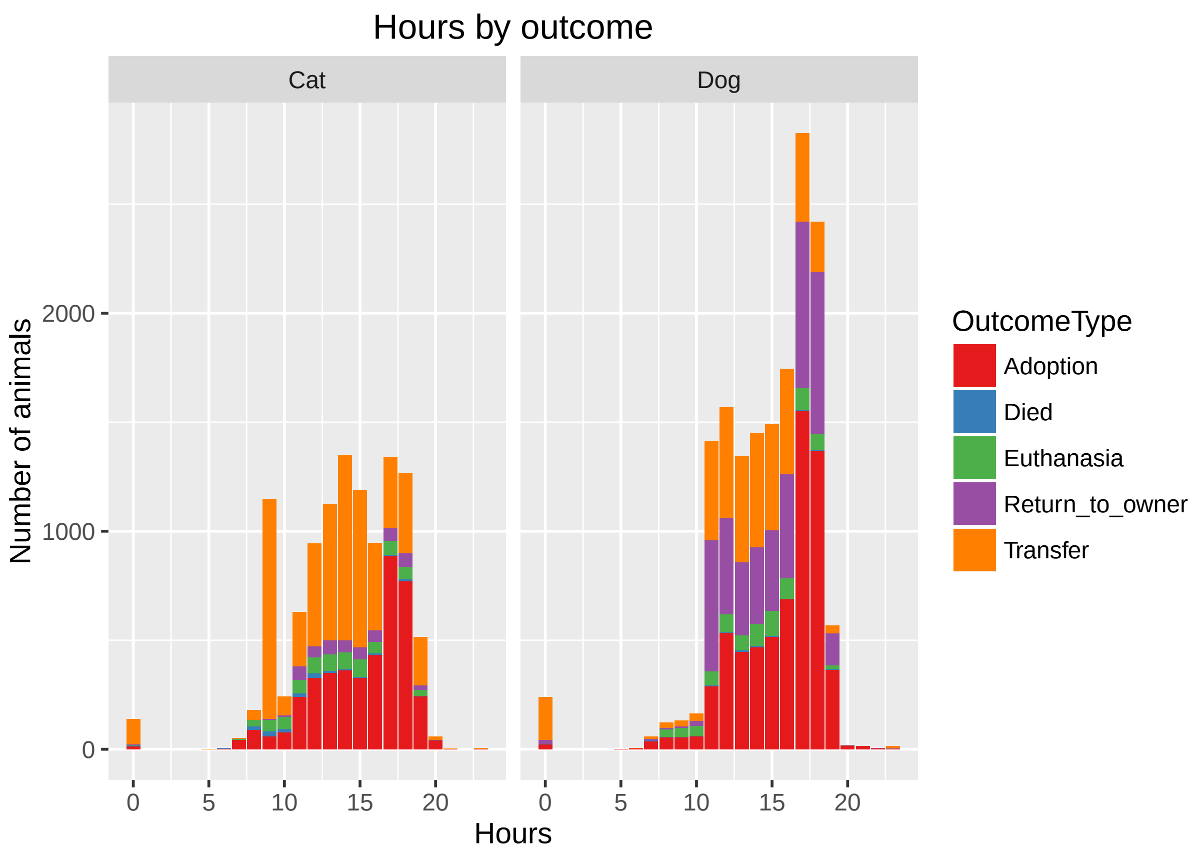
```
train$Sterility <- 0
train$Sterility[grep("Intact", train$SexuponOutcome)] <- 1
train$Sterility[grep("Spayed|Neutered", train$SexuponOutcome)] <- 2

test$Sterility <- 0
test$Sterility[grep("Intact", test$SexuponOutcome)] <- 1
test$Sterility[grep("Spayed|Neutered", test$SexuponOutcome)] <- 2

train$SexuponOutcome <- NULL
test$SexuponOutcome <- NULL
```

Date & Time

The date and time may have a big influance on the outcomes. We have seen that adoptions represent the most popular outcome of the dataset. Our hypothesis is that people will mostly adopt an animal the weekend.

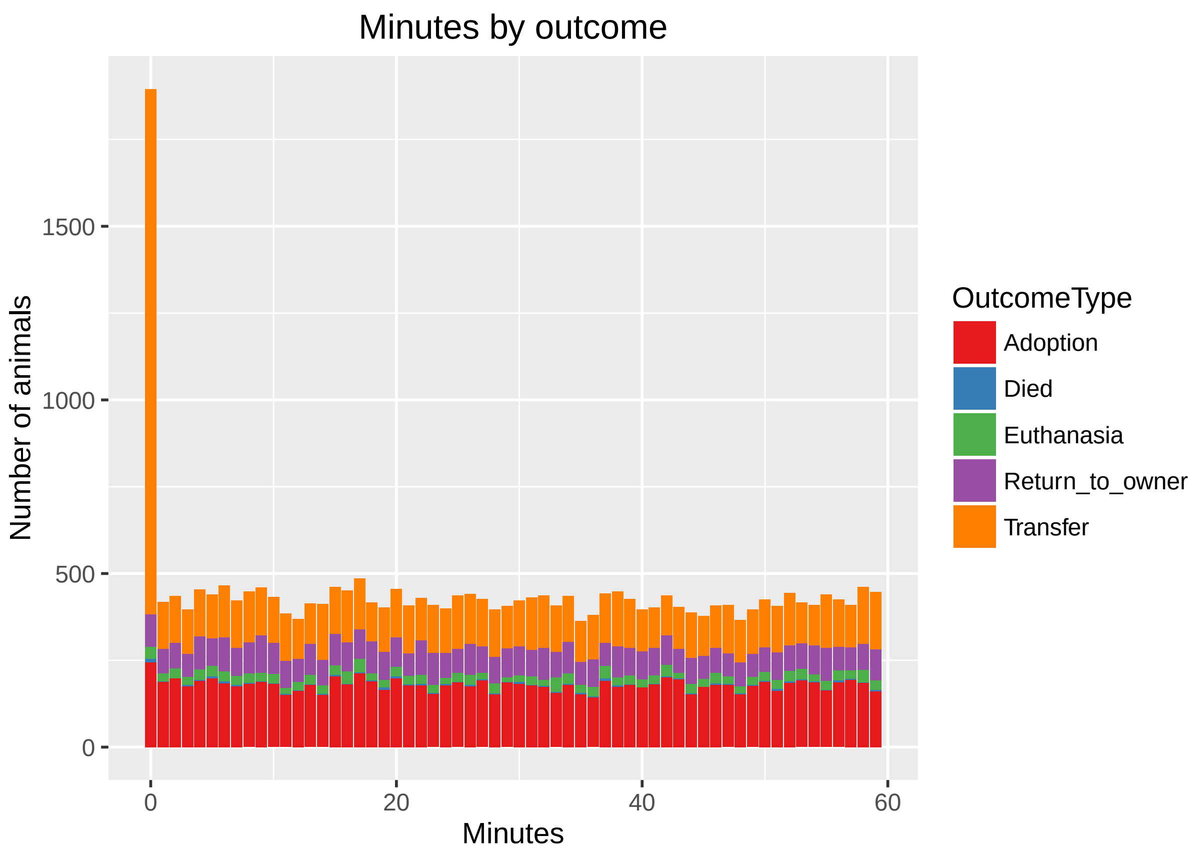


From the bar chart, we can see that animals are mostly adopted the weekend (Saturday and Sunday) which confirm our hypothesis. This makes sense since most of people are working from Monday to Friday. Looking at the website Austin Animal Center, animal adoption claims are from 11am - 7pm every day. From these information, we suppose that extracting the hour from the `DateTime` feature could have influance on the outcomes.



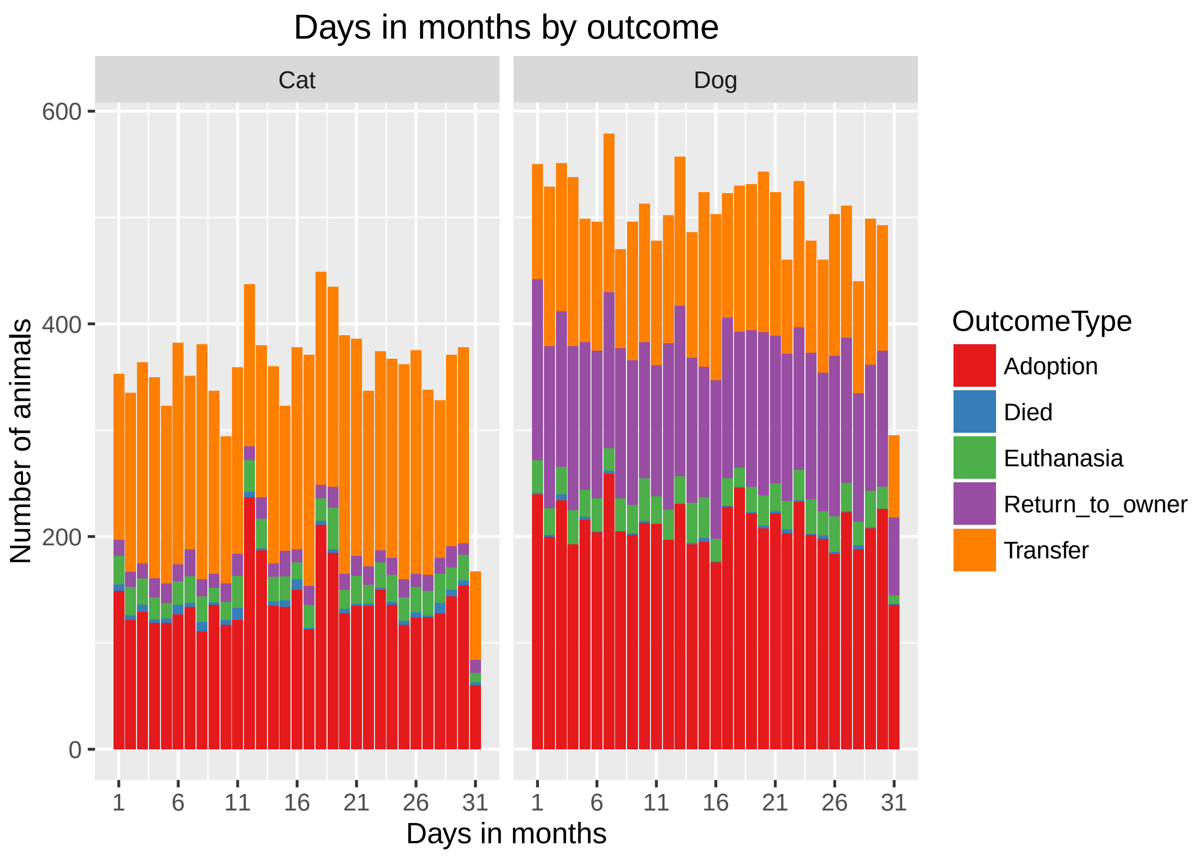
From the bar chart, we see that adoptions and returns to owner occur at 5pm and 6pm. Generally, people finish working around 4:30pm - 5pm so this makes sense. There are no outcomes done between 1am and 4am inclusively. There are transfers done at midnight, and at 9am for cats mostly, but most of them are done between 11am and 7pm which correspond to the open hours of the center. For adoptions and transfers done out of open hours, it may be the following hypothesis:

