GAURAV KUMAR SINGH
7/21/16

CAPSTONE PROJECT: UDACITY MACHINE LEARNING NANODEGREE

Table of Contents

I. DEFINITION

Project Overview

In this project, I utilized my machine learning skills on a publically available dataset provided as part of a Kaggle Competition. Kaggle is a platform for predictive modelling and analytics competitions on which companies and researchers post their data and statisticians and data miners from all over the world compete to produce the best models. Kaggle also hosts recruiting competitions in which data scientists compete for a chance to interview at leading data science companies.

The competition I picked is about the outcomes of the animals in Austin Animal Center. Outcomes represent the status of animals as they leave the Animal Center. The description of the challenge can be found here:

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

As described on Kaggle:

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.

I found this problem particularly interesting because it's about those animals who just can't speak like us, but they do have all the emotions like humans, and the hardships they face in the world. By building a predictive model, I would like to save those unlucky dogs and cats who had to be euthanized. From machine learning point of view, the interest lies in the multi-class nature of the problem, which will help me explore the performance of several classification algorithms where there are more than 2 labels.

Problem Statement

As we notice in our dataset, there are two animals: cats and dogs, at the Austin Care Center. Some of the animals do have names while others do not. The breeds, sex and colors of these animals are provided too. When these animals leave the shelter, they can face one of the following outcomes: Adoption, Died, Euthanasia, Return_to_owner, and Transfer. There has been a log maintained of the outcomes, along with the date and times. The age of the animals upon the outcome is provided too.

Using the data, for every animal from the Austin Care Center, I would like to predict the probabilities of the animal facing each of the five outcomes, whether they will be lucky to be adopted, returned to owner or transferred, or they get euthanized or die in some other fashion when leaving the shelter. Thus our target variable is OutcomeType. The predictors will be the features of the animals. For example certain breeds are more popular are more likely to be in the lucky set of Outcomes, while certain breeds and certain colors are extremely unpopular. Also, people prefer to adopt young animals like puppies and kittens, however, old animals are ignored.

Firstly, I have to establish the feature importances, and if needed create new features too. For example, many animals have no names, but is having a name important? Later, I will show that it is indeed important for animals to have a name. Similarly, instead of having several breed types in our breed predictor, we might want to classify the breeds into pure or mix. I will also notice the significance of the sex of the animals for their outcomes.

After feature selection and extraction procedures, I would employ a bunch of classification algorithms and there performance will be compared against the log-loss metric (to be described in next section). I wish to observe the classification power of those algorithms and thereafter include a voting classifier so as to enhance the prediction by amplifying the merits of the algorithms involved.

Metrics

In this project, we are going to use the multi-class log-loss metric. This error metric is used where we have to predict that something is true or false with a probability (likelihood) ranging from definitely true (1) to equally true (0.5) to definitely false(0). The use of log on the error provides extreme punishments for being both confident and wrong. Our goal will be to minimize the log-loss. The formula is provided below:

$$log\ loss =$$

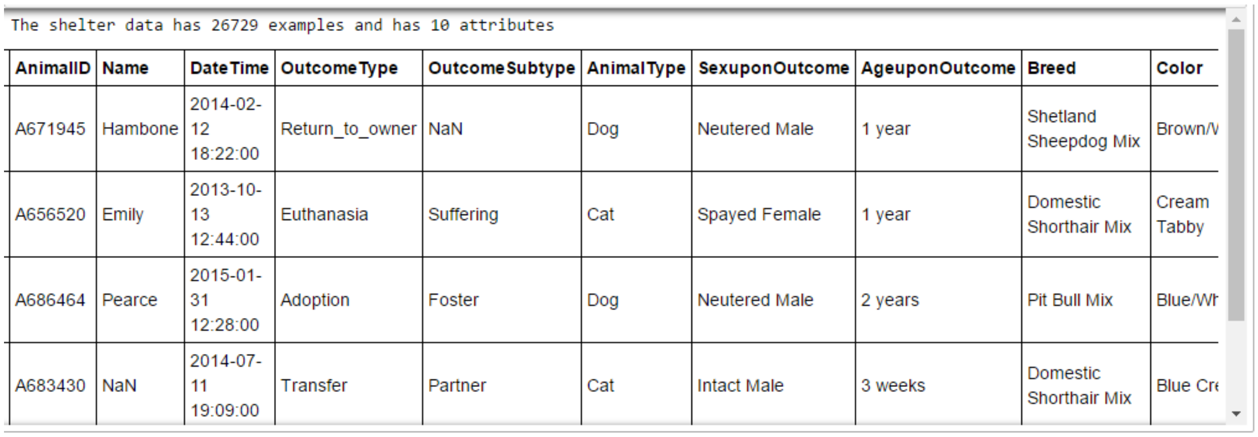
$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}y_{ij}\log p_{ij}$$

