STAT 340 - Final Project Report
Analyzing the Trend in Animals' Outcomes

Irene Nguyen, Sabirah Shuaybi, Nhu Thao Do

December 28, 2018

Abstract

According to the pet statistics in the US, every year, approximately 7.6 millions animals end up in the shelters. They are mostly abandoned by the owners or rescued out of cruelty situations. It is unfortunate that not all animals which are treated cruelly are rescued. That is why animal advocates are working hard to raise awareness and to reduce the number of cases of animal abuse. Since we care about improving the lives of animals, we also want to contribute something to make sure all animals have a home. As for our motivation for this project, we chose this data set on animal outcomes because we want to look for the trend in the outcomes of animals that end up in shelters and therefore, predict the outcome of an animal given their attributes: color, breed, sex, and age. This will help shelters focus their energy on certain kinds of animal that need more help to find a new home.

1 Data Overview and Data Cleaning

The dataset, which is retrieved from Kaggle, has 10 major variables and 26729 observations. The variables are Animal ID, Name, Date and Time, Outcome Type, Outcome Subtype, Animal Type, Sex upon Outcome, Age upon Outcome, Breed, and Color. We chose to focus on the Animal Outcome as our response variable. We also did not use all of the variables as the predictors since some of them are not so relevant in predicting the animal outcome. There are five major animal outcomes: adoption, return to owner, transfer, died, and euthanasia. There are two animal types: cat and dog. As for Sex upon Outcome, there are neutered male, intact male, spayed female, intact female and unknown.

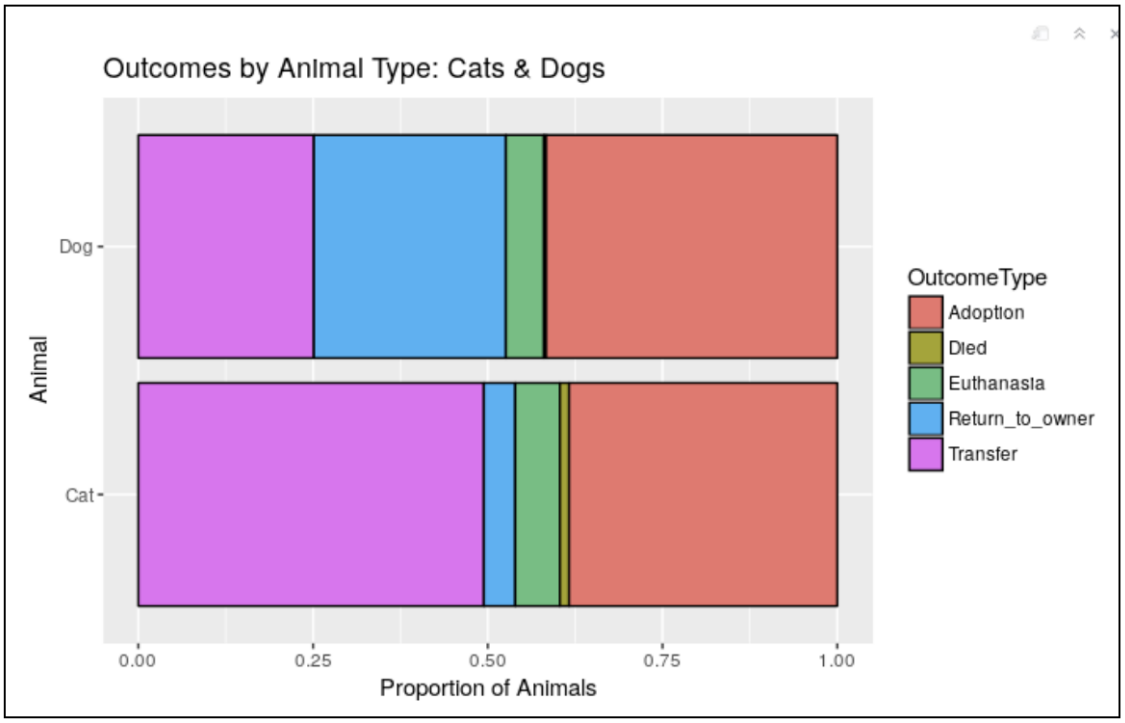
The original dataset was quite messy to work with directly so we did clean the data so it is easier to work with for our purpose. For Age upon Outcome, we turned all the ages in different units (years, months) to days and stored them in a variable called Age in Days. Then we classify animals under one year old to be baby and above that to be adult. Therefore we created a new variable called Life Stage with two level: adult and baby. The variable Intact is coded based on the intactness of the animal, with 1 being intact and 0 being not intact. We also simplified two variables since they have many different levels: Color and Breed to Simple Color and Simple Breed. For Color, some animals have more than 2 different colors

so we just took the first color that appears and set it as the main color for the given animal. For Breed, we did the same thing since some animals are mixed. The variable IsMix was also created with 1 being mixed and 0 being non-mixed.

