Machine Learning Engineer Nanodegree - Udacity

Capstone Project

Kaggle Problem (with modification): Shelter Animal Outcomes

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

By Quan Tran

August 7th 2017

I. Definition

Project Overview

Each year millions of dogs enter animal shelters across the U.S. Many animals are given up as unwanted by their owners, while others are picked up after getting lost or taken out of cruelty situations. Animal entering shelters usually meet one of three fates: they are reclaimed, adopted or euthanized. Approximately 3.2 million shelter animals are adopted each year. Many of these animals find forever families to take them home, but just as many are not so lucky. Each year, approximately 2.6 million shelter animals are euthanized. However, this number has declined to 1.5 millions in 2011.[1]

Therefore, the task of predicting animals' outcomes can gain useful insights to the shelter's trends. Moreover, recognizing animal outcomes beforehand may shape shelter policy. Shelters can focus their energy on specific animals that need a bit more help to find their new home and utilize their resources effectively.

[1] https://www.aspca.org/animal-homelessness/shelter-intake-and-surrender/pet-statistics

We can perform such task by applying machine learning, especially supervised classification, on a dataset from Austin Animal Center (via Kaggle Competition)**.** The animal shelter dataset represent the status of animals as they enter and leave the Animal Center.

Problem Statement

The objective of this project is to predict if an animal is going to be **adopted, died, euthanized, returned to owner** or **transferred.** In order to do this, I perform and evaluate some different classification techniques to the animal dataset and compare and interpret the results by incorporating background knowledge and personal opinions. The best classification will be selected and applied to test set and predict the animal outcomes. The tasks involved in this project are:

- Data cleaning and feature engineering
- Exploratory Data Analysis
- Data Preprocessing
- Applying supervised learning algorithms
- Improvement

Metrics

The evaluation metric is taken from Kaggle Competition - Evaluation section:

The metric is multi-class logarithmic loss. Each incident has been labeled with one true class. For each animal, a set of predicted probabilities (one for every class) will be produced. 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, loglog is the natural logarithm, y_{ij} is 1 if observation i is in outcome j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to outcome j.

The 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

$$max(min(p, 1 - 10^{-15}), 10^{-15}).$$

II. Analysis

Data Exploration

As mentioned above, The dataset that I will be working with is obtain from the Austin Animal Center (via Kaggle competition). This dataset can be obtained from here

There are 10 features in training set, including the class attribute **Outcome Type**. Attribute *Outcome Sub-type* can be excluded since it provides additional information about animal outcome (such as 'suffering' for Euthanasia outcome, or 'foster' for Adoption outcome). This is an example of the data:

	Animal ID	Name	DateTime	Outcome Type	Animal Type	Sex upon Outcome	Age upon Outcome	Breed	Color
0	A741715	*Pebbles	2017-01-11 18:17:00	Adoption	Cat	Spayed Female	10 months	Domestic Shorthair Mix	Calico
1	A658751	Benji	2016-11-13 13:38:00	Return to Owner	Dog	Neutered Male	5 years	Border Terrier Mix	Tan
2	A658751	Benji	2016-05-06 16:48:00	Return to Owner	Dog	Neutered Male	4 years	Border Terrier Mix	Tan
3	A746650	Rose	2017-04-07 11:58:00	Return to Owner	Dog	Intact Female	1 year	Labrador Retriever/Jack Russell Terrier	Yellow
4	A750122	Happy Camper	2017-05-24 18:36:00	Transfer	Dog	Intact Male	1 month	Labrador Retriever Mix	Black

Data Leak Issue

There is an issue about this Kaggle competition, especially about some dataset features. Some of these issues were addressed in this Kaggle competition discussion forum. Features such as **Datetime of outcome**, **Age upon outcome** and **Sex upon Outcome** are achieved when outcome is already determined. Even though these features can be included to make better prediction, any models using these features are useless in real life. Also, using these features violates the rules of a Kaggle competition: "*"Your model should only use information which was available prior to the time for which it is forecasting".* these features can introduce potentially leaked information, which is mentioned in this Kaggle discussion here.

In fact, top 5 highest scores on this competition leaderboard exploited **DateTime upon outcome** attribute to get better log loss score. For example, *Transfer* type can be reliably predicted this way since when animals get transferred, they will leave exactly at the same time. Using this leak, an extreme score of 0.0 can be achieved, which is currently at the top of this competition leaderboard.

Therefore, I decide to exclude these features.

This dataset provides the data when animals leave the shelter. There is also another dataset that provides data when animals enter the shelter, which is called intake dataset.

The intake dataset can be found from the official Austin Animal Shelter website here.

This data introduces the properties of new animal at intake, such as **age upon intake**, **sex**, **breed**, **color** and **date & time of intake**. These intake features will replace those from the outcome dataset. This is how the intake data looks like:

	Animal ID	Name	DateTime	Intake Type	Intake Condition	Animal Type	Sex upon Intake	Age upon Intake	Breed	Color
0	A748291	*Madison	2017-05-01 14:26:00	Stray	Normal	Dog	Intact Female	10 months	Pit Bull Mix	Black
1	A730601	NaN	2016-07-07 12:11:00	Stray	Normal	Cat	Intact Male	7 months	Domestic Shorthair Mix	Blue Tabby
2	A748238	NaN	2017-05-01 10:53:00	Stray	Normal	Dog	Intact Male	3 years	Bichon Frise Mix	White
3	A683644	*Zoey	2014-07-13 11:02:00	Owner Surrender	Nursing	Dog	Intact Female	4 weeks	Border Collie Mix	Brown/White
4	A748635	NaN	2017-05-04 17:56:00	Stray	Normal	Cat	Unknown	9 months	Domestic Shorthair Mix	Blue

This intake dataset is mapped to the outcome dataset appropriately based on the animal ID. Based on domain knowledge of animal, feature engineering will be performed on some categorical features

Feature Engineering

1. Mapping intake data and outcome data

Since there are animals are in the intake data but on in the outcome data and vice versa, an inner join merge will be performed using Animal ID. There are some animals with several entries in the intake data and outcome data because they return to the shelter several times. Because of this, a feature called **'Intake Count'** will be generated to count how many times the animals have come back to the shelter.

