# Shelter Pets Outcome Prediction Project Report

Song Tang, sot13@pitt.edu

Jiale Sun, jis96@pitt.edu

Kyuri Song,

2017.4.12

## Introduction

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.

For better to take care of each animal in shelters and provide help to find a new home for different animals, we will train the data to build multiple models and see which model is more reliable, collect and conclude insights of data to predict others shelter animal's outcome. The train and test dataset both from Kaggle Shelter Animal Outcome. The final prediction result is a csv file contains each animal ID, all animals outcome type, and the probability for each outcome type.

### Overview of Shelter Animal Dataset

```
library(readr)
#inport train and test dataset
train<- read.csv('train.csv', header = T, stringsAsFactors = F, na.strings =
c(""))
test<- read.csv('test.csv', header = T, stringsAsFactors = F, na.strings = c(
""))
names(train)

##  [1] "AnimalID"       "Name"           "DateTime"       "OutcomeType"
##  [5] "OutcomeSubtype" "AnimalType"     "SexuponOutcome" "AgeuponOutcome"
##  [9] "Breed"          "Color"

nrow(train)

## [1] 26729
```

In the shelter animals train dataset, there are ten columns, which are AnimalID, Name, DateTime, OutcomeType, AnimalType, SexuponOutcome, AgeuponOutcome, Breed and Color. Totally there are 26729 rows data in the train dataset.

AnimalID indicates the ID of each animal. Name column shows the name of each animal, which has lots of missing value. DateTime indicates the specific time for each animal accepted by the shelter. OutcomeType is the final outcome for each animal from shelter, only five outcome types: Return to owner, Euthanasia, Adoption, Transfer and died. OutcomeSubtype is a more specific explanation for OutcomeType, also there are lots of missing value in this column. AnimalType column is the type of each animals, dog or cat. SexuponOutcome column demonstrates the sex ability of each animals, including Neutered Male, Intact Male, Spayed Female, Intact Female and unknown. AgeuponOurcome implied the age for each animal when they are transferred from the shelter. Breed column indicates the breed(type) for each animals and color shows the detailed color for each animal.

```
names(test)

## [1] "ID"              "Name"            "DateTime"        "AnimalType"
## [5] "SexuponOutcome" "AgeuponOutcome" "Breed"           "Color"

nrow(test)

## [1] 11456

setdiff(names(train), names(test)) # columns in train but not in test

## [1] "AnimalID"       "OutcomeType"    "OutcomeSubtype"
```

Take a look of test data set, there are 11456 rows and the difference between test and train data set is three columns, most of attributes are matched.

```
#show the sum of null valve in each columns in train data
sapply(train,function(x) sum(is.na(x)))

##       AnimalID           Name       DateTime    OutcomeType OutcomeSubtype

##              0           7691              0              0          13612

##     AnimalType SexuponOutcome AgeuponOutcome          Breed          Color

##              0              1             18              0              0
```

There are lots of missing value in `Name` and `OutcomeSubtype` columns, as well as in `AgeuponOutcome` column.

```
# show the unique value for each column.
sapply(train, function(x) length(unique(x)))

##       AnimalID           Name       DateTime    OutcomeType OutcomeSubtype

##          26729           6375          22918              5             17
##     AnimalType SexuponOutcome AgeuponOutcome          Breed          Color

##              2              6             45           1380            366
```

We can find that there are massive unique values in `Name`, `OutcomeSubtype`, `SexuponOutcome`, `AgeuponOutcome`, `Breed` and `Color` column.

## Import Necessary Packages

```r
#Load all the necessary packages to explore data.
library(ggplot2) #data visualization
library(dplyr) # for dataset manipulation
library(ggthemes)# data visualization
library(lubridate)# modify the format of dates
library(rpart) # rpart for imputation create decision tree
library(randomForest) # classification algorithm
library(rpart.plot)
```

## Data Cleanup and Modify

According to the overview of the dataset, the original data is "dirty" and in “disorder”. Therefore, we need to clean up and narrow down the multiple unique variables later to a set with better organized and classified variables before we start explore dataset and build models.

In this part, we focus on cleaning up the train dataset which have null value and complex unit. We will also clean up the data in test dataset in the same way.

We already find that `Name` and `OutcomeSubtype` have most missing values. Actually, animal's name can be changed as boolean and the outcome subtype is not required in final prediction.

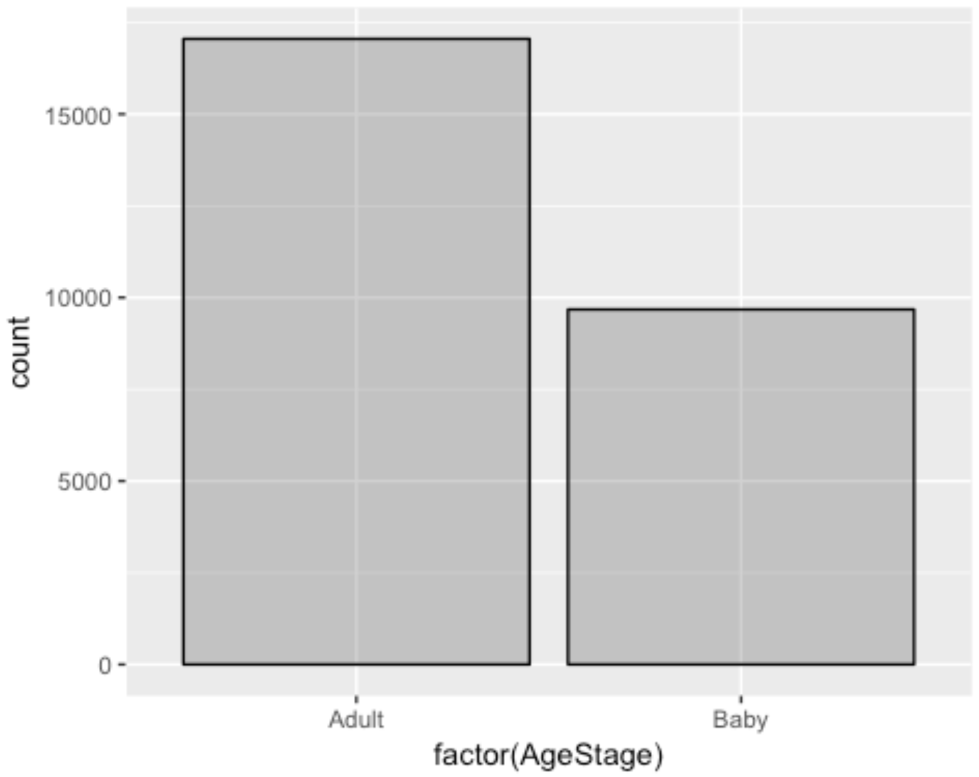
- Eliminate the null value in the `Name` and `OutcomeSubtype` column and replace it with boolean value.

```r
library(readr)
#re-load train dataset with setting in empty value as "0"
#cleanup and eliminate NA value in the train dataset and convert it into 0 or
 1 value
train<- read.csv('train.csv', header = T, stringsAsFactors = F, na = 0)
train$Name = ifelse(nchar(train$Name) == 0, "Nameless", train$Name)
train$Name=ifelse(train$Name == 'Nameless', 0, 1)
#Replace all null value in the OutcomeSubType colum with 'Other' value
train$OutcomeSubtype = ifelse(nchar(train$OutcomeSubtype) == "0", 'Other', tr
ain$OutcomeSubtype)
```

The age of animals is disorderly. We need to change it into a uniformed way.

