Capstone Project

Machine Learning Engineer Nanodegree

Jeffrey lewis
May 22nd, 2016

Definition

Project Overview

We love our pets. Most of us treat them like family, doing all we can to ensure that they are healthy and happy. We get upset and angry when we hear of animals being abused and we act strongly when they are killed. Pets serve as a companion and just being around them can bring a warming, loving feeling in our hearts. Simply put our pets offer us unconditional love and that love feels good to us.

According to the Humane Society of the United States[1], about 65% of U.S. households owns pets and 42% of those households have more than one pet. Unfortunately, as much as we love our animals, we still find that approximately 7.6 million companion animals end up in US shelters every year. Even though most of these animals find owners, 2.7 million animals have to be euthanized each year[2].

This data has been provided by both Kaggle and the Austin Animal Center to help better understand the trends of sheltered animal outcomes. The information and insight gained from this data can help animals who need extra help in finding a forever home. A set of training data has been provided as well as a testing data set to validate on. Kaggle is a popular platform for data science and machine learning challenges with a very active and growing community. It provides a means for you to hone your machine learning skills by attempting to solve complex real-world problems[3].

Problem Statement

The data set provides a large collection of different features that can be trained to determine the probability of an outcome an animal will have. The goal will be to use

1 http://www.humanesociety.org/issues/pet_overpopulation/facts/pet_ownership_statistics.html

2 http://www.aspca.org/animal-homelessness/shelter-intake-and-surrender/pet-statistics

3 https://www.kaggle.com/c/shelter-animal-outcomes

these features to perform supervised learning to predict whether an animal in a shelter will be adopted, transferred, returned to their owner, euthanized, or unfortunately die for some reason. The steps to complete this will be first to explore the dataset and determine whether or not that the data will need to be clean or not, once that is taken care of, training and selecting a predictive model take place, and lastly we'll evaluate the model and predict this model on unseen data.

Metrics

Since the problem is to determine the probability of an outcome for an animal and not simply predict what an outcome will be, the metric that will be used to evaluate this problem will be the multi-class logarithmic loss (log loss). The log loss metric determines the likelihood of the model independently of each observation, which is exactly what is trying to be solved. This makes the use of accuracy in efficient for this problem.

The probability essentially serves as a gauge of confidence. If the true label is 0 but the classifier thinks it belongs to class 1 with probability 0.51, then the classifier would be making a mistake.

Note that the probability for a given animal are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). Also the lower the log loss score, the better.

Analysis

Data Exploration

The data set contains 10 features and 26,729 data points. During my initial inspection of this data I see that some features is missing data, this will be taken care of during the preprocessing stage. I also discover some features that can be transformed into multiple features, such as the DateTime feature and the SexuponOutcome feature. This will also be taken care or during the preprocessing stage. A description of each feature is provided below in Table 1.

FEATURES	TYPE OF DATA	DESCRIPTION
AnimalID	Discrete	ID of the animal
Name	Discrete	Name of the animal
DateTime	Continuous	The date and time of the outcome of an animal.
OutcomeType	Discrete	Target: Outcome of the animal
OutcomeSubtype	Discrete	Reasoning of an animal's outcome
AnimalType	Discrete	Type of animal
SexuponOutcome	Discrete	The sex of an animal and whether or not the animal is neutered.
AgeuponOutcome	Discrete	Animal’s age
Breed	Discrete	Animal’s breed
Color	Discrete	Animal’s color

Table 1 - Description of features

The first five data points of the data set has been provided below in Table 2 to get an idea of what the data looks like before the cleaning of the data.

AnimalID	**Name**	**DateTime**	**Outcome Subtype**	**Animal Type**	**Sexupon Outcome**	**Ageupon Outcome**	**Breed**	**Color**	**Outcome Type**
A671945	Hambone	2014-02-12 18:22:00	NaN	Dog	Neutered Male	1 year	Shetland Sheepdog Mix	Brown/White	Return_to_owner
A656520	Emily	2013-10-13 12:44:00	Suffering	Cat	Spayed Female	1 year	Domestic Shorthair Mix	Cream Tabby	Euthanasia
A686464	Pearce	2015-01-31 12:28:00	Foster	Dog	Neutered Male	2 years	Pit Bull Mix	Blue/White	Adoption
A683430	NaN	2014-07-11 19:09:00	Partner	Cat	Intact Male	3 weeks	Domestic Shorthair Mix	Blue Cream	Transfer
A667013	NaN	2013-11-15 12:52:00	Partner	Dog	Neutered Male	2 years	Lhasa Apso/Miniature Poodle	Tan	Transfer

Table 2 - Data points example

Exploratory Visualization

In this section, some simple visualizations are provided to get a grasp on what the dataset looks like.