II. ANALYSIS

Data Exploration

To start with our data analysis, we first load the training data and observe what our column heads are and how many examples we have. We notice that we have the following 10 columns in our dataset:

- AnimalID: just the ID numbers, not interesting.
- Name: names of the animals, not all the animals have names.
- DateTime: time and date of the outcome.
- OutcomeType: every example has one of the 5 outcomes, the labels for our training model.
- OutcomeSubtype: subtypes of the outcomes, however not part of the testing set.
- AnimalType: two valued variable, dogs or cats.
- SexuponOutcome: Sex of the animal upon the outcome and includes values like neutered and intact animals.
- AgeuponOutcome: Age of the animal upon the outcome.
- Breed: Breed of the animal, includes several pure and mix breeds.
- Color: color of the animals.

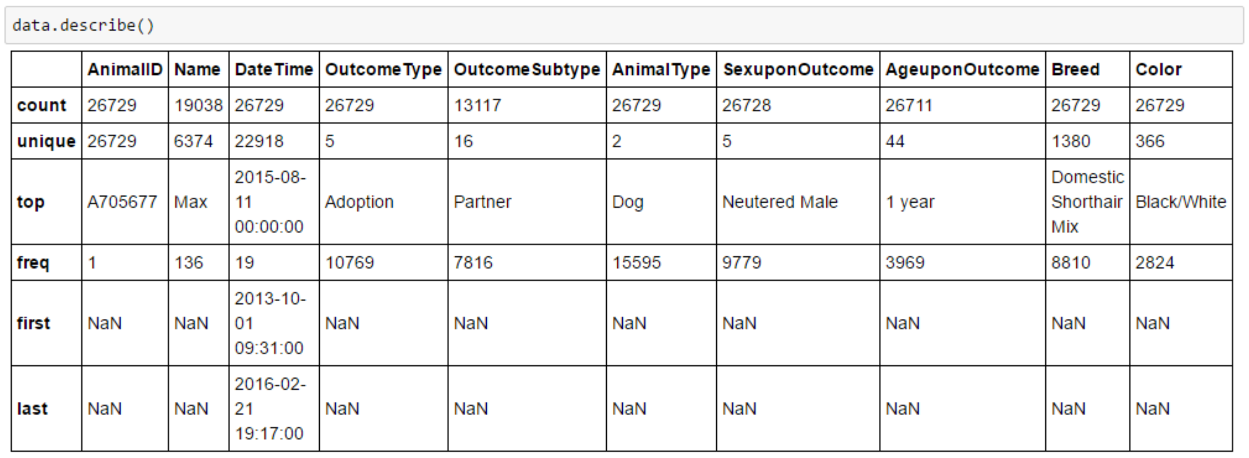


```
The shelter data has 26729 examples and has 10 attributes
```

AnimalID	Name	DateTime	OutcomeType	OutcomeSubtype	AnimalType	SexuponOutcome	AgeuponOutcome	Breed	Color
A671945	Hambone	2014-02-12 18:22:00	Return_to_owner	NaN	Dog	Neutered Male	1 year	Shetland Sheepdog Mix	Brown/V
A656520	Emily	2013-10-13 12:44:00	Euthanasia	Suffering	Cat	Spayed Female	1 year	Domestic Shorthair Mix	Cream Tabby
A686464	Pearce	2015-01-31 12:28:00	Adoption	Foster	Dog	Neutered Male	2 years	Pit Bull Mix	Blue/Wh
A683430	NaN	2014-07-11 19:09:00	Transfer	Partner	Cat	Intact Male	3 weeks	Domestic Shorthair Mix	Blue Cre

Thus, we notice that we have shelter data for 26729 animals and there are 10 attributes including the outcomes of the animals.

Now, we should have knowledge the basic statistics of each of the predictors and labels in our data.



```
data.describe()
```

	AnimalID	Name	DateTime	OutcomeType	OutcomeSubtype	AnimalType	SexuponOutcome	AgeuponOutcome	Breed	Color
count	26729	19038	26729	26729	13117	26729	26728	26711	26729	26729
unique	26729	6374	22918	5	16	2	5	44	1380	366
top	A705677	Max	2015-08-11 00:00:00	Adoption	Partner	Dog	Neutered Male	1 year	Domestic Shorthair Mix	Black/White
freq	1	136	19	10769	7816	15595	9779	3969	8810	2824
first	NaN	NaN	2013-10-01 09:31:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
last	NaN	NaN	2016-02-21 19:17:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN

We should also look whether any of the data is missing or not.