- Modify the `AgeuponOutcome` column to unified day unit in age, and then translate age in day unit to an age slot to identify the status of each animal.

```
# there are 18 missing value in AgeuponOutcome column, we find that the  youn
g age stage animals has
#the most number, so we set the empty value in AgeuponOutcome column is '7 mo
nths', which is for the young stage.
train$AgeuponOutcome=ifelse(nchar(train$AgeuponOutcome) == 0, '7 months', tra
in$AgeuponOutcome)
# retrive the time value
train$TimeValue = sapply(train$AgeuponOutcome, function(x) strsplit(x, split
= ' ')[[1]][1])
# retrive the time unit,like years, months, days
train$UnitofTime = sapply(train$AgeuponOutcome, function(x) strsplit(x, split
 = ' ')[[1]][2])
#modify the plural number into singular. for example 'years' to 'year'
train$UnitofTime = gsub('s', '', train$UnitofTime)
train$UnitofTime = as.factor(train$UnitofTime) # convert the unit of time int
o factor
train$TimeValue = as.numeric(train$TimeValue) #convert the value of time into
 interger

#compute the age of each animal in day unit by converting TimeValue in days u
sing
#the appropriate multiplier based on different unit.
multiplier = ifelse(train$UnitofTime == 'day', 1, ifelse(train$UnitofTime ==
'week', 7,
    ifelse(train$UnitofTime == 'month', 30, ifelse(train$UnitofTime == 'year'
, 365, NA))))
train$AgeinDays = multiplier * train$TimeValue

#then we convert the AgeinDays into more understanable formate.
train$AgeStage <- ifelse(train$AgeinDays > 1 & train$AgeinDays < 180, 'Baby',
'Adult')
ggplot(train, aes(factor(AgeStage)))+geom_bar(col="black", alpha = .3)
```



Note that `train$AgeinDays` is created to store the age in the unit of day, and `train$Ageslot` is created to indicate the status of each animal.

The `DateTime` column indicates the specific time when animals were accepted by shelter. We can't handle it in thousands of separate dates without classification. We assume maybe there are relationships between outcomes with months, weekdays and daytimes, which consists with the fact in reality. So we decide to separate the date time into three other columns: Season, Weekday and Daytime.

- Extract time variables from date (use the "lubridate" package).

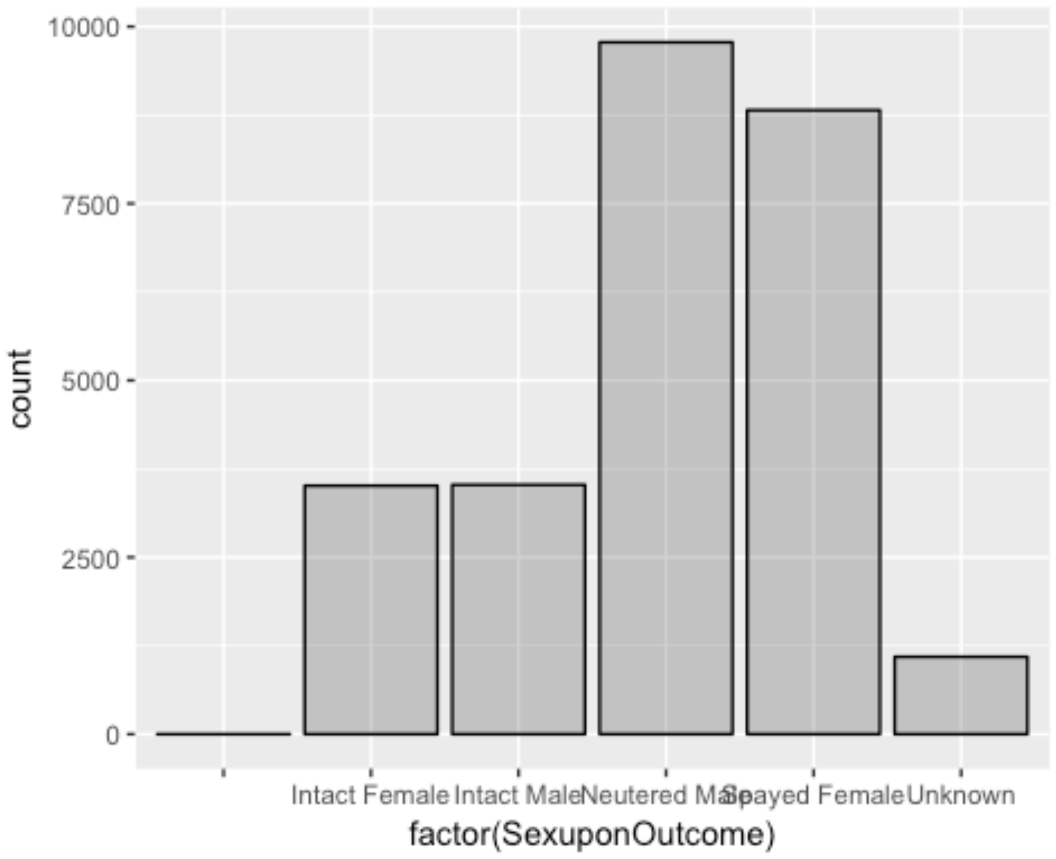
```
train$Hour <- hour(train$DateTime)
train$Month <- month(train$DateTime)
train$Weekdays <- wday(train$DateTime)
train$Year <- year(train$DateTime)

#In this part we extract the Months of DateTime and translate it into the fou
r seasons.
train$AcceptedSeasons <- ifelse(train$Month > 1 & train$Month < 5, 'Spring',
                  ifelse(train$Month > 4 & train$Month < 8 , 'Summer',
                  ifelse(train$Month > 7 & train$Month < 11, 'Autumn', 'Winte
r' )))

#translate the the DateTime columns to the week day value
```

```
train$AcceptedWeekday <- weekdays(as.Date(train$DateTime))

#also, maybe the time in day may affect the outcomes of animals, divide it in
to two types, day and night
train$AcceptedDayTime <- ifelse(train$Hour > 5 & train$Hour < 18, 'day', 'nig
ht')
```

Additionally, we can find the sex attribute is not simply indicates “male” or “female”, but also the intact situation.

- Divide sex into two collumns, `isIntact` and `Sex`.

```
# find the most frequent sex attribute to replace the missing value
ggplot(train, aes(factor(SexuponOutcome)))+geom_bar(col="black", alpha = .3)
```



```
# we conclude that the Neutered Male is the most common one
train$SexuponOutcome = ifelse(train$SexuponOutcome == "Unknown", "Neutered Ma
le",
                              train$SexuponOutcome)
# Use "grepl" to look for "Intact". Neutered and spayed is the same meanings
train$isIntact <- ifelse(grepl('Intact', train$SexuponOutcome), 'Intact',
       ifelse(grepl('Unknown', train$SexuponOutcome), 'Neutered', 'Neutered')
)
```

```
#replace the missing value with the most common Neutered Male.

# Use "grepl" to look for sex
train$Sex <- ifelse(grepl('Male', train$SexuponOutcome), 'Male',
            ifelse(grepl('Female', train$SexuponOutcome), 'Female', 'Male'))
```

We check out that Breed and Color have too many level value. We have to unify the breed and color into a more general standard. Maybe we can ignore some bias value. For example, we just divide the color and breed into two types, Pure and Mix. This method could dramatically decrease the complexity of value but may represent the general attribute of animals.

- Clear color and breed into mix and pure.

```
length(unique(train$Color)) #number of unique colors we're dealing with

## [1] 366

#Use "grepl" method to look for "Mix" and "/" to identify the pure and mix ty
pe.
train$isMixBreed <- ifelse(grepl("Mix", train$Breed), 'MixBreed',
                     ifelse(grepl("/", train$Breed), 'MixBreed', 'PureBreed')
)
train$isMixColor <- ifelse(grepl('/' , train$Color), 'MixColor',
                      ifelse(grepl('Mix' , train$Color), 'MixColor','PureColo
r'))
```

- Finally we write the modified table into csv file and saved to disk.
- **After we finish the cleanup of train dataset, we should also apply the same cleanup method to the test dataset**

## Summary of Statistics (Plots)

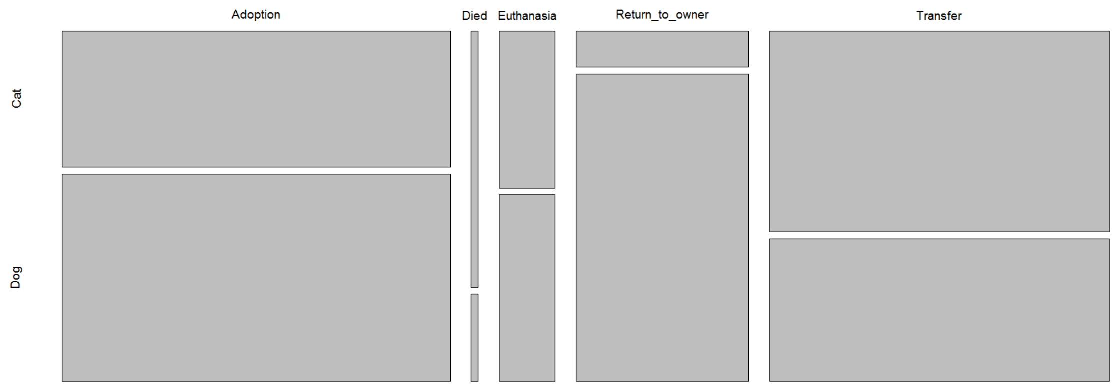
The initial data cleaning up is finished (We will improve that later). To make better intuitions, we should look at the summary of statistics.