Below in figure 1, we can see that there are a significant amount more dogs than there are cats. Also we can see that there are only two types of animals in this dataset

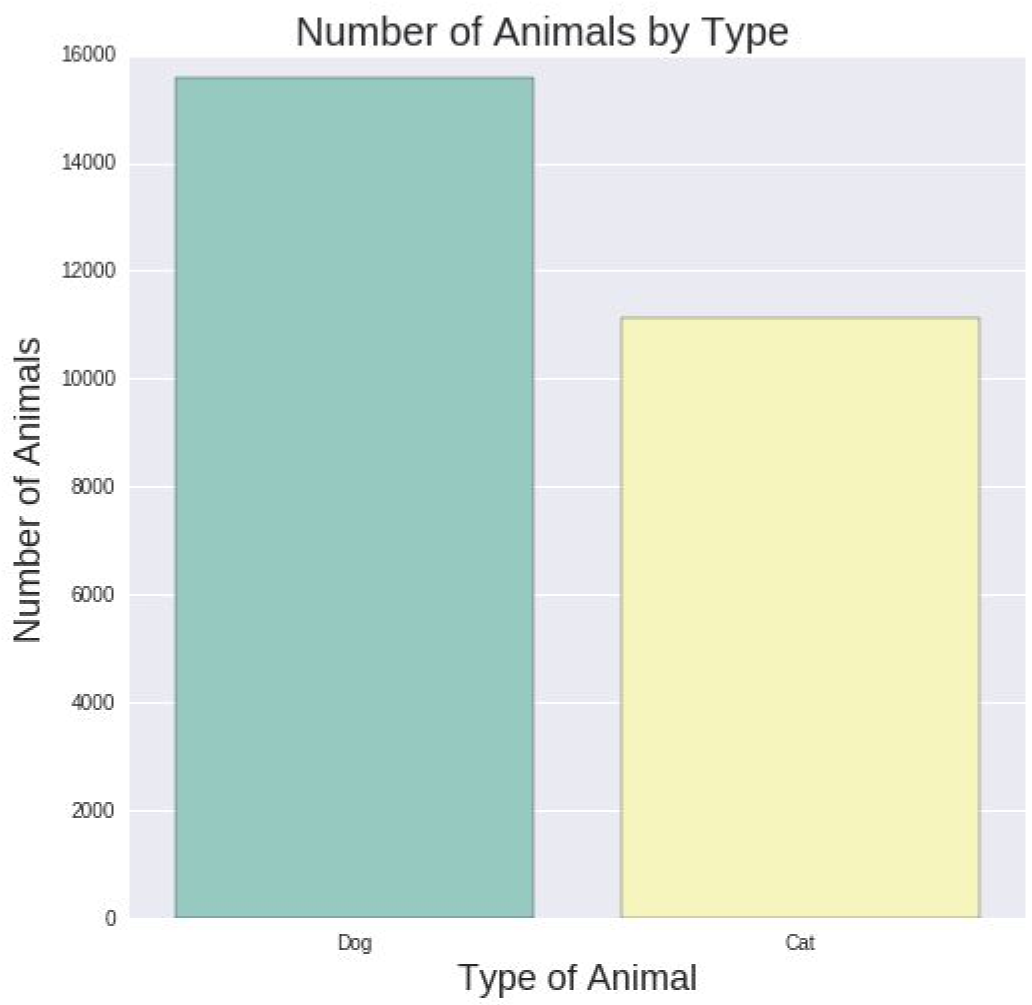

Figure 1 - Number of animals by type

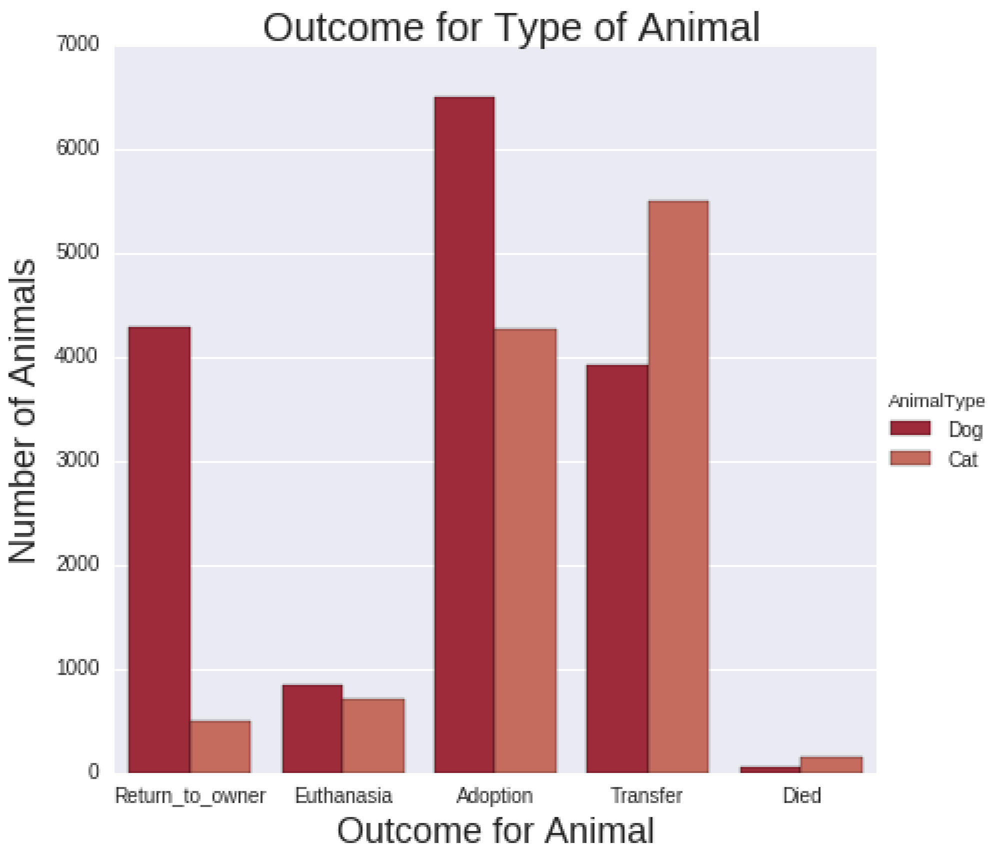

Figure 2 - Outcome for the type of animal

In figure 2, we see that outcome for each type of animal. A significant amount of both dogs and cats are either adopted or transferred. We also that many more dogs than cats are returned to their owners. Its very good to see that the number of animals that are either euthanized or who have died is very low.