For the dataset, it's best to allow only one entry for each animal. For animals with multiple intakes, we will choose the most recent one. For example, this puppy Lucy has 3 intakes, and the most recent one is on 2016-12-18 at 3:53 pm

Animal ID	Name	DateTime	Intake Type	Intake Condition	Animal Type	Sex upon Intake	Age upon Intake	Breed	Color
A740022	*Lucy	2016-12-18 15:53:00	Owner Surrender	Normal	Dog	Spayed Female	1 year	Maltese Mix	White
A740022	*Lucy	2016-12-10 13:37:00	Stray	Normal	Dog	Spayed Female	1 year	Maltese Mix	White
A740022	*Lucy	2016-12-14 17:39:00	Owner Surrender	Normal	Dog	Spayed Female	1 year	Maltese Mix	White

She has 3 outcomes, with the most recent one is 'adopted', on the same day, at 5:25 pm (90 minutes later, which is really fast!)

Animal ID	Name	DateTime	Outcome Type	Animal Type	Sex upon Outcome	Age upon Outcome	Breed	Color
A740022	*Lucy	2016-12-18 17:25:00	Adoption	Dog	Spayed Female	1 year	Maltese Mix	White
A740022	*Lucy	2016-12-15 11:51:00	Adoption	Dog	Spayed Female	1 year	Maltese Mix	White
A740022	*Lucy	2016-12-14 11:40:00	Adoption	Dog	Spayed Female	1 year	Maltese Mix	White

After merging, this should be her entry in our data:

Animal ID	Name_x	DateTime_x	Outcome Type	Animal Type_x	Sex upon Outcome	Age upon Outcome	Breed_x	Color_x	IntakeCount	Name_y	DateTime_y	Intake Type	Intake Condition	A T
A740022	*Lucy	2016-12-18 17:25:00	Adoption	Dog	Spayed Female	1 year	Maltese Mix	White	3	*Lucy	2016-12-18 15:53:00	Owner Surrender	Normal	

(DateTime_x is the outcome time, and DateTime_y is the intake time. Lucy's **IntakeCount** is 3)
After merging, all columns associated with the Outcome Data are dropped since the same information can be taken from the Intake Data

There are few animals that has missing outcome type, age, breed or colors. These records are dropped.

2. Feature Engineering

The **Breed** features contain more than 1000 unique value. Since we are going to perform one-hot encoding on categorical features, this will increase feature space complexity tremendously.

Breed provides several information about the animal. For example, "Chihuahua Shorthair/Dachshund" implies a cross breed between Chihuahua and Dachshund with short hair. Or "American Bulldog Mix" implies American Bulldog with mix breed and unknown hair length. Thus, I create **IsCross**, **IsMix** and **HairType** feature to address this characteristic

Breeds can be grouped together. One example is Beagle, Dachshund and Bloodhound breed can be grouped into Hound group. Thus, feature **BreedGroup** is created to scale down the number of breeds. This reduces Breeds to 7 groups for dogs and 2 groups for cats

Some breeds are 'Miniature', implying the smaller-size version of the breed, such as Miniature Poodle, Miniature American Shepherd. In order to see if size affects outcome, i create another feature **IsMiniature**.

Some dog breeds are known to be aggressive, such as Pit Bull, Rottweiler. Some insurance companies don't provide pet insurance for these breeds[2]. Based on domain knowledge, this breeds are less likely to be adopted. In order to see if this is true, feature **IsAgressive** is created.

For **Color** features, there are 467 unique colors recorded. Some animal has 1, 2 and 3 colors on them. Similar to breed, I group these colors into color groups: Light, Dark, Medium, Light/Dark, Dark/Light, Light/Medium, Medium/Light, Dark/Medium, Medium/Dark and Tricolor.

Sex features provide 2 pieces of information about the animal: whether the animal is neutered/spayed or intact, and whether animal is a male or female. Therefore, 2 features are created: **Intactness** and **AnimalSex.**

[2] https://www.psychologytoday.com/blog/canine-corner/201405/14-dog-breeds-blacklisted-insurance-companies

HasName feature is created to see whether having a name contributes to the outcome. A study says that cage cards information, such as name, license or tag, is used to determine the effect of past ownership on adoption[3]

NameFrequency feature is created to see whether popular animal name affect the outcome

Age feature is converted to numerical feature **Age in Days**

Datetime feature (date and time when animal enters the shelter) is broken down into *year***, month, hour, day, day of the week (or wday)** feature

After performing feature engineering mentioned above, we have a dataset from 2013-10-01 to 2017-07-19 with 57880 entries and 21 features, including the class feature: **'OutcomeType', 'IntakeCount', 'IntakeType', 'IntakeCond', 'AnimalType', 'BreedGroup', 'IsMix', 'IsMiniature', 'HairType', 'IsAggressive', 'IsCross', 'ColorGroup', 'AnimalSex', 'Intactness', 'HasName', 'NameFreq','AgeDays', 'month', 'hour', 'day', 'wday'.** Feature '**year'** is used in Exploratory Data Analysis to study the trend, but is excluded eventually as it does not have practical use in the future.

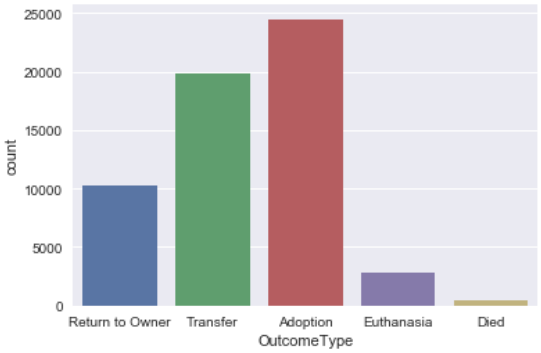
Most of these features for this dataset are categorical. **IntakeCount, NameFreq, AgeDays** and **Hour** are numerical features.