1. They are done for exceptional circumstances.
2. Error when entering the hours in the system.



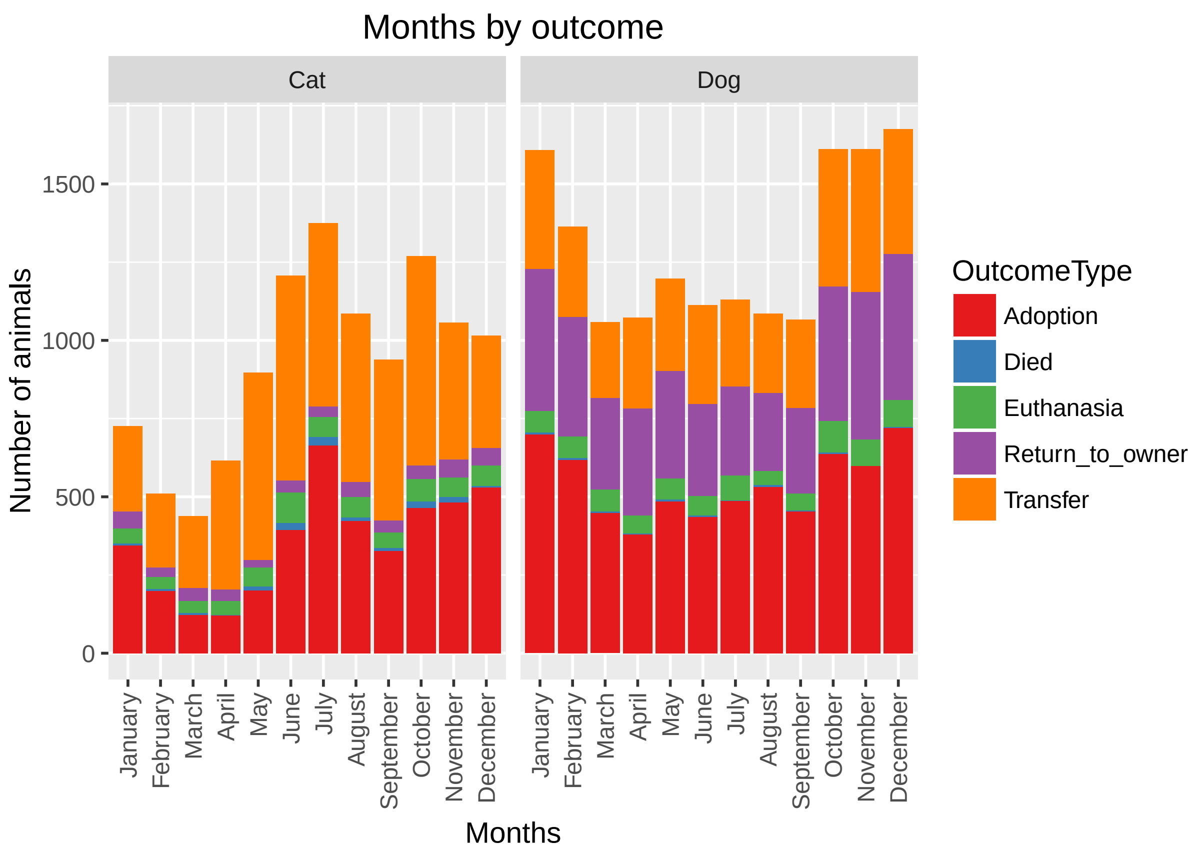
Looking at minutes seems useless since 1 minute is very short and most of us will look at the hour only except if the software records also the minutes and even the seconds. But, from this bar chart, we can see that at 0 minute, there are a lot of transfers done compared to other minutes. From 1 to 59 minutes, the difference is negligible since it approximately follows a uniform distribution as expected. This could be explained by an automatic setting if the user does not enter the minutes when recording a new outcome with the software. The software convert empty minutes by “00” as well as the seconds. Since this is recurrent for transfers at 9:00am, it could be the same employee that records these transfers. This is also the case if the user enters only the date. The hours may be set automatically to 00:00:00. Therefore, it is possible that some entries concerning hours and minutes are not correct.

Looking at the days, we should see more adoptions the weekend since the majority of people are not working.

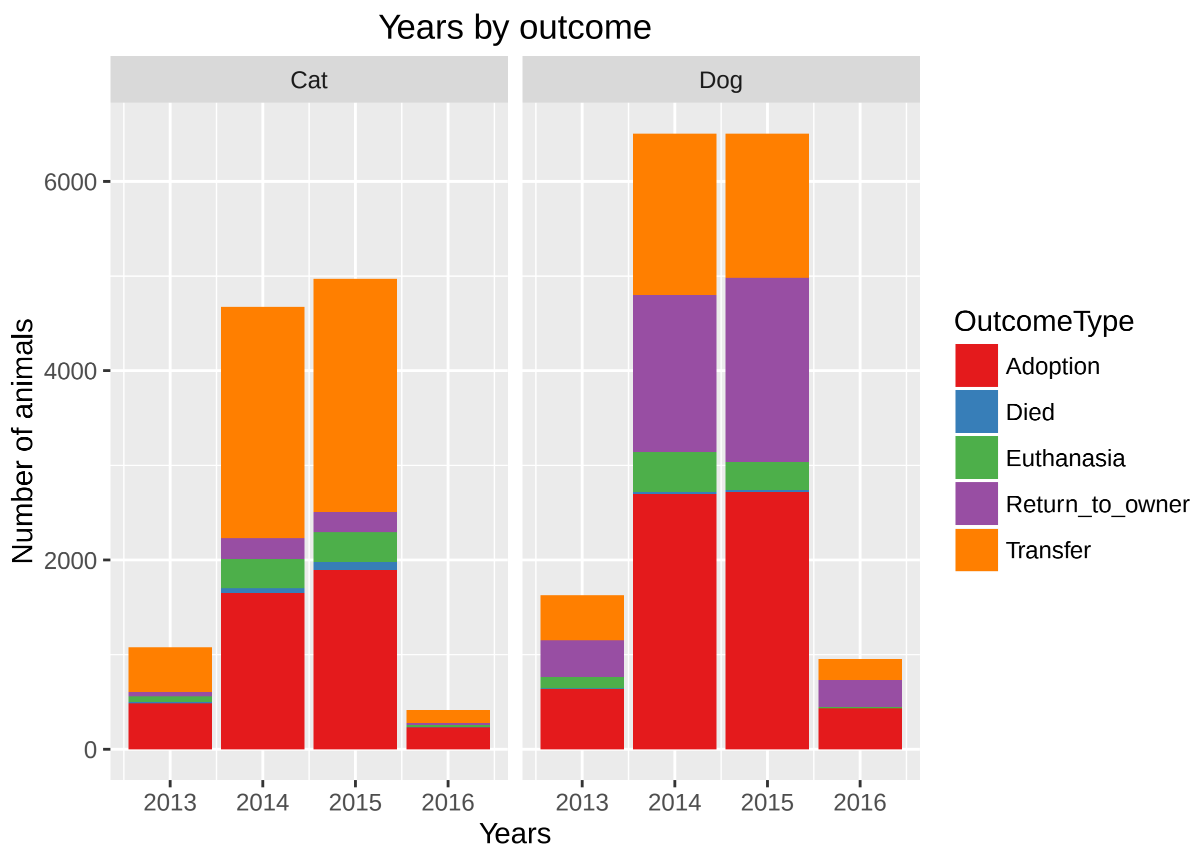


There are 4 days (12, 13, 18, 19) where the number of adopted cats are higher than the other days. Also, note that there are less animals on the day 31. Indeed, since there are only 7 months over 12 having 31 days, this result makes sense.

We look now at the months to check if some of them can reveal important insights.



From the bar chart, we can see that December and January are months having highest number of adoptions of dogs. for the cats, July and December represent the highest number of adoptions. This can be explained by holidays like Christmas and the New Year's day. One possible hypothesis is that children want a dog or a cat for Christmas so they ask their parents to adopt an animal and give them as a Christmas gift.



Years 2013 and 2016 have less outcomes than 2014 and 2015 since the dataset dates start on 2013-10-01 09:31:00 and end on 2016-02-21 19:17:00.

We did not mention the seconds since there is 0 animal where the seconds are different to 0.

From the feature `DateTime` of the dataset, we extract many useful features. Those features are transformed to positive integers. The features we extract are the following:

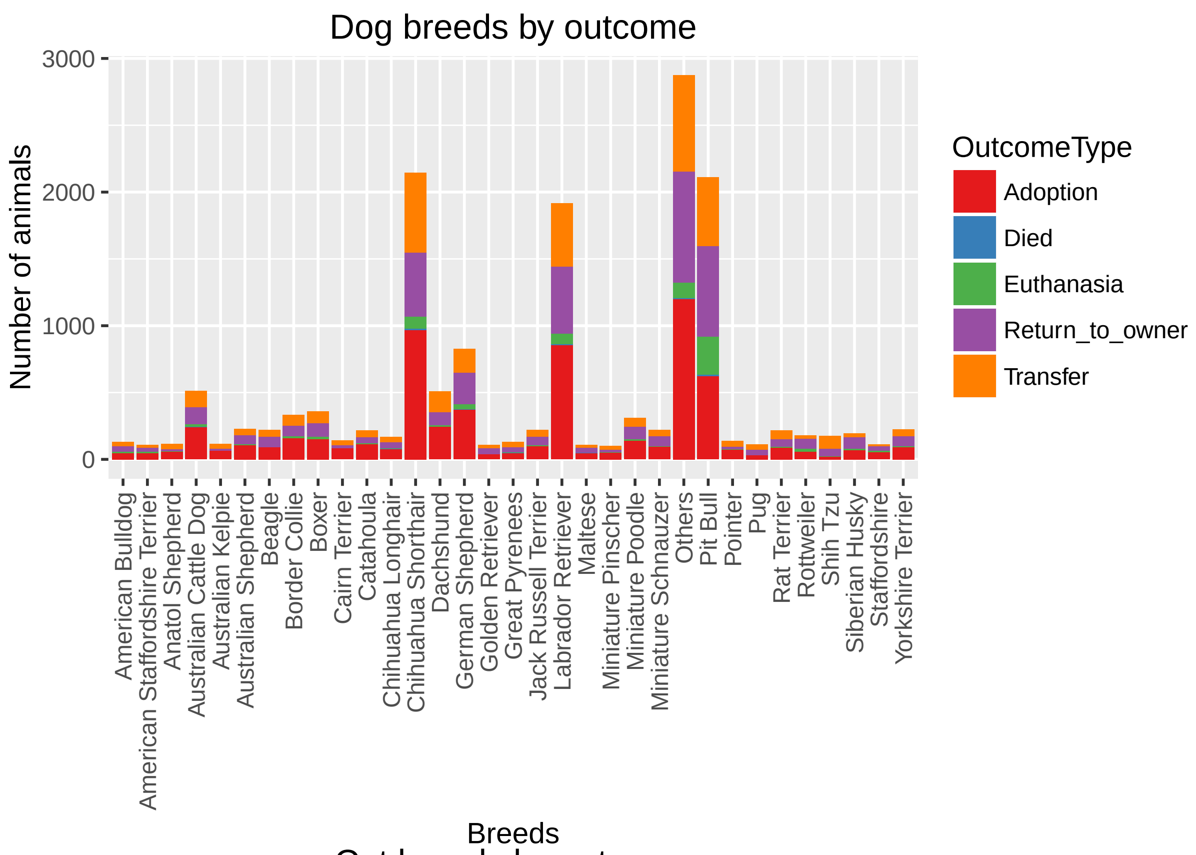
- Day: integer from 1 to 31 depending on the month
- Month: integer from 1 to 12
- Year: integer from 2013 to 2016
- Weekday: integer from 0 to 6 which represent Sunday to Saturday
- DateInDays: Number of days from the oldest date of the dataset
- Hour: integer from 0 to 23 where 0 = midnight
- Time: The time represented by the equation $60h + m$ where h is the hours and m the minutes.

