# Machine Learning Hackathon

*Animal shelter outcome*

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

## Team Members

Sravanti Nomula (MT2018524)

Sarvesh Nandkar (MT2018519)

## Problem statement

To predict the outcome of the animal as they leave the Animal center.

These outcomes include:

- Adoption
- Died
- Euthanasia
- Return to owner
- Transfer

## ABSTRACT

We considered the problem of predicting the outcome of an animal from shelter. We explored relationship between various features and validated the effectiveness of the features on different classifier models including Logistic Regression, Naïve Bayes, Random Forest, XGBoost and LightGBM. We also discovered some interesting phenomenon of the dataset.

## INTRODUCTION

In the Kaggle competition, the goal was to use a dataset on shelter animals to do two things: gain insights that can potentially improve their outcome, and to develop a classification model which predicts the outcome of animals.

We believe that the shelter would use the model when a new animal arrives. This is the time when the shelter wants to know the likely outcome of the new animal. A classifier that predicts the outcome of animals at the time of their intake can help in several ways.

With this information, shelters all around the world can now determine the best course of action to maximize their animal adoptions.

## DATA EXPLORATION AND PREPROCESSING

Time period of the dataset covers is from year 2013 to 2016 with 26729 data points in total. The features are:

- Datetime
- Outcome Type
- Outcome Sub Type
- Animal Type
- Sex upon Outcome
- Age upon Outcome
- Color
- Breed

Our initial look at the data reveals 26,729 rows. *Name* is not included for 7,000 cases, but is included for 71.2% of the data. A large quantity of *OutcomeSubtype* data is missing, which we worked to quantify below. Aside from these two columns, we had at least 99.9% of the data in every other column.

| | Column Count | % Percent of Total Data |
|---|---|---|
| AnimalID | 26729 | 100.0 |
| Name | 19038 | 71.2 |
| OutcomeType | 26729 | 100.0 |
| OutcomeSubtype | 13117 | 49.1 |
| AnimalType | 26729 | 100.0 |
| SexuponOutcome | 26728 | 100.0 |
| AgeuponOutcome | 26711 | 99.9 |
| Breed | 26729 | 100.0 |
| Color | 26729 | 100.0 |
| Year | 26729 | 100.0 |
| Month | 26729 | 100.0 |
| Date | 26729 | 100.0 |
| Minute | 26729 | 100.0 |
| Hour | 26729 | 100.0 |

Table:1

## Name

We feel that this level of sparsity within *Name* will not help us predict animal outcomes. This makes logical sense as well, as *Name* seems unlikely to be the deciding factor in whether an animal is adopted. Animal owners regularly change animals' names following adoption as well.

So, we decided to name the unnamed animals as TIMON and imputed the same into the train data.

### OutcomeType and OutcomeSubtype

In fig1, we see a large imbalance between the outcome classes. The outcomes lean heavily toward Adoption, Transfer, and Return to Owner. There is a smaller, but notable, portion of Euthanasia outcomes, and a very small number of Died outcomes.

Partner, which we see below is within the Transfer outcome, is the most common subtype; this presumably means that the animal was transferred to a partner organization or shelter.

While the Outcome Subtype variable is also missing for many animals, we do not need to predict based on this variable, so this does not pose a problem for feature evaluation. We cannot use the outcome subtype as a feature, as it would bias the dataset. However, we do want to investigate outcome subtype more fully in an effort to understand the data.