The next step after cleaning the data and creating new variables was to look at the missing data. There are 18 missing data points in the Age in Days variable and several missing data points in Name and Outcome Subtype. Since we would not be using Name and Outcome Subtype, we only imputed the missing data in Age in Days using the median age of the animals.

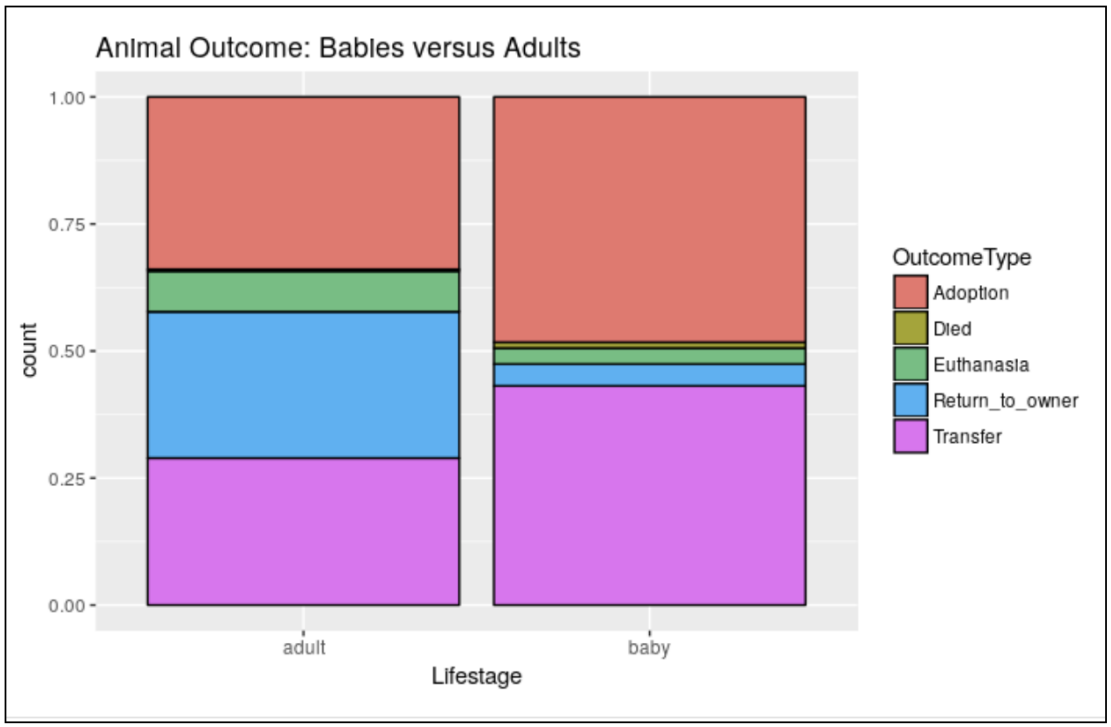
2 Data Visualization

There are some variables that we are interested in investigating further their effects on the animal outcomes: Animal Type, Sex upon Outcome, Lifestage, Breed, and Color. For Breed and Color, there are more than 100 levels so we will not visualize them since the graphs are complicated and not very helpful.