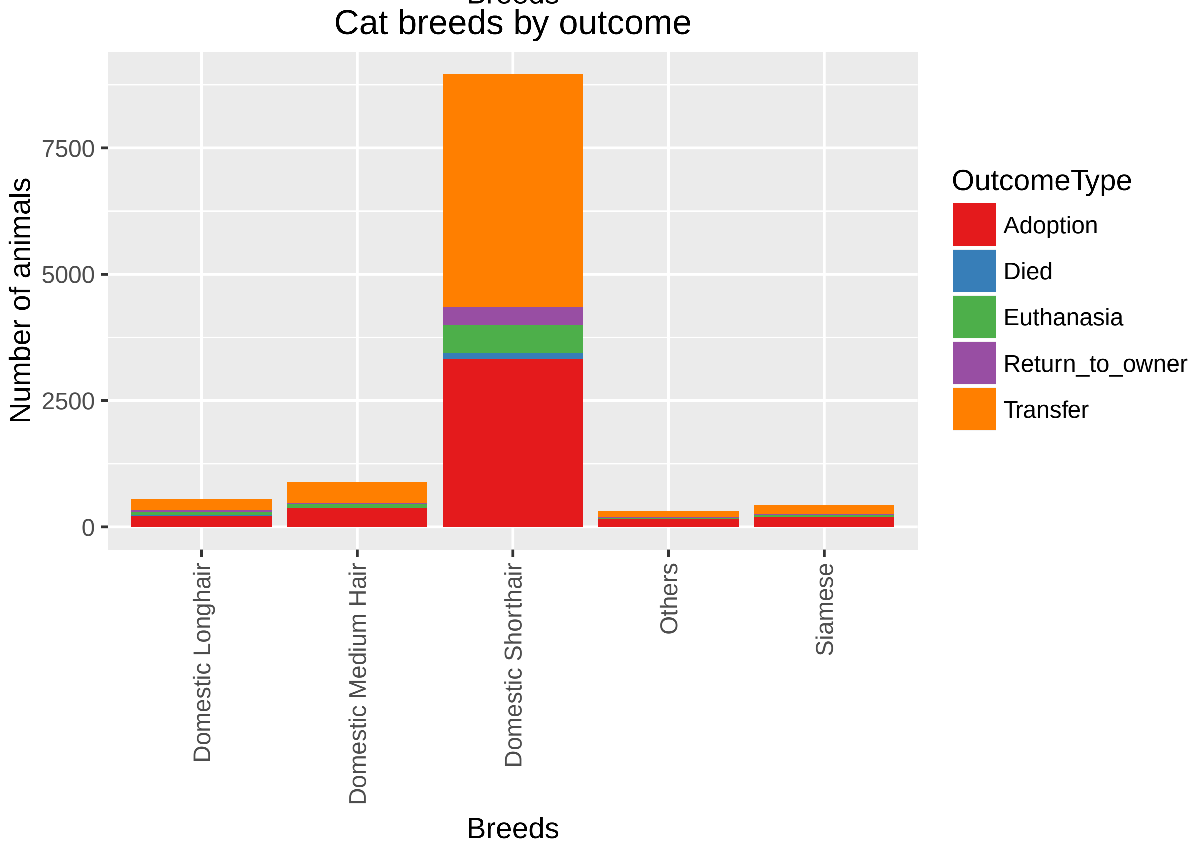
```
train$Weekday <- as.POSIXlt(train$DateTime)$wday
test$Weekday <- as.POSIXlt(test$DateTime)$wday

train.date <- ymd_hms(train$DateTime)
test.date <- ymd_hms(test$DateTime)

## The year starts at 0 which is the minimum year of the dataset.
train$Year <- year(train.date)
train$Month <- month(train.date)
train$Day <- day(train.date)
train$Hour <- hour(train.date)
```

```
train$DateInDays <- difftime(as.Date(train.date,'%Y/%m/%d'),
                             as.Date(min(train.date),'%Y/%m/%d'),
                             units = c("days"))
train$Time <- hour(train.date) * 60 + minute(train.date)

test$Year <- year(test.date)
test$Month <- month(test.date)
test$Day <- day(test.date)
test$Hour <- hour(test.date)
test$DateInDays <- difftime(as.Date(test.date,'%Y/%m/%d'),
                            as.Date(min(test.date),'%Y/%m/%d'),
                            units = c("days"))
test$Time <- hour(test.date) * 60 + minute(test.date)

train$DateTime <- NULL
test$DateTime <- NULL
```

Animal's Breed

Since we have many breeds, we will create a breed class named 'Other' where breeds having less than 100 animals will be in this class. The breed should be important since people do not want to adopt an agressive animal for example.



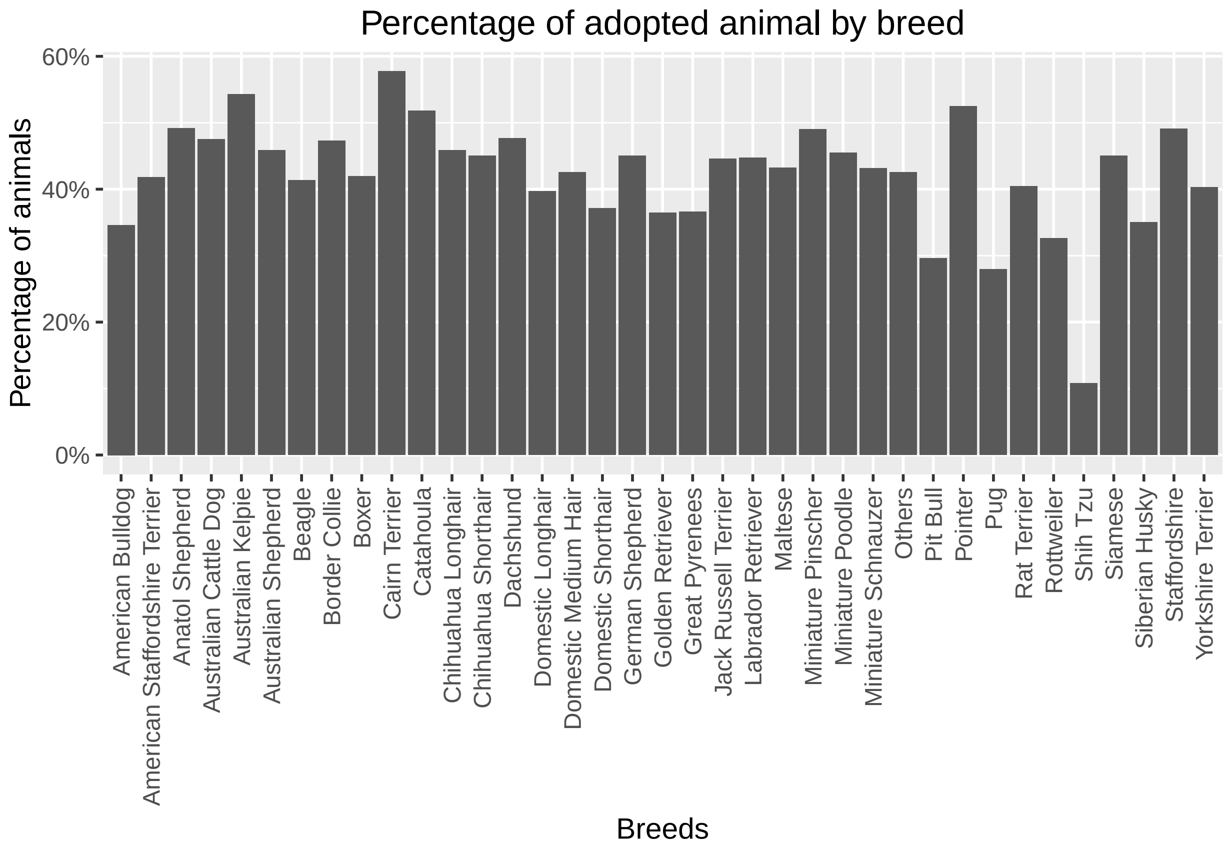


For the dogs, Shih Tzu are mostly transferred and less adopted. Pit bulls, Labrador Retreiver and Chihuahua

Shorthair are the most common dogs for outcomes. For cats, the domestic shorthair is the most frequent and represent 80.5730196 % of the cats in the train set.