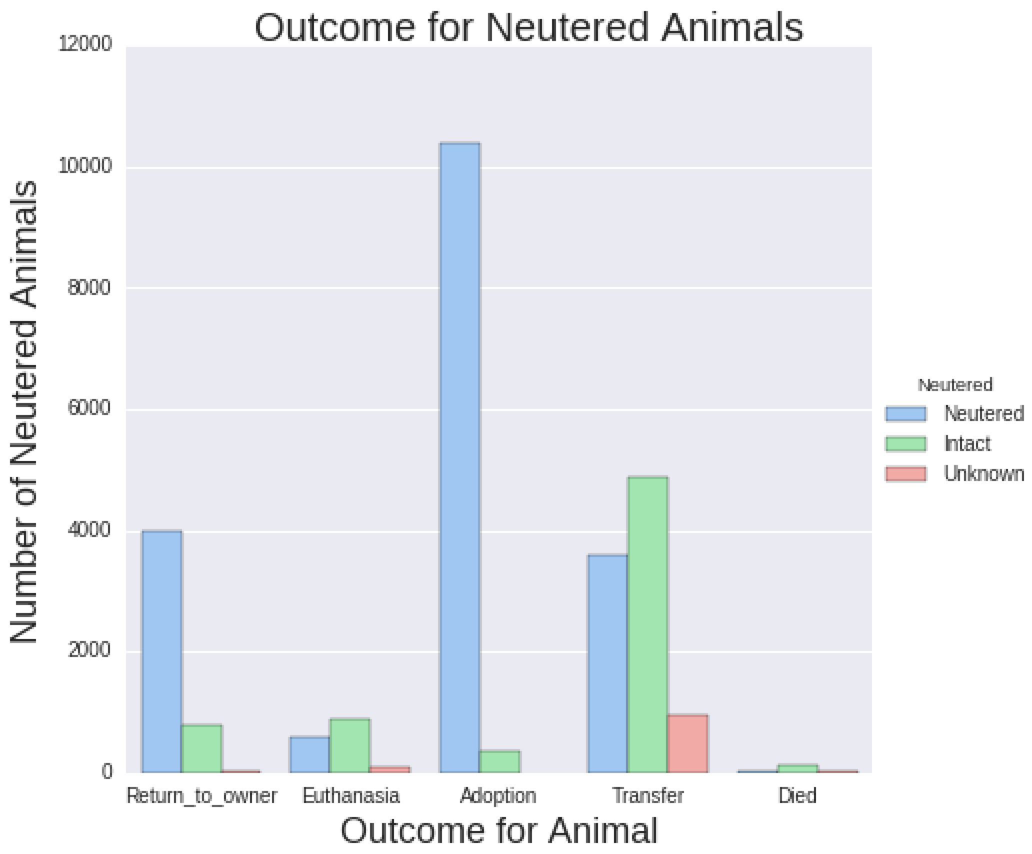

Figure 3 - Outcome for neutered animals

Figure 3 shows the outcome based off of its sterilized state. According to the chart, we can see that an animal who is neutered has a very high chance of being adopted.

Algorithm and Techniques

Since the problem is one of supervised learning, there are a numerous amount of algorithms that can be used to train the classifier. There is a fair share of data points (over 21,000) and not that many features (10), knowing this as well as the nature of the problem will help decide what algorithms are to be used. The algorithms that are going to used and described will be listed below and will be used with their default parameters.

1. **Naive Bayes**
    a. **Description:** A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on

each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple[4]. Naive Bayes model is easy to build and particularly useful for very large data sets.

2. **Random Forest**
    a. Description: Random Forest is a type of ensemble learning method, where a group of weak models combine to form a powerful model. It is a versatile machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration, and does a fairly good job. Random Forest classifies a new object based on attributes grown from multiple trees and each tree gets to vote for that class[5].
3. **Gradient Boosting**
    a. Gradient Boosting is basically a sequential technique which works on the principle of ensemble. It combines a set of weak learners and delivers improved prediction accuracy. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher[6].
4. **K-Nearest Neighbors**
    a. K-Nearest Neighbors is an algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function[7].
5. **Logistic Regression**
    a. Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function[8].

[4] http://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/
[5] http://www.analyticsvidhya.com/blog/2015/09/random-forest-algorithm-multiple-challenges/
[6]http://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/
[7] http://www.saedsayad.com/k_nearest_neighbors.htm
[8] http://www.analyticsvidhya.com/blog/2015/11/beginners-guide-on-logistic-regression-in-r/

Benchmark

The adopted benchmark that I will be using for this problem will be a log loss score of 20.251. Since this is a project from a Kaggle contest, I find it appropriate to use the benchmark that this particular contest has provided. The benchmark score is the score obtained when predicting that all animals in the test set are adopted.

Each incident has been labeled with one true class. For each animal, you must submit a set of predicted probabilities (one for every class). The formula is then,

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

where N is the number of animals in the test set, M is the number of outcomes, *log*log is the natural logarithm, *yij*yij is 1 if observation *i*i is in outcome *j*j and 0 otherwise, and *pij*pijis the predicted probability that observation *i*i belongs to outcome *j*j.

The submitted probabilities for a given animal are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the log function, predicted probabilities are replaced with *max*(*min*(*p*,1−10−15),10−15)max(min(p,1−10−15),10−15)[9].

Methodology

Data Preprocessing

Since there was some missing data, I thought it would be best to take care of that problem first. The features that contained missing values were Name, OutcomeSubtype, SexuponOutcome, and AgeuponOutcome. To determine the how to fill SexuponOutcome and AgeuponOutcome, I simply used the value that occurred the most which was “Neutered Male” and “1 year ” respectively. How I approached the

[9] https://www.kaggle.com/c/shelter-animal-outcomes/details/evaluation

Name feature was to change whether or not animal had a name. I filled the missing data with the value “No Name” and if an animal had one, their value was changed to “Has Name.” I dropped the feature of OutcomeSubtype for a couple of reasons. Not only was the feature missing more than half of its values, the test set that will be used to evaluate the final model does not have OutcomeSubtype as one of its features. This makes this feature irrelevant in terms of building a model.

