

Animal Shelter Outcome

Definition

Project Overview

The data is from Kaggle Shelter Animal Outcome competition and you can find information from <https://www.kaggle.com/c/shelter>animal>outcomes>.

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.

Project Statement

Goal

- a. Using a dataset of intake information including breed, color, sex, and age from the Austin Animal Center, participants are expected to predict the outcome for each animal.
- b. The Austin Animal Center also believes this dataset can help it understand trends in animal outcomes. These insights could help shelters focus their energy on specific animals who need a little extra help finding a new home.

2

How to solve?

We aim to predict the outcomes of animals and understand trends in animal outcomes. Thus, it is a classification problem. I tend to use Decision Tree, Random Forest, Naïve Bayes, and other classification algorithms. Some visualization graphs are essential to show the trends in animal outcomes. I am going to group the animal by different groups and draw charts to show how The Austin Animal Center can distribute their efforts to help those who are least likely to be adopted.

$$\text{logloss} = -\frac{1}{N} \sum_{i=1}^N (y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

N is the number of data, i is an animal and j is the probable outcome of the animal. In this case probable outcomes are return_to_owner, euthanasia, adoption, transfer, and died and thus M is 5. P_{ij} is the probability of animal i ends up with the outcome j.

Let the true labels for a set of samples be encoded as a 1-of-K binary indicator matrix, i.e., if sample has label taken from a set of labels. If the animal is classified correctly, log loss will be close to 0, and it penalizes incorrect classification by a positive log loss. What's more, log loss function penalizes incorrect classifications which you've assigned a high probability more seriously than incorrect classifications which you've assigned a low probability. Minimize multi class log loss can be combined with maximum likelihood estimation. I need to predict the probability of each outcome type for each animal in test data and the small logloss valuation is, the better the model will be.

Analysis

Data Exploration

There are 26729 data in training data set, and 11456 data in testing data set.

ID	Name	OutcomeType	SexuponOutcome	train/test
A671945	Hambone	Return_to_owner	Neutered Male	train
A656520	Emily	Euthanasia	Spayed Female	train
A686464	Pearce	Adoption	Neutered Male	train
A683430	NaN	Transfer	Intact Male	train
A667013	NaN	Transfer	Neutered Male	train

ID: the number of the animal which should be dropped in analysis because ID of each animal is unique and it does not make any sense in prediction

Name: The name of animal.

OutcomeType: That is what we are going to predict.

SexuponOutcome: This column includes two features, sex and intact type. The animals are divided by Female and Male. Also they are divided by neutered, spayed or intact pets.

Train/test: In order to process both train and test data more easily, I use `pd.concat` to combine `train.csv` and `test.csv` and add a new column called 'train/test' to show where the data is originally from.

	Breed	Color	breedmix	colormix
0	Shetland Sheepdog Mix	Brown/White	mixed	mixed
1	Domestic Shorthair Mix	Cream Tabby	mixed	pure
2	Pit Bull Mix	Blue/White	mixed	mixed
3	Domestic Shorthair Mix	Blue Cream	mixed	pure
4	Lhasa Apso/MMiniature Poodle	Tan	mixed	pure
5	Cairn Terrier/Chihuahua Shorthair	Black/Tan	mixed	mixed
6	Domestic Shorthair Mix	Blue Tabby	mixed	pure
7	Domestic Shorthair Mix	Brown Tabby	mixed	pure
8	American Pit Bull Terrier Mix	Red/White	mixed	mixed
9	Cairn Terrier	White	pure	pure

Age: The original data is string and it should be changed into number.

AnimalType: Is the animal cat or dog?

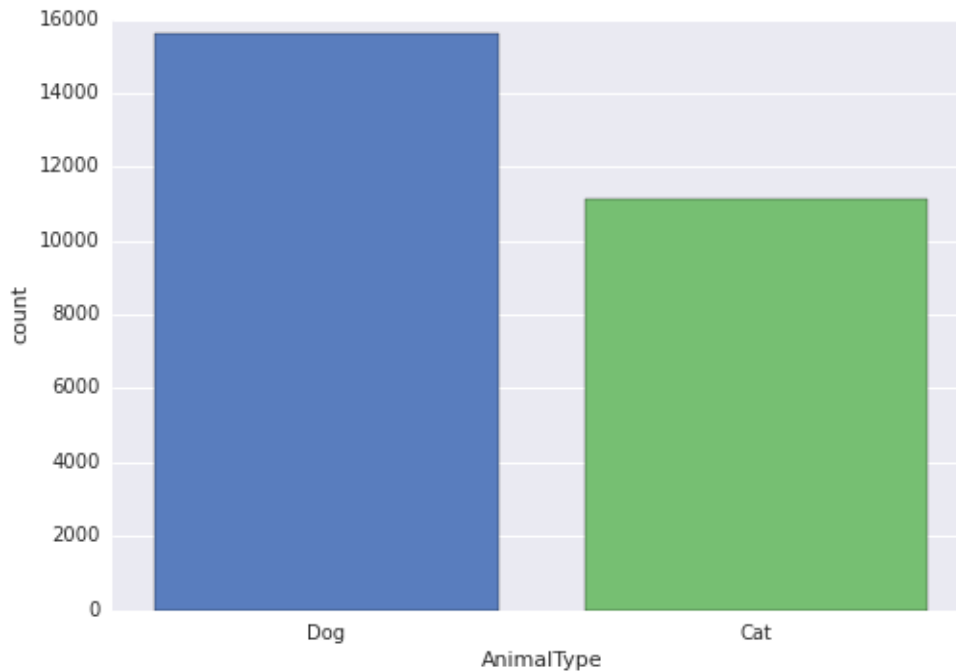
Breed: Animal breed. There are more than 1300 animal breeds into training data and