The original train and test set from Kaggle set are chosen randomly, regardless to datetime. To better suit my goal which is to predict outcomes for shelter animals in the future, animals with later datetime intake will be included in test set. Training set includes animal entering the shelter no later than 2016-09-01, test set includes ones after 2016-09-01 (78% / 22% split)

[3] Animal Shelter Dogs: Factors Predicting Adoption Versus Euthanasia
http://soar.wichita.edu/bitstream/handle/10057/3647/d10022_DeLeeuw.pdf

Exploratory Visualization

There are 5 outcomes and here is the distribution of them

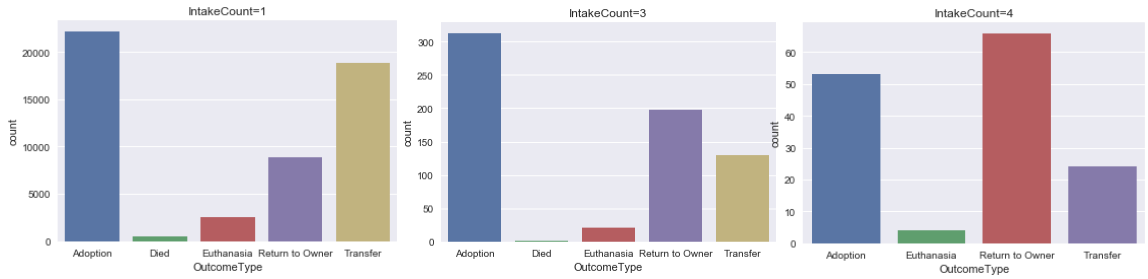


Adoption	24490 (42.311679 %)
Transfer	19856 (34.305460 %)
Return to Owner	10311 (17.814444 %)
Euthanasia	2750 (4.751209 %)
Died	473 (0.817208 %)

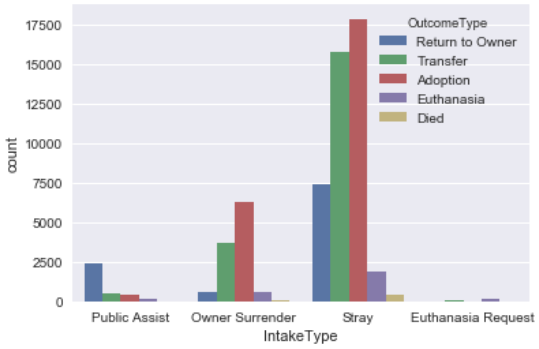
Few standout features are plotted and discussed.

- Intake Count

For new animals, they are more likely to be adopted or transferred. They still have a high chance to be adopted if they return to the shelter within 3 times. For more than 3 times, they are more likely to be returned to owner

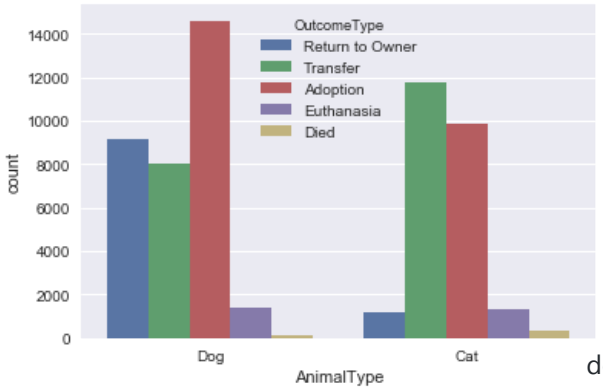


- Intake Type



Majority of animals entering the shelter are Stray animals, followed by Owner Surrender. Animals are more likely to be returned to owner if there intake type is Public Assist.

- Animal Type:

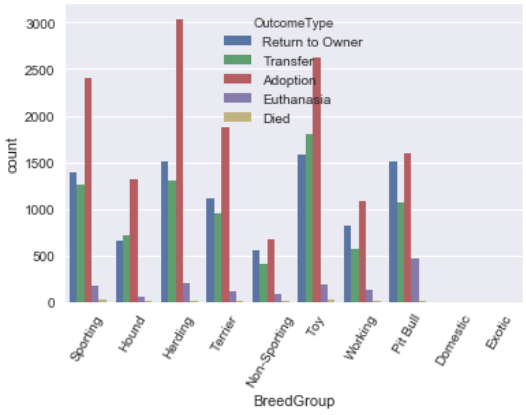


There are more dogs than cats (58% vs 42%) in the shelter. Dogs are more likely to be returned to owner than cat. Combining with other features, this feature can be a useful to predict 'Returned to owner' outcome.

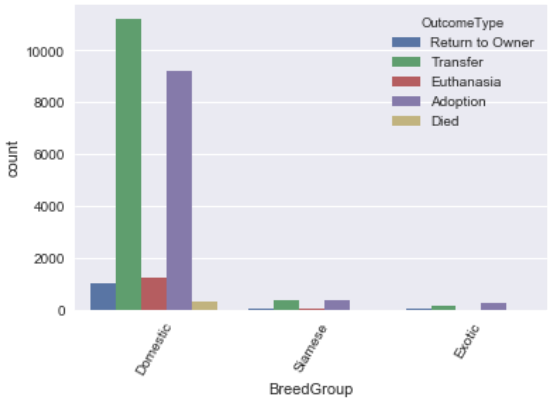
Also, cat are more likely to be transferred and dogs have higher chance to be adopted.