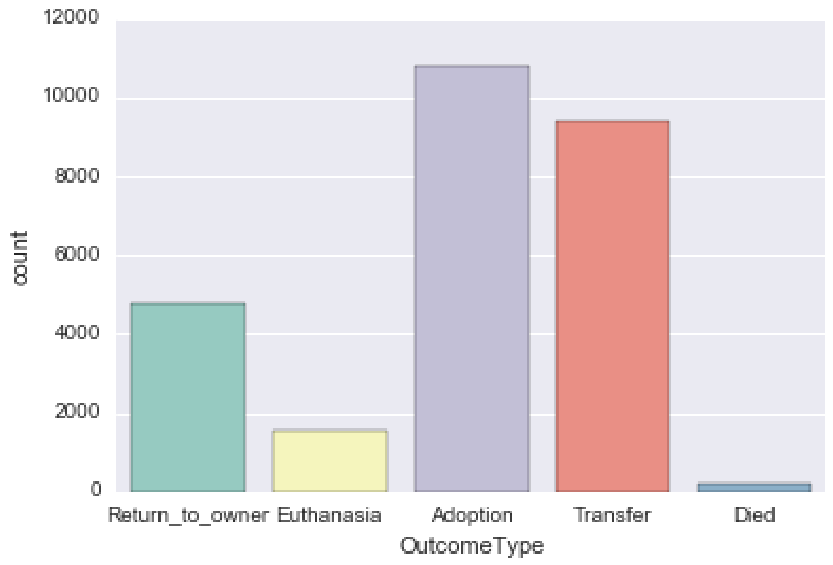
```
Missing Numbers in Training Set?

AnimalID              0
Name               7691
DateTime              0
OutcomeType           0
OutcomeSubtype    13612
AnimalType            0
SexuponOutcome        1
AgeuponOutcome       18
Breed                 0
Color                 0
      dtype: int64
```

We can see that many of the animals have missing names. Although, names don't matter as such, however later we will show that whether the animal is named or not has an effect on its outcome. The many missing OutcomeSubtype values should also not matter since we are only trying to predict the OutcomeType. For missing values for AgeuponOutcome, I am replacing that with the most frequent value "1 year". I have decided to drop AnimalId, Color, DateTime, and OutcomeSubtype from our training model as I think, these features will not be affecting the outcomes of the animals.

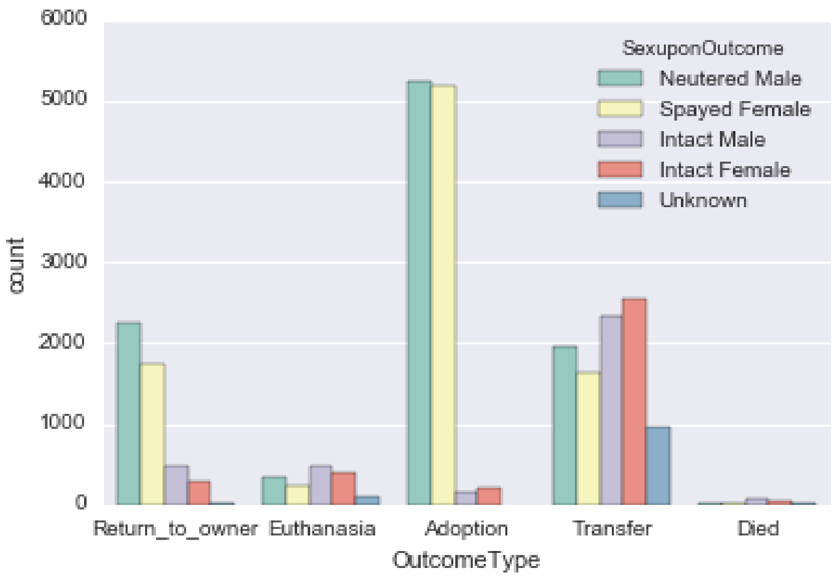
Exploratory Visualization

Firstly, we would like to know the distribution of the outcomes of animals:



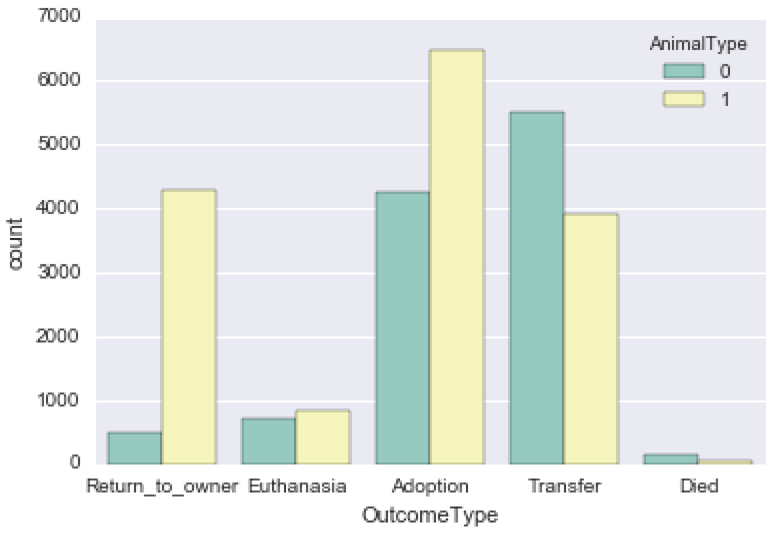
The most common outcome we observe is Adoption, while animals dying is the least probable outcome. Most of the animals seem to be a part of the set of the "lucky" outcomes.