it is impossible to use all 1300 breeds. The most common breed is Domestic Shorthair Mix.

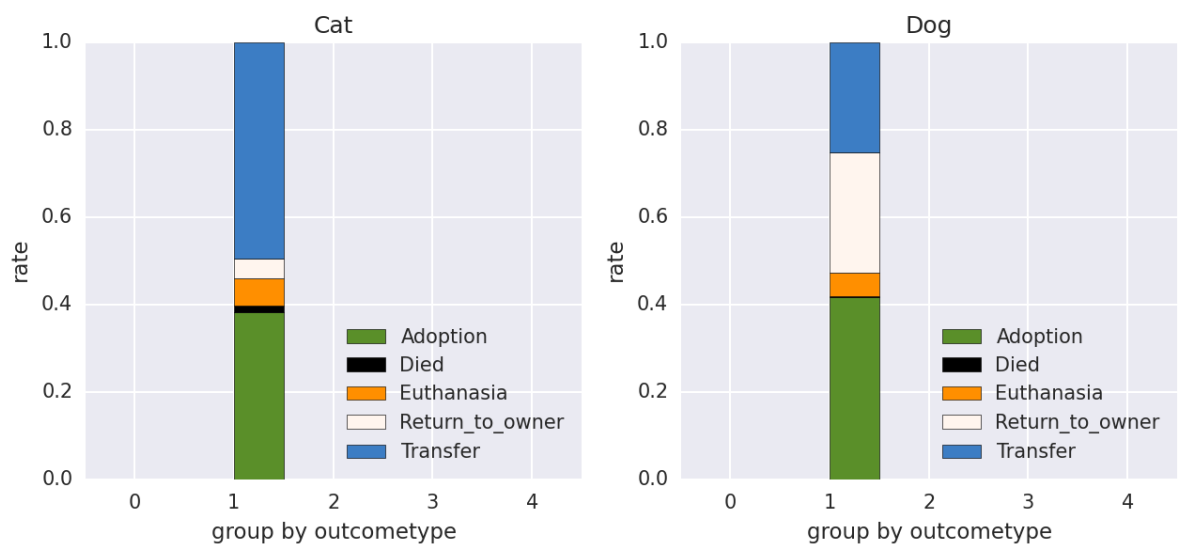
Color:Color reflects both fur type and color type but not all animal has a pattern of fur type. The most common color is Black/White.

DateTime:The information of year, month, date, hour, weekday can be extract from date and time.

Exploratory Visualization

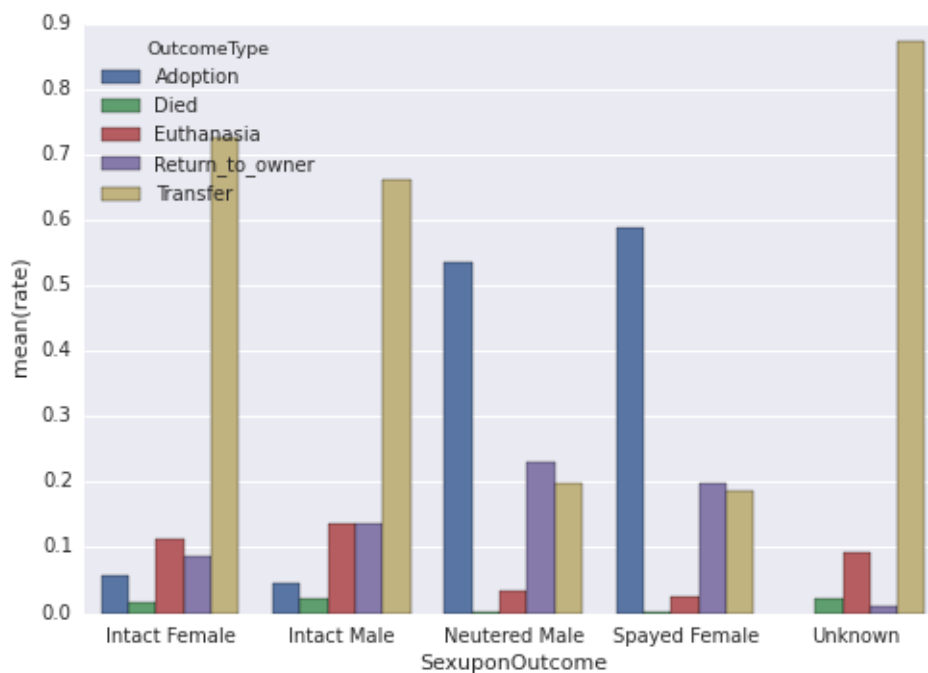


The first count plot shows the number of Dog and Cat in train data and the number of dog is about 5000 more than that of cat.