We compare the proportions of the outcomes. The table and graph stated below show the result. We could find that about 6% of the animals died. Cats are more likely to have died compared to dogs. Most of animals are transferred or adopted.

```
table(train$OutcomeType)
```

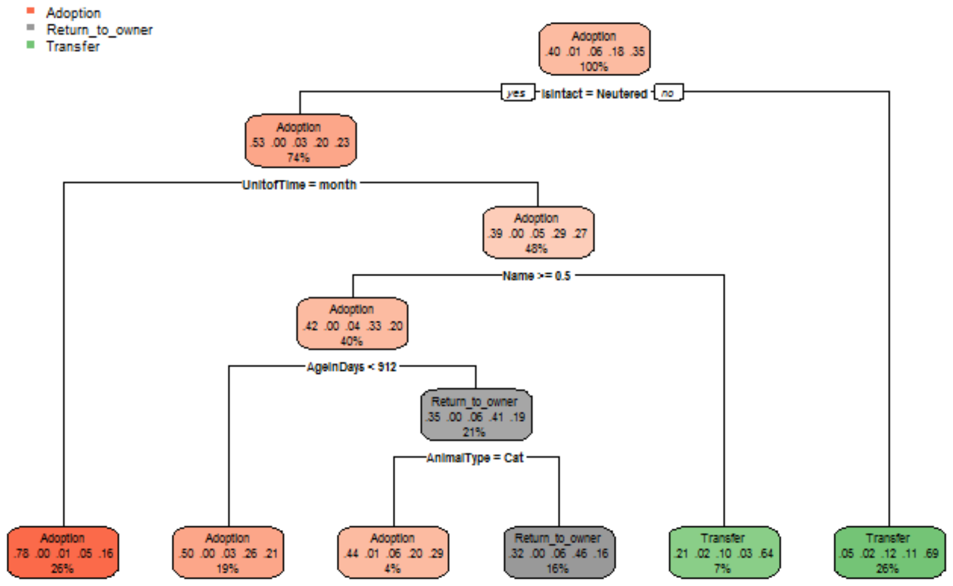
```
       Adoption            Died      Euthanasia Return_to_owner        Transfer
          10769             197            1555            4786            9422
```

```
mosaicplot(table(train$OutcomeType, train$AnimalType), main="", cex=1.2)
```



Next, we study which variables determined the outcome. As mentioned above, we simply uniform data through defining missing values and dividing each variable as several groups. We test the factors by using a binary tree model to understand the importance of each values for the prediction.

```
fit <- rpart(OutcomeType ~ Name + AnimalType + Breed + Sex + Color + UnitofTi
me + AgeinDays + Month + AgeStage + AcceptedSeasons + AcceptedDayTime + inInt
act, data=train, method="class")
rpart.plot(fit)
```


🟥 Adoption
▨ Return_to_owner
🟩 Transfer
Adoption .40 .01 .06 .18 .35 100%
yes
IsIntact = Neutered
no
Adoption .53 .00 .03 .20 .23 74%
UnitofTime = month
Adoption .39 .00 .05 .29 .27 48%
Name >= 0.5
Adoption .42 .00 .04 .33 .20 40%
AgeinDays < 912
Return_to_owner .35 .00 .06 .41 .19 21%
AnimalType = Cat
Adoption .78 .00 .01 .05 .16 26%
Adoption .50 .00 .03 .26 .21 19%
Adoption .44 .01 .06 .20 .29 4%
Return_to_owner .32 .00 .06 .46 .16 16%
Transfer .21 .02 .10 .03 .64 7%
Transfer .05 .02 .12 .11 .69 26%

The statistic below describes which factor is more effective than others on the prediction. It shows that intactness is the most significant determinant of outcomes than others. 'Age' and 'has name' might be more important variables than accepted daytime, month, and season. The next step describes detailed information and analysis methodologies used to create a prediction.

```
fit$variable.importance
```

```
      isIntact        AgeinDays       UnitofTime         AgeStage             Name       AnimalType 
   2383.651028      1687.608681      1500.526010       890.385529       698.670181       157.964207 
AcceptedDayTime            Color            Month 
      4.215374         1.696127         1.172323 
```

### Intuitions

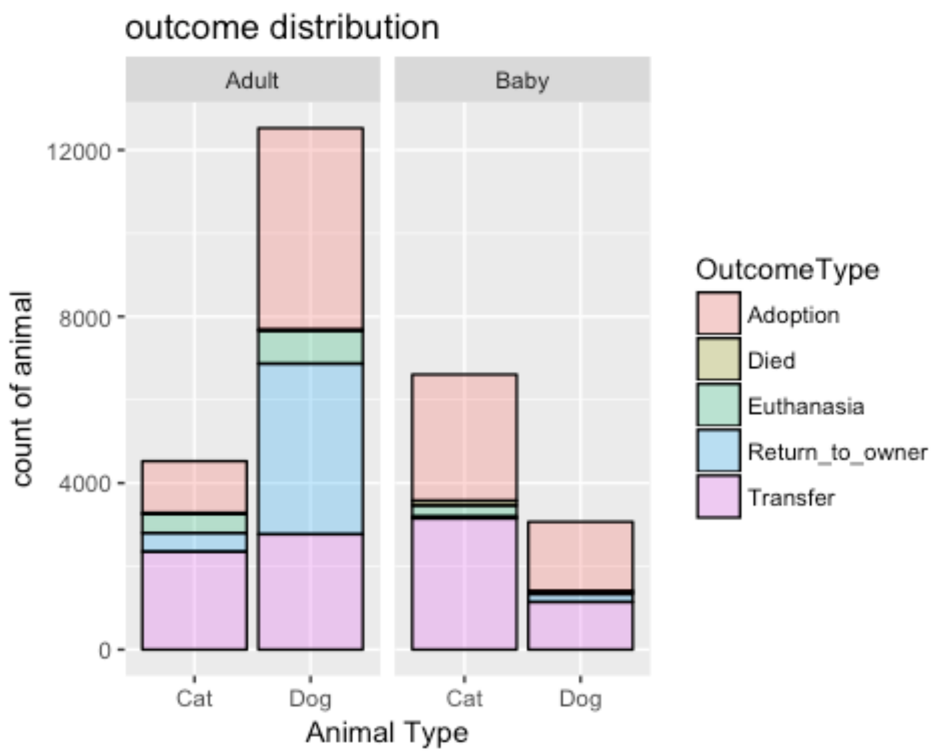
Therefore, based on data cleaning and summary, the `isIntsct`, `AgeinDays`, `UnitofTime`, `Name` and `AnimalType` are much more important than other factors. We think these factors may relate to the health conditions and temperment for each animal. These results would be helpful in predictors than some of the raw features provided in the data.

## *Data Story Telling*

Before we start to create model and predict the outcome of animals. We should explore the data distribution and tell a data story.

- Plot the relation between animal type and the outcome of animals.

```
# plot the outcome of different type of animals
ggplot(train, aes(factor(AnimalType),fill = OutcomeType))+
  geom_bar(col="black", alpha = .3)+facet_wrap(~AgeStage)+
  labs(x="Animal Type", title="outcome distribution", y="count of animal")
```