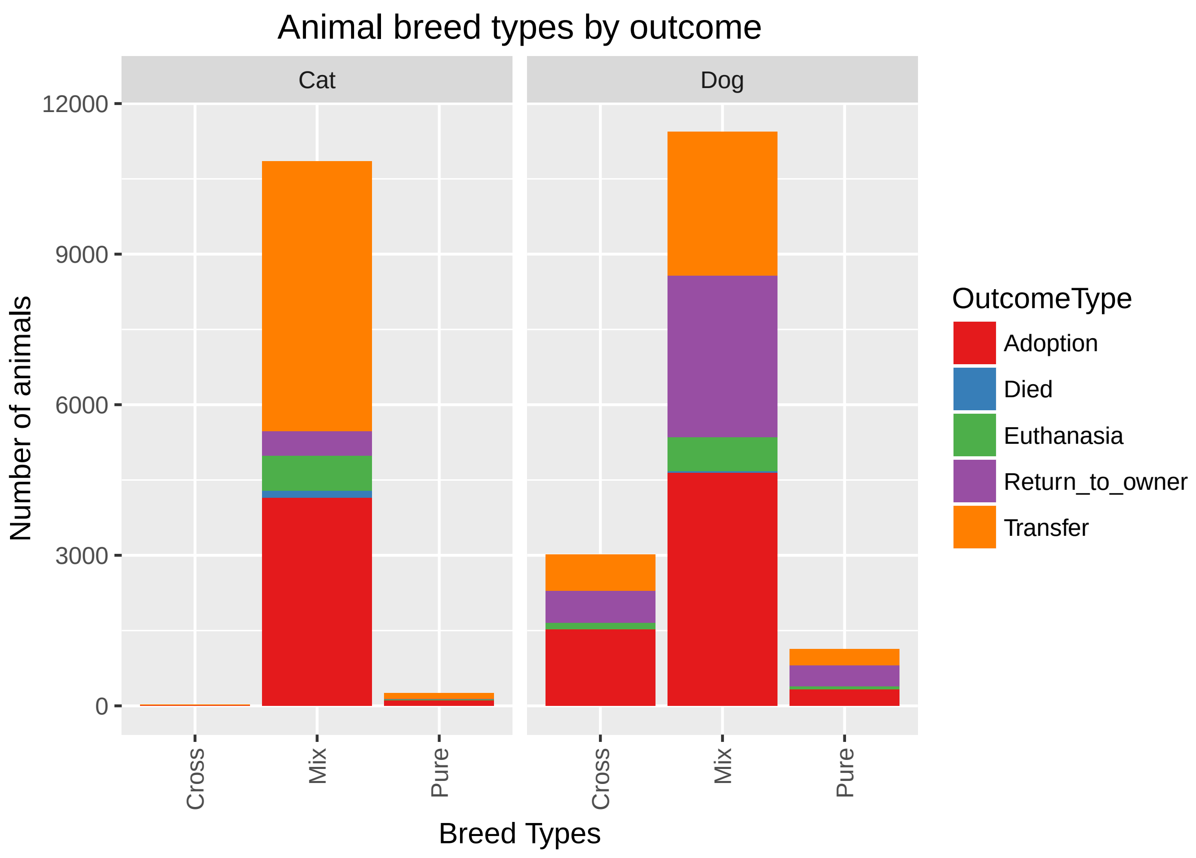
Lets summarize the number of adopted animals and their percentage grouped by the breed. In this way, we will see which animals have the lowest ratio for adoption.

```
## Source: local data frame [37 x 4]
## Groups: Breed1 [37]
##
##               Breed1 OutcomeType Total Percentage
##                (chr)       (chr) (int)      (dbl)
## 1           Shih Tzu    Adoption    19  0.1079545
## 2                Pug    Adoption    31  0.2792793
## 3           Pit Bull    Adoption   625  0.2957880
## 4         Rottweiler    Adoption    59  0.3259669
## 5    American Bulldog    Adoption    45  0.3461538
## 6      Siberian Husky    Adoption    68  0.3505155
## 7    Golden Retriever    Adoption    39  0.3644860
## 8      Great Pyrenees    Adoption    48  0.3664122
## 9  Domestic Shorthair    Adoption  3328  0.3715115
## 10  Domestic Longhair    Adoption   217  0.3967093
## ..                ...         ...   ...        ...
```



From the bar chart and summary, we see that Shih Tzu, Pug, Pit Bull are breeds with lowest percentage of adoption.

We also check if the breed types (Mix, Cross or Pure) have a significant impact on the outcomes.



We extract this information from the `Breed` feature and add a new feature `BreedType` where all purebred animals are identified with the value 0. The mixed breed are identified with the value 2 and crossed breed with value 1.

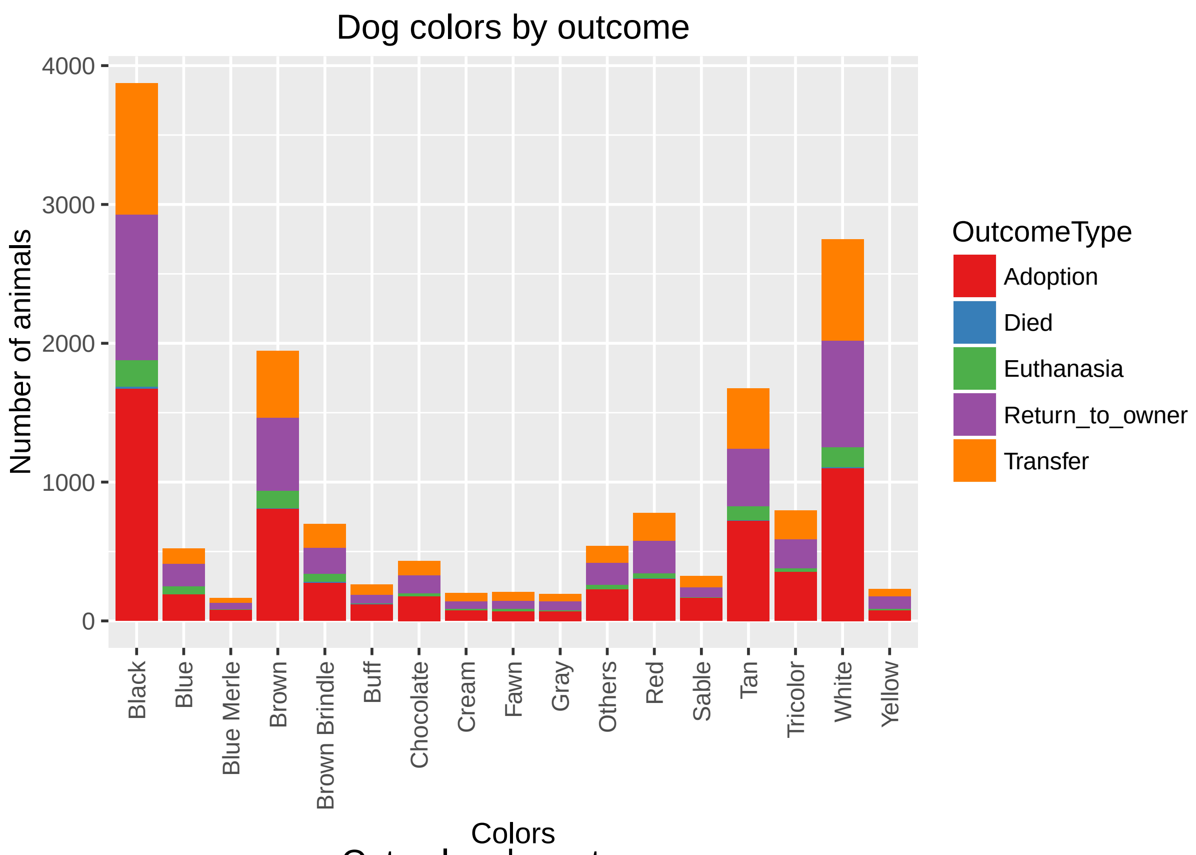
```
train$BreedType <- 0
train$BreedType[train.breed.cross.list] <- 1
train$BreedType[train.breed.mix.list] <- 2

test.breed.mix.list <- grep(" Mix", test$Breed)
test.breed.cross.list <- grep("/", test$Breed)
test$BreedType <- 0
test$BreedType[test.breed.cross.list] <- 1
test$BreedType[test.breed.mix.list] <- 2
```

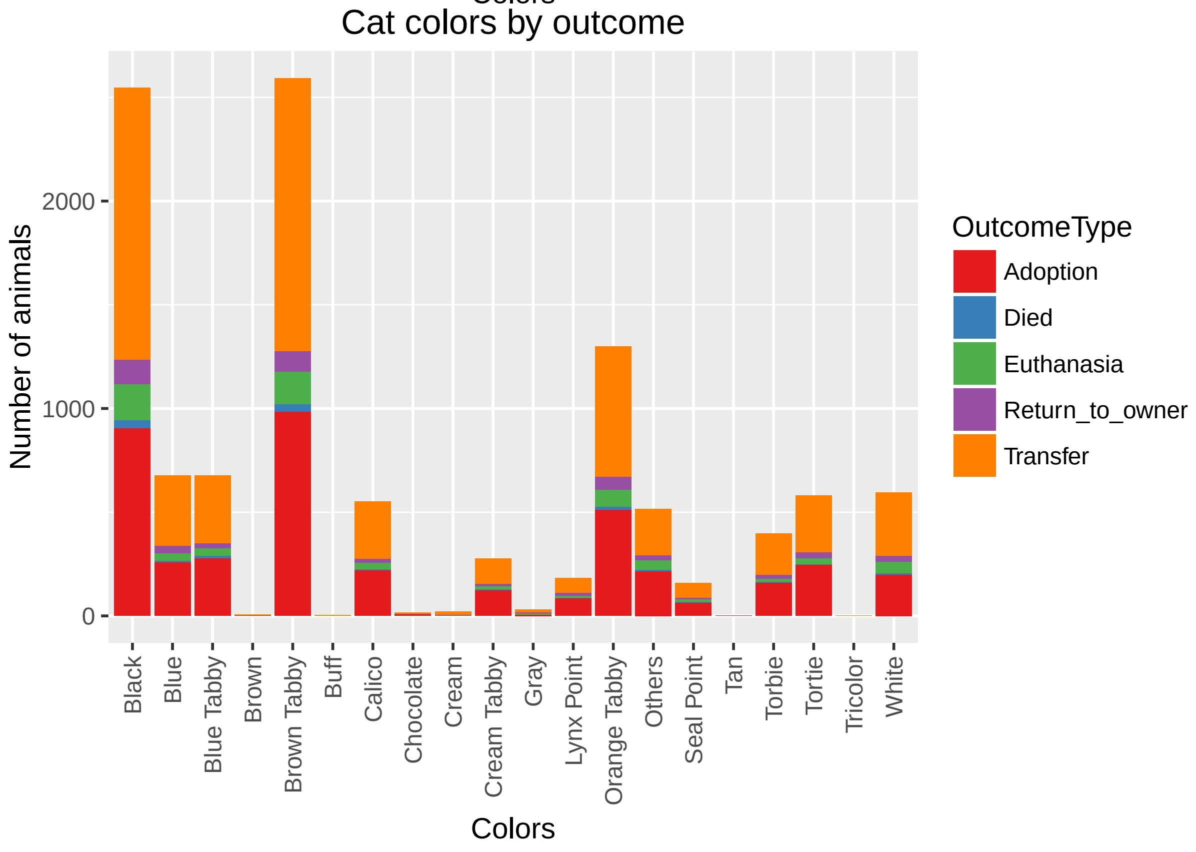
Special cases with Pit Bull, Shih Tzu and Pug are considered since they are less adopted than the others.

```
breeds <- c("Pit Bull", "Shih Tzu", "Pug")
train$CommonBreeds <- GetIntegerFeatureFromGroups(train$Breed, breeds)
test$CommonBreeds <- GetIntegerFeatureFromGroups(test$Breed, breeds)

train$Breed <- NULL
test$Breed <- NULL
```