Next up, we would like to know that the effect of the SexUponOutcome predictor on the outcomes of the animals. Usually castrated animals have a better chance to be adopted. Let us just verify that.



As expected, neutered male and spayed females do have better chance of getting adopted and returned to owner than intact males and females, however, for other outcomes, there is no clear distinction.

Next up, we want to know the distribution of outcomes based on animal type. We have assigned 'cats': 0 and 'dogs': 1



Interestingly, we observe that, the dogs will most likely to be adopted. While cats are most likely to be transferred. Both dogs and cats have good chances to be adopted. Also, it appears that returning to owner is very likely for dogs while for cats - not so much. My guess would be shorter lifespan of dogs.

Next we find out how breeds of animals affect their outcomes. Certain breeds are considered dangerous while certain others breeds are extremely popular as pets.

```
len(data["Breed"].unique())
```

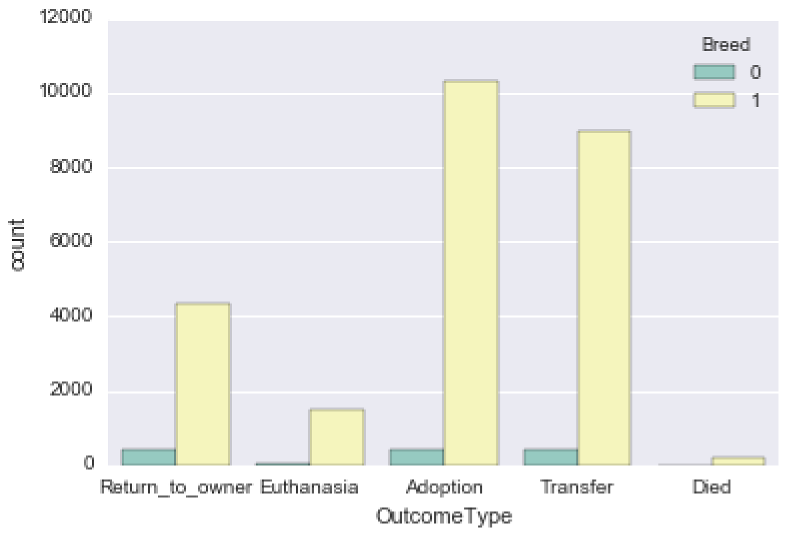
```
1380
```

1380 is a huge number, and finding distribution of the outcomes over these many breeds doesn't seem a good idea. Very few animals have a pure breed (for example'Cairn Terrier') and a lot of animals are mix. Let's replace all variations of mix by "Mix".

```
def mix(x):
    x = str(x)
    if x.find('Mix')> -1 or x.find('mix')> -1 or x.find("/") > -1: return "Mix"
    else: return x
data['Breed'] = data.Breed.apply(mix)
print(len(data["Breed"].unique()))
```

```
140
```

140 breeds of dogs and cats make up too many classes. If we were to convert all of these we could suffer from the curse of dimensionality and we could also highly over fit our dataset. A possible class division can be made by calling Mix breeds as 1 and others 0. Now, we are ready to observe the distribution of the outcomes by splitting on whether the animal is pure breed or mix breed.



Since the target variable appears to be imbalanced, choosing log-loss as a metric seems to be a good decision. As confident misclassifications will be penalized more.

Algorithms and Techniques

Since this project requires us to predict outcomes of the animal based on other predictors, we would be using few classification algorithm with multi-class capabilities. Also, with log loss as metric it will be better if we predict probabilities of animals having certain outcomes, because incorrect predictions with confidence will lead to large penalties. I have chosen the following classification algorithms for our purposes:

- **KNeighborsClassifier**

  Neighbors-based classification is a type of *instance-based learning* or *non-generalizing learning*: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point. KNeighborsClassifier implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user.

  KNeighborsClassifier is a good starting model, when the dataset is not huge and it can be assumed that the test set has similar examples as in the train set.

- **Logistic regression**

  Logistic regression, also known as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier, and is a misnomer as it is a linear model for classification rather than regression. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function. The implementation of logistic regression in scikit-learn can be accessed from class LogisticRegression. This implementation can fit a multiclass (one-vs-rest) logistic regression with optional L2 or L1 regularization.