Obviously, most of cats are transferred and adoption is the second most common outcome for cats. Most of dogs are adopted and rates of adoption of cats and dogs are similar. About 20% of dogs are returned to their owners while that rate of cats is only

5%. Moreover, rate of died accounts for the least proportion in both animals. In conclusion, animal type should be an influenced factor because bars above show different trends in two animals.



From the bar chart above, I consider 'sexuponcome' as an important factor in deciding the outcome of an animal. As you can see in the count plot, there are 5 types of sex upon outcome, and it can be divided into sex type and fertility type. Neutered Male and Spayed Female animals have highest adoption rate and Return_to_owner rate, they are least likely to be died and euthanasia. If an animal's sexuponoutcome is unknown, this animal has no chance to be adopted.



Algorithms and Techniques

It is a classification problem and the feature is not constant and thus, random forest is appropriate to be used in this case.

However, I decide to use xgboost algorithms because xgboost utilize gradient boosting and random forest algorithms and more importantly, by using multiple threads, the data will be trained quickly. Xgboost is based on Gradient Boosting and Random Forest, it gives different weights on data. For each time, xgboost gives higher weight on those that was classified incorrectly, and those classified correctly will weight less importantly. In this way, xgboost has better accuracy with less trees.

How to measure correct classification or incorrect ones? Precisely, xgboost measure training loss plus complexity of model.

$$Obj = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

 Training loss
 Complexity of the Trees

If the objective is large, that means loss is large and xgboost gives more weights on it. It is similar to our revision. When preparing examinations, we will do more exercise on wrong answers.

Another reason is that, according to experience of many kaggle winners, xgboost is a robust method in classification and is the one they will try first in training data. And that is what I learn from reading many kaggle winner interview and decide to choose xgboost. Although, neural network is another robust algorithm in classifying, I don't understand this method so far. SVM seems to also be a applicable algorithm and thus I will try it.

Parameters to be fitted including:

param['objective']: the outcome type has 5 categories and we need log loss to evaluate

outcome and thus 'multiprob' is used

param['eta']: the learning rate of xgboost

param['max_depth']: the depth of decision tree to avoid overfitting problem.

param['num_class'] = 5

param['eval_metric'] = "mlogloss"

num_round: how many times does xgboost take to train data

Benchmark

As the data is extracted from Kaggle competitions, I suppose to use the other kagglers' loss results as the benchmark. used the median of kaggle learderboard top 500(0.73148) 10 days before the end of completion as the benchmark. Leaderboard is a board to present participants' model performance on testing data and it is ranked from the best to the worser. The lower the LEADERBORAD score is , the better your model is.

Methodology

Data Preprocessing

Features in the data are:

```
Index([u'AgeuponOutcome', u'AnimalType', u'Breed', u'Color', u'DateTime',  
       u'ID', u'Name', u'OutcomeType', u'SexuponOutcome',  
       u'traintest'],  
      dtype='object')
```

1.AgeuponOutcome

I regard year as 365 days, months as 30 days, week as 7 days. There are 24 missing

value in AgeuponOutcome and the missing value is replaced by 0 days.

The new number of age is stored into a new columns call 'age'

2.AnimalType

Do not need to change

3.Breed:

The data has more than 1300 unique features including pure type, mixed type,

and '/' type. I am thinking about two methods to deal with breed.

Method 1 is to identify unique features and mixed feature. For example, if the

breed is

Shetland Sheepdog Mix, column['Shetland Sheepdog'] is 1, column['Mix']

is 1, the other columns are 0. If the breed is Lhasa Apso/Miniature Poodle,

column['Lhasa Apso'] is 1, column['Miniature Poodle'] is 1, the other columns are 0.

Method 2 is much simpler. If breed contains 'Mix' or '/', it should be classified as

'mixed', otherwise, it is 'pure'.

4.Color

Method 1 and Method 2 above also apply to color feature.

5.DateTime

Library datetime provides a convenient way to extract year, month, day,

weekday, hour.

6.ID

Drop ID

7.Name

If the pet has name, it should be 1 and otherwise it is 0. The new data is in new

column called 'named'

8. SexuponOutcome:

Noticeably, some animals' SexuponOutcome feature is unknown and there are two

ways to deal with the miss value. Method 1 is to replace the missing value as

'Unknown' and classify it as an unique category. Method 2 is to replace the missing

value of sex as 'Male' in dog and 'Female' in cat, and the missing value of intact

type as 'Neutered' as these are the most common types.

cat

Neutered Male	4457
Spayed Female	4320
Intact Female	3002
Intact Male	2718
Unknown	1437

dog

Neutered Male	9557
Spayed Female	8313
Intact Male	2267
Intact Female	2002
Unknown	111

However, after analyzing SexuponOutcome, I don't think Method 2 should be implemented because for cats, the number of unknown sex cat are

more than a quarter of the most common Neutered Male cats. Moreover, the number of female&male and the number of neutered&spayed type in both cats and dogs are just 10% different and thus it is arbitrary to use Method 2.