Animal’s Color

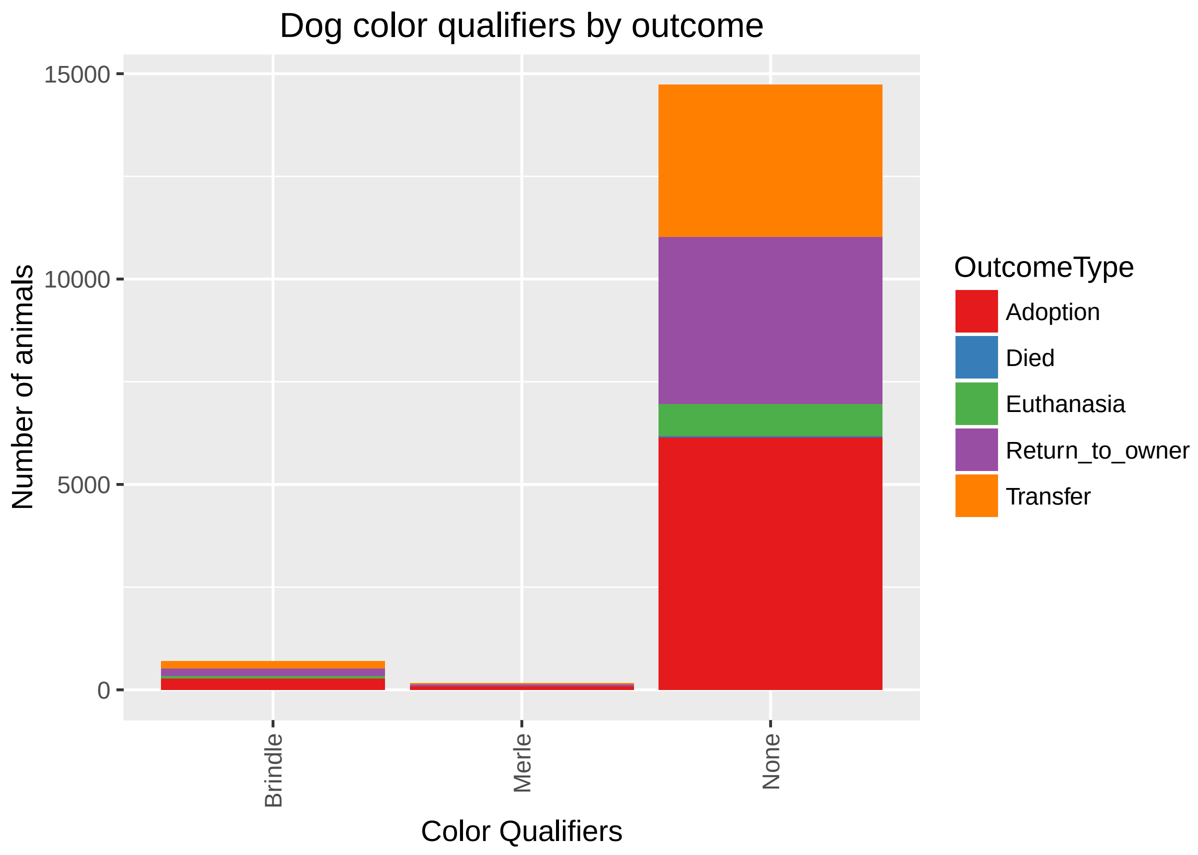
For each outcome type, we want to know which animal’s color has the greatest and lowest count. For colors having less than 100 animals, we categorize them as `Others` meaning that these colors are not common.

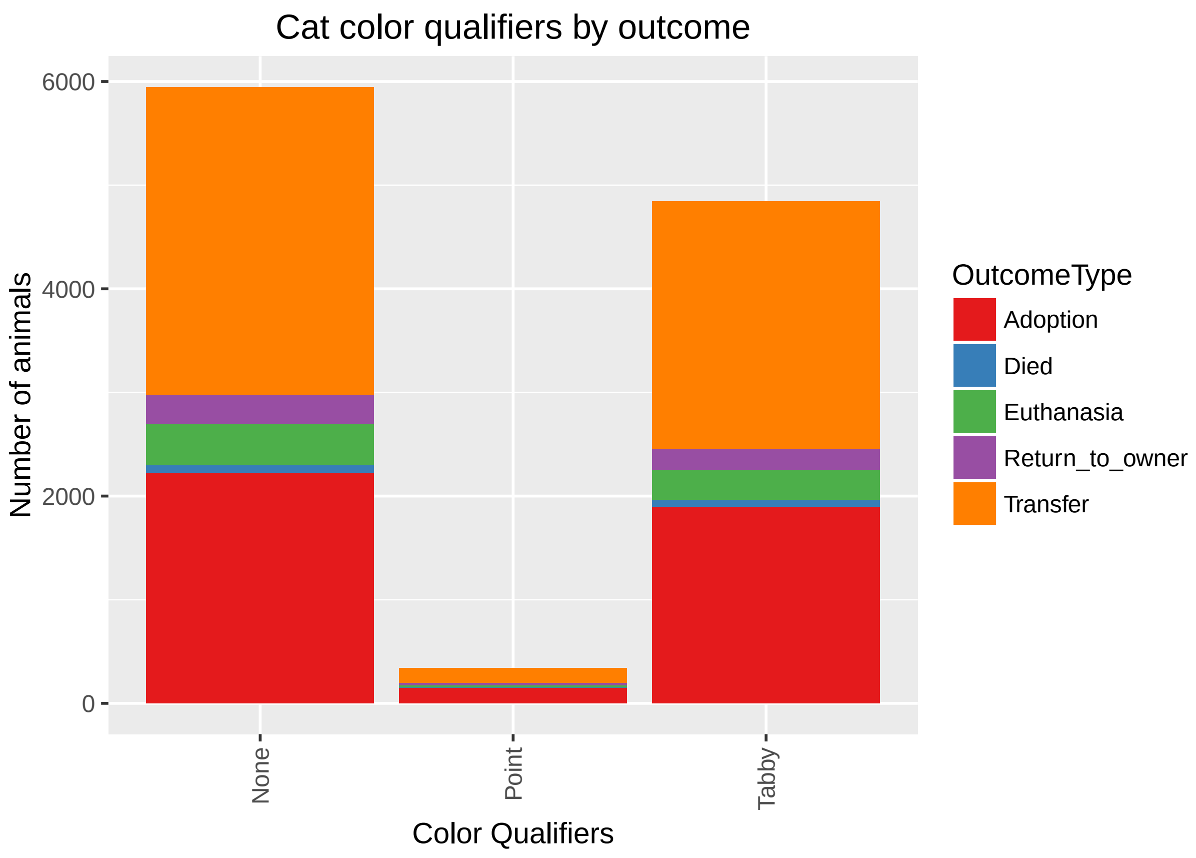




For the cats, we can see that tan cats are always adopted and tricolor cats are always transferred. Another important insight is that brown cats have not died, are not euthanasied or are not returned to their owner. However, the number of cats for of those colors is not big. Black color is the most popular for cats and dogs following this bar chart.

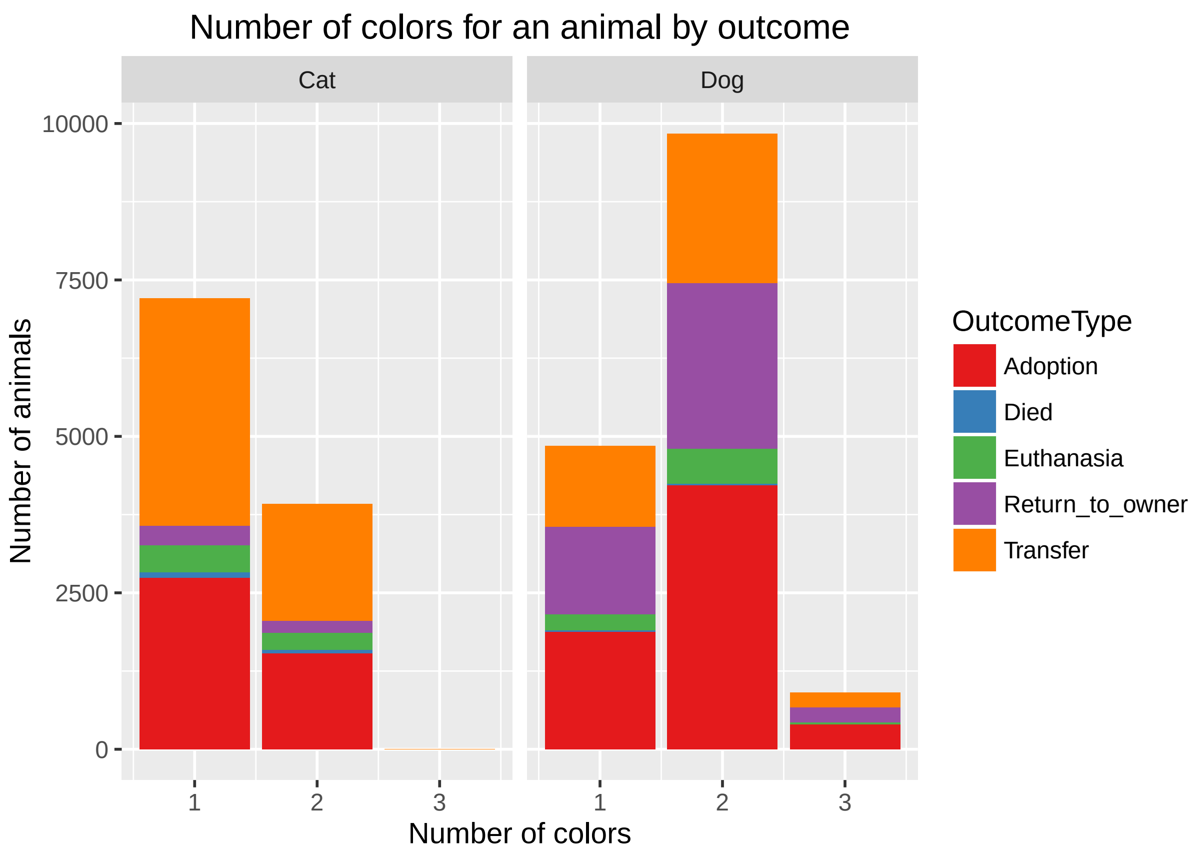
Some of colors have a qualifier but these should not have big influance on the outcomes. We take the feature `Color1` to determine the qualifiers and see what happens.





We can verify with the bar chart above shows that qualifiers has no significant influance on the outcomes for dogs. For cats, tabby is frequent.

With the color feature, we can extract the number of colors. If we find a slash `/` character which we define as a `separator`, then we have 2 colors. If we find `Tricolor`, then this means that we have 3 colors.



Modeling and Interpretation

In this section, we present what algorithm is used, why are we using it and how it is used. The objective is to build the final submission file containing the probabilities for each outcome type of each animal given in the test set.

Extreme Gradient Boosting Trees is a good choice to modelize our data. Each tree is a decision tree representing a group of animals where the probabilities of each outcome are approximately the same. For example, first level could be if the animal is sterile, intact or unknown. The second level could be if an animal has a name or not.

Fine Tuning Parameters

We prepare the parameters and matrices for the cross-validation and final prediction. Since we need to get the probabilities for each class, then the objective used is `multi:softprob`. This metric requires the number of classes `num_class` to predict which is 5 in our case. We have found that the evaluation metric used is the multi-class log loss function which is `mlogloss` for the `XGBoost` algorithm.