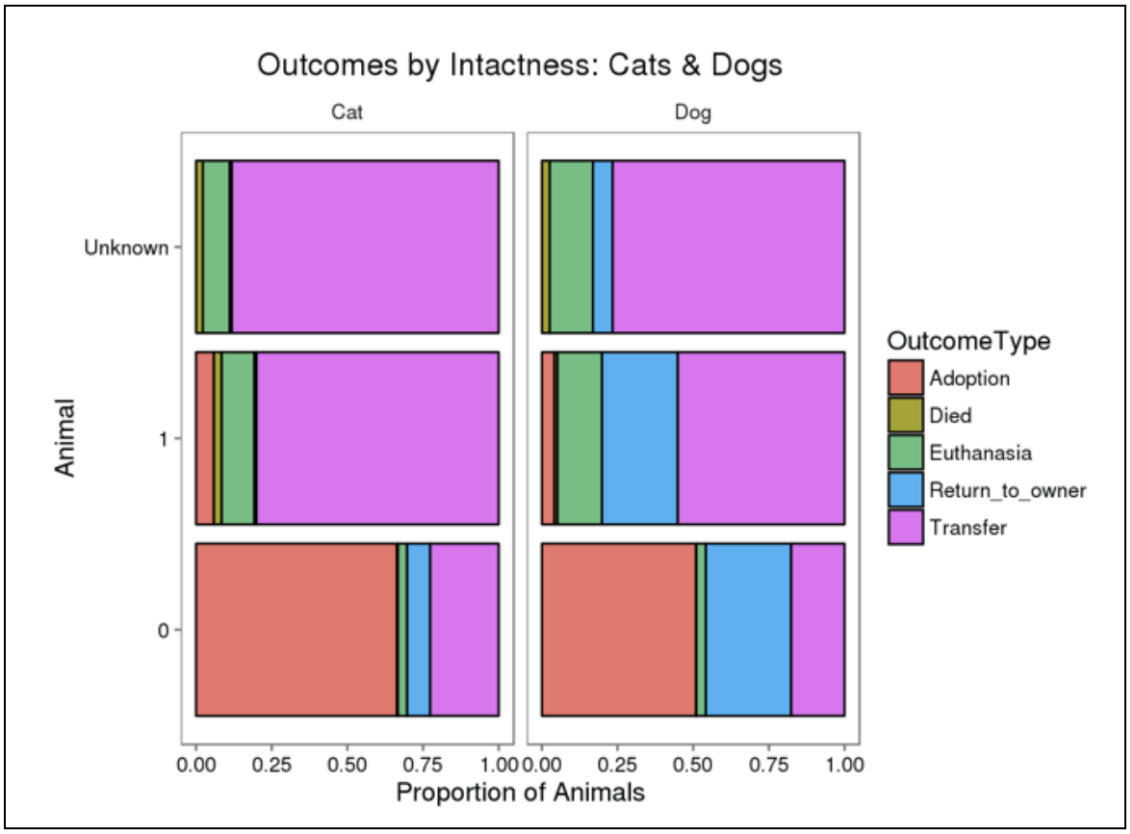
The first factor that we want to look at is the animal type: cat and dog. This graph visualizes the outcomes of cats and dogs. Both cats and dogs are equally likely to be adopted. However, cats are twice as likely to be transferred to different shelters than dogs. Dogs are approximately four times more likely to be returned to owners than cats. The numbers of

cats and dogs that got euthanized are approximately the same.



Life stage is also an important factor in determining the outcomes of the animals. The three major outcomes for adult animals are adoption, return to owner and transfer. For baby animals, there are two dominating outcomes: adoption and transfer. Baby animals are almost twice as likely to be adopted compared to adult animals. However, baby animals are also twice more likely to be transferred compared to adult animals. Approximately a third of the adult animals are being returned to owner whereas, for the baby animals, this

proportion is very small.



The intactness of the animals is also a vital factor that affects the outcomes. The variable Intact has 3 levels: 1 (intact), 0 (neutered or spayed), and Unknown. Animals which are neutered or spayed are more likely to be adopted compared to intact animals. A higher proportion of intact animals is transferred compare to non-intact animals.

3 Methods and Results

3.1 Gradient Tree Boosting

Our first statistical method was the use of Gradient Tree Boosting (Classification). The reason we went with gradient boosting algorithm as opposed to a random forest fit is because 1) It takes less time and was thus the more feasible option due to time-constraints. 2) boosting is an effective ensemble method where predictors are not made independently/randomly (such as in random forests) but sequentially. This is one way to ensure that the component models are distinct rather than hoping that they will be. For our first gradient tree boosting model, we experimented with 5 predictors: Animal Type, Lifestage, Is Mix, Sex, and Simple color. The test set error rate for our final model is quite high, about 0.522.

Because in gradient boosting, the new predictors are learning from mistakes committed by previous predictors, it takes less time/iterations to reach close to actual predictions. But

we have to choose the stopping criteria carefully or it could lead to overfitting on training data. The first tuning parameter is nrounds, which is the number of component models. We tried different values from 5, 10, 20 to 100 for this parameter. The second tuning parameter we manipulated was learning rate. If learning rate of the algorithm is set to be too high, this can result to overfitting. The learning rates that we experimented with were: 0.5, 0.6, and 0,7 and they are just a little higher than the default learning rate. The final parameter that we manipulated was subsample, which is the proportion of observations to use to grow each tree. The subsample values that we tried were 0.5, 0.9, 1. For max depth, we experimented with all values from 1 to 5.