9. Train and test

After preprocessing the data, train and test data are separated to fit and predict data. At the end, all features that are not float, int or bool type are transformed and fit by preprocessing.LabelEncoder().

Implementation

As mentioned above, I try to use SVM to train data. However, it takes me more than 5 minutes to fit the data in a simple try using whole features. That is time-consuming and not efficient to apply in reality. Thus, I drop SVM and continue using XGBOOST.

To be more clearly, I will illustrate more detailedly what is Method1 and Method2 in features selection.

Method 1

As there are over 1300 types of breed and over 300 types of color in the dataset, it is unwise to get dummies variable of every breed type. Firstly, I get the unique type of breeds. I can't not use value_counts() in this case because there are many animals are mixed breed. I split the breed by 'Mix' or '/', get names of pure breed.

Secondly, new columns are created and named by unique pure breed name and 'Mix'.

Thirdly, apply function to original data breed columns.

	Breed	train	Pug	English Pointer	Bull Terrier M	Unknown	Bedlington Terr	Smooth Fox Terrier	Bengal	Black Mouth Cur	...	Blue Point	Gold	Tricolor	Calico Point	Yellow	Silver
0	Shetland Sheepdog Mix	train	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
1	Domestic Shorthair Mix	train	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
2	Pit Bull Mix	train	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
3	Domestic Shorthair Mix	train	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
4	Lhasa Apso/MMiniature Poodle	train	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False

From the table above, you can get what I mean. For index0,

Breed is Shetland Sheepdog Mix, then Mix is True, Shetland Sheepdog is True and other columns is False. For index 4, Lhasa Apso and MMiniature Poodle are True, the other is False. Specially, the original breed of index 4 in Lhasa Apso/Miniature Poodle, when I want to separate breed contained 'Mix' or not, I split breed by 'Mi', but 'Mi' is also in 'Miniature', and in order not to split 'Mi' in 'Miniature', I add an 'M' in 'Miniature'. The same logic applies to Color.

Method 2

Method2 is much simpler than Method1. I just simply classified Breed as 'pure' or 'mixed' according to whether 'Mix' or '/' in breed. The same to color.

	Breed	Color	breedmix	colormix
0	Shetland Sheepdog Mix	Brown/White	mixed	mixed
1	Domestic Shorthair Mix	Cream Tabby	mixed	pure
2	Pit Bull Mix	Blue/White	mixed	mixed
3	Domestic Shorthair Mix	Blue Cream	mixed	pure
4	Lhasa Apso/MMiniature Poodle	Tan	mixed	pure
5	Cairn Terrier/Chihuahua Shorthair	Black/Tan	mixed	mixed
6	Domestic Shorthair Mix	Blue Tabby	mixed	pure
7	Domestic Shorthair Mix	Brown Tabby	mixed	pure
8	American Pit Bull Terrier Mix	Red/White	mixed	mixed
9	Cairn Terrier	White	pure	pure

Actually, features selection is very important and I try to find the best features. As I have mentioned above, there are 2 ways to deal with breed and color. To make it more clearly, I will iterate what is method 1 and method 2. Method 1 is to get dummies variables of breed, if it is Domestic Shorthair Mix, then Domestic shorthair is 1, Mix is 1, and other is 0 and so does color. Method 2 is simply clarify mix or not mix, if '/' or 'mix' in the Breed or Color, then it should be 'mix', otherwise, it should be 'non mixed'. In order to select best features, I run 5 models.

For all the 5 models, I use the default parameters.

According to Xgboost official guide, general parameters, booster parameters and learning parameters are the 3 main parameters in Xgboost.

General Parameters

- 1) **Booster**[default=gbtree/gblinear]
gbtree:tree-based models
gblinear:linear-models
- 2) **silent** [default=0]:
Silent mode is activated is set to 1, i.e. no running messages will be printed. It's generally good to keep it 0 as the messages might help in understanding the model.

Booster Parameters

- 1) **eta** [default=0.3]
 - a) Analogous to learning rate in GBM
 - b) Makes the model more robust by shrinking the weights on each step
 - c) Typical final values to be used: 0.01-0.2
- 2) **min_child_weight** [default=1]
 - a) Defines the minimum sum of weights of all observations required in a child.
 - b) This is similar to **min_child_leaf** in GBM but not exactly. This refers to min "sum of weights" of observations while GBM has min "number of observations".
 - c) Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
 - d) Too high values can lead to under-fitting hence, it should be tuned using CV.
- 3) **max_depth** [default=6]
 - a) The maximum depth of a tree, same as GBM.
 - b) Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
 - c) Should be tuned using CV.
 - d) Typical values: 3-10
- 4) **max_leaf_nodes**

- a) The maximum number of terminal nodes or leaves in a tree.
- b) Can be defined in place of max_depth. Since binary trees are created, a depth of 'n' would produce a maximum of 2^n leaves.
- c) If this is defined, GBM will ignore max_depth.