- Animal Breed group

Dog

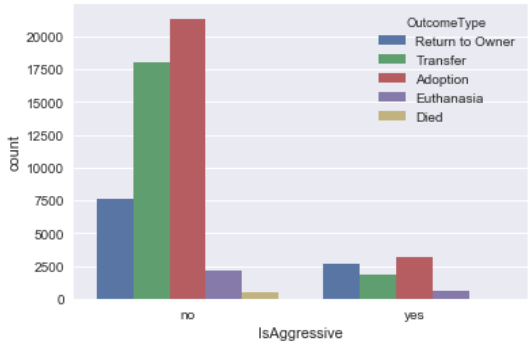


Cat



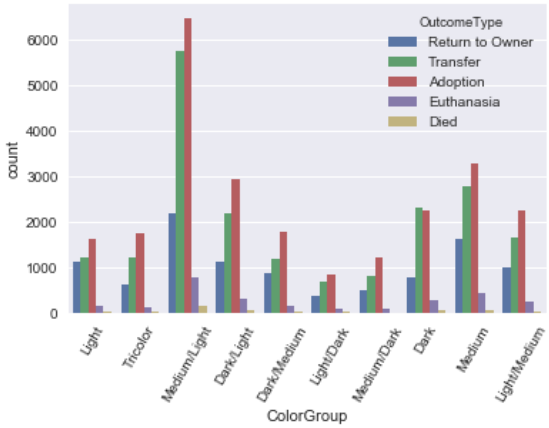
For dogs, the majority of breed group is 'Herding'. For cats, it is 'Domestic' group. All of them have highest probability in adoption, few dog breeds have higher transfer rate than return rate and vice versa, but they don't differ significantly, except for Pitbull: this group has high chance to be returned to owner, almost as high as Adoption chance. Sadly, Pitbull group has highest chance to be euthanized, comparing to all other groups

- Aggressiveness:



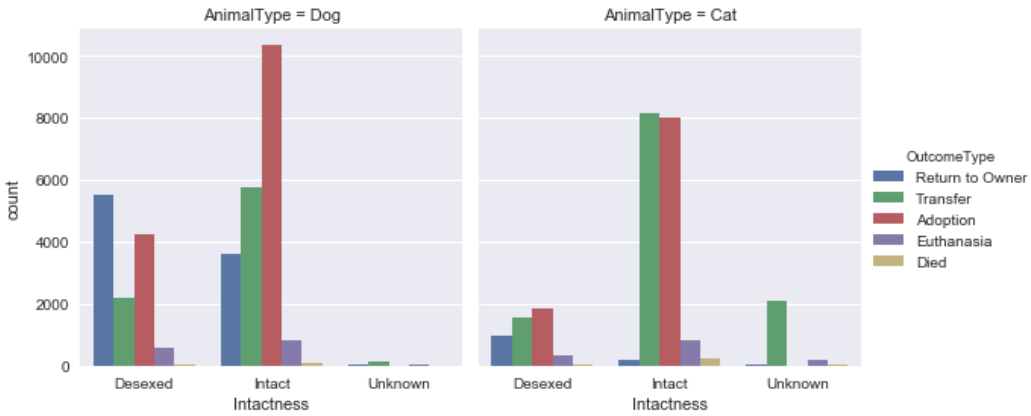
Not many animals are aggressive. For aggressive breed, they have a fairly same chance to be either returned or adopted. They are also less likely to get adopted.

- Color group



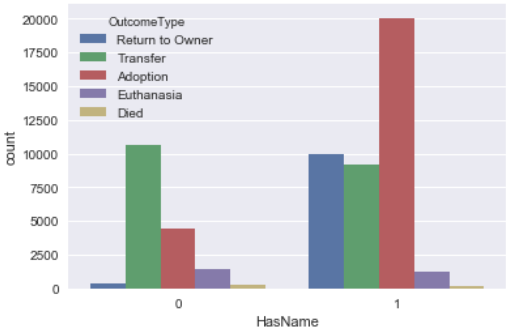
Most animals have medium/light features, and dark colored animal have the same chance to be either transferred or adopted. Same with Breed group attribute, not much information can be drawn from Color group

- Intactness and Animal Type



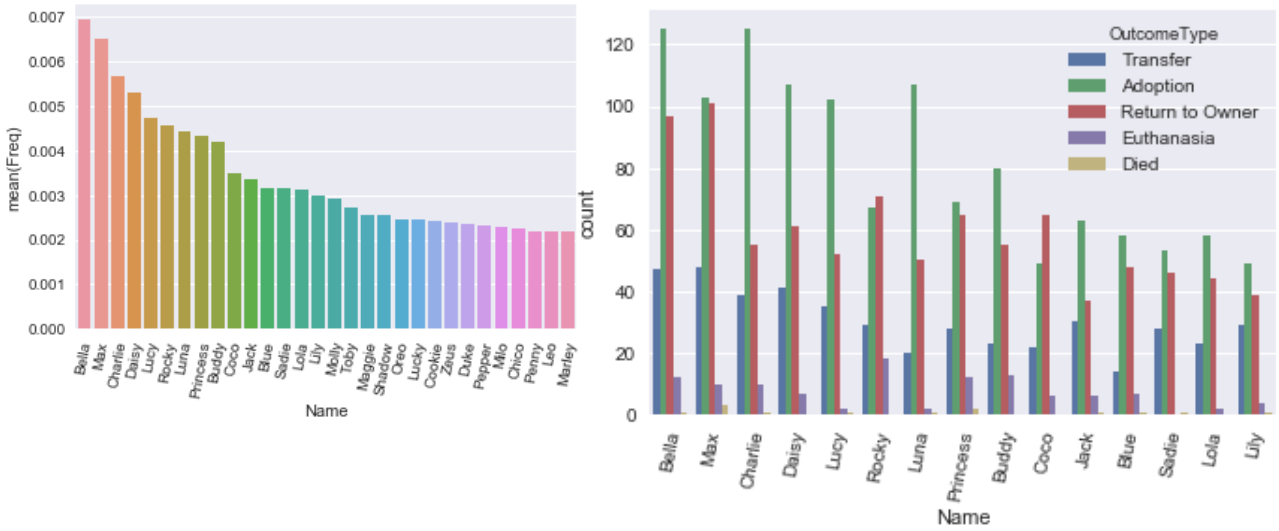
Intact animal are more likely to be adopted or transferred. Intact dog are more likely to be adopted, meanwhile intact cats have the same chance to be transferred or adopted. Cat with unknown intactness are more likely to be transferred.