After experimenting with all the parameters, we chose the model with highest cross validation accuracy. This model has 10 component models, learning rate of 0.5, subsample of 0.5 and max depth of 3.

nrounds	double [1]	10
max_depth	integer [1]	3
eta	double [1]	0.5
gamma	double [1]	0
colsample_bytree	double [1]	1
min_child_weight	double [1]	1
subsample	double [1]	0.5

Since one of our goal is to predict the outcomes and that the first model did not do so well, we added more explanatory variables to see whether the test set performance improves. Our second model contained the original 5 predictors from the first model (AnimalType, Lifestage, IsMix, Sex, and Simple Color) as well as SimpleBreed. And the result showed that it performed better than model 1 with a new test set error rate of 0.378. We also experimented with the different values for the tuning parameters and chose the model with highest accuracy.

nrounds	double [1]	20
max_depth	integer [1]	4
eta	double [1]	0.6
gamma	double [1]	0
colsample_bytree	double [1]	1
min_child_weight	double [1]	1
subsample	double [1]	0.9

Compared to the last model, this model has higher number of component models, higher max depth, higher learning rate and subsample.

3.2 Suppor Vector Machines

We referred to [1] for this whole section.

The second statistical method we decided to apply to our dataset was support vector machines. In order to introduce this method, we first discuss some related methods that we will see naturally lead to support vector machines.

3.2.1 Maximal Margin Classifier

In general, if our data can be separated using a hyperplane, then there in fact exist an infinite number of such hyperplanes, because we can "perturb" a given separating hyperplan just a little bit and it still does not come into contact with any of the observations. A natural choice among infinitely many of these hyperplanes is the *maximal margin hyperplane*, which is the separating hyperplane that is farthest from the training observations. We call the smallest distance from all training observations to the hyperplane the "margin". After identifying the maximal margin hyperplane, we can then classify a test observation based on which side of the maximal margin hyperplane it lies.

The training observations whose's perpendicular distance to the separating hyperplane is exactly the margin are called "support vectors". The maximal margin hyperplane depends directly on the support vectors, but not on the other observations: a movement to any of the other observations would not affect the separating hyperplane, provided that the observations movement does not cause it to cross the boundary set by the margin.

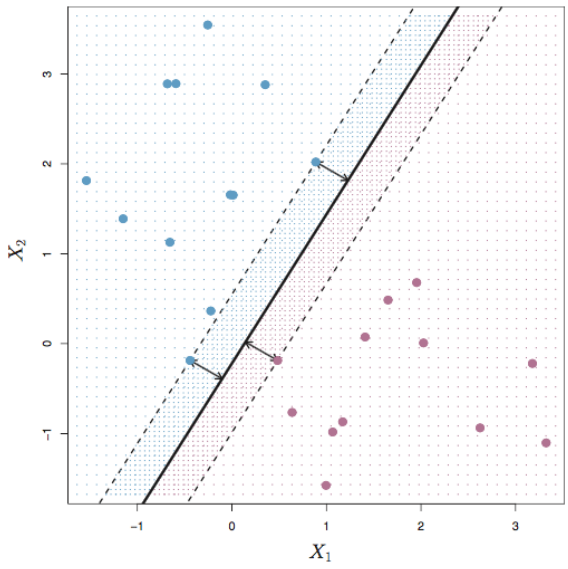


FIGURE 9.3. *There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the hyperplane is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.*

(Figure taken from [1])

Consider the task of constructing the maximal margin hyperplane based on a set of n training observations $x_1, \ldots, x_n \in \mathbb{R}^p$ and associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$. The maximal margin hyperplane is the solution to the optimization problem:

$$\begin{aligned}
&\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \; M \\
&\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \\
&y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n.
\end{aligned}$$

where M represents the margin of our hyperplane. This problem can be solved efficiently by common optimization techniques.

The maximal margin classifier is a natural way to perform classification, if a separating hyperplane exists. In cases where we cannot exactly separate the two classes, we can use a generalization of the maximal margin classifer, known as the *support vector classifier*.

3.2.2 Support Vector Classifiers

When using a classifer base on a separating hyperplane, adding a single observation can lead to a dramatic change in the maximal margin hyperplane.