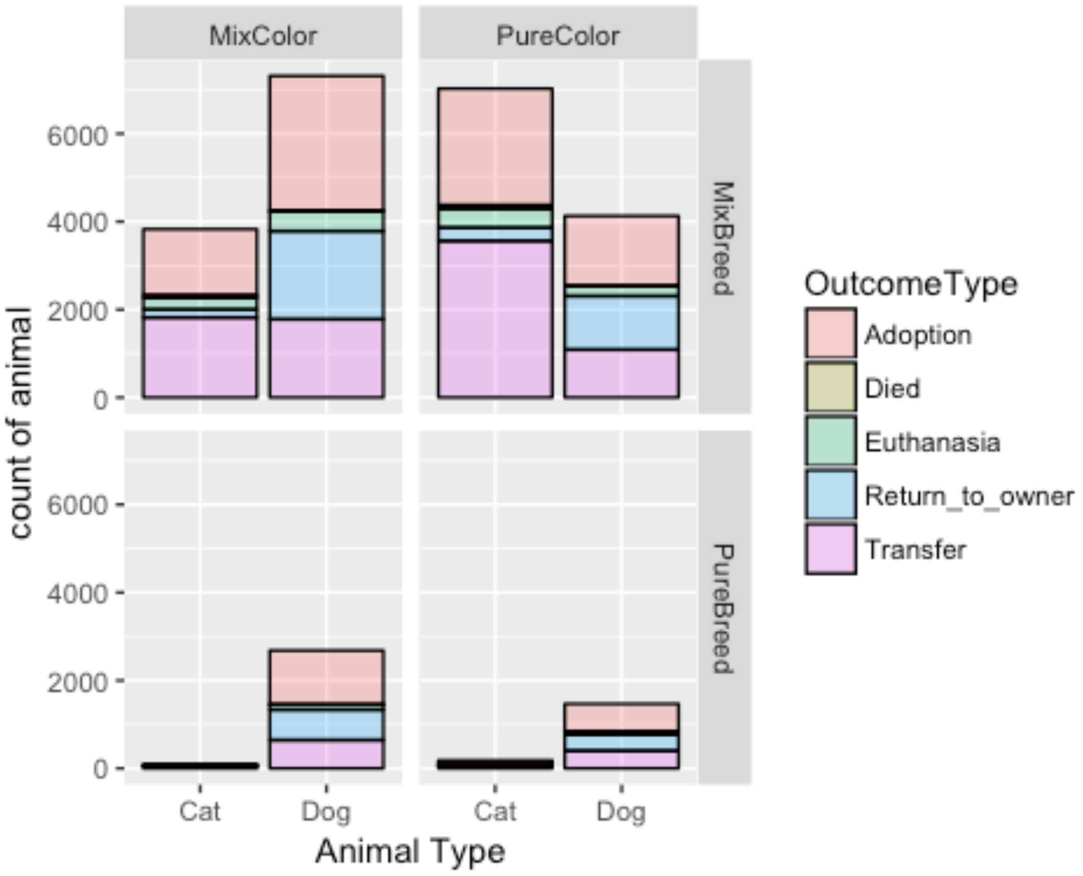
As for animal type, the number of dogs are more than the number of cats, also the dog has more possible to be returned to their original owner and be adopted, while cats are more likely to be transferd. Generally dogs are more popular in adoption than the cats.

Also, we find most of the animals are adults (more than six months old), especially as adult dogs. Adults animals are more likely to be returned to owner than baby animals. But the partition of baby animals are more than adults to be adopted. And fortunately, animals are seldomly euthanasiazed or died.

---

- Next we want know how the breed, color and the animal type affect the outcomes of shelter animals.

```
newtrain <- read_csv("~/Documents/2017Spring/Data Analytic/term-project/newtr
ain.csv")
ggplot(newtrain, aes(factor(AnimalType),fill = OutcomeType))+
  geom_bar(col="black", alpha = .3)+facet_grid(Breed ~ Color)+
  labs(x="Animal Type", y="count of animal")
```



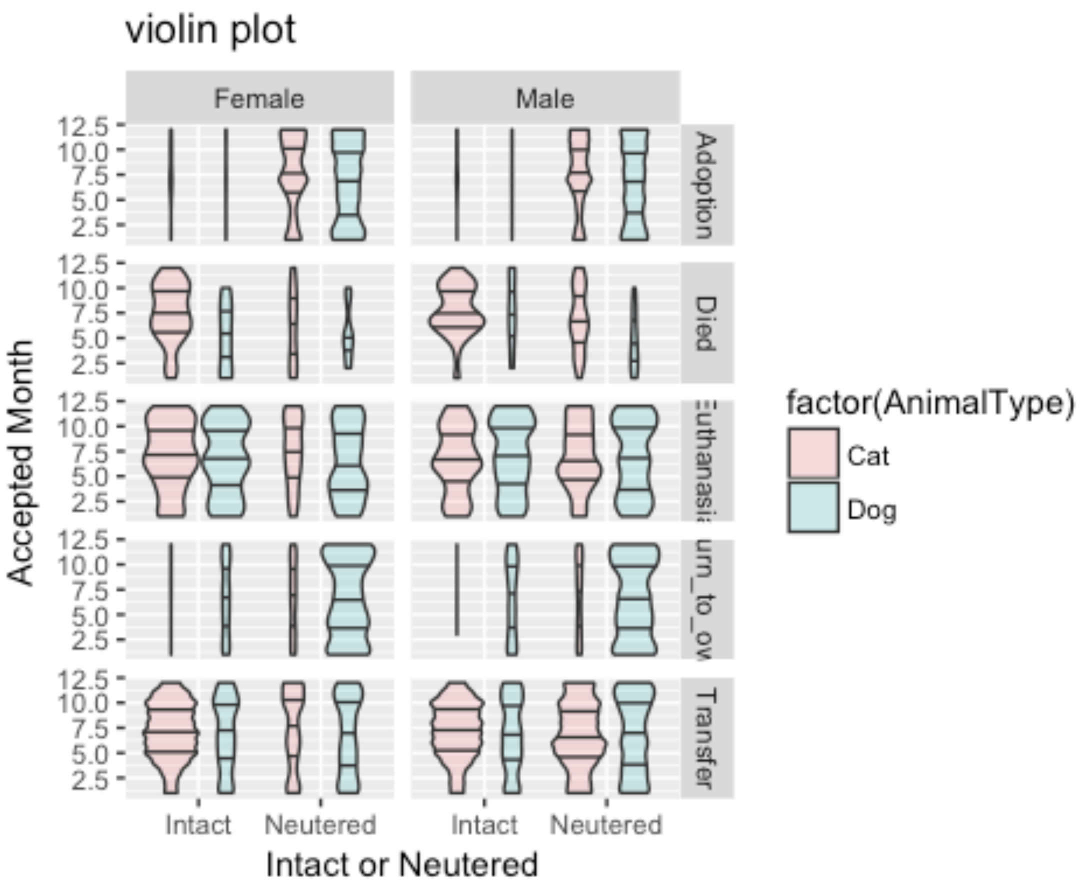
From the above plot, most of animals are mix breed rather than pure breed.

There are more dogs in mix color than pure color while there are more cats in pure color than mix. It seems the mixed color animals are more popular in adoption, especially for mixed color dogs, there are larger possibility for them to be adopted or returned to owner.

---

- To explore more detailed information about the outcome distribution with attributes, we apply violin plot.

```
ggplot(train, aes(x=factor(isIntact), y=Month))+ylim(1,12)+
  geom_violin(scale = "count",adjust= .8, aes(fill=factor(AnimalType)),
              draw_quantiles = c(0.25, 0.5, 0.75), alpha= .2)+facet_grid(Outc
omeType ~ Sex)+
  labs(x="Intact or Neutered")+labs(title="violin plot")+labs(y="Accepted Mon
th")
```