- Name



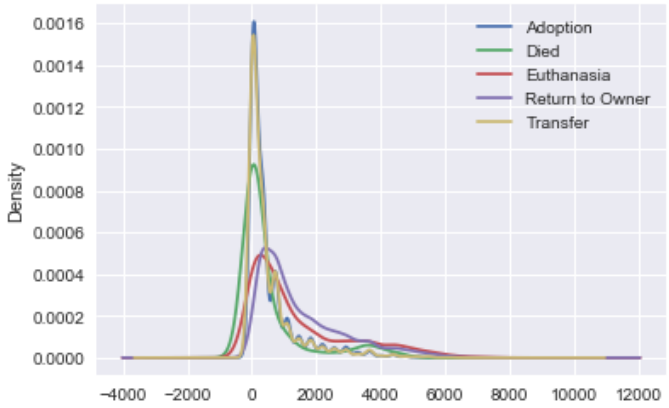
Animal with a name are more likely to be adopted. Also they are more likely to be returned to owner than animal with no name. This can be a useful feature to distinguish 'Return to Owner' type

- Name Frequency



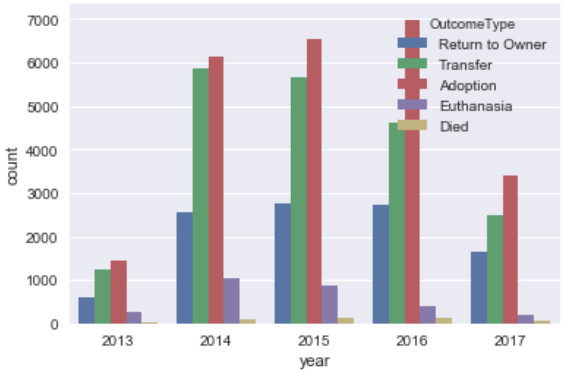
Bella seems to be the most popular name in the shelter. Animal with name 'Charlie' (3rd popular name) , Daisy (4th), Lucy (5th) or Luna (6th) seems to have a much higher chance to be adopted. Animal with names such as Max, Rocky or Princess have similar adoption and return rate, and Rocky seems to have the highest euthanasia rate. There are some patterns for outcome based on animals' names, thus this feature can be combined with other to determine the outcomes.

- Age in Days

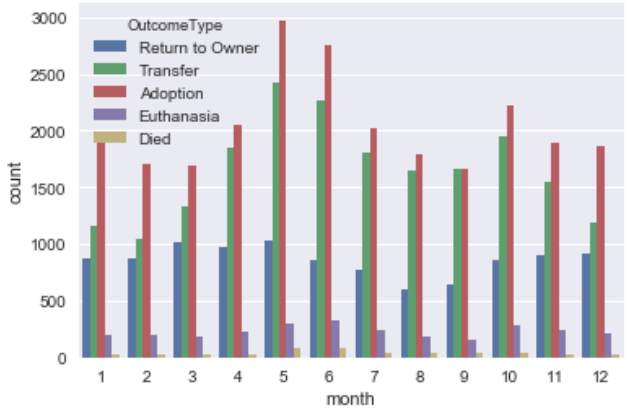


Majority of animals are less than 1000 days old (2.7 years old). Based on kernel density estimation (KDE) plot, lots of animal die young (~100 days old). The older the animal, the less likely he/she is adopted or transferred.

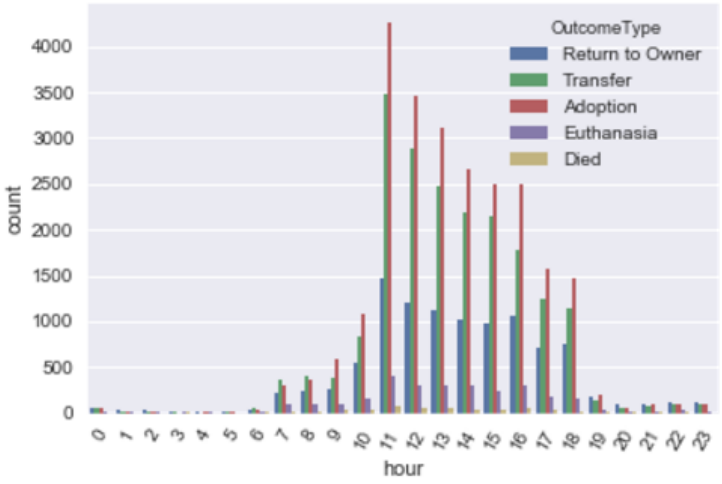
- Date Time



It's good to see number of adoption grows (about 500 animals are adopted each year between 2014 and 2016) and number of euthanasia decreases over the year. They are more likely to be adopted in this period as well.



Shelter receives lots of animal in Summer, especially in May and June, and they have higher chance to be adopted.



Most of animal enters the shelter in the afternoon, and most busiest hours are between 11 am and 3 pm. More than 8000 animals are adopted if they enter the shelter between 11 am and 1 pm.

There are some useful features, and there are some features not standing out. To see the correlation between features, Numpy's correlation coefficient matrix is constructed on the dataset after performing one-hot encoding (see Data Preprocessing) on features. There are few features that are highly correlated (with > 0.8 correlation coefficient), and some of them are: IsMix vs IsCross, AnimalType vs Domestic breed group, Domestic breed group vs Unknown hair type, etc.

