PetFinder Adoption Prediction

George Cruz Karim Hammoud Maliat Islam Gabriella Martinez Ken Popkin

Date: 2021-12-12 Due: 2021-12-12

Abstract

There are millions of stray pets around the world, some of which are fortunate enough to be adopted while many others are not. While adoption of a pet is often the definition of success, the rate at which a pet is adopted is also a key success factor - pets that take a long time to adopt contribute to over-crowded animal shelters and can prevent taking on new strays. Sadly, pets that are not adopted eventually need to be euthanized.

Contents

1 Overview

There are millions of stray pets around the world, some of which are fortunate enough to be adopted while many others are not. While adoption of a pet is often the definition of success, the rate at which a pet is adopted is also a key success factor - pets that take a long time to adopt contribute to over-crowded animal shelters and can prevent taking on new strays. Sadly, pets that are not adopted eventually need to be euthanized.

1.1 Learn more about the data [1]

Variable	Description
PetID	Unique hash ID of pet profile
AdoptionSpeed	**response variable** Categorical speed of adoption. Lower is faster.
Type	Type of animal (1 = Dog, 2 = Cat)
Name	Name of pet (Empty if not named)
Age	Age of pet when listed, in months
Breed1	Primary breed of pet (Refer to BreedLabels dictionary)
Breed2	Secondary breed of pet, if pet is of mixed breed (Refer to BreedLabels dictionary)
Gender	Gender of pet (1 = Male, 2 = Female, 3 = Mixed, if profile represents group of pets)
Color1	Color 1 of pet (Refer to ColorLabels dictionary)
Color2	Color 2 of pet (Refer to ColorLabels dictionary)
Color3	Color 3 of pet (Refer to ColorLabels dictionary)
MaturitySize	Size at maturity (1 = Small, 2 = Medium, 3 = Large, 4 = Extra Large, 0 = Not Specified)
FurLength	Fur length (1 = Short, 2 = Medium, 3 = Long, 0 = Not Specified)
Vaccinated	Pet has been vaccinated (1 = Yes, 2 = No, 3 = Not Sure)
Dewormed	Pet has been dewormed (1 = Yes, 2 = No, 3 = Not Sure)
Sterilized	Pet has been spayed / neutered (1 = Yes, 2 = No, 3 = Not Sure)
Health	Health Condition (1 = Healthy, 2 = Minor Injury, 3 = Serious Injury, 0 = Not Specified)
Quantity	Number of pets represented in profile
Fee	Adoption fee (0 = Free)

[1] https://www.kaggle.com/c/petfinder-adoption-prediction/data

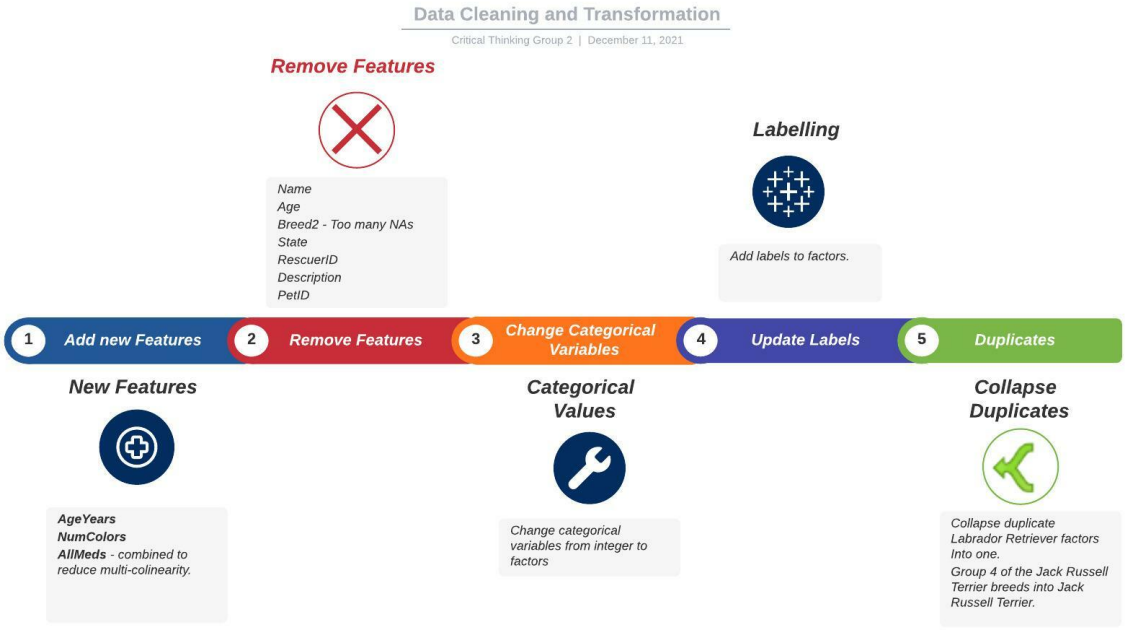
Variable	Description
State	State location in Malaysia (Refer to StateLabels dictionary)
RescuerID	Unique hash ID of rescuer
VideoAmt	Total uploaded videos for this pet
PhotoAmt	Total uploaded photos for this pet
Description	Profile write

1.2 What to Predict?

Predictor (Adoption Speed)
Description: Predict how quickly, if at all, a pet is adopted.

The values are determined in the following way:
- 0. Pet was adopted on the same day as it was listed.
- 1. Pet was adopted between 1 and 7 days (1st week) after being listed.
- 2. Pet was adopted between 8 and 30 days (1st month) after being listed.
- 3. Pet was adopted between 31 and 90 days (2nd & 3rd month) after being listed.
- 4. No adoption after 100 days of being listed.