We use the `subsample` parameter which is the subsample ratio of the training instances. The parameter `eta` control the learning rate since the XGBoost uses the gradient descent which needs a learning rate parameter.

```r
outcomes <- unique(train$OutcomeType)
outcome.type <- CategoryToInteger(train$OutcomeType) - 1

train$OutcomeSubtype <- NULL
```

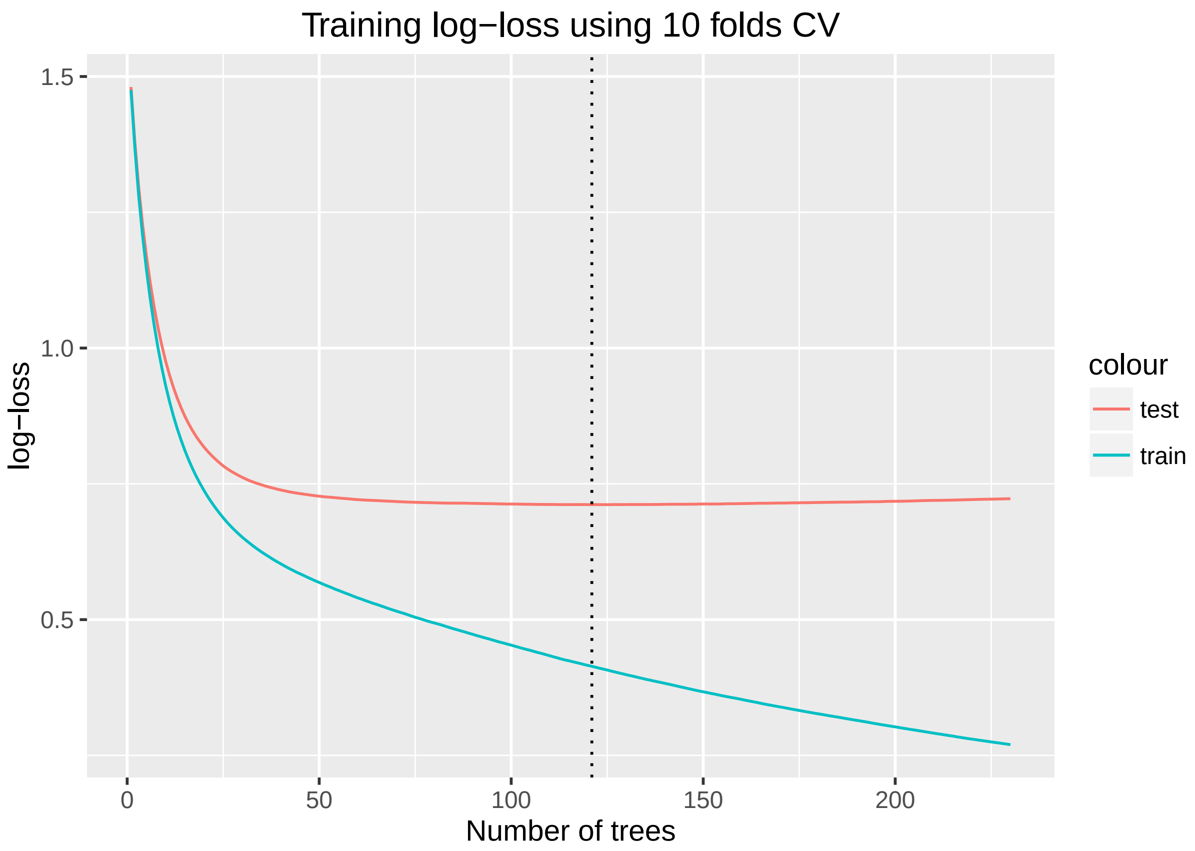
```r
train$OutcomeType <- NULL

outcome.class.num <- length(outcomes)
param <- list(objective        = "multi:softprob",
              num_class        = outcome.class.num,
              eta              = 0.12,   # Control the learning rate
              subsample        = 0.8,
              max_depth        = 8,
              colsample_bytree = 0.9,
              eval_metric      = "mlogloss")
```

Cross-Validation

We proceed to a 10-fold cross-validation to get the optimal number of trees and multi-class log-loss score. We use randomly subsamples representing 80% of the training set. Since we don't have too many observations (animals) and features, we can use 10 folds instead of 5 folds. Thus, the training set will be split in 10 test samples where each test sample has 2672.9 observations.

For each tree, we will have the average of 10 error estimates to obtain a more robust estimate of the true prediction error. This is done for all trees and we get the optimal number of trees to use for the test set.

We also display 2 curves indicating the test and train multi-Logloss mean progression. The vertical dotted line is the optimal number of trees. This plot shows if the model overfit or underfit.



```
##      train.mlogloss.mean train.mlogloss.std test.mlogloss.mean
##   1:            1.475111           0.003918           1.480649
##   2:           1.368849          0.003043          1.379403
##   3:           1.280670          0.002463          1.296582
##   4:           1.207875          0.003879          1.228346
##   5:           1.145335          0.005523          1.170455
##  ---
## 226:           0.273856          0.002023          0.721984
## 227:           0.272875          0.002011          0.722153
## 228:           0.271873          0.002015          0.722307
## 229:           0.270858          0.001929          0.722459
## 230:           0.269838          0.001999          0.722582
##      test.mlogloss.std names
##   1:          0.004783     1
##   2:          0.004409     2
##   3:          0.004844     3
##   4:          0.005615     4
##   5:          0.006325     5
##  ---
## 226:          0.019219   226
## 227:          0.019200   227
## 228:          0.019143   228
## 229:          0.019240   229
## 230:          0.019318   230
```

```
##
## Optimal testing set Log-Loss score: 0.711664
```

```
##
## Associated training set Log-Loss score: 0.41399
```

```
##
## Interval testing set Log-Loss score: [ 0.695175 ,  0.728153 ].
```

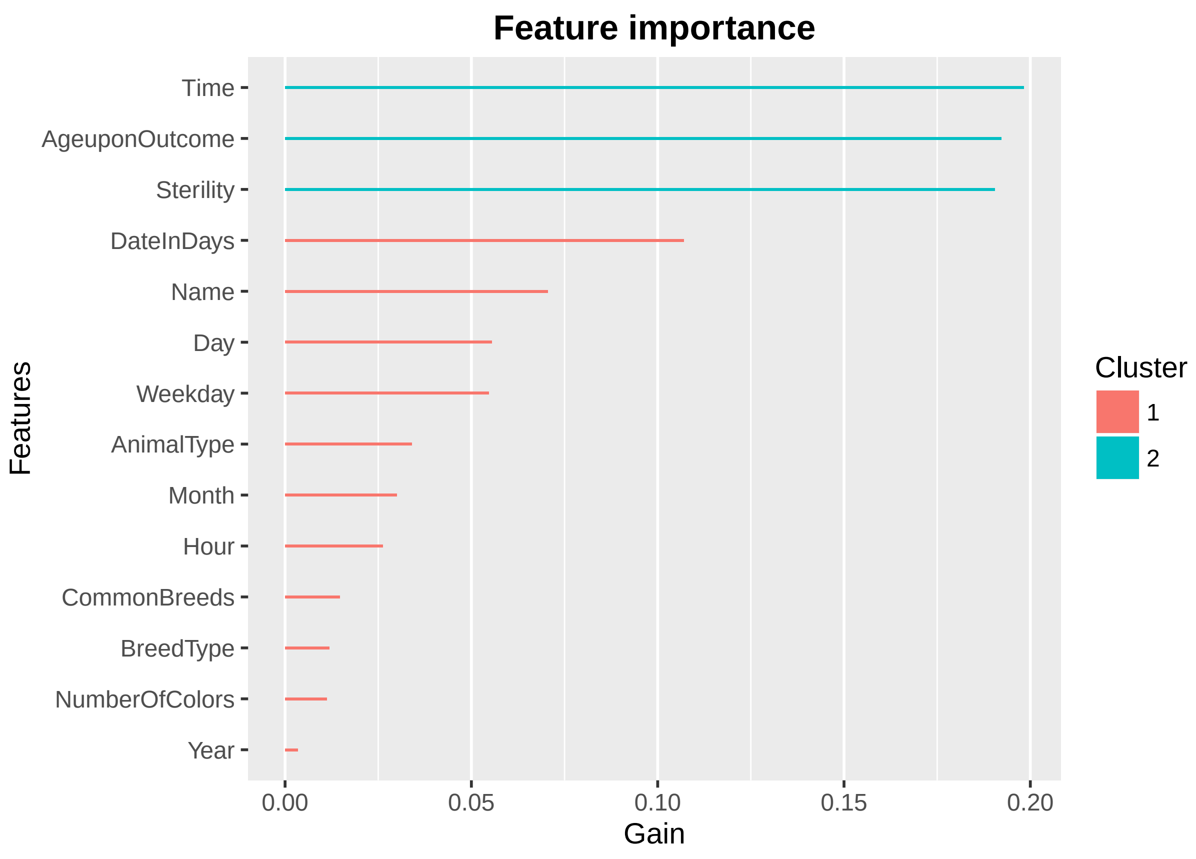
```
##
## Difference between optimal training and testing sets Log-Loss: 0.297674
```

```
##
## Optimal number of trees: 121
```

Prediction

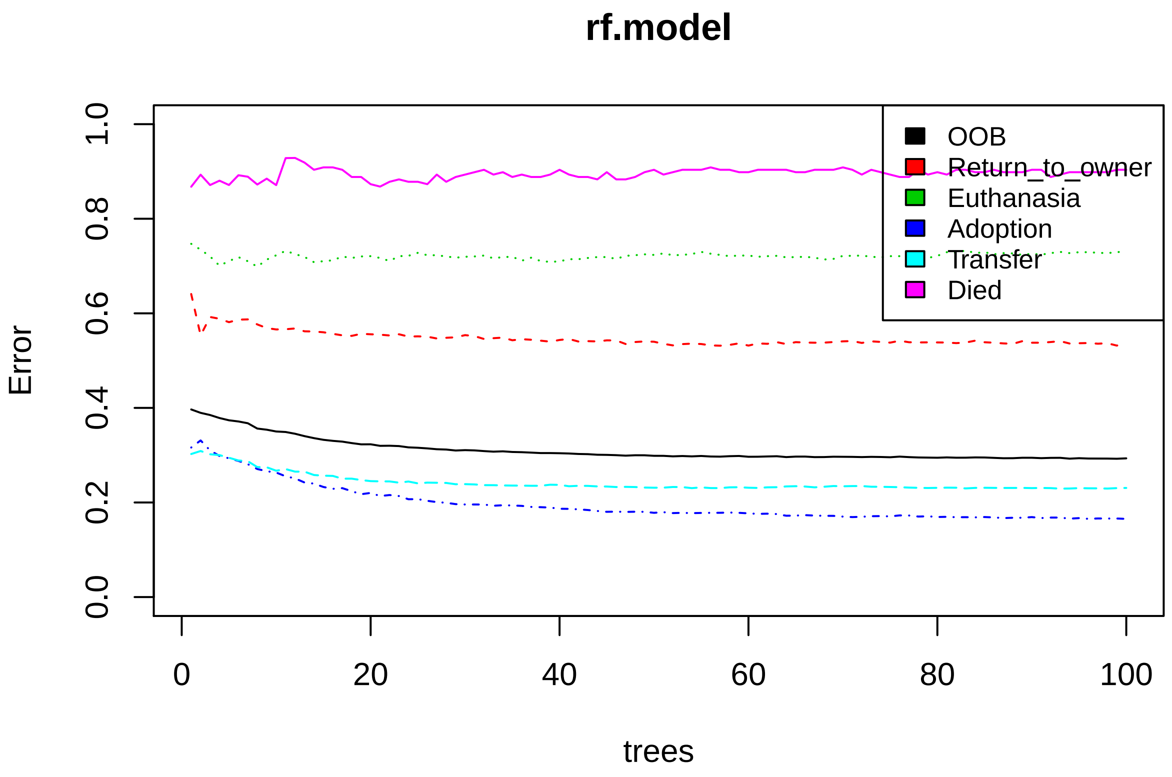
We proceed to the predictions of the test set and show the features importance.

```
##            Feature        Gain       Cover   Frequence
##  1:           Time 0.198258341 0.264619918 0.244790460
##  2: AgeuponOutcome 0.192224067 0.166925665 0.114355006
##  3:      Sterility 0.190454481 0.057478354 0.017254429
##  4:     DateInDays 0.107002818 0.204292857 0.185272719
##  5:           Name 0.070571185 0.033988858 0.018057680
##  6:            Day 0.055456406 0.076373826 0.122866367
##  7:        Weekday 0.054656494 0.052554562 0.094134730
##  8:     AnimalType 0.033959505 0.025629085 0.023325146
##  9:          Month 0.029993976 0.032399469 0.063951064
## 10:           Hour 0.026217875 0.019854356 0.037505600
## 11:   CommonBreeds 0.014744520 0.031161746 0.018706458
## 12:      BreedType 0.011909365 0.019964272 0.023402382
## 13: NumberOfColors 0.011178811 0.012346745 0.027310502
## 14:           Year 0.003372156 0.002410288 0.009067458
```