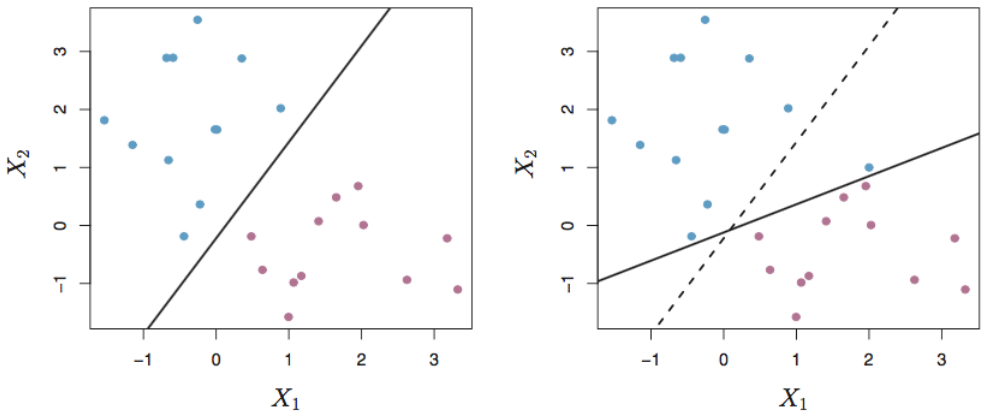


FIGURE 9.5. Left: *Two classes of observations are shown in blue and in purple, along with the maximal margin hyperplane.* Right: *An additional blue observation has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line. The dashed line indicates the maximal margin hyperplane that was obtained in the absence of this additional point.*

Figure taken from [1]

The maximal margin hyperplane is extremely sensitive to a change in a single observation, so it may have overfit the training data. In this case, it could be worthwhile to misclassify a few training observations in order to do a better job in classifying the remaining observations.

The *support vector classifier* does exactly this. Rather than seeking the largest possible margin so that every observation is not only on the correct side of the hyperplane but also on the correct side of the margin, we instead allow some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane.

The support vector classifier classifies a test observation depending on which side of a hyperplane it lies. The hyperplane is chosen to correctly separate most of the training observations into two classes, but may misclassify a few observations. It is the solution to the optimization problem

$$\begin{aligned}
&\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M \\
&\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \\
&y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \\
&\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C,
\end{aligned}$$

where C is a nonnegative tuning parameter, M is the width of the margin, and $\epsilon_1, \ldots, \epsilon_n$ are slack variables. If $\epsilon_i = 0$ then the ith observation is on the correct side of the margin. If $\epsilon_i > 0$ then the ith observation is on the wrong side of the margin, and the ith observation

has violated the margin. If $\epsilon_i > 1$ then it is on the wrong side of the hyperplane.

We can think of C as a budget for the amount that the margin can be violated by the n observations. If $C = 0$ then there i no budget for violations to the margin, and it must be the case that $\epsilon_1 = \ldots = \ldots \epsilon_n = 0$. For $C > 0$ no more than C observations can be on the wrong side of the hyperplane, because if an observation is on the wrong side of the hyperplane then $\epsilon_i > 1$, and we must have $\sum_{i=1}^{n} \epsilon_i \leq C$. As the budget C increases, we become more tolerant of violations to the margin, and so the margin will widen, and vice versa.

This optimization problem has a very interesting property: it turns out that only observations that either lie on the margin or that violate the margin will affect the hyperplane. Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as *support vectors*. These observations do affect the support vector classifier. The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle \tag{1}$$

where S is the collection of indicies of the support points, α_i are the parameters, and $\langle x, x_i \rangle$ denotes the inner product between the new point x and the training point x_i.

The fact that the support vector classifier's decision rule is based only on a potentially small subset of the training observations (the support vectors) means that it is quite robust to the behavior of observations that are far away from the hyperplane.

3.2.3 Support Vector Machines

The support vector classifier is a natural approach for classification in the two-class setting, if the boundary between the two classes is linear. If we are faced with non-linear class boundaries, we can consider enlarging the feature space using functions of the predictors, such as quadratic and cubic terms, in order to address this non-linearity.

The optimization problem we need to solve becomes:

$$\underset{\beta_0, \beta_{11}, \beta_{12}, \ldots, \beta_{p1}, \beta_{p2}, \epsilon_1, \ldots, \epsilon_n, M}{\text{maximize}} M \tag{2}$$

$$\text{subject to } y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i)$$

$$\sum_{i=1}^{n} \epsilon_i \leq C, \epsilon_i \geq 0, \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1$$

In the original feature space, the decision boundary is of the form $q(x) = 0$, where q is a quadratic polynomial, and its solutions are generally non-linear, and that is why this lead to a non-linear decision boundary.

The *support vector machine* is an extension of the support vector classifier that results from enlarging the feature space using *kernels*, which we can think of as a generalization of the inner product appears in (1), or in a calculation of the solution for the support vector classifier, of the form $K(x_i, x_{i'})$ where K is some function that we refer to as *kernel*. The support vector classifier uses the *linear* kernel $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$. One could replace every such instance with the quantity

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d \tag{3}$$

which is known as a *polynomial kernel* of degree d, where $d \in \mathbb{Z}^+$. Using such a kernel with $d > 1$ amounts to fitting a support vector classifier in a higher-dimensional space involving polynomials of degree d, rather than in the original feature space, which leads to a much more flexible decision boundary.