2 Data Cleaning and Transformation



2.1 New Features

We added the following new features:

Variable	Description
AgeYears	provided in months this is too many variations. Decided to convert to new feature of AgeYears and drop Age
NumColors	derived by counting the number of colors a pet has using the `Color1`,`Color2`, and `Color3` features.
AllMeds	the sum of Vaccinated, Dewormed, and Sterilized. Combining these into one feature could potentially reduce multi-collinearity

2.2 Remove Features

We decided to remove these features from the data set as inconsequential to our analysis:

Variable	Description
Name	this is a text field and new owners can (and usually do) rename their pets, so removing this feature
Age	replaced with AgeYears *(see above)*
Breed2	$10,700$ of almost $15,000$ rows are populated with 0 *(unknown)*, so this doesn't seem like a good feature to keep
State	initially kept this, but the correlation to AdoptionSpeed is only about 2%. Decided to remove it
RescuerID	common sense is that this field will not have any predictive value for adoption rate
Description	this will have value in future analysis for NLP, but this will be evaluated differently in another notebook
PetID	same reason as RescuerID

2.3 Other data cleaning and transformation

We performed some other data cleaning and transformation tasks like:

1. Change categorical variables from integer to factors.

2. Add Labels by recoding certain values:
   2.1. **Type**: "1" = "Dog", "2" = "Cat"
   2.2. Import the color labels to use the names of the colors.
   2.3. Adding labels to `Gender` variable per definitions assigned to create `GenderLabel`.
   2.3. Adding labels to `MaturitySize`, `Health` and `FurLength`.
   2.4. Adding labels to `Vaccinated`, `Dewormed`, `Sterilized`. A number of observations in these columns are `Not Sure` which can be interpreted as `NA`. Similarly, the `VideoAmt` variable contains 14419 observations with 0 videos so the 0s have been converted to `NA`.
   2.5. Converting the `breedname` and `AllMeds` variables to factor since the original data points used to create these were factors.

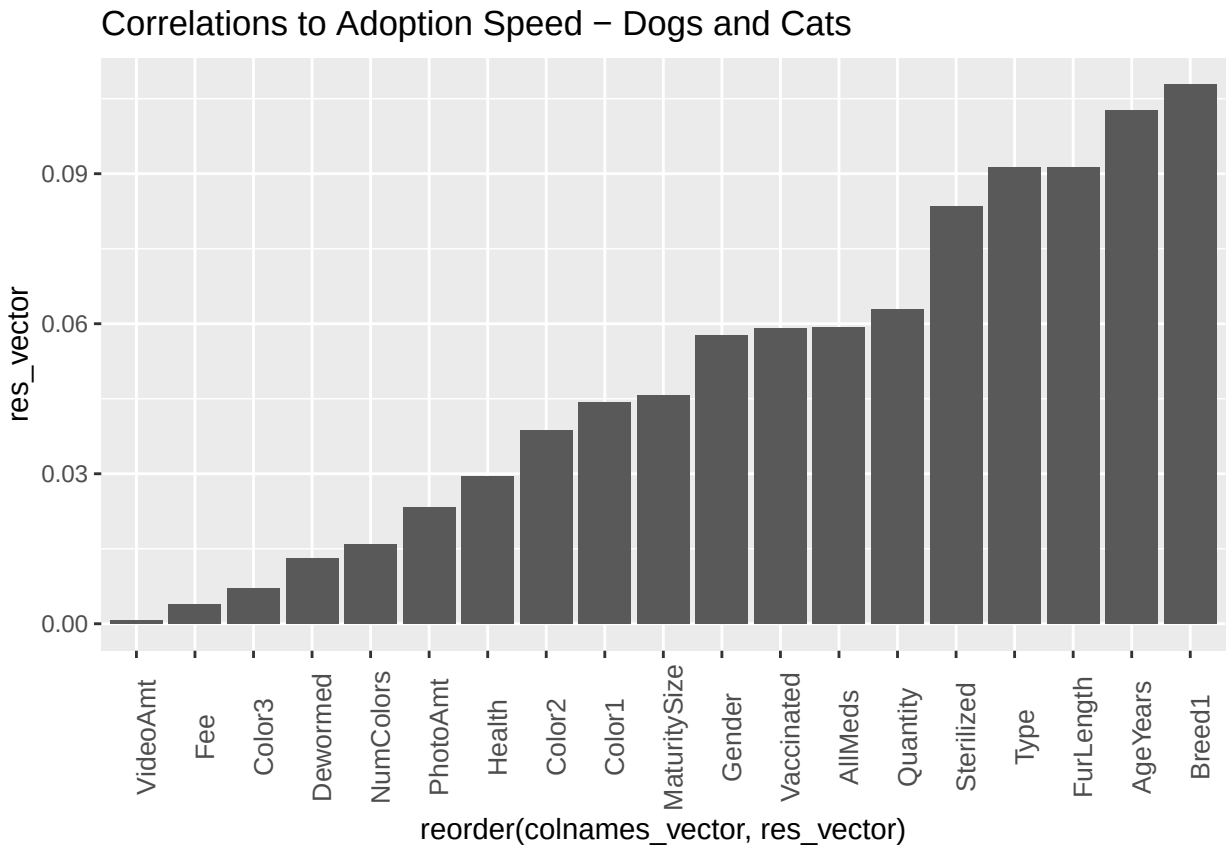
3. We collapsed duplicate Labrador Retriever factors (Black, Chocolate, and Yellow) into one as they are not considered to be separate ^https://www.akc.org/expert-advice/dog-breeds/retriever-breeds/].
4. We also grouped 4 of the Jack Russell Terrier (Parson Russell Terrier) breeds into Jack Russell Terrier.