As mentioned in the Data Exploration section, there are some features that I would like to split into multiple features. SexuponOutcome has values such as “Neutered Male” and “Spayed Female,” so I separated this feature into two different features “Sex” and “Neutered.” The Sex features contains whether the animal was male or female, while the Neutered feature contain the values “Neutered”, “Intact”, and “Unknown.” Spayed animals were lopped into the “Neutered” category. The other feature that I converted into multiple features was DateTime. These values were put into their own Year, Month, Day, Hour, and Minute features.

The last couple of changes that were made were to the features AgeuponOutcome, Breed, and Color. Age values were converted into numeric age values of weeks. Breed had 1,380 unique values, which is a lot. To simplify this, all the animals who had the word mix or the character “/” in it was converted into the values “Hybrid,” all others were converted into the value “Purebred.” Just like Breed, Color had a significant amount of unique colors (366). How I approached this was, if an animal was described to have two different colors, the first color that was described was assumed to be the dominant color. This approach reduced the amount of unique colors to 29.

Implementation

The first step of the implementation process is to split the data into a training set and testing set. By splitting the data into a training and testing set, we can ensure that the model generalizes well to predict accurately. We use the training set to train and optimize our machine learning model, while we keep the test set until the very end to evaluate the final model. A 70-30 split was used, 70% used for the training set, while 30% is reserved for the test set. This split was with sklearn function train_test_split[10]. Now that the data set has been split into a testing and training set, the model can be built and trained with a scoring function of log loss. The model is fit on the training set and is then used to predict the outcome. There were no issues due to the fact that we

[10] http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.train_test_split.html

are using a plug and play method with our classifiers, since we are only using the default parameters.

Table 3 illustrates the performance metrics obtained from each classifier used. As mentioned before, on the classifier's default parameters were used. Training time is included in this chart, because if a classifier take a long time to train, in can become incredibly painful. The reason being is that training will be done many times and one cannot progress further until the results have been given, making training time very important.

Classifier	**Training Time**	**Training Log_loss Scores**	**Test Log_loss Scores**
Naive Bayes	0.032	1.352	1.423
Random Forest	0.144	2.440	2.682
Gradient Boosting	4.833	0.796	0.806
K-Nearest Neighbors	0.029	5.478	6.188
Logistic Regression	0.625	0.972	0.9769

Table-3 Classifiers log loss scores

In terms of performance of log_loss score, the classifiers rank in the following from best to worst:

1. Gradient Boosting
2. Logistic Regression
3. Naive Bayes
4. Random Forest
5. K-Nearest Neighbor

KNN performed the worst by a significant margin. The default K parameter for neighbors to be used is five. Maybe by either increasing or decreasing K, the model may perform a lot better. If I were to take a guess, I would say increasing K would make this model better. KNN is typically computationally expensive when K is a large value, but KNN performed the best as far as training time is concerned. A small K typically means that noise will have a higher influence on the result.

Random forest needs to be carefully built, because of its tendency to overfit. By tweaking some of the parameters like the number of trees could help this model perform better than what it. Naive Bayes is very good at making the most use of small number of samples and does not overfit as much as other classifiers. Since this is not necessarily a large dataset, this could be a reason as to why Naive Baye performed better than Random Forest. GBC and Logistic Regression performed the best, with GBC being slightly better than Logistic Regression.

Refinement

Gradient Boost was selected as the best model to use and refine based off the scores that were produced. Although the Gradient Boost had a significantly higher training time compared to the other classifiers, its score is significantly lower than theirs.

Table 4 illustrates the step taken to get a fined tuned model. The possible parameter values were fed to sklearn's gridsearchCV. Gradsearch does an exhaustive search over specified parameter values for an estimator. These values came about by simply finding the range of the best value of that parameter and then just narrowing it down to the exact best fit, according to gridsearch.