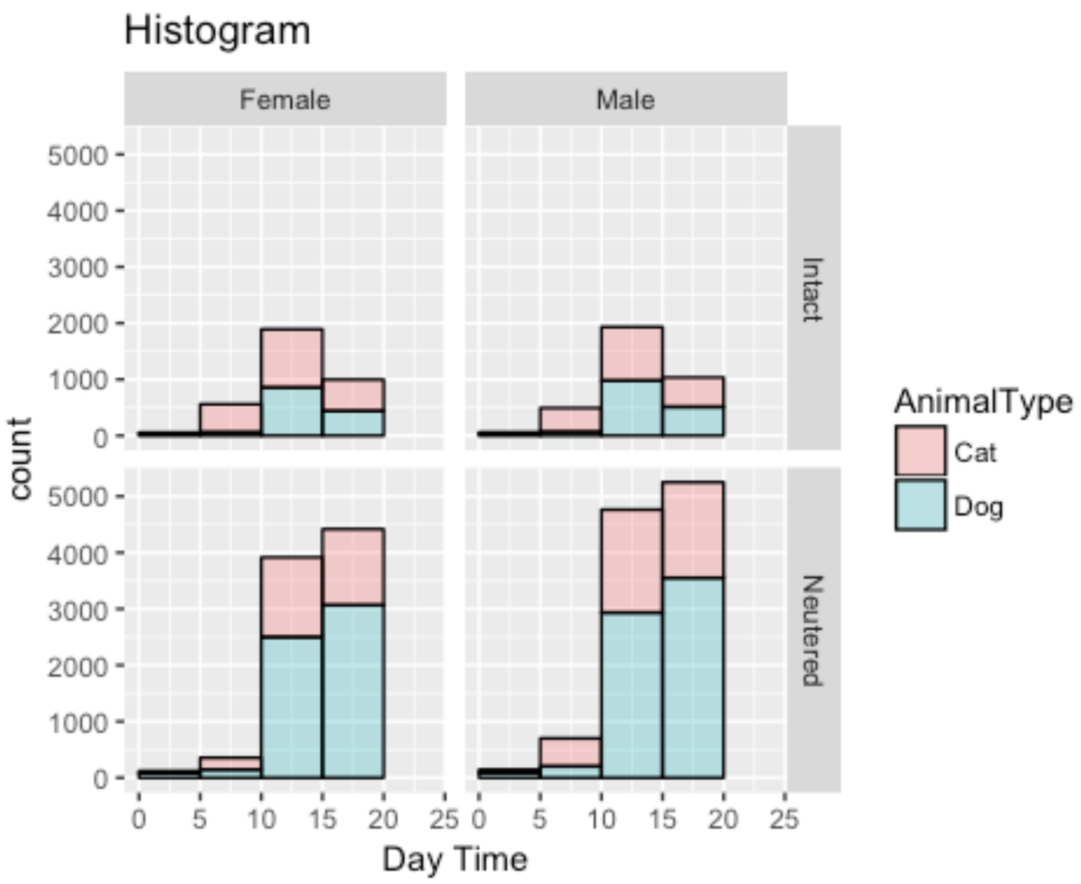


We divided this plot into four attributes. First, we find that nearly all animals adopted are neutered rather rather than intact. Second,  there is not direct relationship in sex(male or female) towards to the final outcome. Third,  less animals are accepted by shelter in Winter. Fourth, dogs are more likely to be returned and adopted than cats as mentioned above.

---

- To explore the outcome with daytime.

```
ggplot(train, aes(x=Hour,fill = AnimalType))+xlim(0,24)+
  geom_histogram(breaks=seq(0, 100, by = 5), col="black", alpha = .3)+facet_g
rid(isIntact~Sex)+
  labs(x="Day Time", title="Histogram")
```

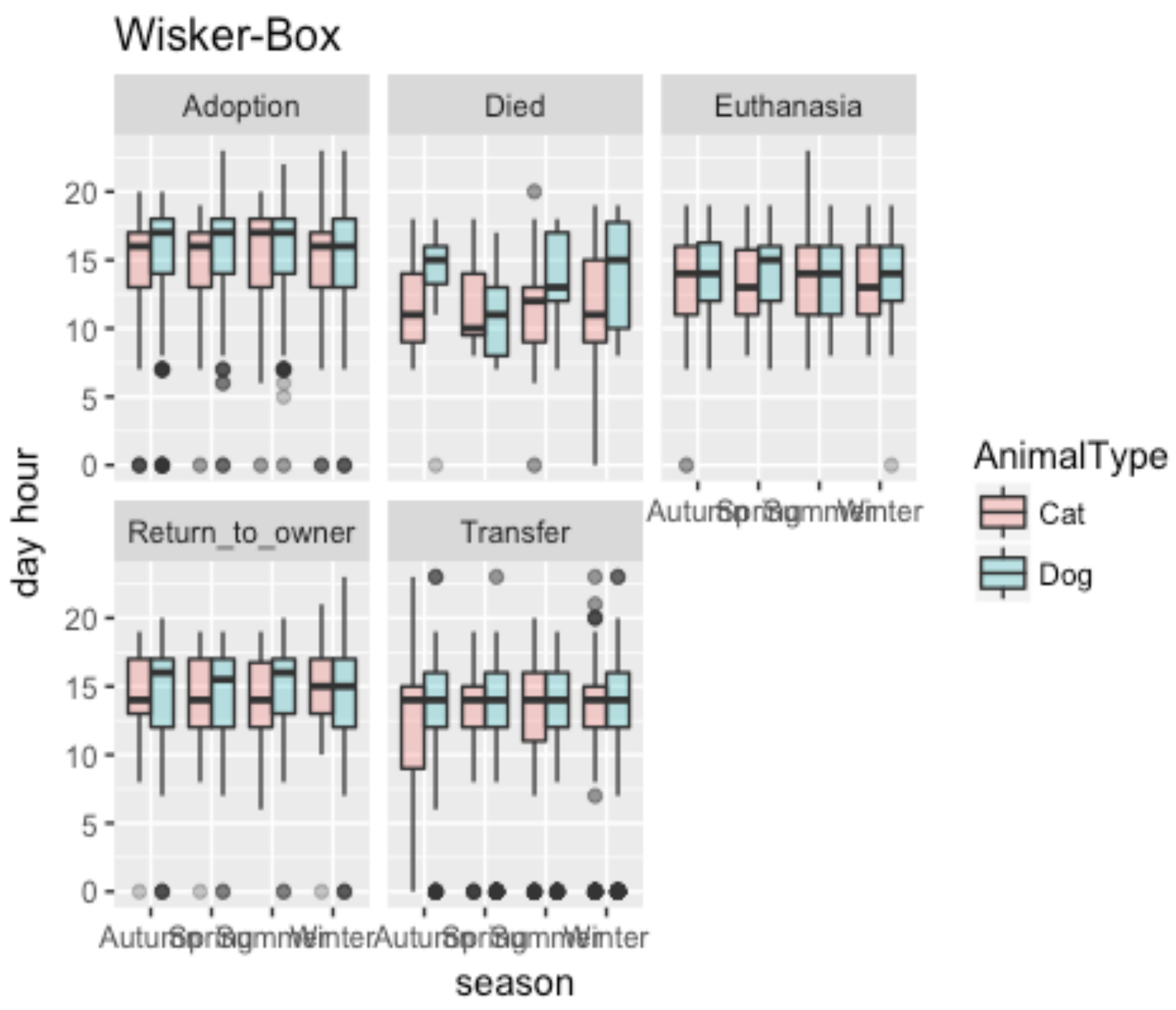


It is obvious that most of animals are accepted in afternoon, and very few on early morning. It is easy understand that people seldom work so early to accept animals.

---

- To explore more on daytime and season in outcome.

```
ggplot(train, aes(x=factor(AcceptedSeasons),y=Hour))+geom_boxplot(aes(fill =
AnimalType), alpha=.3)+facet_wrap(~OutcomeType)+
labs(title="Wisker-Box")+labs(y="day hour")+labs(x="season")
```