3 Data Analysis and visualization

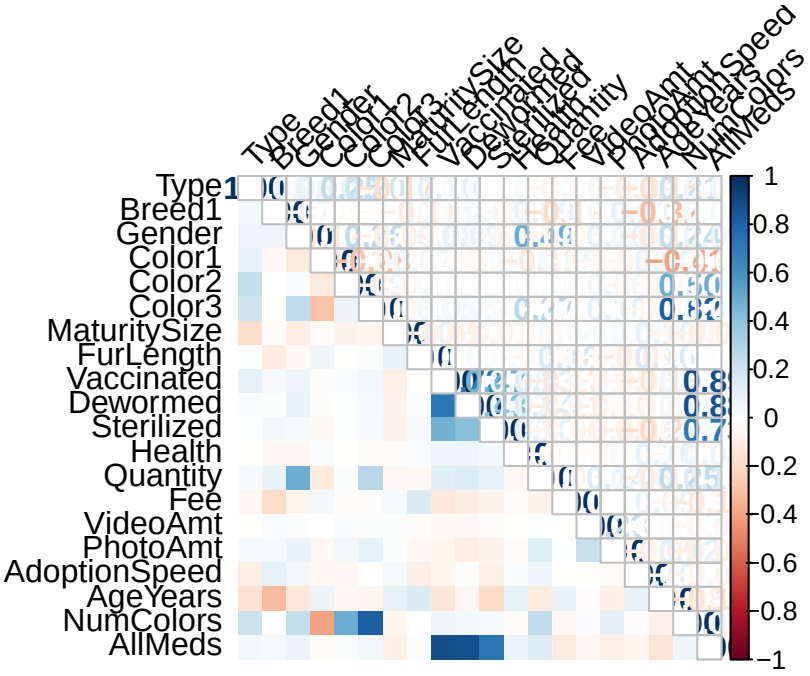
We explore our data to investigate, among other things, the correlation among variables in the data that could be used in our models.

3.1 Feature Correlation

3.1.1 Correlation of each feature to AdoptionSpeed



3.1.2 Visual Correlation between features



3.2 Dog Breed Word Cloud

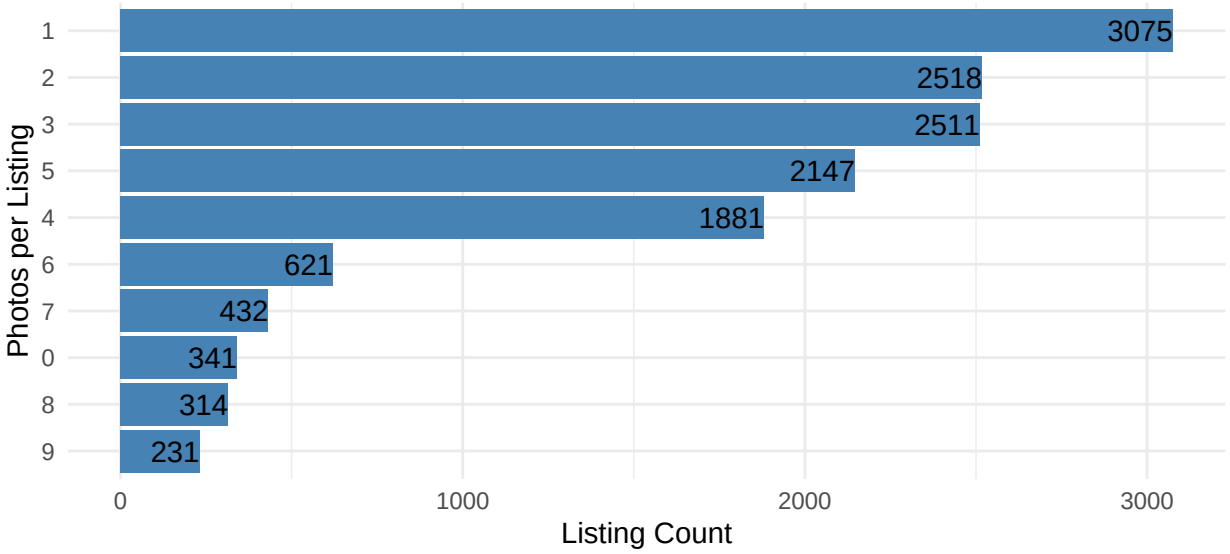
Below we use `wordcloud` to visualize the dog breeds in our data. Note the Mixed Breed has been removed as it was skewing our data with the highest number of observations with 5923 total.

3.3 Cat Breed Word Cloud

Below we use `wordcloud` to visualize the cat breeds in our data.