  Logistic Regression is one of the most basic classification algorithms and works great in many applications. Another reason to pick this algorithm is that the Logistic regression is intrinsically simple, it has low variance and so is less prone to over-fitting.

- **Random Forests**

  Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model. Another reason to use this algorithm in our project is that splitting reveals the importance of features in decreasing order.

- **AdaBoostClassifier**

  The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying weights w_1, w_2, ..., w_N to each of the training samples. Initially, those weights are all set to w_i = 1/N, so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted

model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence.
Selecting a DecisionTreeClassifier as weak learner, can be a good way to estimate the power of this algorithm in this project.

- **Gaussian Naive Bayes**

  Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality. GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:
  The main advantage of using GaussianNB in our project is that it is outrightly simple, fast and doesn't assume anything about the data.

- **CalibratedClassifierCV**

  CalibratedClassifierCV is something we didn't encounter in the Nanodegree program. As per sklearn's page on Calibrated Classification:
  "When performing classification you often want to predict not only the class label, but also the associated probability. This probability gives you some kind of confidence on the prediction. However, not all classifiers provide well-calibrated probabilities, some being over-confident while others being under-confident." This does sound like what we should be using as we have to predict the probabilities of the outcomes for the animals.

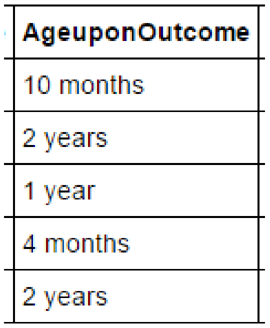
Benchmarks

The benchmark log-loss score for this data is adopted to be 20.25113. This score is obtained if we predict Adoption, the most frequent outcome, for all the animals.

III. METHODOLOGY

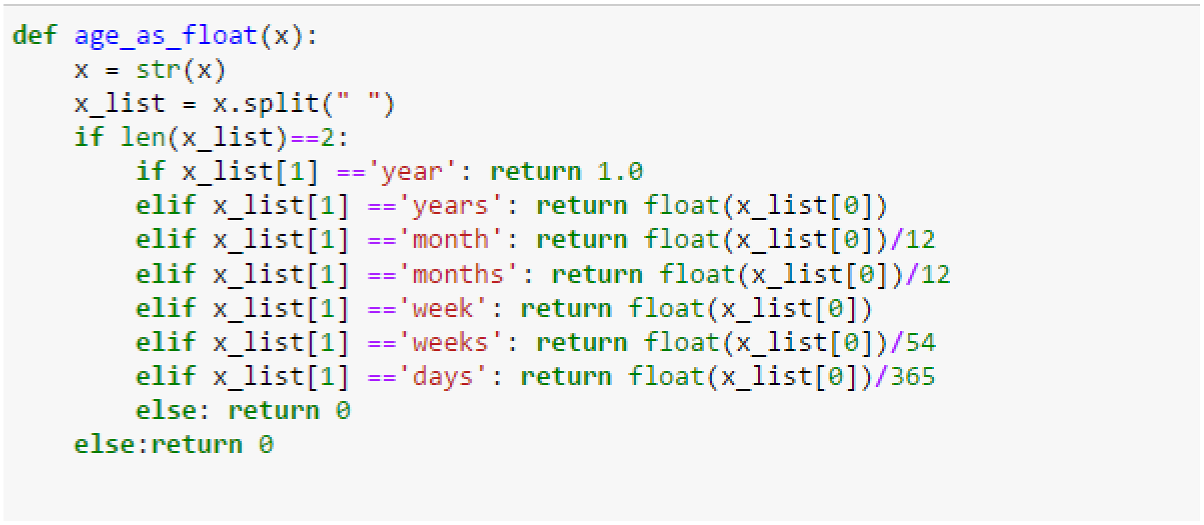
Data Preprocessing

- The AgeUponOutcome Variable is given in all possible units: years months and weeks



AgeuponOutcome
10 months
2 years
1 year
4 months
2 years

To standardize the AgeUponOutcomeVariable, I converted all the ages into years by using the following script:



```
def age_as_float(x):
    x = str(x)
    x_list = x.split(" ")
    if len(x_list)==2:
        if x_list[1] =='year': return 1.0
        elif x_list[1] =='years': return float(x_list[0])
        elif x_list[1] =='month': return float(x_list[0])/12
        elif x_list[1] =='months': return float(x_list[0])/12
        elif x_list[1] =='week': return float(x_list[0])
        elif x_list[1] =='weeks': return float(x_list[0])/54
        elif x_list[1] =='days': return float(x_list[0])/365
        else: return 0
    else:return 0
```

- We also more had 5 values in the SexUponOutcome Variable : Neutered Male, Intact Male, Spayed Female, Intact Female, Unknown. Using get_dummies function by pandas, new features are generated: SexuponOutcome_Intact_Female, SexuponOutcome_Intact_Male, SexuponOutcome_Neutered_Male, SexuponOutcome_Spayed Female, SexuponOutcome_Unknown.
- We have already converted all the breeds into either pure or mix breed in previous section.
- Similarly, AnimalType is made binary-valued by assigning 1 to dogs and 0 to cats.