5) **gamma [default=0]**

- a) A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split.
- b) Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned.

6) **max_delta_step [default=0]**

- a) In maximum delta step we allow each tree's weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative.
- b) Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced.
- c) This is generally not used but you can explore further if you wish.

7) **subsample [default=1]**

- a) Same as the subsample of GBM. Denotes the fraction of observations to be randomly samples for each tree.
- b) Lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting.
- c) Typical values: 0.5-1

8) **colsample_bytree [default=1]**

- a) Similar to max_features in GBM. Denotes the fraction of columns to be randomly samples for each tree.
- b) Typical values: 0.5-1

9) **colsample_bylevel [default=1]**

- a) Denotes the subsample ratio of columns for each split, in each level.
- b) I don't use this often because subsample and colsample_bytree will do the job for you. but you can explore further if you feel so.

10) **lambda [default=1]**

- a) L2 regularization term on weights (analogous to Ridge regression)
- b) This used to handle the regularization part of XGBoost. Though many data scientists don't use it often, it should be explored to reduce overfitting.

11) **alpha [default=0]**

- a) L1 regularization term on weight (analogous to Lasso regression)
- b) Can be used in case of very high dimensionality so that the algorithm runs faster when implemented

12) **scale_pos_weight [default=1]**

- a) A value greater than 0 should be used in case of high class imbalance as it helps in faster convergence.

Learning Parameters:

1) **objective [default=reg:linear]**

- a) This defines the loss function to be minimized. Mostly used values are:
 - i. **binary:logistic** –logistic regression for binary classification, returns predicted probability (not class)
 - ii. **multi:softmax** –multiclass classification using the softmax objective, returns predicted class (not probabilities)
 - 1. you also need to set an additional **num_class** (number of classes) parameter defining the number of unique classes
 - iii. **multi:softprob** –same as softmax, but returns predicted probability of each data point belonging to each class.

2) **eval_metric [default according to objective]**

- a) The metric to be used for validation data.
- b) The default values are rmse for regression and error for classification.
- c) Typical values are:
 - i. **rmse** – root mean square error
 - ii. **mae** – mean absolute error
 - iii. **logloss** – negative log-likelihood
 - iv. **error** – Binary classification error rate (0.5 threshold)
 - v. **merror** – Multiclass classification error rate
 - vi. **mlogloss** – Multiclass logloss
 - vii. **auc**: Area under the curve

3) **seed [default=0]**

- a) The random number seed.
- b) Can be used for generating reproducible results and also for parameter tuning.

For Feature2 to Feature4, other include [u'AnimalType', u'year', u'month', u'day', u'weekday', u'hour', u'age', u'sex', u'intact']

	Features	evals_result
Feature1	All columns	0.74225
Feature2	Breed:Method1 Color:Method1+other	0.73668
Feature3	Breed:Method2 Color:Method1+other	0.74997
Feature4	Breed:Method1 Color:Method2+other	0.74997
Feature5	Breed:Method2 Color:Method2+other	0.74759

In conclusion, Features2 has best results and I will select features in Features2 to find the best estimator.

Refinement

Firstly, I want to tune is max_depth. In order to get high accuracy, learning step should be small and train data for many rounds.

```
param['eta'] = 0.02
```

```
param['silent'] = 1
```

```
param['nthread'] = 4
```

```
param['num_class'] = 5
```

```
param['eval_metric'] = 'mlogloss'
```

```
param['subsample']=0.8
```

```
param['verbose'] = False
```

```
param['early_stopping_rounds'] = 10
```

Max_depth	Test_error
8	0.733149
9	0.729826
10	0.727181
11	0.725523
12	0.726005

Max_depth = 11 is the best.

Results

Model Evaluation and Validation

Since the data is from kaggle completion, the best way to validate the robustness of the model is to use the model to predict test data provided in the competition and make a submission to check leaderboard score.

	eta	round	depth	Test error	LEADERBORAD(score)
Para1	0.02	700	11	0.72972	0.70988
Para2	0.02	650	11	0.72924	0.70927
Para3	0.02	800	11	0.73139	0.71115
Para4	0.015	1000	11	0.72996	0.71086
Para5	0.02	500	11	0.72864	0.71115
Para6	0.01	1200	11	0.72680	0.7094

Para1:

```
param['eta'] = 0.02
```

```
param['silent'] = 1
```

```
param['nthread'] = 4
```

```
param['num_class'] = 5
```

```
param['eval_metric'] = 'mlogloss'
```

```
param['subsample']=0.8
```

```
param['verbose'] = False
```

```
param['early_stopping_rounds'] = 10
```

```
param['round']=700
```

```
param['max_depth']=11
```

para2:

```
param['eta'] = 0.02
```

```
param['silent'] = 1
```

```
param['nthread'] = 4
```

```
param['num_class'] = 5
```

```
param['eval_metric'] = 'mlogloss'
```

```
param['subsample']=0.8
```

```
param['verbose'] = False
```

```
param['early_stopping_rounds'] = 10
```

```
param['round']=650
```

```
param['max_depth']=11
```