Algorithms and Techniques

As mentioned above, decision trees algorithm is applied due to the nature of this dataset; some features are highly correlated, and there might be redundant attributes or irrelevant features. In order to deal with decision tree's disadvantages such as: not robust with noise, easy to overfit and low generalization accuracy, decision trees are used as weak learners to build a more robust model. This method is called Ensemble Methods. Two methods are used in this dataset are sklearn's RandomForest and XGBoost

- Random Forest is an averaging algorithms based on randomized decision trees. Each tree is built from a sample drawn with replacement from training set (bootstrap sample). Node is split based on the best split among a random subset of the features instead of all the features (without replacement). As mentioned in sklearn documentation on Random Forest, due to a result of this randomness, the bias of the forest usually slightly increases

but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.[4]

- XGBoost is an algorithm that uses Gradient Boosting method. The idea behind Gradient Boosting is when a weak learner can become better. The difference between gradient boosting and Random Forest is in gradient boosting, If a learner misclassified subset of the data, the importance of those incorrectly predicted data points will be 'boosted'. This process is run through several iteration until no further improvements can be made. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. There are some other good Gradient Boosting method such as AdaBoost, but XGBoost is chosen because it uses a more regularized model formalization to deal with overfitting. Also, XGBoost algorithm pushes the computations resources for boosted tree algorithms, thus it runs fast (faster than other gradient boosting) and reduced overall training time.

After applying Ensemble Methods, feature importance can be drawn via attribute *feature_importance_* to determine which features provide the most predictive power. This can provide more useful information about the dataset
Stratified Shuffle Split is used for cross-validating the training set due to its unbalanced nature. When splitting, a stratified fold has the same percentage of samples for each labels. For each Ensemble method, grid search cross validation is applied to tune hyper parameters.

Benchmark

Initially, Kaggle leaderboard benchmark is used to benchmark my models. However, as mentioned above, there is data leak in the outcome dataset and some users already exploit this leak to get the top scores on the leaderboard, so this leaderboard might not be accurate to use as the benchmark.

Instead, Sklearn's DummyClassifier with 'prior' strategy is used as the basic benchmark. "Prior" means the model will always predict the class that maximizes the class prior (the most frequent

[4] http://scikit-learn.org/stable/modules/ensemble.html#forest

label in the training set, in this dataset it is 'Adoption') and uses function *predict_proba* to return log probability estimates for the test vectors in order to calculate log loss metric. Using this classifier, we achieve a log score of 1.2302 on training set, and 1.1971 on test set.

As this benchmark is based on a naïve guess, RandomForest classifier and XGBoost performance will be compared to this log loss score to see if these 2 models can actually learn anything from the dataset

III. Methodology

Data Preprocessing

For all categorical features, sklearn's one-hot encoding is used to create dummy variable for each possible category of each of these features. Also, non-numeric class label 'Outcome Type' is converted to numeric class:
"Adoption"-> 0
"Died" -> 1
"Euthanasia" -> 2
"Return to Owner" -> 3
"Transfer" -> 4

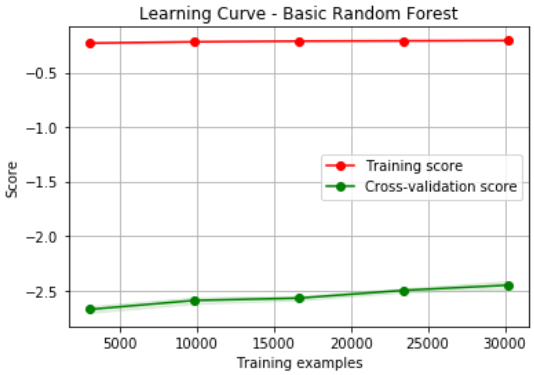
After performing one hot encoding, the dataset has 106 features. Then, dataset is split into training set (animals that enters no later than 2016-09-01) and testing set based (animals entering after 2016-09-01). Training set is then ready to be used to train and cross-validated, and log loss score will be calculated on test set.
Since we are using decision tree learning, numerical features does not have to be scaled or transformed.

Implementation

1. Random Forest

Sklearn's Random Forest is run with default setting along with cross validation + learning curve in order to see how well it performs on the training set.

Using our default Random Forest classifier with Stratified CV on the training set, we can draw the learning curve for this classifier



Default Random Forest shows sign of overfitting when training score is steadily low but cross-validation score is very high in comparison, thus create a distinctive gap indicating overfitting. Log loss score of this classifier on the test set is 2.5299, which is worse than our naïve benchmark score of 1.1971.

2. XGBoost

Similar to Random Forest, a default XGBoost is used to fit the training set. Evaluation metric is set to 'mlogloss' for log loss metric evaluation. One useful factor about XGBoost library is XGBoost model can evaluate and report on the performance on a test set during training. Thus training can be stopped once there is no improvement seen on the validation. Here is the code snippet used to train the model using default XGBoost with early stopping. Early stopping rounds is set to 10 indicating that validation error needs to decrease every 10 rounds to continue training

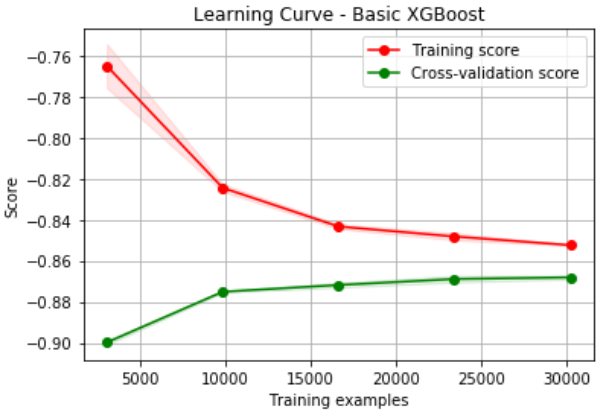
```
clf_xgb = XGBClassifier(objective='multi:softprob')

clf_xgb = clf_xgb.fit(train,y_train,eval_metric='mlogloss',\

        early_stopping_rounds=10,\

        eval_set = [(test,y_test)],verbose=True)
```

Here is the learning curve of XGBoost



XGBoost generalizes the dataset better than Random Forest as training score increases rapidly as the model sees more data (For log loss we will ignore the negative sign). Cross-validation score for this model is ~0.865. When using this model to predict the test set, log score of 0.8572 is achieved, which is better than both Random Forest and the benchmark model.