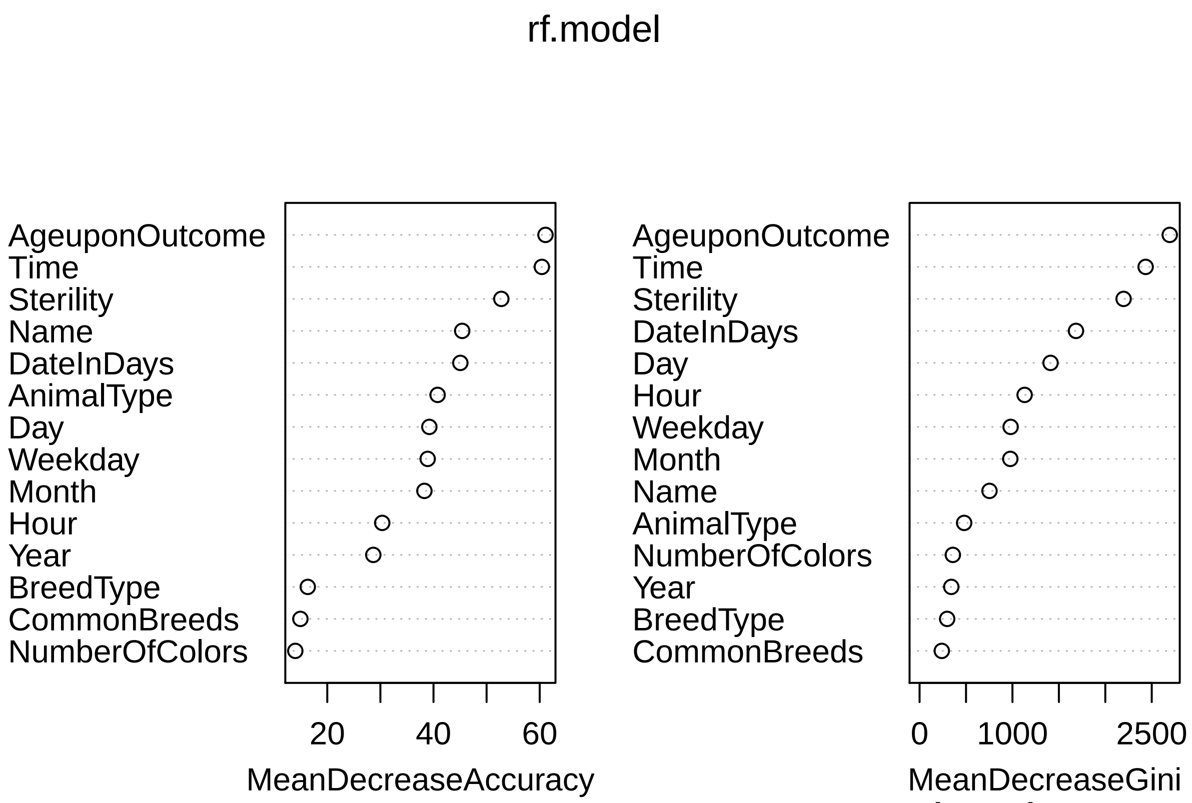


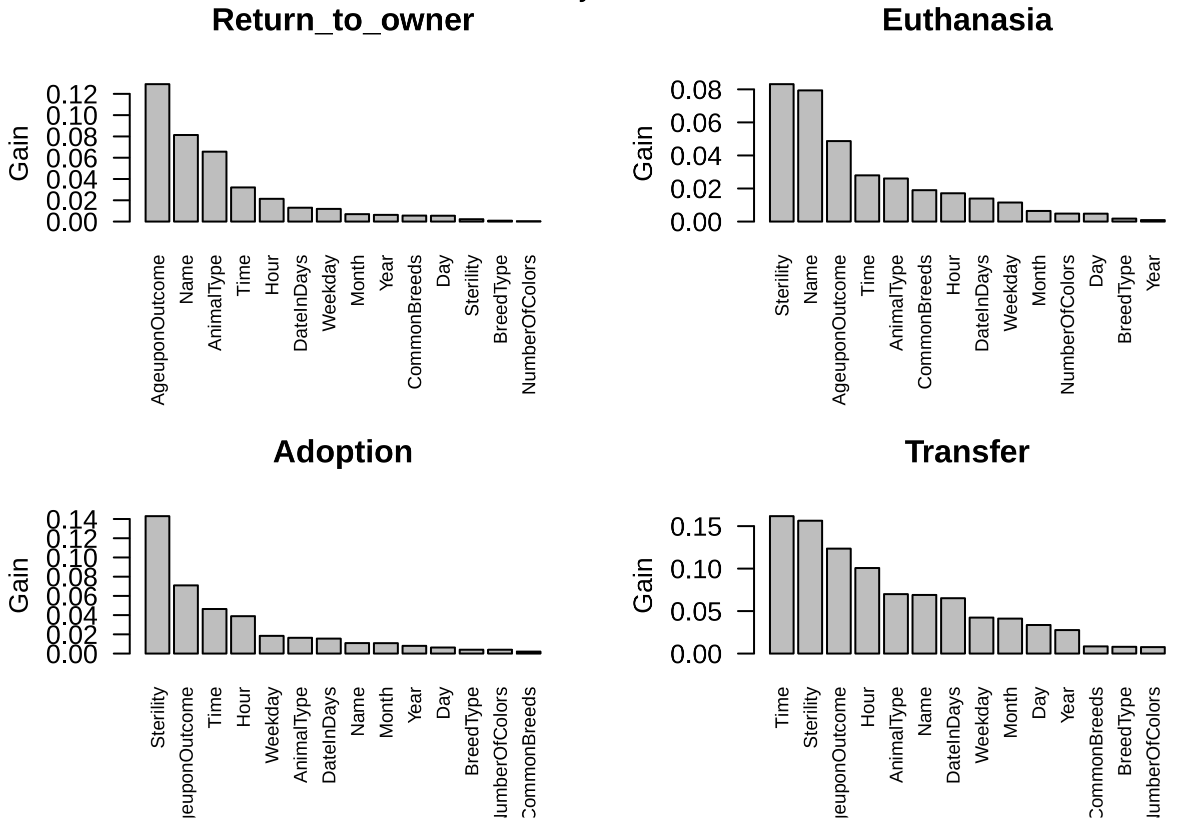
Lets try with the Random Forest algorithm.

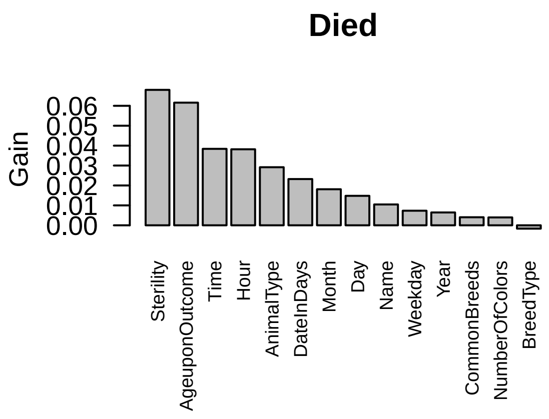
```
## ntree      OOB      1      2      3      4      5
##     5:  37.38% 58.13% 71.05% 29.34% 29.46% 87.13%
##    10:  35.01% 56.59% 72.27% 26.33% 26.72% 87.11%
##    15:  33.25% 55.99% 71.02% 23.28% 25.67% 90.86%
##    20:  32.30% 55.57% 72.09% 22.02% 24.52% 87.31%
##    25:  31.57% 55.13% 72.80% 20.71% 24.04% 87.82%
##    30:  31.07% 55.39% 71.96% 19.57% 23.88% 89.34%
##    35:  30.68% 54.30% 71.96% 19.37% 23.57% 88.83%
##    40:  30.42% 54.37% 71.00% 18.67% 23.72% 90.36%
##    45:  30.06% 54.28% 71.90% 18.02% 23.37% 89.85%
##    50:  29.87% 53.99% 72.35% 17.81% 23.12% 90.36%
##    55:  29.84% 53.51% 72.99% 17.79% 23.19% 90.36%
##    60:  29.66% 53.18% 72.22% 17.70% 23.12% 89.85%
##    65:  29.70% 53.91% 71.90% 17.23% 23.45% 89.85%
##    70:  29.66% 54.07% 72.15% 17.01% 23.42% 90.86%
##    75:  29.56% 53.80% 72.09% 17.06% 23.28% 89.34%
##    80:  29.46% 53.84% 72.15% 16.93% 23.09% 89.85%
##    85:  29.51% 53.89% 72.93% 16.92% 23.11% 89.85%
##    90:  29.45% 53.78% 72.60% 16.91% 23.04% 90.36%
##    95:  29.37% 53.68% 72.93% 16.70% 23.04% 89.85%
##   100:  29.34% 53.74% 73.12% 16.53% 23.07% 90.36%
```



```
##
## Call:
##  randomForest(x = train, y = as.factor(outcome.type), ntree = 100,      importance = TRUE, do.trace =
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 29.34%
## Confusion matrix:
##      0   1    2    3  4 class.error
## 0 2214  41 1924  607  0   0.5374008
## 1  270 418  251  610  6   0.7311897
## 2 1068  30 8989  680  2   0.1652893
## 3  729 119 1324 7248  2   0.2307366
## 4   11  24   21  122 19   0.9035533
```


rf.model
AgeuponOutcome
Time
Sterility
Name
DateInDays
AnimalType
Day
Weekday
Month
Hour
Year
BreedType
CommonBreeds
NumberOfColors
20
40
60
MeanDecreaseAccuracy
AgeuponOutcome
Time
Sterility
DateInDays
Day
Hour
Weekday
Month
Name
AnimalType
NumberOfColors
Year
BreedType
CommonBreeds
0
1000
2500
MeanDecreaseGini


Return_to_owner
Gain
0.12
0.10
0.08
0.06
0.04
0.02
0.00
AgeuponOutcome
Name
AnimalType
Time
Hour
DateInDays
Weekday
Month
Year
CommonBreeds
Day
Sterility
BreedType
NumberOfColors
Euthanasia
Gain
0.08
0.06
0.04
0.02
0.00
Sterility
Name
AgeuponOutcome
Time
AnimalType
CommonBreeds
Hour
DateInDays
Weekday
Month
NumberOfColors
Day
BreedType
Year
Adoption
Gain
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Sterility
ıeuponOutcome
Time
Hour
Weekday
AnimalType
DateInDays
Name
Month
Year
Day
BreedType
lumberOfColors
CommonBreeds
Transfer
Gain
0.15
0.10
0.05
0.00
Time
Sterility
ıeuponOutcome
Hour
AnimalType
Name
DateInDays
Weekday
Month
Day
Year
CommonBreeds
BreedType
lumberOfColors



Note that normally, we would have used a threshold to identify each animal associated to an outcome. Since this is a Kaggle competition, it is not required.