Implementation

- Since, the solutions of the testing set are not up yet, I have decided to split the train.csv data into training data (75%) and testing data (25%). Stratified Shuffle Split is used to tackle the problem of class imbalance.
- I chose my features to be: ['Name','AnimalType', 'AgeuponOutcome', 'Breed', 'SexuponOutcome_Neutered Male', 'SexuponOutcome_Spayed Female', 'SexuponOutcome_Unknown']
- Remember that the intact and neutered/spayed Sexuponoutcomes are complementary.

Now we are ready to find a model which minimizes the log-loss score and thus predict the outcomes of the animals way better than the benchmark solution. Other than creating dummy variables for SexUponOutcome feature, no other complications are observed.

- Using kNeighborsClassifier, the best result obtained was using 5 neighbors and the log-lo ss score was 4.670 which is indeed better than the benchmark score, but not good enough.
- Using LogisticRegression, the minimum log-loss score obtained was 0.917 which is a dec ent score.
- Using RandomForestClassifier, the minimum log-loss score I obtained was 1.263.
- Using AdaBoostClassifier, with weak-learner being DecisionTreeClassifier, the minimum log-loss score I obtained was 1.555.
- Using GaussianNB, the minimum log-loss score I obtained was 3.925.
- Using CalibratedClassifierCV, the minimum log-loss score I obtained was 0.922.

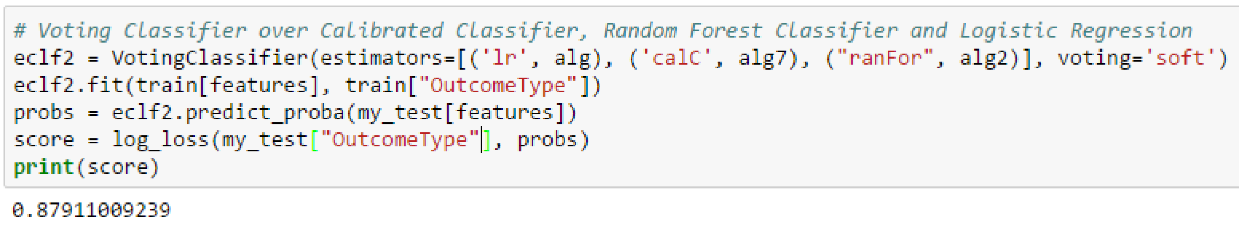
Refinement

So far, we have tried some of the great classification methods and some of them perform as well as others. Interestingly, instead of selecting a particular algorithm as my model, I have decide to use a combination of 3 best performing models and let them vote for which outcome an animal would get. I will be utilizing the Voting Classifier functionality of ensemble methods of the sklearn. As per sklearn's page on Voting Classifier:

The idea behind the voting classifier implementation is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses.

In majority voting, the predicted class label for a particular sample is the class label that represents the majority (mode) of the class labels predicted by each individual classifier. In contrast to majority voting (hard voting), soft voting returns the class label as argmax of the sum of predicted probabilities. With log-loss score as the metric, and our goal of predicting probabilities of the outcomes, ‘soft voting’ is chosen.

The best 3 models for me were Calibrated Classifier, Random Forest Classifier and Logistic Regression.



```
# Voting Classifier over Calibrated Classifier, Random Forest Classifier and Logistic Regression
eclf2 = VotingClassifier(estimators=[('lr', alg), ('calC', alg7), ("ranFor", alg2)], voting='soft')
eclf2.fit(train[features], train["OutcomeType"])
probs = eclf2.predict_proba(my_test[features])
score = log_loss(my_test["OutcomeType"], probs)
print(score)
```