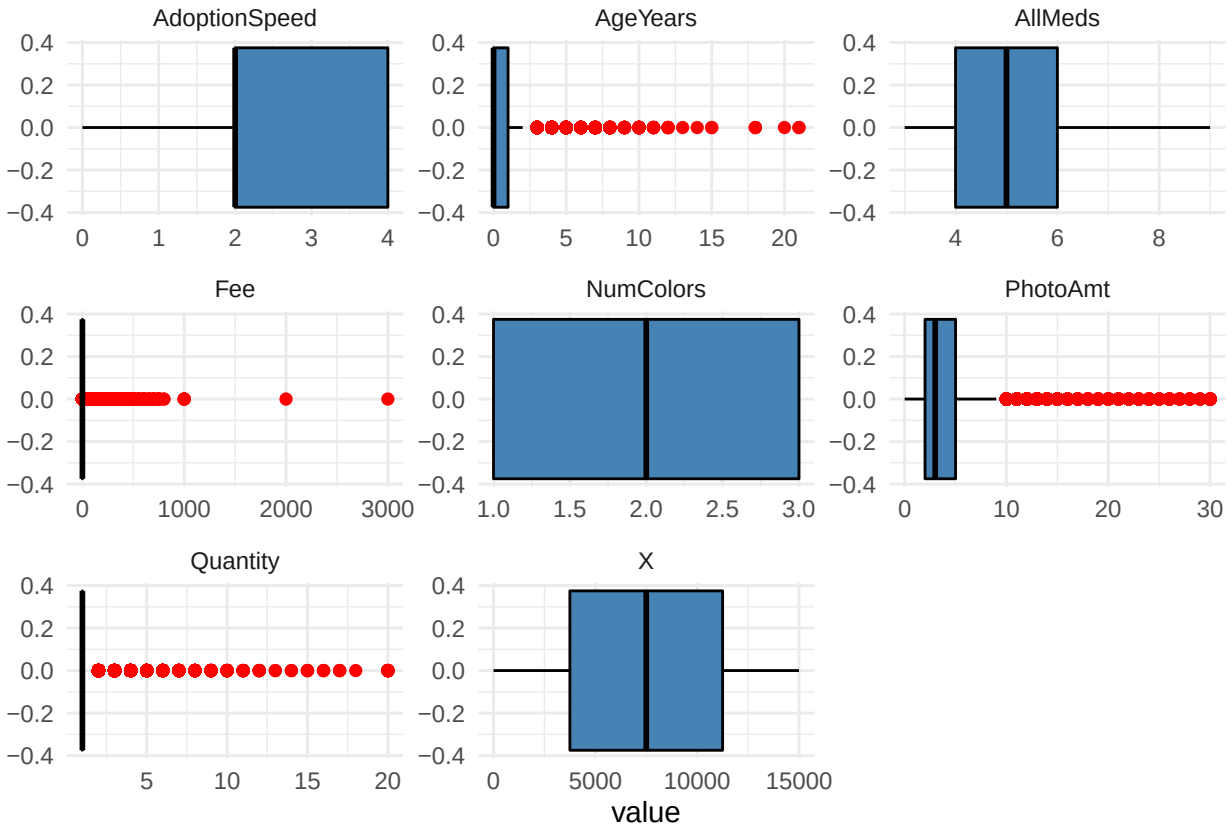
3.4 Bar plots

As part of exploring the data, below we look into our response variable and see which level of `AdoptionSpeed` has the most occurrences.



3.5 Distributions of Numeric Variables

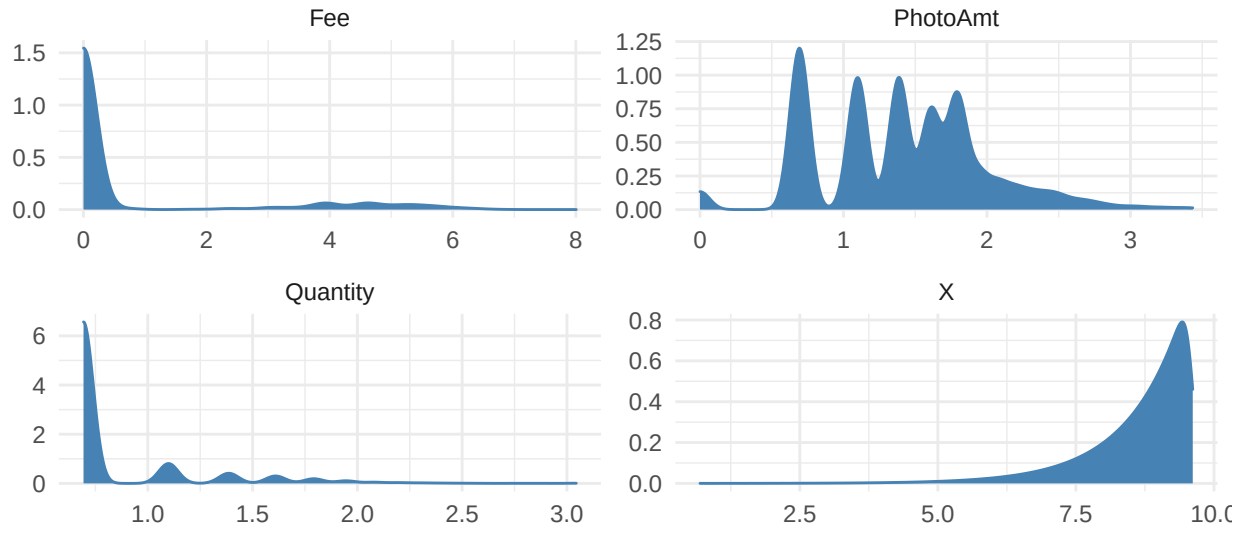
We subset our numeric predictor variables to check their distributions.



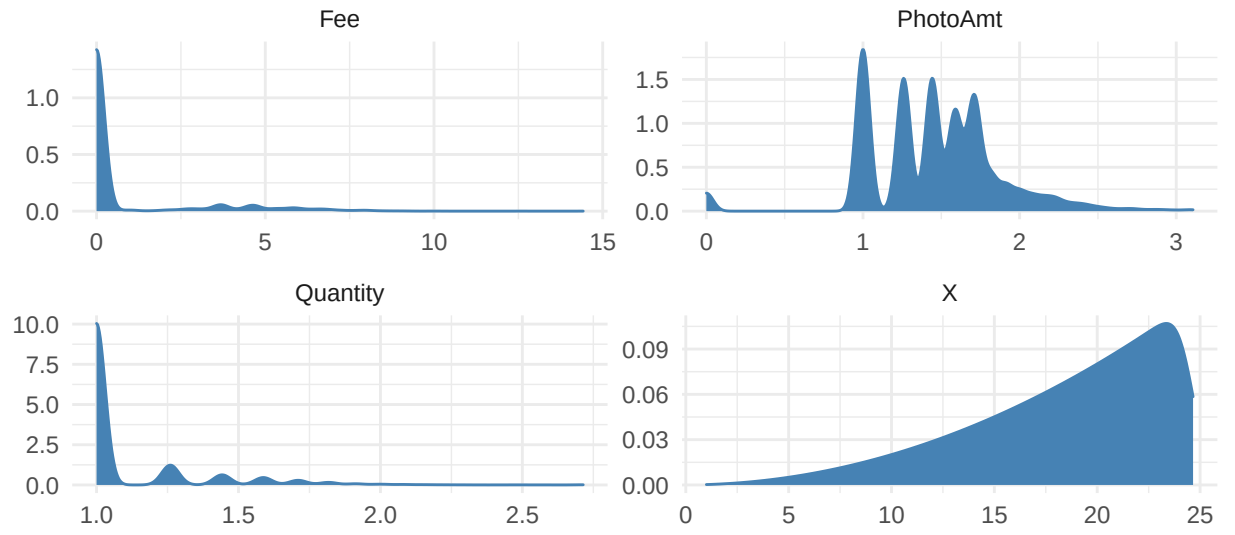
3.6 Transformations of numeric variables

Next we will take transformations of the variables reviewed above to see if the transformed variables are worth looking into and using for our model.