Multi-class Log-Loss

We can verify how our predictions score under the multi-class log-loss function. We take our predictions applied to the train set and we compare to the real `OutcomeType` values of the train set.

```
## [1] 0.4252021
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction        Adoption Died Euthanasia Return_to_owner Transfer
##   Adoption            9955    8        110            1022      821
##   Died                   0  164          0               0        0
##   Euthanasia             4    0       1096              11       19
##   Return_to_owner      507    6        119            3494      425
##   Transfer             303   19        230             259     8157
##
## Overall Statistics
##
##                Accuracy : 0.8555
##                  95% CI : (0.8512, 0.8597)
##     No Information Rate : 0.4029
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.784
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Adoption Class: Died Class: Euthanasia
## Sensitivity                   0.9244    0.832487           0.70482
## Specificity                   0.8771    1.000000           0.99865
## Pos Pred Value                0.8354    1.000000           0.96991
## Neg Pred Value                0.9450    0.998758           0.98207
## Prevalence                    0.4029    0.007370           0.05818
## Detection Rate                0.3724    0.006136           0.04100
## Detection Prevalence          0.4458    0.006136           0.04228
## Balanced Accuracy              0.9008    0.916244           0.85174
##                      Class: Return_to_owner Class: Transfer
## Sensitivity                          0.7300          0.8657
## Specificity                          0.9518          0.9531
## Pos Pred Value                       0.7677          0.9096
## Neg Pred Value                       0.9417          0.9288
## Prevalence                           0.1791          0.3525
## Detection Rate                       0.1307          0.3052
## Detection Prevalence                 0.1703          0.3355
## Balanced Accuracy                    0.8409          0.9094
```

Submission

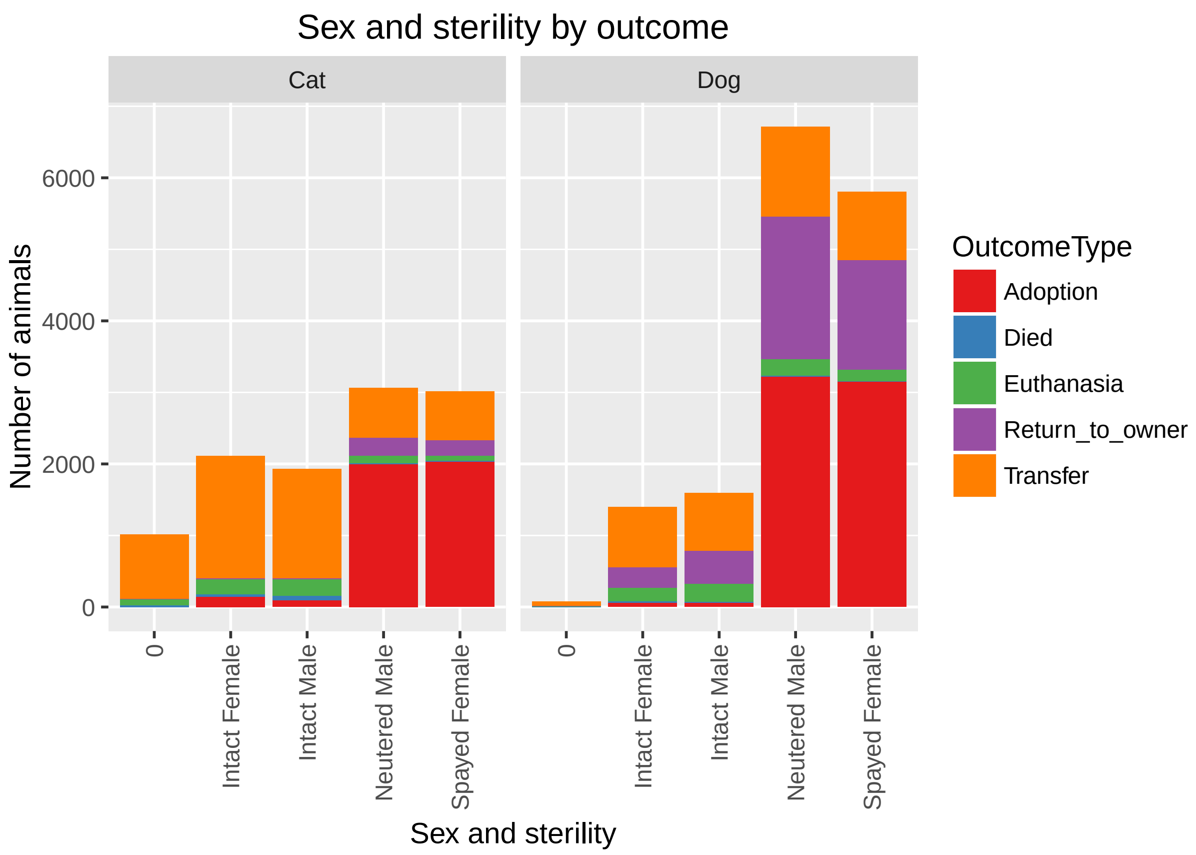
We write the `ID` and the predicted outcome classes in the submission file.

```
prediction.test <- data.frame(matrix(prediction.test,
                                     ncol = outcome.class.num,
                                     byrow = TRUE))[, order(outcomes)]
colnames(prediction.test) <- sort(outcomes)

submission <- cbind(data.frame(ID = test.id), prediction.test)
write.csv(submission, "Submission.csv", row.names = FALSE)
```

Recommandations

The objective is to understand trends in animal outcomes. These insights could help shelters focus their energy on specific animals who need a little extra help finding a new home. Therefore, we need insights to help increasing the number of animals adopted.

An important insight we found in the dataset is if the animal is sterile, not sterile or unknown.



The bar chart shows that animals with no information about its sterility are never adopted. They represent 4.0929328 % of the dataset where we already know the outcomes. Our predictions go the same way where the probabilities are very low as the following summary shows the results.

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
## 0.0000598 0.0001487 0.0002488 0.0018640 0.0005468 0.1966000
```

Intact animals have a low percentage of adoptions. They represent 5.1307561 % of 7036 intact animals.

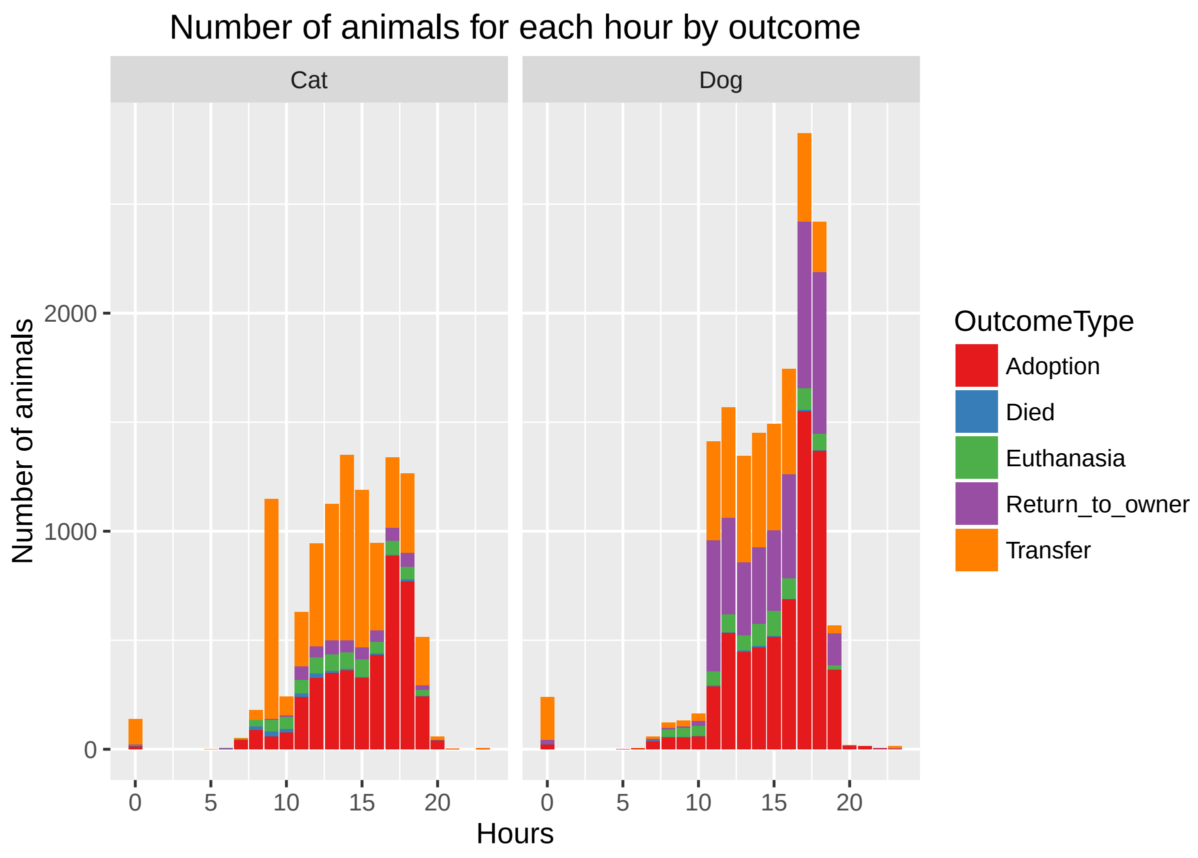
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
## 0.0000671 0.0005453 0.0068780 0.0512300 0.0372900 0.9444000
```

Sterile animals show a better adoption ratio. They represent 55.9599978 % of 18599 sterile animals. This is an increase of 50.8292417 % compared to the intact ones.

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
## 0.002305 0.322800 0.571500 0.570000 0.842600 0.998200
```

Since the adoption ratio is by far much better for sterile animals, we recommand to sterilize them when received to the Animal Center. This will greatly increase their chance to be adopted. Indeed, regarding the web site, the steps of adoption take much more time if the animal is not sterilized. Poeple have to come to the center again to complete the adoption.

Regarding the open hours of the Animal Center which is between 11am and 7pm during the work days, we observed that there are more adoptions between 5pm and 7pm. Since people normally finish working between 4h30pm and 5h30pm, this result makes sense.



We have also seen that the weekend (Saturday and Sunday), there are more adoptions, specially between 5pm and 7pm as for work days. Therefore, we recommand to keep these open hours for adoption claim everyday and extend them for the weekend as for the work days to allow people coming to the Animal Center later. Adding one hours (11am to 8pm) may help.