When the support vector classifier is combined with a non-linear kernel such as (3), the resulting classifier is known as a support vector machine. Another popular choice is the *radial kernel*, which takes the form

$$K(x_i, x_{i'}) = exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right) \tag{4}$$

where λ is a positive constant. If a given test observation $x^* = (x_1*, \ldots, x_p*)^T$ is far from a training observation x_i in terms of Euclidean distance, then $\sum_{j=1}^{p} (x_j^* - x_{ij})^2$ will be large, and so $K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_j^* - x_{ij})^2)$ will be very tiny. This means that in (1), x_i will play virtually no role in $f(x^*)$. In other words, training observations that are far from x^* will play essentially no role in the predicted class label for x^*. This means that the radial kernel has very *local* behavior, in the sense that only nearby training observations have an effect on the class label of a test observation. This property resembles the **k-nearest neighbors** classifier we have learned in class.

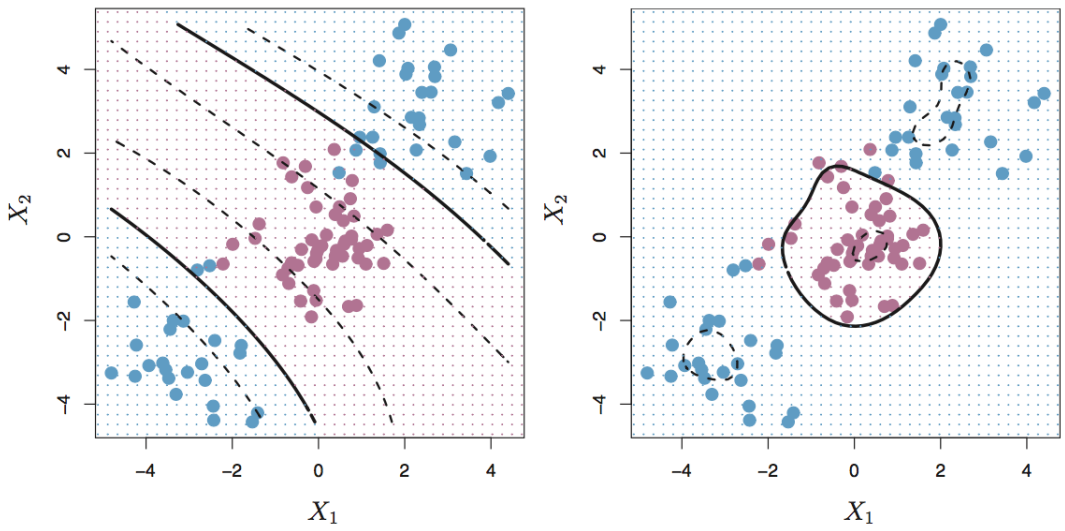


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

(Figure taken from [1])

One advantage of using a kernel rather than simply enlarging the feature space using functions of the original features, as in (2) is computational, and it amounts to the fact that using kernels, one need only compute $K(x_i, x_{i'})$ for all $\binom{n}{2}$ distinct pairs i, i'. This can be done without explicitly working in the enlarged feature space, which is important because in many applications of SVMs, the enlarged feature space is so large that computations are intractible.

3.2.4 SVMs with more than two classes

So far, our discussion is in the two-class setting. The two most popular extensions of SVMs to the K-class case are the *one-versus-one* and *one-versus-all* approaches.

1. **One-Versus-One Classification**
   Suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes. A *one-versus-one* or *all-pairs* approach constructs $\binom{K}{2}$ SVMs, each of which compares a pair of classes. The final classification is performed by assigning the test observation to the class to which it was most frequently assigned in these $\binom{K}{2}$ pairwise classifications.

2. **One-Versus-All Classification** In this approach, we fit K SVMs, each comparing one of the K classes to the remaining $K-1$ classes. Let $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the kth class to the others. Let x^* denote a test observation. We assign the observation to the class for which $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \ldots + \beta_{pk}x_p^*$ is largest.

3.2.5 Resemblance of SVM to KNN Classification

As discussed in 3.2.3.