Para6

```
param['eta'] = 0.01
```

```
param['silent'] = 1
```

```
param['nthread'] = 4
```

```
param['num_class'] = 5
```

```
param['eval_metric'] = 'mlogloss'
```

```
param['subsample']=0.8
```

```
param['verbose'] = False
```

```
param['early_stopping_rounds'] = 10
```

```
param['round']=1200
```

```
param['max_depth']=11
```

para1,2,6 refers to different parameters I test.

Test error is the log loss for testing data split from the train data and Leaderboard score is the log loss error of test data which is shown after submission in Kaggle.

In terms of test error, para5 and para 6 is the best parameters and in terms of Leaderboard score, para2 and para 6 is the best model. I will compare the para2 and para6 by statistic hypothesis.

Ho:log_loss of para2 > log_loss of para 6

H1:log_loss of para2 <= log_loss of para 6

```
Ttest_indResult(statistic=-3.0647963780112253, pvalue=0.0027937437371030193)
```

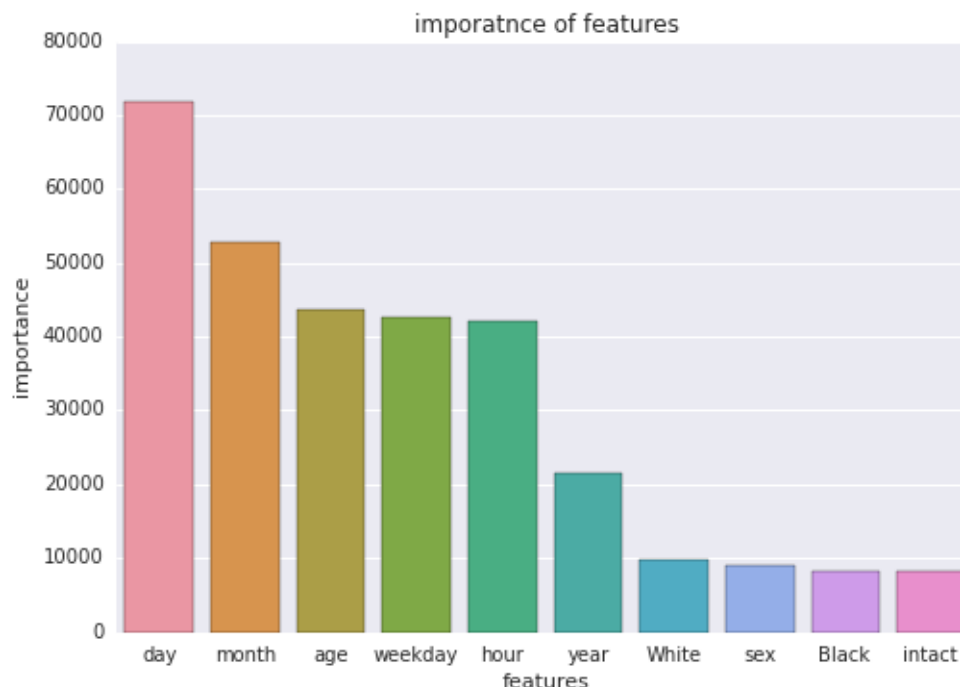
pvalue is less than 5% and thus H0 can be rejected. So log loss of para2 is less than log loss of para6, and model with para2(eta=0.02, round=650) is the best model.

Justification

The benchmark is 0.73418 and it is much better than it. Parameter 2 has the best performance in leaderboard. Since the best team got 0.32 multi log loss score, I think my model is good but it still has a long way to solve the problem.

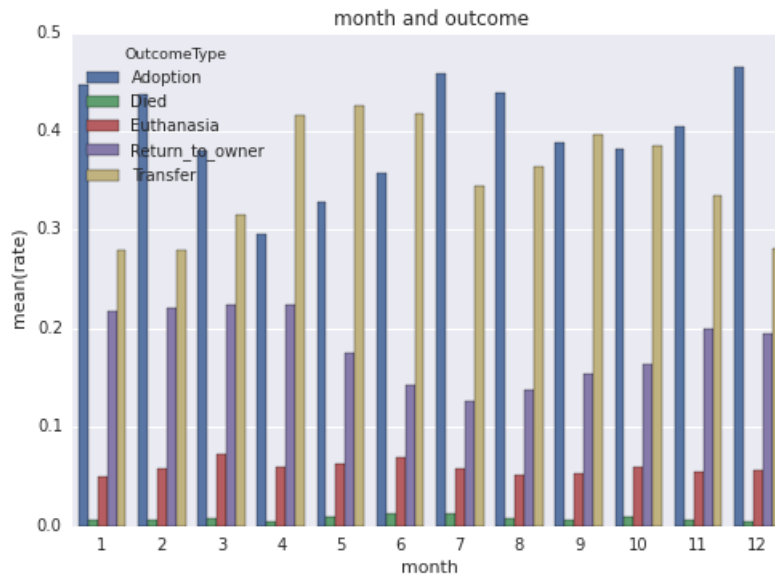
Conclusion

Free-From Visualization



I select top 10 most important features in the model. The date time is the most important feature and animals' age, sex and fertility type are also important. Noticeably, black and white color are