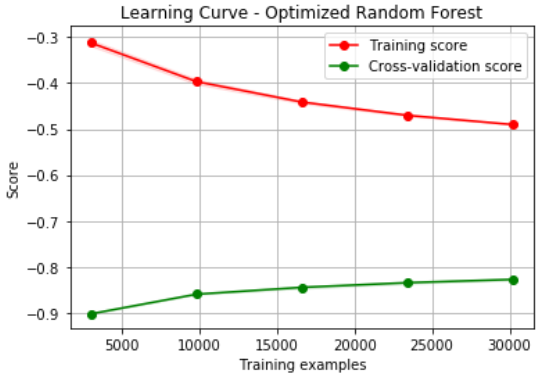
However, the learning curve graph also suggests that XGBoost might be underfitting.

Refinement

1. Random Forest

Sklearn's GridSearchCV is used to tune Random Forest hyperparameters. Three important parameters are the number of trees to build (n_estimators), tree depth (max_depth) and number of features are considered at each split (max_features). Stratified cross validation for 5 folds are used. Here is the hyper parameters space:

```
params = {
        'n_estimators': [100,500,1000],
        'max_features':['auto','log2'],
        'max_depth': [10,15,20,30]
    }
```



Grid search results in 1000 trees, 'auto' max features and max depth of 20 as best values for the model. There is a huge improvement in cross-validation score comparing to the default Random

Forest model and the model generalizes the dataset better. Log loss score on test set is 0.8523.

2. XGBoost

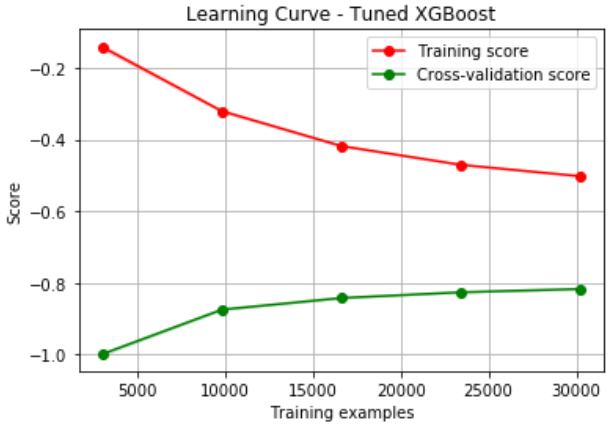
Sklearn's GridSearchCV is used to do a grid search on these XGBoost parameters:

- Learning_rate: decreasing this can lead to better log loss, but learning time will be too long.
- Max_depth: increasing max depth increases complexity of the tree. A high value of max_depth can lead to overfitting.
- N_estimators: number of trees. More trees results in better log loss score in exchange of running time
- Colsample_bytree: Denotes the fraction of columns to be randomly samples for each tree.
- Reg_lambda: L2 regularization term on weight, to control overfitting

Similar to Random Forest tuning, 5 stratified folds of the data are used for cross validation, and here is the hyper parameter space, including all the values considered for tuning.

```
params = {
        'n_estimators': [200, 400, 800, 1000],
        "learning_rate": [0.03 ,0.07 ,0.1],
        "max_depth": [2,4, 6 ,8],
        "objective":['multi:softprob'],
        "colsample_bytree": [0.4,0.6,0.8,1],
        'reg_lambda':[0.1, 0.3, 1, 3]
    }
```

This is a large hyper parameter space, thus I perform tuning on smaller hyper parameter space one at a time, each time I tune only 2 or 3 parameters in order to see the intermediate results and adjust the hyper parameter space better. Here is the learning curve for optimized XGBoost.



Grid search CV is able to pick a higher number of trees (400) and higher max depth (8), which increases the complexity of the model. Cross validation score of tuned XGB tree is just slightly better than Random Forest's (as it's closer to 0.8).

Log loss score on test set is 0.8409, which is ~0.01 lower than Random Forest's, making this XGB tree the best model for this dataset.

IV. Results

Model Evaluation and Validation

The optimized XGBoost is chosen based on its performance among other supervised models.

This model includes these hyperparameters:

colsample_bylevel=1	colsample_bytree=0.4
gamma=0,	learning_rate=0.07
max_depth=8	n_estimators=400
objective='multi:softprob'	reg_alpha=0
reg_lambda=1	subsample=1

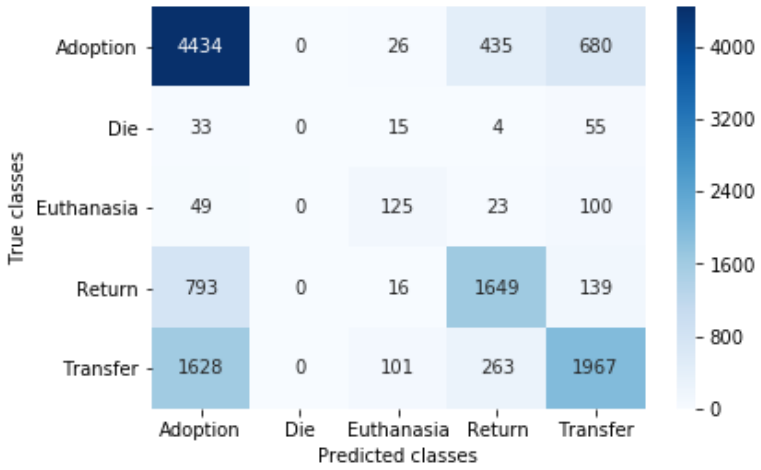
The final model has been optimized using grid search to find the best combination of hyper parameters and then cross-validated with 5 stratified CV folds, which provides an unbiased estimate of how good the model and its performance at runtime. The model is then used to predict the test set, which it has never seen before, and its log loss score on test set is close to its validation log loss score (log loss score on validation set) and is better than other models. Therefore the model generalizes well to unseen data and fits my expectation.