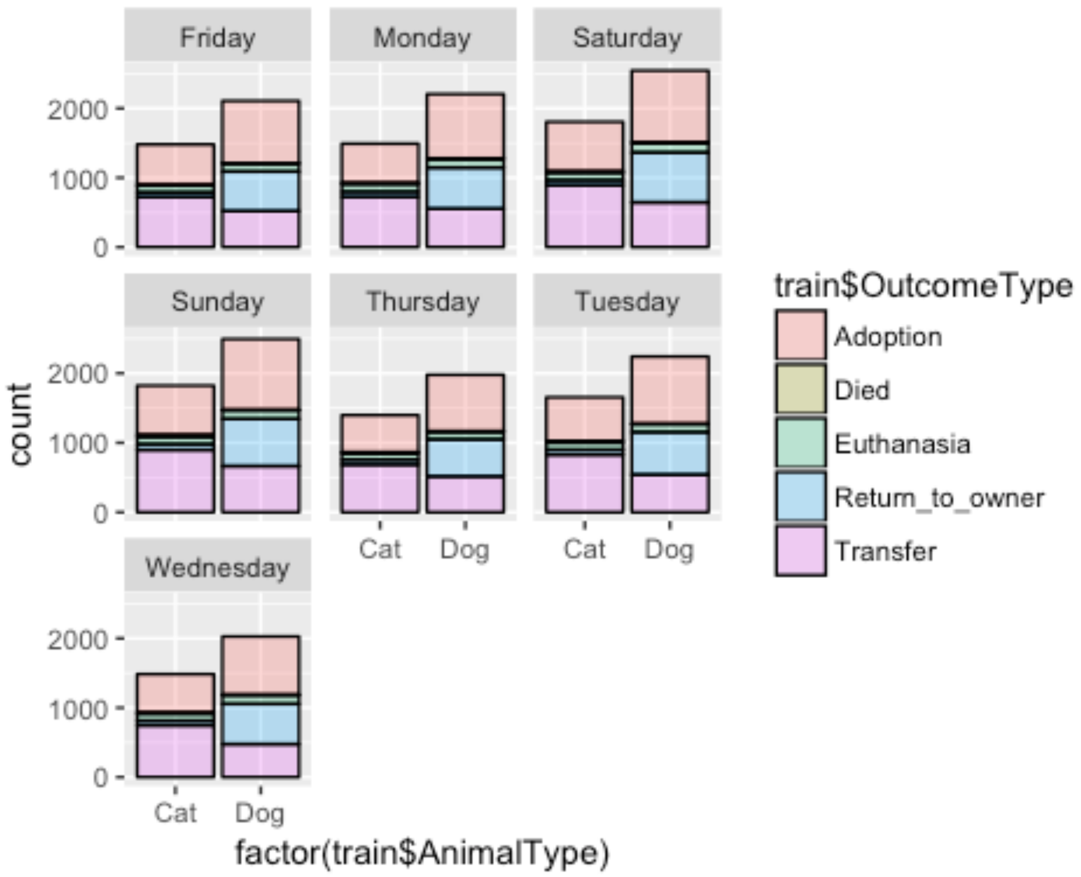


Actually, there are not apparent relations between seasons in outcome, which means maybe month is not a so important factor in influencing the outcome. And similarly, most of animals are accepted in afternoon, and very few on early morning.

---

- To explore the relations among outcome type, accepted weekday and the type of animals.

```
ggplot(train, aes(factor(train$AnimalType),fill = train$OutcomeType))+facet_w
rap(~train$AcceptedWeekday)+
  geom_bar(col="black", alpha = .3)
```



It is obvious that when accepted on weekends (Saturday and Sunday), animals are more likely to be adopted. Very few cats are returned to owner in fact, also cat are more likely transferred then adopted.

## *Create Model and Hypothesis*

### Modified Dataset Overview

- It is the overview of the data we have already cleaned before.

```
#show the unique value in each column
sapply(train, function(x) length(unique(x)))

##        AnimalID            Name        DateTime     OutcomeType
##           26729               2           22918               5
##  OutcomeSubtype      AnimalType  SexuponOutcome  AgeuponOutcome
##              17               2               5              44
##           Breed           Color       TimeValue      UnitofTime
##            1380             366              21               4
##       AgeinDays        AgeStage            Hour           Month
##              43               2              20              12
##        Weekdays            Year AcceptedSeasons AcceptedWeekday
##                 7                 4                 4                 7
## AcceptedDayTime          isIntact               Sex        isMixBreed
##                 2                 2                 2                 2
##       isMixColor
##                 2
```

```
#There are no empty value in train dataset
sapply(train,function(x) sum(is.na(x)))
```

```
##          AnimalID              Name          DateTime       OutcomeType
##                 0                 0                 0                 0
##   OutcomeSubtype        AnimalType   SexuponOutcome   AgeuponOutcome
##                 0                 0                 0                 0
##             Breed             Color         TimeValue       UnitofTime
##                 0                 0                 0                 0
##         AgeinDays          AgeStage              Hour             Month
##                 0                 0                 0                 0
##          Weekdays              Year AcceptedSeasons AcceptedWeekday
##                 0                 0                 0                 0
## AcceptedDayTime          isIntact               Sex        isMixBreed
##                 0                 0                 0                 0
##       isMixColor
##                 0
```

```
nrow(train) # we still have 26729 rows in the train data set after cleanup
```

```
## [1] 26729
```

We can delete some of the original columns after cleanup

```
#we just keep the avliable columns.
train$AnimalID<-NULL
train$AgeuponOutcome<-NULL
train$SexuponOutcome<-NULL
train$OutcomeSubtype<-NULL
```

***Do the same deletion to the modified test dataset.**

## Data Lineage

To analyze the data, we have cleaned up the original data set mentioned above.

- Defining missing values in three methods: Transfer it as Boolean- 1 or 0, such as Name; Or delete it directly such as the useless factor – OutcomeSubtype; Or fill in the missing value by the most popular attribute such as make the N/A Sex as male.
- Separate each variable as sub groups. For example, categorize the column 'DateTime' as three column: 1)AcceptedSeasons, 2)AcceptedWeekends, and 3)AcceptedDaytime; categorize the column 'SexuponOutcome' as two column: 1)Sex, 2)IsIntac

- Reclassified animals’ attributes: divide Colors and Breeds as Mix and Pure.

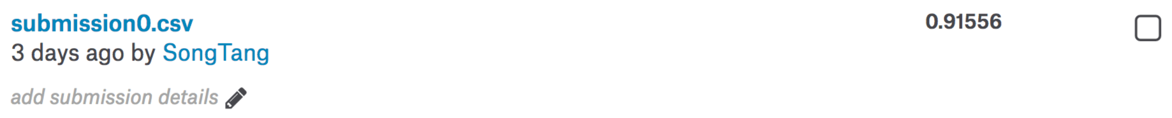
## Building Model

### Model 1 :Ramdom forest

Random Forest model is a common model in quick prediction. We just load the random forest package in R studio and train the method `randomForest` with existing attributes in data set.

```r
library(randomForest)
library(e1071)
set.seed(731) #set a random seed
newtrain <- read_csv("~/Documents/2017Spring/Data Analytic/term-project/newtr
ain.csv")
newtest <- read_csv("~/Documents/2017Spring/Data Analytic/term-project/newtes
t.csv")
newtrain$OutcomeType=as.factor(newtrain$OutcomeType)
modle4 <- randomForest(OutcomeType ~ Name + AnimalType + Breed + Color + AgeS
tage + isIntact + Sex + AcceptedWeekday + AcceptedDayTime,data=newtrain)
pre <- predict(modelrf, newtest)
prob5 <- data.frame(ID = newtest$ID, pre)
#write.csv(prob5, file = "submission0.csv")
```

Result score: not ideal result.


submission0.csv
3 days ago by SongTang
add submission details
0.91556

### Modle 2: Neural Network

The second model we chosen is a feed-forward neural network with a single hidden layer, and multinomial log-linear models. Neural Network model calculates weight of each attribute in data set, and improve weight continually by training feedback. `nnet` is a neural network package in R. The method `multinom` in package nnet generates multinomially distributed random number vectors via neural network, and compute multinomial probabilities to predict the outcome.

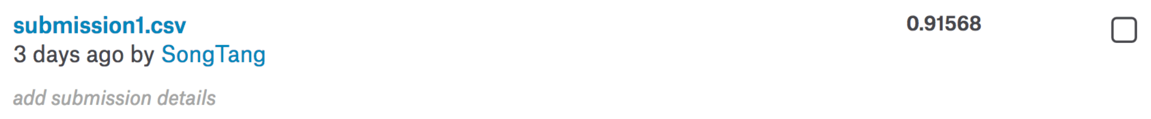
```r
library(nnet)
library(reshape2)
newtest <- read_csv("~/Documents/2017Spring/Data Analytic/term-project/newtes
t.csv")
newtrain <- read_csv("~/Documents/2017Spring/Data Analytic/term-project/newtr
ain.csv")
```

```
newtrain<- read.csv('newtrain.csv', header = T, stringsAsFactors = T)
newtest<- read.csv('newtest.csv', header = T, stringsAsFactors = T)
model <- multinom(OutcomeType ~ Name + AnimalType + Breed + Color + AgeStage
+ isIntact + Sex + AcceptedWeekday + AcceptedDayTime, data = newtrain)
```

```
## # weights:  105 (80 variable)
## initial  value 43018.665961
## iter  10 value 26892.544294
## iter  20 value 25834.700988
## iter  30 value 25300.540473
## iter  40 value 24545.098574
## iter  50 value 24065.329242
## iter  60 value 23873.899506
## iter  70 value 23797.441710
## iter  80 value 23734.009794
## iter  90 value 23682.245849
## iter 100 value 23681.995359
## final  value 23681.995359
## stopped after 100 iterations
```

```
pred <- predict(model, newdata = newtest, "probs")
sub <- data.frame(ID = newtest$ID, pred)
#write.csv(sub, file="submission1.csv")
```

Result score:


submission1.csv
3 days ago by SongTang
add submission details
0.91568

The score doesn't improve much, and we still keep trying.