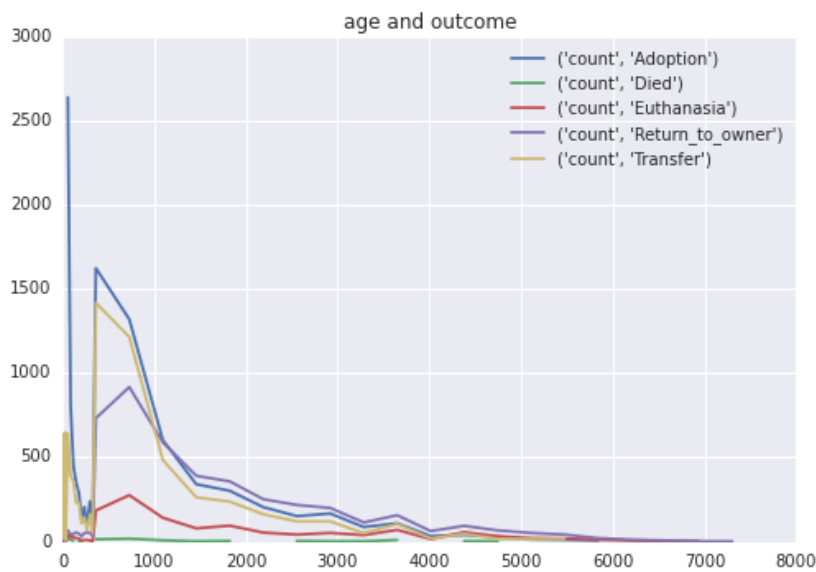
included in the top 10 important features. People may have special preference in these two types of color animal. Surprisingly, animal type is not included in top 10, it is ranked 12. It means people may not care about cat or dog as much as I suppose to be.



Graph month and outcome shows that although adoption rate is highest among 5 outcome types almost all the year round, the adoption rate fluctuated through the year. In April, May, June and September, transfer rate reach the highest and it drops significantly in July. Return to owner rate is flat in the first 3 month and decreases in the second quarter, reaching the bottom in July and then climbs up slightly in the second half year. Euthanasia and Transfer rate change slightly in the year.

Recommendation 1

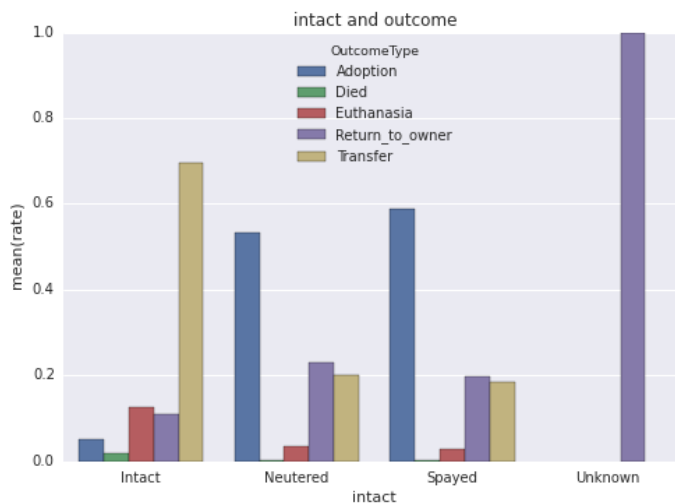
In the second and third quarter, Austin Animal Shelter should concentrate more resources in returning animals to their owner and avoid transferring too much animals. The best outcome for animals is to be adopted or returned to owner, it does not make any sense to their lives just change from one shelter to another one.



Graph age and outcome shows that the younger the animals are, the more likely they are to be adopted.

Recommendation 2

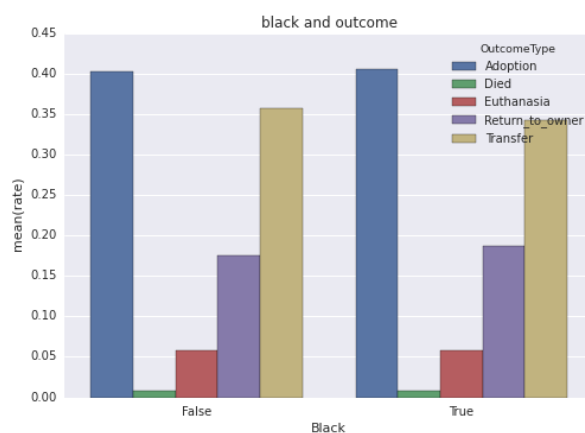
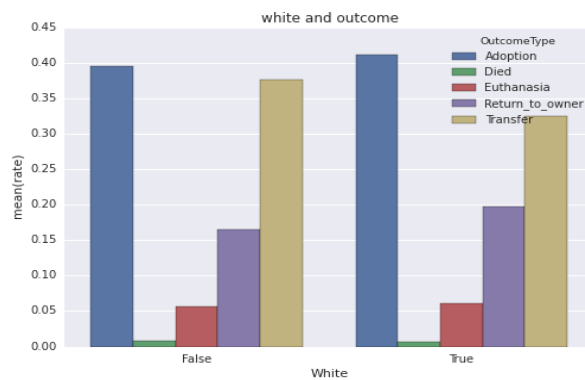
Focus on helping those who are older than 1 year