Justification

Log loss score on test set will be the metric to compare models. And as discussed above, Optimized XGBoost model is the one with the best score on test set.

Metric	Benchmark	Unoptimized Random Forest	Optimized Random Forest	Unoptimized XGBoost	Optimized XGBoost
Log loss on test set	1.1971	2.5299	0.8523	0.8572	**0.8409**

To analyze the model prediction on test set, confusion matrix which is generated on the test set is used.



We can take a closer look at each of the label prediction by calculating its precision and recall

- Adoption: 63.9% precision, 79.5% recall. This is high compared to other labels. The model did a good job on predicting adopted animals.

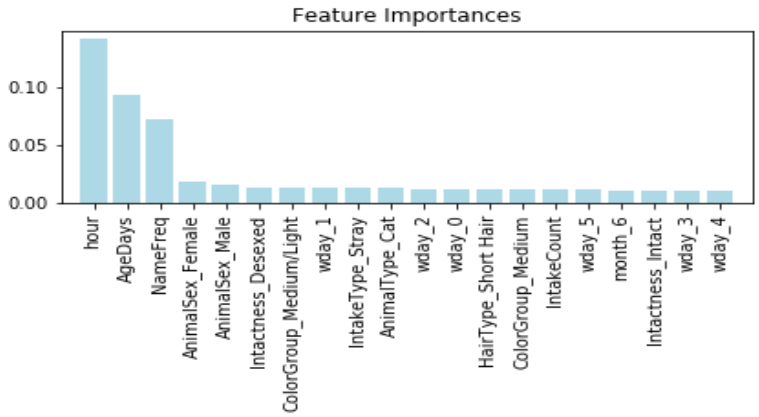
- Die: 0% precision, 0% recall. The model did the worst on predicting this label. Lots of 'Die' animals are misclassified as 'Transfer' and 'Adoption'. There are only 8% animals that die in the dataset, and this can be the reason for poor prediction on Die animal
- Euthanasia: 44.2% precision, 42% recall. The model also did poorly on this class. 34% of 'Euthanasia' animals are misclassified as 'Transfer'
- Return To Owner: 69.4% precision, 63.4% recall. This is the second best label the model predicted. Lots of 'Return to owner' animals (30%) are misclassified as 'Adoption'
- Transfer: 66.9% precision, 49.68% recall. The model has the hard time distinguishing between 'Transfer' and 'Adoption', as 41% of 'Transfer' are misclassified as 'Adoption'

In conclusion, Adoption is usually predicted accurately, followed by Return-To-Owner. However this is not a good model to predict other labels such as Euthanasia or Die.

V. Conclusion

Free-Form Visualization

Here is the first 20 features ranked on importance, from left to right.



Intake hour, name frequency, age in days, animal sex and color (medium/light) are five most important features. Surprisingly, features I thought that could make an impact such as aggressiveness, animal type or breed are less important.

As mentioned in Confusion matrix of the optimized XGB, even though the model does not solve the problems defined in Problem Statement section, it performs best on predicting Adoption, which aligns with the goal of this project: to predict their outcomes in order to determine which

animal has low adoption probability so shelter employees can focus on them and give them extra help finding their new home. Employees can pay more attention to their age, their name or their sex, not as definitive measures but as helpful characteristics to assist the animals

Reflection

Here is the project outline

- Research Kaggle animal competition and analyze the data leak feature problems
- Obtain datasets from Austin animal shelters website and perform dataset merging
- Feature engineering to break down existing features to more useful ones
- Exploratory Data Analysis (EDA)
- Analyze two appropriate supervised models for the dataset based on EDA
- Train the models
- Hyper parameter tuning and pick the best model
- Evaluate and validate model on test set.

Some interesting aspects of the project are: doing feature engineering and see how each features affect the outcome during EDA. Unfortunately, some of the features which take lots of time to generate, such as breed and color, are not important in predicting the outcomes. Some features that are thought to be irrelevant, such as intake hour and name frequency, are important.

Besides feature engineering, the most difficult aspect of this projects is tuning XGBoost to get the best result. Initially a randomized search is performed on large hyper parameters but the model is overfitting and log loss score on test set is just a little better than the default XGBoost. The best model is then built with few parameters tuned one at a time using grid search, iteratively. This task is very time-consuming, thus I have to run it on Amazon Web Service EC2 Instance which is more powerful. This process helps me understand each of XGBoost parameters and how they impact the overall performance, which is a useful practice that I can apply to other Kaggle competitions I am planning to take. Also I learn how to run it on AWS which will come in handy for my future projects.

The final model is helpful to predict Adoption label, but it did poorly on predicting other labels, therefore it can only be used in a limited setting. However, it fits my expectation: to help animals who have a hard time to get adopted. More on this is discussed in Improvement

Improvement

Our final model still shows sign of overfitting, and log loss score seems to be above 0.8 even with multiple tries on hyper parameter tuning and regularization. To achieve a better performance, one solution is to get more data. We can contact the shelter to get data prior to 2013 and train this data to see whether it helps overcome the overfitting problem.

Since this model is limited to Austin shelters, it would be best to obtain data from other cities near Austin to test its performance. We can apply transfer learning to build up and transfer knowledge of the dataset to new dataset, which can save training time and improve the model.

Another way to improve the model is to change the problem's definition. The model did well on predicting adoption (and perhaps return-to-owner), so we can limit the labels it needs to predict. After all, the most important goal of any animal shelter is to provide the best outcome to the animals, which is getting adopted. By limiting the labels to three (Adoption, Return and Neither), or making it a binary classification (Adopted and Not Adopted), we can focus on predicting the important label(s) and this could make the model improve its performance.