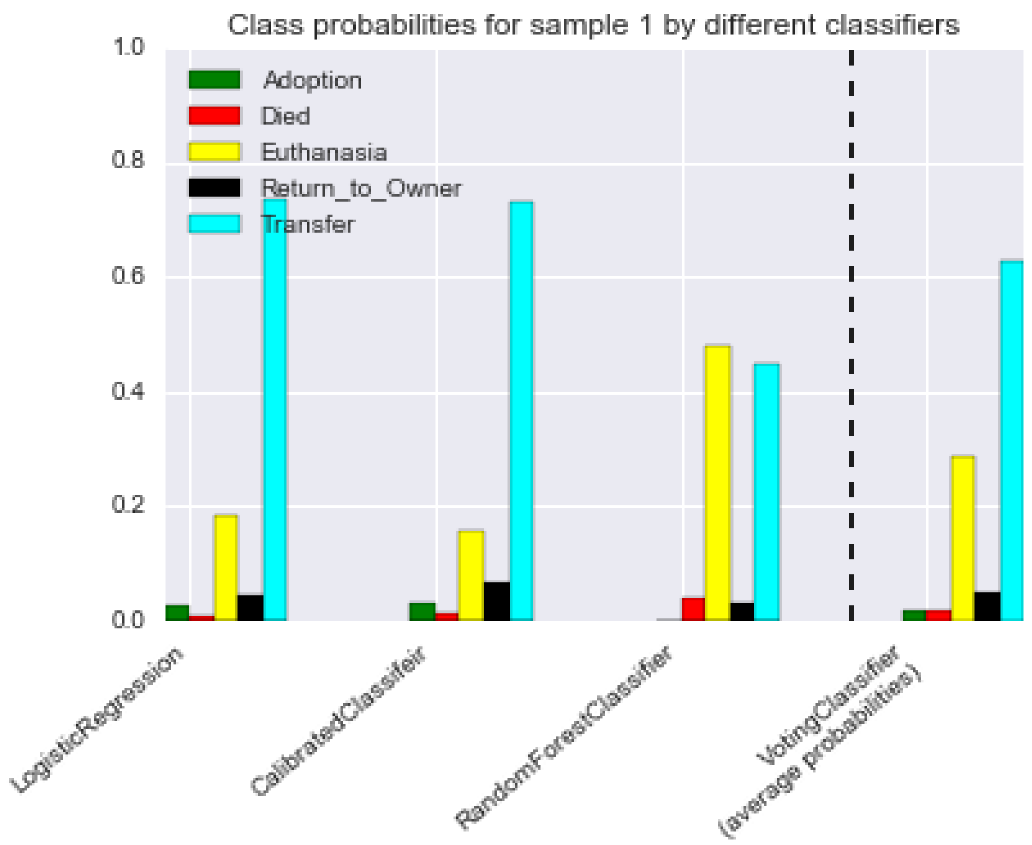
```
0.87911009239
```

The log loss score of 0.879 is the best so far and hence selected as the final model.

IV. RESULTS

Model Evaluation and Validation

Using the final model – a combination of Calibrated Classifier, Random Forest Classifier and Logistic Regression, stitched through Voting Classifier, I received a log-loss score of 0.91876 on the entirely unseen test set, provided by Kaggle, which has been my best score so far. An example of the voting classifier working is shown below:



Justification

Since, the obtained log-loss score is way lower than the benchmark score of 20.25113 established in a previous section, the model can be assumed to be performing great, and hence can be approved. Also, a test score of 0.879 and 0.91876 submission score by Kaggle showcases that our model is generalizing well. Training score of 0.8921 and testing score of 0.879 shows that we are not overfitting the training set.

V. CONCLUSION

So far we have:

- Pre-processed data, remove missing values, and created dummy variables for better feature-set.

- Used several classifiers :Calibrated Classifier, Random Forest Classifier, Decision Tree Classifier, AdaBoost Classifier and Logistic Regression.
- Used VotingClassifier from sklearn to obtain better results.