## Model 3: Logistic Regression

The third model is Logistic Regression, which is a statistical regression model for prediction. We choose the method `vglm.` It is used to fit vector generalized linear models (VGLMs) in `library(VGAM)`

```
newtrain<- read.csv('newtrain.csv', header = T, stringsAsFactors = T)
modle2 <- vglm(OutcomeType ~ Name + AnimalType + Breed + Color + AgeStage +
isIntact + Sex + AcceptedWeekday +
                 AcceptedDayTime,family=multinomial,data=newtrain)
```

```
prob <- predict(modle2,newdata=newtest,type="response")
```

```
sub2 <- data.frame(ID = newtest$ID, prob)
```

```
#write.csv(sub2, file="submission2.csv")
```

Result score:

final2.csv
a day ago by SongTang
add submission details
0.88866

## Model 4: Initial Xgboost Model

### *First attempt version*

The last model we use is xgboost. Xgboost is a effective machine learning model in training. In this model, we need transfer all attributes into a Boolean matrix. The model iterates the training data consistently, and every iteration will form a new booster model. The parameters of the model should be invoked and tried many times. Then by fitting the process to find the optimum result.

```r
#first part is to convert newtrain table and newtest table to numeric matrix
with binary value except the OutcomeType column
library(readr)
newtrain <- read_csv("~/Documents/2017Spring/Data Analytic/term-project/newtr
ain.csv")
newtest <- read_csv("~/Documents/2017Spring/Data Analytic/term-project/newtes
t.csv")
newtrain$OutcomeType <- ifelse(newtrain$OutcomeType=='Adoption', 1,
                                ifelse(newtrain$OutcomeType=='Died', 2,
                                       ifelse(newtrain$OutcomeType == 'Euthan
asia', 3,
                                              ifelse(newtrain$OutcomeType ==
'Return_to_owner',4, 5))))
newtrain$AnimalType <- ifelse(newtrain$AnimalType=='Dog',1,0)
newtrain$Breed <- ifelse(newtrain$Breed=='MixBreed',1,0)
newtrain$Color <- ifelse(newtrain$Color=='MixColor',1,0)
newtrain$AgeStage <- ifelse(newtrain$AgeStage=='Baby', 1,0)
newtrain$isIntact <- ifelse(newtrain$isIntact=='Intact',1,0)
newtrain$Sex <- ifelse(newtrain$Sex=='Male',1,0)
newtrain$AcceptedWeekday <- ifelse(newtrain$AcceptedWeekday=='Saturday', 0,
                                   ifelse(newtrain$AcceptedWeekday=='Sunday',
 0, 1))
newtrain$AcceptedDayTime <- ifelse(newtrain$AcceptedDayTime=='night', 0,
                                   ifelse(newtrain$AcceptedDayTime=='mid-nigh
t', 0, 1))
```

```r
#convert newtest table to binary table
newtest$AnimalType <- ifelse(newtest$AnimalType=='Dog',1,0)
newtest$Breed <- ifelse(newtest$Breed=='MixBreed',1,0)
```

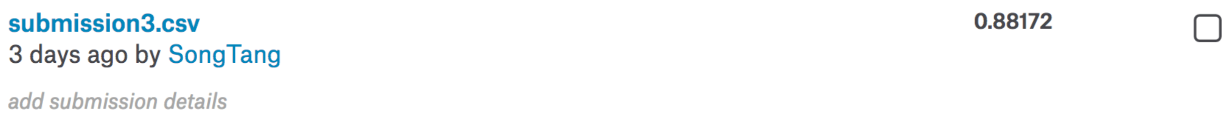
```
newtest$Color <- ifelse(newtest$Color=='MixColor',1,0)
newtest$AgeStage <- ifelse(newtest$AgeStage=='Baby', 1, 0)
newtest$isIntact <- ifelse(newtest$isIntact=='Intact',1,0)
newtest$Sex <- ifelse(newtest$Sex=='Male',1,0)
newtest$AcceptedWeekday <- ifelse(newtest$AcceptedWeekday=='Saturday', 0,
                                  ifelse(newtest$AcceptedWeekday=='Sunday', 0
, 1))
newtest$AcceptedDayTime <- ifelse(newtest$AcceptedDayTime=='night', 0,
                                  ifelse(newtest$AcceptedDayTime=='mid-night'
, 0, 1))
```

- Following is to submit the train and test matrix to build xgboost model.

```
# submission code xgboost model
library(xgboost)
set.seed(121)
targets <- newtrain$OutcomeType
newtest$ID<-NULL
full_targets_train <- as.numeric(targets)-1
newtrain$OutcomeType<-NULL
full_train_matrix = matrix(as.numeric(data.matrix(newtrain)),ncol=length(newt
rain))
View(full_train_matrix)
test_matrix <- matrix(as.numeric(data.matrix(newtest)),ncol=length(newtest))
xgb_model_test <- xgboost(data=full_train_matrix,
                    label=full_targets_train,
                    nrounds=150,
                    verbose=1,
                    eta=0.05,
                    max_depth=8,
                    subsample=0.85,
                    colsample_bytree=0.85,
                    objective="multi:softprob",
                    eval_metric="mlogloss",
                    num_class=5)

test_preds <- predict(xgb_model_test, test_matrix)
test_preds_frame <- data.frame(matrix(test_preds, ncol = 5, byrow=TRUE))
#write.csv(test_preds_frame, file = "submission3.csv")
```

Unfortunately, no matter how parameters we change, the attempts of this model is not so effective. Combined with the above three models, none of them are got a satisfied result yet.

The final result for first attempt Xgboost is improve a little bit, but still not ideal.



submission3.csv
3 days ago by SongTang

add submission details

0.88172

**We get into trouble, and wonder do we make mistakes in the data cleaning part? Since the quality of data plays a vital role leading to the final outcome. We should come back again to check the data cleanup.**

### *Refine both dataset to better fit. (Re-do building Xgboost model)*

At first, we though the massive color and breed may impede the outcome, since there are hundreds of color and breed for animals to be classified. After practice, this idea is proved to be inaccurate.

We decide to clean up the data again and keep all unique attribute in color and breed. First, we reclean up color and breed. When the color is mixed by several description words of color, we separate it one by one ("/" or "blank") to make sure that every word to describ color is a separate column. The breed is processed as well. Meanwhile, the attribute of "is Mix" and "pure" is still kept to increase accuracy.