3.7 Log Transformations



3.8 Cube root transformations



Based on the above, all variables are still heavily right skewed with the exception of `PhotoAmt` which made some improvement, but still not normally distributed.

4 Data Modeling

4.1 Ordinary Logistic Regression Model

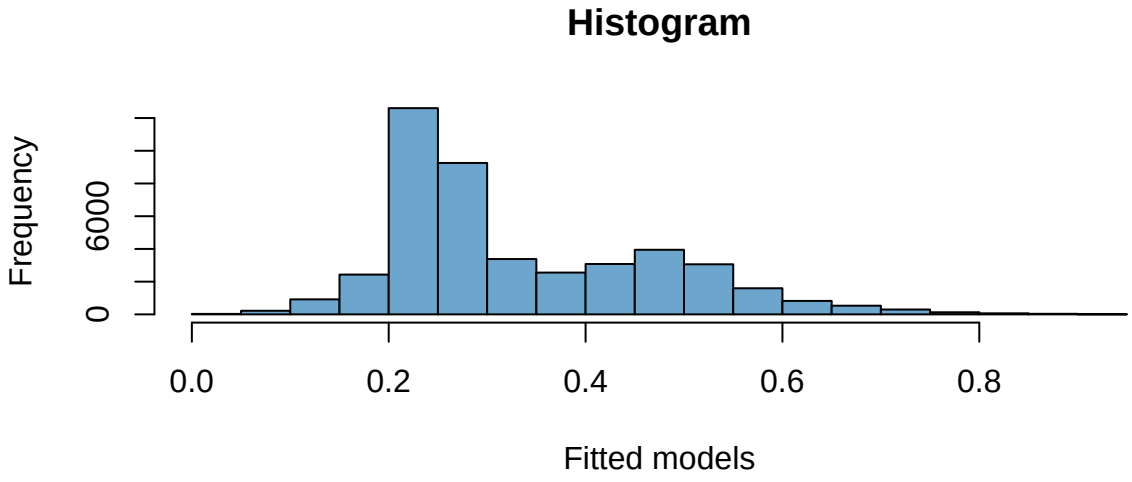
We created several models using ordinal logistic regression since response variable can be considered an ordinal factor. Ordinary Logistic Regression

```
## Call:
## polr(formula = euth_risk ~ Type + (Gender + Color1 + MaturitySize +
##     FurLength + Health + Quantity + Fee + PhotoAmt + AgeYears +
##     NumColors + purebreed + AllMeds), data = train_data2, Hess = TRUE)
##
## Coefficients:
##                           Value Std. Error t value
## TypeDog                0.6465752  0.0539215 11.9910
## GenderMale             0.0372710  0.0344366  1.0823
## GenderMixed            0.0082918  0.0618420  0.1341
## Color1Brown           -0.0076983  0.0402321 -0.1913
## Color1Cream            0.1009893  0.0696028  1.4509
## Color1Golden           0.0965965  0.0676829  1.4272
## Color1Gray             0.0995334  0.0797473  1.2481
## Color1White            0.1525216  0.0835323  1.8259
## Color1Yellow          -0.0930997  0.0796500 -1.1689
## MaturitySizeLarge     -0.9301085  0.3582734 -2.5961
## MaturitySizeMedium    -0.7983518  0.3556510 -2.2448
## MaturitySizeSmall     -0.8899883  0.3564297 -2.4970
## FurLengthMedium       -0.2220177  0.0715048 -3.1049
## FurLengthShort        -0.2108798  0.0706601 -2.9844
## HealthMinor Injury    -0.1949782  0.0892682 -2.1842
## HealthSerious Injury -0.0990709  0.3472700 -0.2853
## Quantity              -0.0880270  0.0153361 -5.7399
## Fee                   -0.0006377  0.0002148 -2.9696
## PhotoAmt               0.0804304  0.0052383 15.3543
## AgeYears              -0.1227725  0.0124293 -9.8776
## NumColors2             0.1539906  0.0410001  3.7559
## NumColors3             0.1028811  0.0486792  2.1135
## purebreedYes           0.6861984  0.0554029 12.3856
## AllMeds4               0.5020650  0.0551127  9.1098
## AllMeds5               0.3858815  0.0564949  6.8304
## AllMeds6               0.4401485  0.0537605  8.1872
## AllMeds7               0.1436685  0.0948477  1.5147
## AllMeds8               0.7181419  0.1009836  7.1115
## AllMeds9              -0.1911094  0.0732132 -2.6103
##
```

```
## Intercepts:
##             Value   Std. Error t value
## High|Low   -0.7194  0.3711    -1.9386
## Low|Medium  0.3275  0.3711     0.8827
##
## Residual Deviance: 30510.98
## AIC: 30572.98
## (5 observations deleted due to missingness)
```

4.2 OLR Histogram

After dealing with multicollinearity and insignificant values, we see that the model with the lowest AIC value of the models created is `model_2` with an AIC of 29952.42.