3.2.6 SVM on our model

We use the 'e1071' package to train a support vector machine on our data set. The predictors are AnimalType, Lifestage, IsMix, SimpleBreed, SexuponOutcome, and SimpleColor. In this SVM, the radial kernel is used in training and predicting, and the one-against-one approach is used for multiclass-classification with $k > 2$ levels (in this case, $k = 5$). The test set error rate in this model is 0.387, which is a little bigger than that obtained from the 2nd gradient tree boosting model.

```
Call:
svm.default(x = x_train, y = y_train, data = animals_train)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.00390625

Number of Support Vectors:  15175

 ( 3829 1244 5446 4498 158 )


Number of Classes:  5
```

4 Variable Importance

After implementing the chosen statistical approaches, fitting the models and fine-tuning them, we decided to perform variable importance analysis on our xgb model fits. This provides some transparency and insight to the modeling process by revealing which variables are most influential in the models in predicting the animals outcome. Because our motivation for this project was to be able to identify which areas animal shelters can focus on to ensure better outcomes for these animals, conducting this type of variable importance analysis will lead us one step closer to this goal.

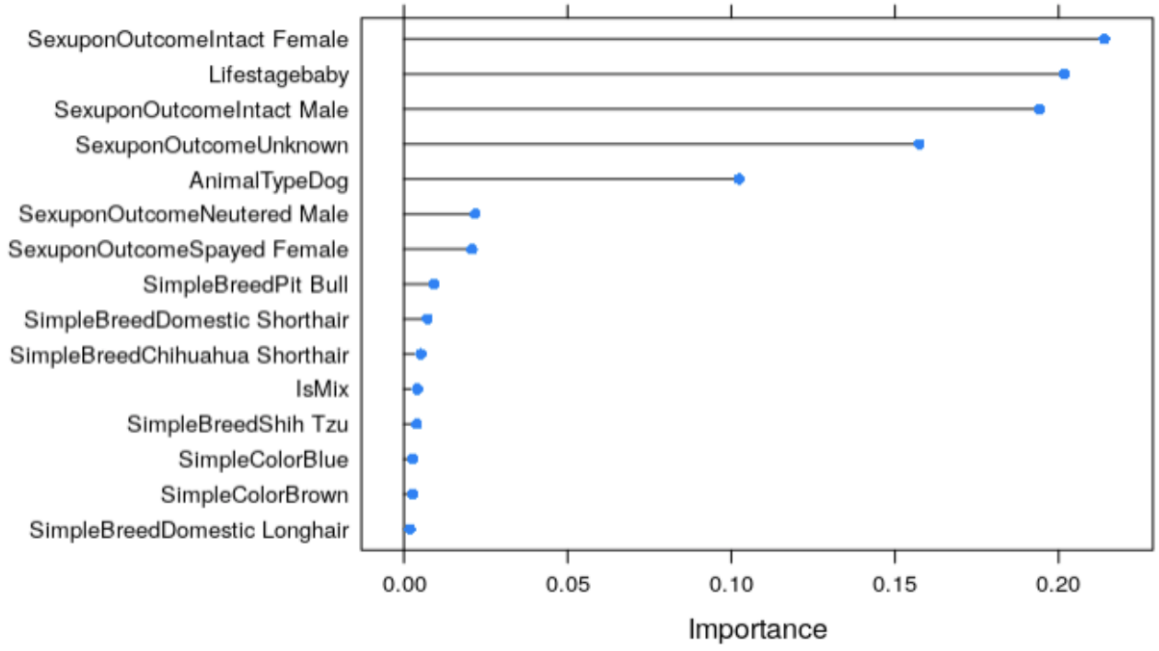
Although we conducted variable importance assessment on both out gradient tree boosting models, we will be focusing on the results corresponding to the second, and better model fit. We used the package caret::VarImp() to achieve this. This function measures variable importance by summing prediction accuracy over each boosting iteration. It uses three factors to ultimately measure a variables importance in the model.