Similarly, we reclean the AcceptedDaytime. We also made the mistake that changing the specific date into a wide range of time. Now, we keep every hour, every weekend, every month and every year instead of time period such as season.

```
#re-clean up the dataset with train dataset and translate each variable to nu
merical value.
library(lubridate)
train<- read.csv('train.csv', header = T, stringsAsFactors = F, na = 0)
train$Hour <- hour(train$DateTime)
train$Month <- month(train$DateTime)
train$Weekdays <- wday(train$DateTime)
train$Year <- year(train$DateTime)


train$Name = ifelse(nchar(train$Name) == 0, "Nameless", train$Name)
train$Name=ifelse(train$Name == 'Nameless', 0, 1)
train$OutcomeSubtype<-NULL
train$AnimalID<-NULL

train$TimeValue = sapply(train$AgeuponOutcome, function(x) strsplit(x, split
= ' ')[[1]][1])
# retrive the time unit,like years, months, days
train$UnitofTime = sapply(train$AgeuponOutcome, function(x) strsplit(x, split
 = ' ')[[1]][2])
#modify the plural number into singular. for example 'years' to 'year'
train$UnitofTime = gsub('s', '', train$UnitofTime)
train$UnitofTime = as.factor(train$UnitofTime) # convert the unit of time int
o factor
train$TimeValue = as.numeric(train$TimeValue) #convert the value of time into
 interger
#compute the age of each animal in day unit by converting TimeValue in days u
sing
#the appropriate multiplier based on different unit.
multiplier = ifelse(train$UnitofTime == 'day', 1, ifelse(train$UnitofTime ==
'week', 7,
                                                         ifelse(train$UnitofT
ime == 'month', 30, ifelse(train$UnitofTime == 'year', 365, NA))))
train$AgeinDays = multiplier * train$TimeValue

#Sex translate, 1 for male, 0 for female
train$Sex <- ifelse(grepl('Male', train$SexuponOutcome), 'Male',
                    ifelse(grepl('Female', train$SexuponOutcome), 'Female', '
Male'))

train$Sex <- ifelse(train$Sex=='Male',1,0)


# find the mix breed or pure breed
train$isMixBreed <- ifelse(grepl("Mix", train$Breed), 'MixBreed',
                           ifelse(grepl("/", train$Breed), 'MixBreed', 'PureB
reed'))
train$isMixBreed <- ifelse(train$isMixBreed=='MixBreed',1,0)
train$isMixColor <- ifelse(grepl('/' , train$Color), 'MixColor',
                           ifelse(grepl('Mix' , train$Color), 'MixColor','Pur
eColor'))
train$isMixColor <- ifelse(train$isMixColor=='MixColor',1,0)

#intact or neutered
train$isIntact <- ifelse(grepl('Intact', train$SexuponOutcome), 'Intact',
                         ifelse(grepl('Unknown', train$SexuponOutcome), 'Neut
ered', 'Neutered'))
train$isIntact <- ifelse(train$isIntact=='Intact',1,0)
train$AnimalType <- ifelse(train$AnimalType=='Dog',1,0)

#Outcome translatetion
train$OutcomeType <- ifelse(train$OutcomeType=='Adoption', 1,
                               ifelse(train$OutcomeType=='Died', 2,
                                      ifelse(train$OutcomeType == 'Euthanasia
', 3,
                                             ifelse(train$OutcomeType == 'Ret
urn_to_owner',4, 5))))

#Delete extra variables
train$TimeValue<-NULL
train$UnitofTime<-NULL
train$AgeuponOutcome<-NULL
train$SexuponOutcome<-NULL
train$DateTime=as.POSIXct(train$DateTime) #date time

#cleanup breed. find all the breed value and each one as a variable, expand t
he train dataset to 300+ columns
all_breeds <- unique(train$Breed)
breed_words <- gsub('/',' ',all_breeds,ignore.case=T)
```

```
breed_words2 <- gsub('Mix','',breed_words,ignore.case=T)
breed_words1 <- unique(unlist(strsplit(breed_words2, " ")))
for (breed in breed_words1){
  train[breed] <- as.numeric(grepl(breed, train$Breed))
}
trainframe=data.frame(train)
train$Breed = NULL


#cleanup color. find all the uniqe color value and each one as a variable, ex
pand the train dataset to 300+ columns
all_train_colors <- unique(train$Color)
color_words <- gsub('/',' ',all_train_colors,ignore.case=T)
color_words1 <- unique(unlist(strsplit(color_words, " ")))
for (color in color_words1){
  train[color] <- as.numeric(grepl(color, train$Color))
}

train$Breed<-NULL
train$Color<-NULL
```

**The following is the refined test dataset. Use the similar method implemented in train dataset cleanup,**

```
library(readr)
test <- read_csv("~/Documents/2017Spring/Data Analytic/term-project/test.csv"
,
                 na = "0")
test$Hour <- hour(test$DateTime)
test$Month <- month(test$DateTime)
test$Weekdays <- wday(test$DateTime)
test$Year <- year(test$DateTime)

#test<- read.csv('test.csv', header = T, stringsAsFactors = F, na = 0)
test$Name = ifelse(nchar(test$Name) == 0, "Nameless", test$Name)
test$Name=ifelse(test$Name == 'Nameless', 0, 1)



test$TimeValue = sapply(test$AgeuponOutcome, function(x) strsplit(x, split =
' ')[[1]][1])
# retrive the time unit,like years, months, days
test$UnitofTime = sapply(test$AgeuponOutcome, function(x) strsplit(x, split =
 ' ')[[1]][2])
#modify the plural number into singular. for example 'years' to 'year'
test$UnitofTime = gsub('s', '', test$UnitofTime)
test$UnitofTime = as.factor(test$UnitofTime) # convert the unit of time into
factor
test$TimeValue = as.numeric(test$TimeValue) #convert the value of time into i
nterger
#compute the age of each animal in day unit by converting TimeValue in days u
```

```r
sing
#the appropriate multiplier based on different unit.
multiplier = ifelse(test$UnitofTime == 'day', 1, ifelse(test$UnitofTime == 'w
eek', 7,
                                                          ifelse(test$UnitofTi
me == 'month', 30, ifelse(test$UnitofTime == 'year', 365, NA))))
test$AgeinDays = multiplier * test$TimeValue

#Sex translate
test$Sex <- ifelse(grepl('Male', test$SexuponOutcome), 'Male',
                    ifelse(grepl('Female', test$SexuponOutcome), 'Female', 'M
ale'))
test$Sex <- ifelse(test$Sex=='Male',1,0)



# mix or pure
test$isMixBreed <- ifelse(grepl("Mix", test$Breed), 'MixBreed',
                           ifelse(grepl("/", test$Breed), 'MixBreed', 'PureBr
eed'))
test$isMixBreed <- ifelse(test$isMixBreed=='MixBreed',1,0)
test$isMixColor <- ifelse(grepl('/' , test$Color), 'MixColor',
                           ifelse(grepl('Mix' , test$Color), 'MixColor','Pure
Color'))
test$isMixColor <- ifelse(test$isMixColor=='MixColor',1,0)

#intact or neutered
test$isIntact <- ifelse(grepl('Intact', test$SexuponOutcome), 'Intact',
                         ifelse(grepl('Unknown', test$SexuponOutcome), 'Neute
red', 'Neutered'))
test$isIntact <- ifelse(test$isIntact=='Intact',1,0)

test$AnimalType <- ifelse(test$AnimalType=='Dog',1,0)

#Delete unuseful variables
test$TimeValue<-NULL
test$UnitofTime<-NULL
test$AgeuponOutcome<-NULL
test$SexuponOutcome<-NULL
test$DateTime=as.POSIXct(test$DateTime)

#breed cleanup (work!!!)
all_breeds <- unique(test$Breed)
breed_words <- gsub('/',' ',all_breeds,ignore.case=T)
breed_words2 <- gsub('Mix','',breed_words,ignore.case=T)
breed_words1 <- unique(unlist(strsplit(breed_words2, " ")))
for (breed in breed_words1){
  test[breed] <- as.numeric(grepl(breed, test$Breed))
}
#trainframe=data.frame(train)
```

```
#train$Breed = NULL


#color clean up
all_test_colors <- unique(test$Color)
color_words <- gsub('/',' ',all_test_colors,ignore.case=T)
color_words1 <- unique(unlist(strsplit(color_words, " ")))
for (color in color_words1){
  test[color] <- as.numeric(grepl(color, test$Color))
}

test$Breed<-NULL
test$Color<-NULL
```

After redo the dataset cleanup part, we have 305 columns for new train dataset and 290 columns for new test dataset. In this way, we could predict the outcome type for each animals in test dataset with higher accuracy.

### Model 5: Improved Xgboost Model

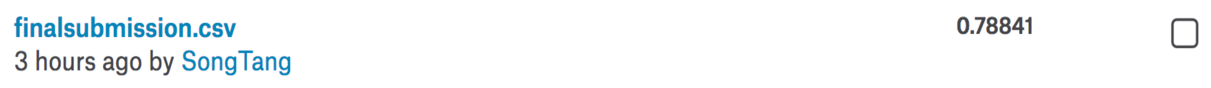
We build a new XGB model with the new cleaned dataset. Before we try to build the model, first we should convert the dataset into numeric matrix format.

```
library(xgboost)
set.seed(121)
targets <- train$OutcomeType
targets_train <- as.numeric(targets)-1 #Outcome Type variable
train$OutcomeType<-NULL
full_train_matrix = matrix(as.numeric(data.matrix(train)),ncol=length(train))
test$ID<-NULL #Delete irrelevant variables
test_matrix <- matrix(as.numeric(data.matrix(test)),ncol=length(test))
model = xgboost(data=full_train_matrix,
                label=targets_train,
                nrounds=200,  #200 is the best fit
                verbose=1,
                eta=0.03,
                max_depth=8,
                subsample=1,
                colsample_bytree=0.85,
                objective="multi:softprob",
                eval_metric="mlogloss",
                num_class=5)

## [1]  train-mlogloss:1.574576
## [2]  train-mlogloss:1.541791
## [3]  train-mlogloss:1.510994
## [4]  train-mlogloss:1.481862
## [5]  train-mlogloss:1.454260

#omit the rest
```

```r
#following is to use that model to create prediction in test matrix

test_preds <- predict(model, test_matrix)
test_preds_frame <- data.frame(matrix(test_preds, ncol = 5, byrow=TRUE))
write.csv(test_preds_frame, file="finalsubmission.csv")
```

The result shows that it does works and way more better than the former ones!

### Result snapshoot



finalsubmission.csv
3 hours ago by SongTang

0.78841

## *Conclusion*