4.3 OLR Predictions

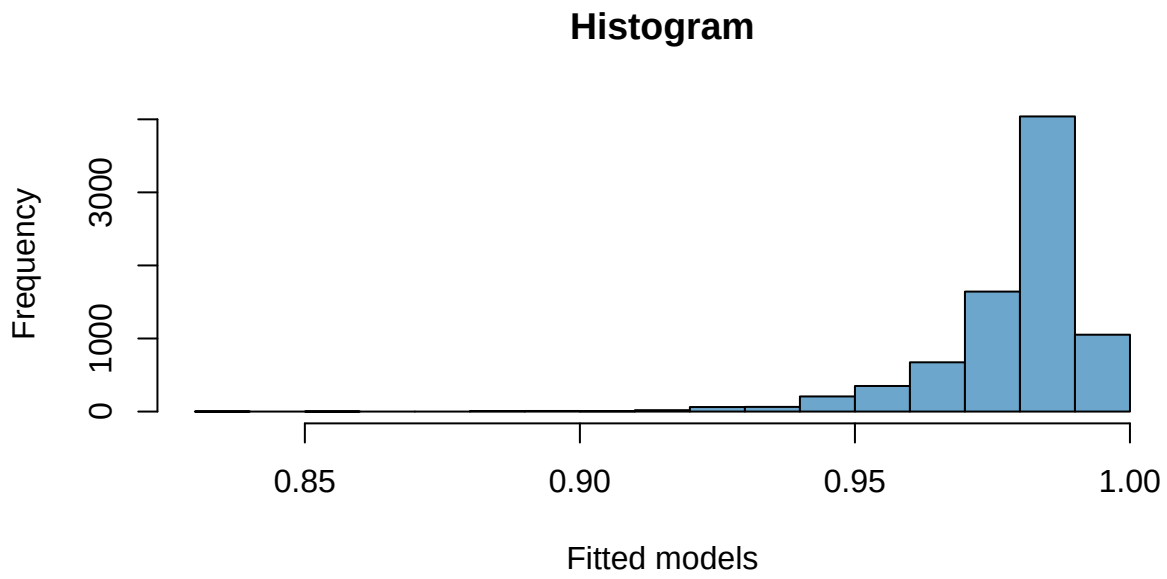
Below we generate our predictions with the `predict` function Toward Data Science - OLR.

```
##        High       Low    Medium
## 1 0.2930491 0.2484305 0.4585204
```

This model predicts that our Evaluation_Pet will be a medium risk of euthanasia where it will adopted in either months 1, 2 or 3 of being listed.

4.4 Binomial Logistic Regression Model

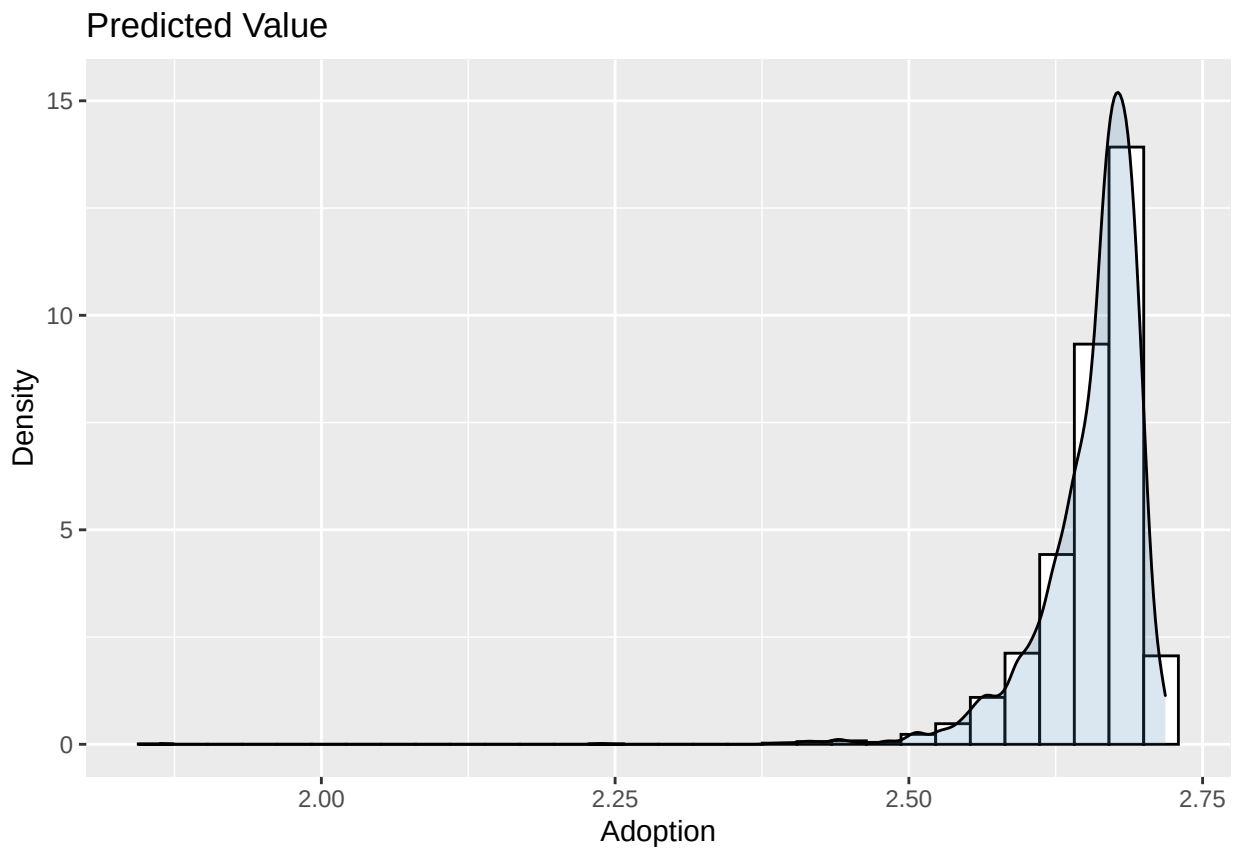
We built a Binomial Logistic Regression Model using a subset of the training data of only dogs. After using the Step Method we arrived at the best fitted model.



Here we can see the model works really well on the train data, we need to use it on the evaluation data and check the accuracy of the model on it.

4.5 BLR Predictions

When we apply the model to the evaluation data we get the following results.