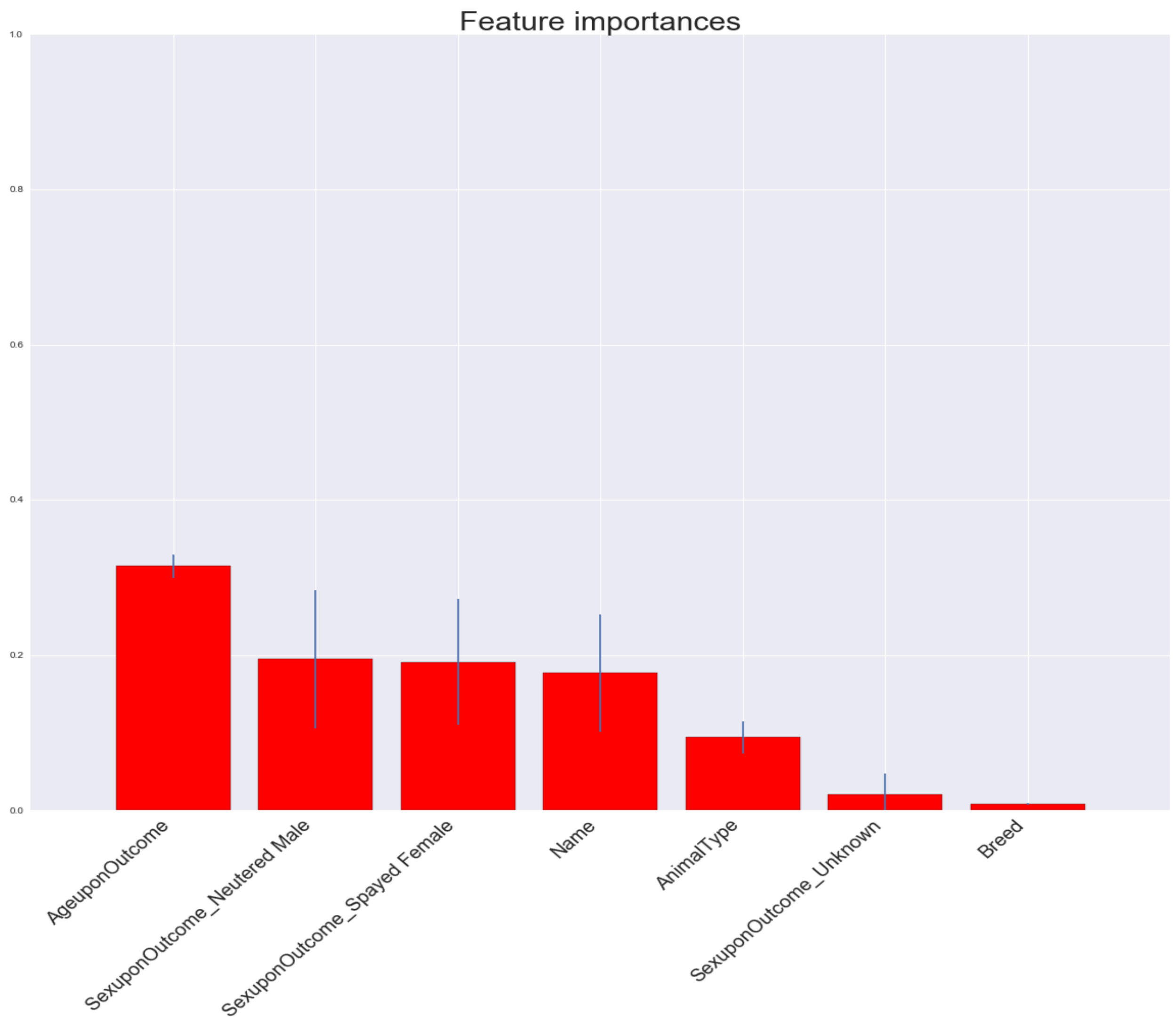
Before we wrap up, let's see what features proved to be the most important ones in solving this multi-class classification problem.

Free-Form Visualization

The feature scores obtained using feature_importances_ method of ensemble classifiers are :

```
Feature ranking:
1. AgeuponOutcome (0.317164)
2. SexuponOutcome_Spayed Female (0.195914)
3. SexuponOutcome_Neutered Male (0.188271)
4. Name (0.174260)
5. AnimalType (0.095136)
6. SexuponOutcome_Unknown (0.021530)
7. Breed (0.007725)
```

Using the feature importances plot (shown below), we notice that AgeUponOutcome is the most relevant feature determining the outcome of the animals. It is expected because, younger animals are more likely to be adopted, transferred and returned to the owner than the older animals. We have already discussed that neutered and spayed animals have more chances to face the good set of outcomes in our exploratory visualization section. Similarly, we discussed, how having a name was important. Surprisingly, the breed feature seems to be the least important,. This just means that our method of classifying breed into pure or mix wasn't very effective. The result doesn't imply that we throw away the breed feature. Instead, we should find a better classification way for breed attribute, something like whether the animal is considered dangerous or not.



Reflection

- Exploratory visualization proved to be very useful, as it guided our feature selection.
- Using some clever tricks like creating dummy variables, we managed to bring the log-loss score down by a huge margin.
- Feature importance plots are found to be advantageous as we get to know the most relevant features in terms of impact on the outcomes of the animals.
- Classifying breed into pure and mix didn't bear fruits, but this didn't render the breed attribute useless.
- Voting Classifier helped to merge the predictions of the chosen classifiers. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses.

- Neutered Male and Spayed Female have a great chance to be adopted, and a pretty decent chance to be returned to the owner. However, it appears harder to find new family for Intact Male and Intact Female!
- We have shown how important it is for an animal to have an ID tag. The animals with unknown names have almost no chance to return to owners and have less chance to be adopted.

Improvement

- We can find out which breeds are considered dangerous which will be more likely to be euthanized; similarly, some breeds, and animal colors are very popular and very likely to be adopted. However, more information from several other definite sources have to be obtained to perform this analysis.
- DateTime variable might be interesting. Maybe Euthanizations are performed early morning, while adoptions happen more on weekends. However, this will require much more data wrangling and less of new machine learning concepts.
- Utilizaing the above schemes appear to be decreasing the log-loss score by some margin. In particular, DateTime attribute seems highly important. If it's euthanization time, then all other features will be rendered useless. If this is indeed true, then exploiting it doesn't seem fair to me.