1. Gain - measures the improvement in accuracy achieved by a feature to the branches it is on.

2. Cover - measures the relative quantity of observations concerned by a feature.

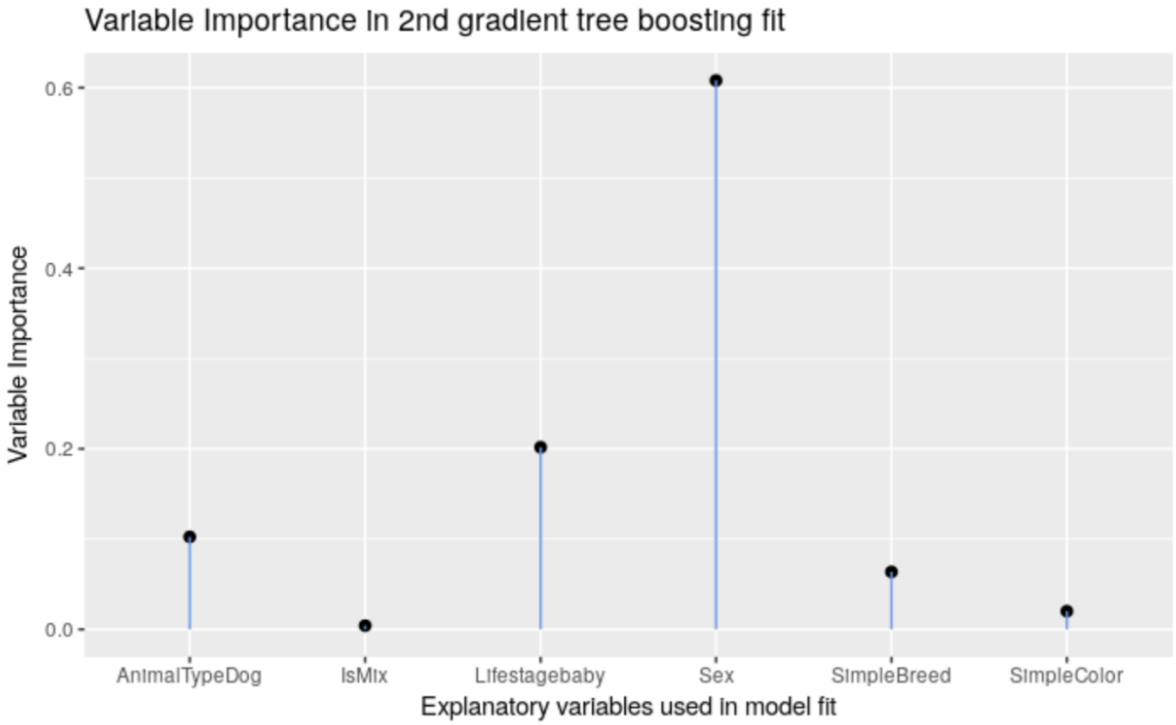
3. Frequency - the number of times a feature is used in all the generated trees.

Plotting varImp() on our second model fit, gave us the following plot. This plot gives a visual aid for comparing each level of each variable used in model and its relative variable importance.



To plot only the variables (without individual levels) in terms of their importance, we then summed all the levels for each variable and plotted the aggregate to get the final variable

importance plot below.



Finally, we were then able to assess that the following predictors were most influential in predicting the animals outcome:

- Sex
- Lifestage
- AnimalType
- SimpleBreed

Identifying this ranking is the primary step in identifying potential areas of focus when seeking to improve the outcome of these sheltered cats and dogs, especially those who are not likely to be adopted or returned to owners. This is simply the primary step. Future steps in terms of variable importance will be addressed in the next section.

5 Discussion: Limitations and Future Work

One future exploration step to take with the support vector machine fit is exploring different kernels in our svm model in addition to the radial kernel (such as the polynomial kernel). Using a linear kernel may not have given us the best results since it works bests for data that can take on a linear decision boundary. However, using a polynomial kernel with a positive degree (degree ¿ 1) leads to a more flexible decision boundary. In the future, we would ideally experiment with a polynomial kernel and assess if this results in a better svm

fit (since the test set performance for our svm fit could definitely be improved).

In our analysis, we conducted variable importance analysis on our gradient tree boosting fits as well as calculated the test set error rate for these fits. Implementing both these tools gave us a better understanding of not only how accurate our model was predicting animal outcomes, but insight into which subset of variables were contributing most to this accuracy. While we were able to identify certain influential predictors in the models (Lifestage, AnimalType, etc.), we did not conduct further analysis as to which levels of these variables were most influential in particular. In other words, establishing that the variable Life Stage is influential does not in itself indicate which of the levels was related to the outcomes. Are baby animals more likely to be adopted? Or adults? Are adults more likely to be euthanized? Or baby animals? This was a limitation of our analysis and would be an area of future enhancement.

Thus, in the future, we would in theory, select one level out of the variables deemed important and isolate all data points equal to that level and evaluate/predict the outcome for that level to see if there was an observed difference in shelter outcome. For instance, we would take all the baby animals from Lifestage variable (as it was identified to be a significant predictor in the model) and predict the outcome of these babies and contrast that with the average outcome of adults to see which level of Lifestage was influential in the outcome or had a better or worse outcome. We would repeat this process over all the important predictors to acquire a more meaningful and in depth understanding of the distribution of importance across the various levels of the predictors.

References

[1] James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning. Vol. 112. New York: springer, 2013.

[2] (Kaggle) Megan Risdal Notebook - Quick and Dirty Random Forest