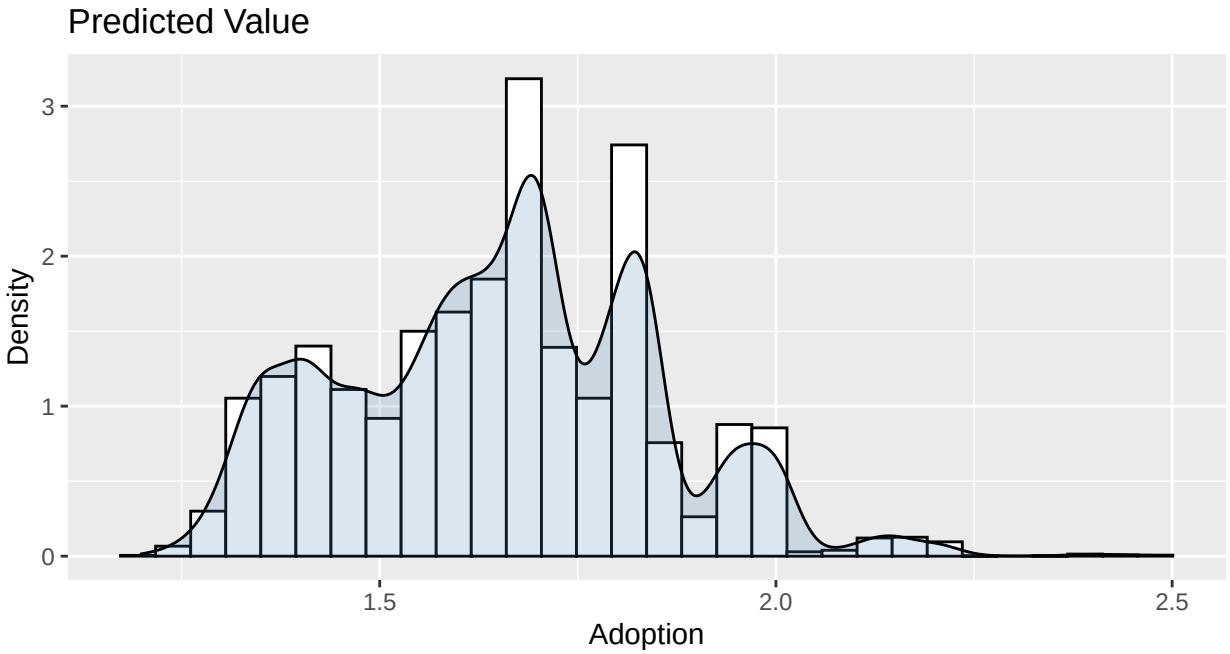


We can see that the adoption values is more skewed to the right, the density of the adoption rate is more around 2.5 to 2.75

4.6 Negative Binomial Model

We also trained several Negative Binomial Models and found the best fit would be the one that uses a few significant variables.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1813  0.4076  0.5103  0.4952  0.5852  0.9092
```

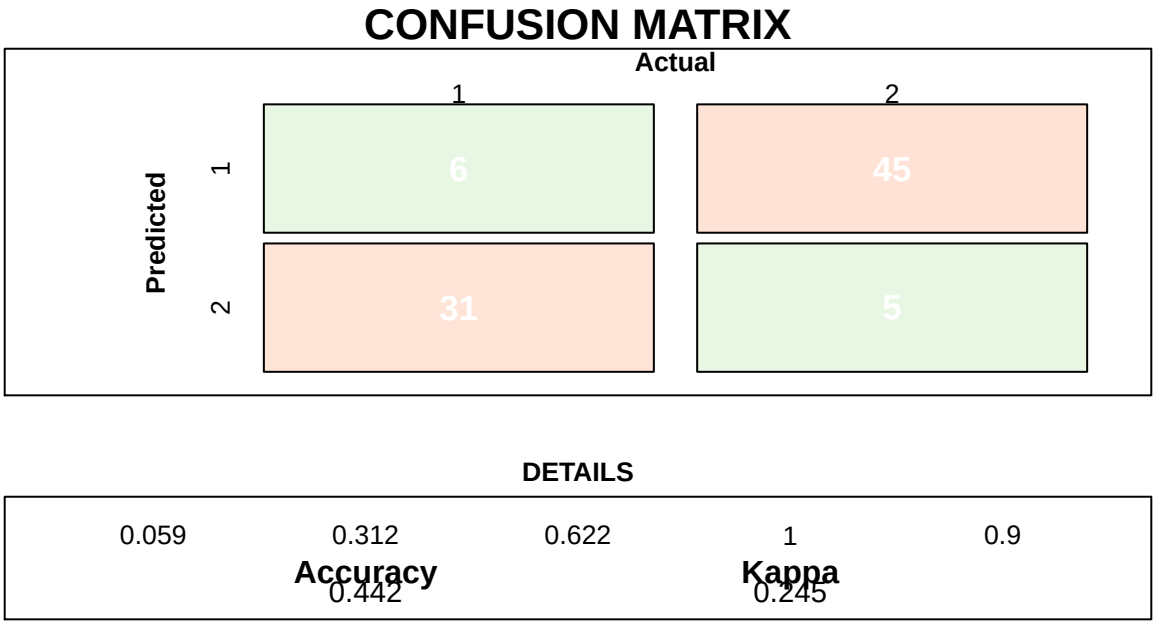


It can be observed that based on the prediction that Mean is 0.4953 and Median is 0.518.

4.7 Random Forest, XGBoost

Helpful article from TowardsDataScience

We explored other models like Random Forest and XGBoost. However, we were not able to get a better fit than with Binomial or OLR. Here a confusion matrix from one of our RF Models:



5 Conclusions

We tried several models:
- Ordinary Logistic Regression
- Binomial Logistic Regression
- Negative Binomial
- Linear Modeling
- Random Forest
- XGBoost

With the data provided, our best fitted models were the **OLR** and the **BLR**.

6 References

Kaggle - About the data. "https://www.kaggle.com/c/petfinder-adoption-prediction/data". Dec, 2021.
UCLA - Ordinary Logistic Regression. "https://stats.idre.ucla.edu/r/dae/ordinal-logistic-regression/". Dec 2021.
TowardsDataScience - Random Forest in R. "https://towardsdatascience.com/random-forest-in-r-f66adf80ec9". Dec 2021.
Toward Data Science - OLR. "https://towardsdatascience.com/implementing-and-interpreting-ordinal-logistic-regression-1ee699274cf5". Dec 2021.

7 Appendix A.

7.1 Notable Code.