Step Number	**Parameter**	**Possible Parameter Values**	**Best Value**	**Log loss score**
Step 1	n_estimators	10-100	93	0.780
Step 2	max_depth	5,7,9,11,13,15	7	0.777
Step 3	min_samples_split	10-1000	170	0.776
Step 4	min_samples_leaf	30,40,50,60	40	0.776
Step 5	max_features	1,3,5,7,9,11	3	0.776
Step 6	subsample	0.01-1	0.96	0.774
Step 7	learning_rate	0.1, 0.05,0.01	0.05	0.758
Step 7	n_estimators	93,186,600	186	0.758

Table 4 - Gradient Boost refinement

The final score that was received from the final model is a 0.758, nearly 0.05 points better than the model that was used with default parameters.

Results

Model Evaluation and Validation

The final parameters that were used for the final Gradient Boost classifier were:

1. Learning_rate = 0.05
2. Max_depth = 7,
3. Max_features = 3
4. Min_samples_leaf = 40
5. Min_samples_split = 170
6. N_estimators = 186
7. Subsample = 0.96

The number of boosting stages to perform. Gradient boosting is fairly robust to overfitting so a large number usually results in better performance.The learning rate shrinks the contribution of each tree by learning_rate. There is a trade-off between learning_rate and n_estimators[11]. So finding the best balance between these two can improve the overall score.

Max_depth represents the maximum depth of each individual regressor and limits the number of nodes in a tree. Max_features just simply looks for the number of features to consider when looking to split. Min_sample leaf represents the minimum of samples to be at a leaf node and min_sample split represents the minimum number of samples required to split a node.

The final model was then tested on a separate testing data set that was provided by kaggle and sent for evaluation. The log loss score that was returned was a score of 0.746. The link has been provided for verification of the screen https://www.kaggle.com/jakur88/results. Since the score was very similar and better

[11] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

than the score used for training, it’s safe to say that the model generalizes well to unseen data. Since the final score came from Kaggle itself, I can say that the final model can be trusted and used on other datasets.

Justification

Since the benchmark value for this problem was 20.251 and the final model achieved a log_loss score of 0.746, it’s safe to say that the final model results are significantly better than the benchmark. Based off the difference of the benchmark score and the final models score, I can say that model is significant enough to have solved the problem.

Conclusion

Free-Form Visualization

A bar graph was produced that demonstrated the importance of each feature that was used to train the data. The top five most important features are name, animalType, ageuponOutcome, breed, and color. It would appear from this graph, that whether an animal had a name or not holds significant weight. It seems like time isn’t really that important. I would have thought maybe the time of year would impact that outcome in a way.