Graph intact and outcome indicates how lucky unknown fertility animals are because all of this kind of animals are returned to owners. However, intact animals are less lucky than them. The Died rate and Euthanasia rate of intact cats and dogs are the highest in 5 groups. I suppose intact animals are more likely to be ill.

Recommendation 3

Pay special attention to intact animals, try to take more lovely photos of this type and encourage people to adopt more intact animals.



Graph named and outcome suggests the importance to name animal. The adoption rate of named animals is more than twice of unnamed ones. Return to owner rate of named animals are significant higher than that of unnamed. On the contrary, unnamed pets are more likely to be transferred, died and euthanasia than named ones. People like to name their animals and they may think named animals would be more friendly to be their family members. It seems that the animal is unique once they get their names.

Recommendation 4

Try to name every animal in Austin Animal Shelter. Show people they are unique and they deserve to be cherished.

Reflection

Step 1

Simple Analysis

What features are in the data, what does it mean, how the outcome distribute according to some features and think about how to deal with outliers, missing value.

Step 2

Data Preprocessing

We cannot use features given in the original data directly, we have to clean the data and transform the data. For example, in the original data, age is given by 1 year, 2months, 5 days, etc., computer cannot identify what does it mean so I convert this column into days and shown as integer. Another example is feature breed, there are so many features in the data, and we must think about how to reduce dimensions.

Step 3

Select model & features

According to our understanding of data and goal, we select the fitted model and then, we have to select features. Not all features contribute to improve the final score and in order to test which features are included. I split train data into train_X, train_Y, test_X, test_Y, fit the model in train_X, train_Y, predict test_X, and finally, evaluate multi log loss by test_Y. By comparing score of different models with different combinations of features, we can find the ideal features and models.

Step 4

Tune Parameters

Tuning parameters is very important to improve performance, especially in kaggle competition. A tiny change in parameters can make you move up 100 in leaderboard. Usually, we can use GridSearchCV to tune parameters and find the best ones.

It is really very interesting to participate into kaggle competition. Sometimes, a very tiny change will make the classification result much much better. For example, there are five types of 'SexuponOutcome' It is commonsense to separate female and male, but I have no idea about spayed, neutered, and intact. I firstly classified intact as intact and the other type(spayed,neutered) as unintacted. The log loss on validation data is 0.78 but when I classified intact as intact, spayed as spayed, neutered as neutered, unknown as unknown, the validation data has been improved to 0.73. It is a great improvement and proves that sexuponoutcome is a very important feature in predicting outcome.

Another lessons I learn is that after finding the best parameters, don't forget to fit the model using the whole train data. I firstly just train_X data split from train data to fit and predict test data, and get 0.72 in kaggle leaderboard and when I realized I had made a mistake I feed the model by whole data, performance was improved significantly, reaching 0.70927 in leaderboard.

This is my first time to participate kaggle competition, and I have never learned machine learning before taking udacity Nandodegree. I met many challenges during the competition. Most of the difficulties arise from preprocessing data. I am not familiar with pandas and it takes me a long time to transform data. Another difficulty comes from xgboost. It is a new machine learning algorithms and I know it is used commonly in kaggle competition. I have never learned this algorithm before and thus, I have to find many information online. But luckily, I always find many useful guideline and tutorial. Actually, I still have no idea about the mathematics deduction behind xgboost, what I know so far is how to apply it.

I set the benchmark 0.73148 at the beginning, and my final LEADERBORAD score is 0.70927. That is much better than benchmark. And I get rank 72 in LEADERBORAD, that is top 10% of all teams. This is my first time to participate Kaggle and I am really satisfied with this achievement. It can be used in a general setting to solve these types of problems. It is efficient and robust to predict the final outcome.

Improvement

In the Color feature, both color type and fur type are given, but I do not consider fur in the model. Although only some animals' fur type are given, maybe I can utilize this information and make a more accurate prediction.

According to many kagglers' interview, xgboost seems to be the best algorithm in many classification problems, and I think so as well. But if I know exactly what does each parameter in xgboost mean, I can make better parameter predictions and if GridSearchCV can be applied in xgboost, it will be more convenient to tune parameters.

I rank

85 in the LEADERBORAD and the best LEADERBORAD score is 0.32. There must

exist some better feature selection methods and tuning parameter methods in predicting data.