```
### Function to Create Confusion Matrix

create_rf_confusion_matrix <- function(pred, test){

  #Creates vectors having data points
  AdoptionSpeed_test = as.numeric(unlist(test$AdoptionSpeed))
  expected_value <- factor(c(AdoptionSpeed_test))

  predicted_value <- factor(c(pred))

  #Creating confusion matrix
  rf_confusion_matrix <- confusionMatrix(data=predicted_value,
                                         reference = expected_value,
                                         positive='1')

  #Results
  return(rf_confusion_matrix)
}

### Function to draw confusion matrix
### Credit: https://stackoverflow.com/questions/23891140/r-how-to-visualize-confusion-
draw_confusion_matrix <- function(cm) {

  total <- sum(cm$table)
  res <- as.numeric(cm$table)

  # Generate color gradients. Palettes come from RColorBrewer.
  greenPalette <- c("#F7FCF5",
                    "#E5F5E0",
                    "#C7E9C0",
                    "#A1D99B","#74C476",
                    "#41AB5D","#238B45",
                    "#006D2C","#00441B")
  redPalette <- c("#FFF5F0","#FEE0D2",
                  "#FCBBA1","#FC9272",
                  "#FB6A4A","#EF3B2C",
                  "#CB181D","#A50F15",
                  "#67000D")
  getColor <- function (greenOrRed = "green", amount = 0) {
```

```
  if (amount == 0)
    return("#FFFFFF")
  palette <- greenPalette
  if (greenOrRed == "red")
    palette <- redPalette
  colorRampPalette(palette)(100)[10 + ceiling(90 * amount / total)]
}

# set the basic layout
layout(matrix(c(1,1,2)))
par(mar=c(2,2,2,2))
plot(c(100, 345), c(300, 450), type = "n",
     xlab="", ylab="", xaxt='n', yaxt='n')
title('CONFUSION MATRIX', cex.main=2)

# create the matrix
classes = colnames(cm$table)
rect(150, 430, 240, 370, col=getColor("green", res[1]))
text(195, 435, classes[1], cex=1.2)
rect(250, 430, 340, 370, col=getColor("red", res[3]))
text(295, 435, classes[2], cex=1.2)
text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
text(245, 450, 'Actual', cex=1.3, font=2)
rect(150, 305, 240, 365, col=getColor("red", res[2]))
rect(250, 305, 340, 365, col=getColor("green", res[4]))
text(140, 400, classes[1], cex=1.2, srt=90)
text(140, 335, classes[2], cex=1.2, srt=90)

# add in the cm results
text(195, 400, res[1], cex=1.6, font=2, col='white')
text(195, 335, res[2], cex=1.6, font=2, col='white')
text(295, 400, res[3], cex=1.6, font=2, col='white')
text(295, 335, res[4], cex=1.6, font=2, col='white')

# add in the specifics
plot(c(100, 0), c(100, 0), type = "n", xlab="",
     ylab="", main = "DETAILS", xaxt='n', yaxt='n')
text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
```

```
  text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
  text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
  text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

  # add in the accuracy information
  text(30, 35, names(cm$overall[1]), cex=1.5, font=2)
  text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)
  text(70, 35, names(cm$overall[2]), cex=1.5, font=2)
  text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)
}

#### Function to create train and test

partition_data <- function(data, isTest) {
  sample <- sample.split(data$AdoptionSpeed,
                         SplitRatio = .75)
  train <- subset(data,
                  sample == isTest)

  return(train)
}

#Create train data
create_train_data  <- function(data) {

  new_data <- partition_data(data, TRUE)

  return(new_data)
}

#Create test data
create_test_data  <- function(data) {

  new_data <- partition_data(data, FALSE)

  return(new_data)
}


rf_model_func <- function(data) {
  #train, test data for RF model 1
  rf_train = create_train_data(data)
  rf_test = create_test_data(data)
```

```
  #create RF model 1
  rf <- randomForest(
    AdoptionSpeed ~ .,
    data=rf_train
  )

  #Predict AdoptionSpeed with RF model 1
  rf_pred = predict(rf, newdata=rf_test)

  #Create a confusion matrix for RF model 1
  rf_confusion_matrix = create_rf_confusion_matrix(rf_pred, rf_test)
  return(rf_confusion_matrix)
}

xg_model_func <- function(data) {
  #train, test data for XG model 1
  xg_train = as.matrix(create_train_data(data))
  xg_test = as.matrix(create_test_data(data))

  #separate out the label
  adoption_speed_train = xg_train[ , ncol(xg_train)]
  adoption_speed_test = xg_test[ , ncol(xg_test)]

  #drop AdoptionSpeed from train and test
  xg_train = subset(xg_train, select=-c(ncol(xg_train)))
  xg_test = subset(xg_test, select=-c(ncol(xg_test)))

  #XGBoost requires a dgc matrix
  xg_train=as(xg_train, "dgCMatrix")
  xg_test=as(xg_test, "dgCMatrix")

  #Create the model
  xg <- xgboost(data = xg_train,
                label = adoption_speed_train,
                nrounds = 300, verbose=0)

  #Predict
  xg_pred <- predict(xg, xg_test)

  #Create a confusion matrix for XG model 1
  xg_pred = factor(as.integer(xg_pred))
  adoption_speed_test = factor(as.integer(adoption_speed_test))

  xg_confusion_matrix = confusionMatrix(xg_pred,
```

```
                                          adoption_speed_test)
  return(xg_confusion_matrix)
}
```

...