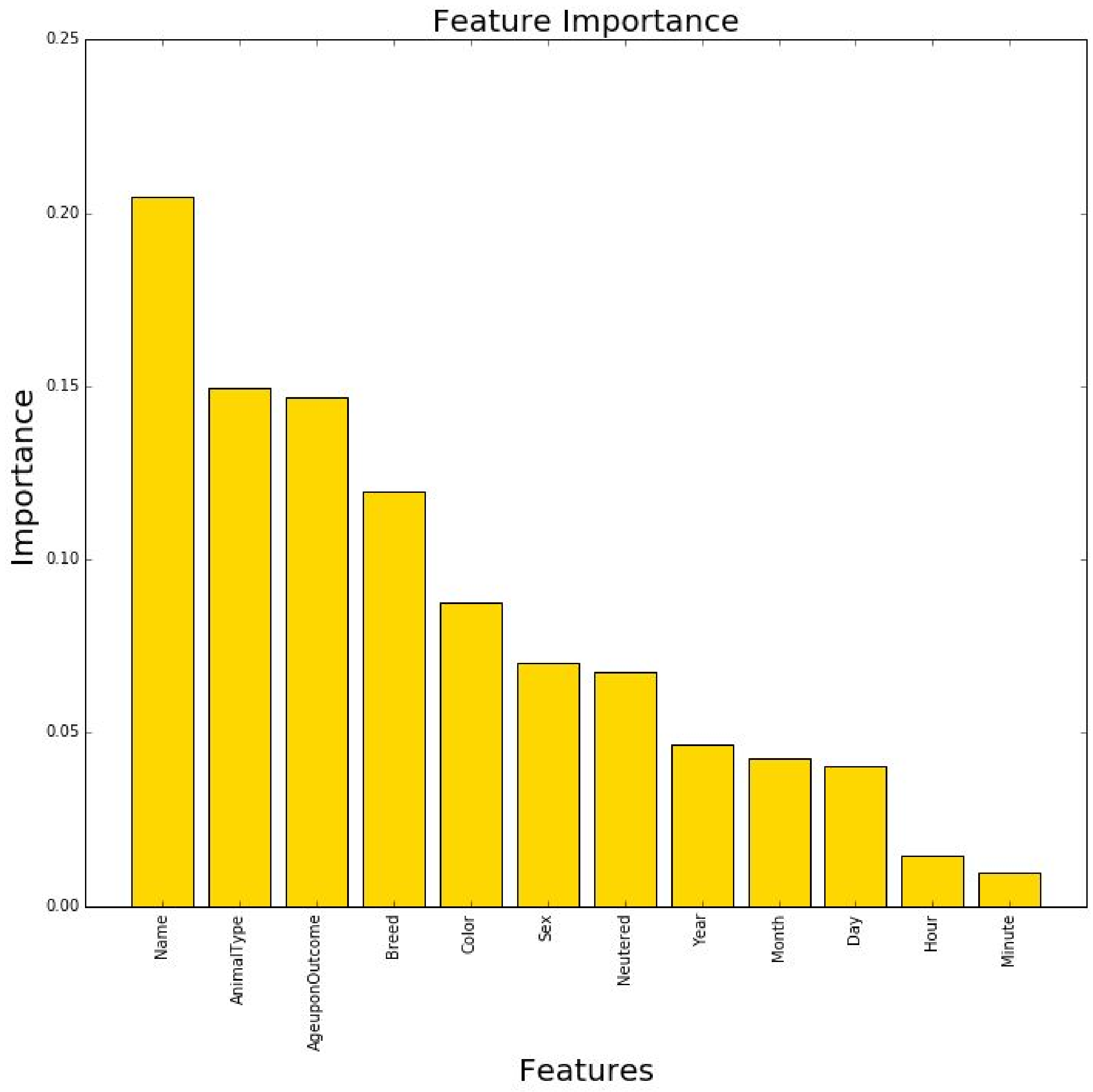

Table 5 - Probabilities of final model on testing data

Reflection

The project began with an exploration the dataset. During this phase we found that several features was missing some data and that this would need to be resolved. Also during this exploration, it was determined that some of the features needed to be cleaned up some in some form or fashion. All this was done during during the preprocessing stage the data that was missing was either dropped or filled with the most frequent value. Some features were split into multiple features and other features had their values converted into much more simplistic values. Once the preprocessing was done, it was time to split the data into a training and testing test. This was done by splitting the with 70% of it being trained and 30% of it used to test the model. Next several different models were trained using their default parameters and the model with the best results (Gradient Boost) was chosen for fine tuning. Once tuned it was determined that the final model exceeded the benchmark that was provided significantly. Maybe with more preprocessing and parameter tuning, the model may be able to perform even better.

There weren't any difficult aspects to this problem. If I had to choose one, it would be the tuning of the parameters for the final model. In addition to the selection of the parameters to search, it was necessary to come up with a set of possible values for the parameters. There are several improvements to the dataset that I would like to see, but that will be discussed in the improvement section.

Improvement

As mentioned in the previous section, one thing that I would like to improve would be on the dataset. Implementing features such as the animal's size, temperament, and the animal's group in terms of whether the animal is belongs in the working group, sporting group, etc. All though the changes largely relate to dogs, having these features may provide more insight this problem. Another improvement would be to find a dataset that have other types of animals such birds, reptiles, and hamsters.

Another improvement that could have been implemented, would be to find the outliers and possibly remove them for from the dataset. The removal of these outliers may impact results and maybe comparing if they do or not could be explored. Of course, if the outliers do not change the results they should be kept in the dataset.

An algorithm that I would like to use would be a Support Vector Machine (SVM). SVMs work by attempting to find a separating hyperplane with the largest possible gap between the data points. A complex mathematical procedure dubbed "the kernel trick" allows this method to be used for data sets that have a very high dimensional space (i.e. have a large number of features). SVM's take a long time to train compared to other algorithms, but can yield terrific results.

If score obtained by the final model was used as a benchmark, there is most certainly an even better solution out there. Since the project is currently being used in as a kaggle competition, currently the best score is a 0.451, which is 0.3 points better than the score that I was able to achieve.