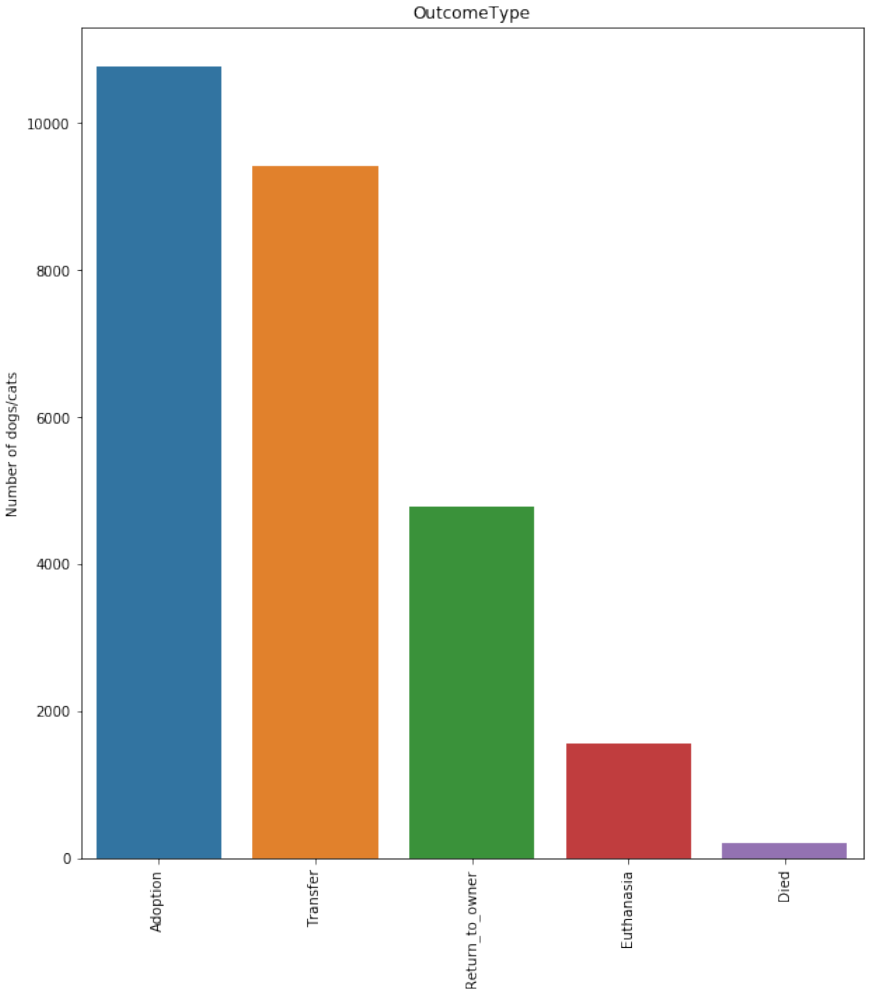
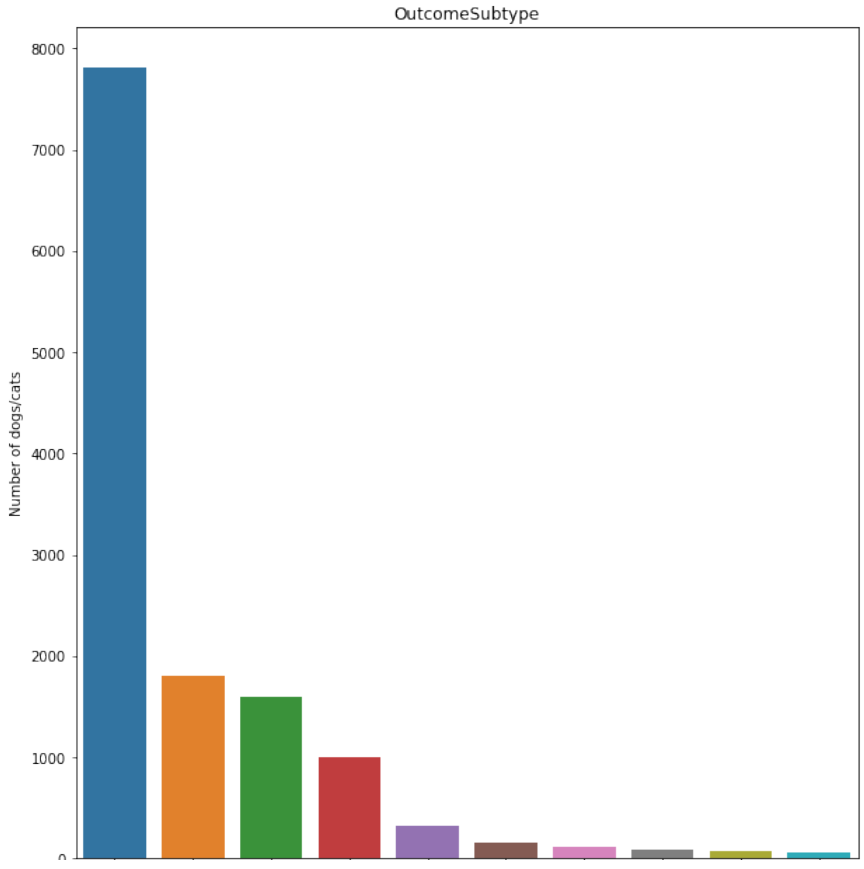

OutcomeType
Number of dogs/cats
10000
8000
6000
4000
2000
0
Adoption
Transfer
Return_to_owner
Euthanasia
Died

Fig1


OutcomeSubtype
Number of dogs/cats
8000
7000
6000
5000
4000
3000
2000
1000
0

## Exploring OutcomeSubtype further,

We note a large quantity of null values in the Adoption outcome. The only subtypes labelled in this category are Foster, Offsite, and in one case, Barn. We feel an important next step is to dig further into the null data.
Key Outcome Subtype definitions are:

- SCRP = Stray Cat Return Program, a catch-and-release program to spay/neuter stray cats
- Barn = Barn Cat Program, which places cats from euthanasia lists that "cannot be adopted as traditional pets" into safe, appropriate adoption environments, such as a barn, stable, garage, or warehouse. They have access to shelter, food, and water.

| | OutcomeType | OutcomeSubtype | Count |
|---|---|---|---|
| 0 | Adoption | Barn | 1 |
| 1 | Adoption | Foster | 1800 |
| 2 | Adoption | Offsite | 165 |
| 3 | Died | At Vet | 4 |
| 4 | Died | Enroute | 8 |
| 5 | Died | In Foster | 52 |
| 6 | Died | In Kennel | 114 |
| 7 | Died | In Surgery | 3 |
| 8 | Euthanasia | Aggressive | 320 |
| 9 | Euthanasia | Behavior | 86 |
| 10 | Euthanasia | Court/Investigation | 6 |
| 11 | Euthanasia | Medical | 66 |
| 12 | Euthanasia | Rabies Risk | 74 |
| 13 | Euthanasia | Suffering | 1002 |
| 14 | Transfer | Barn | 1 |
| 15 | Transfer | Partner | 7816 |
| 16 | Transfer | SCRP | 1599 |

Table2

Table3

| | OutcomeType | Count |
|---|---|---|
| 0 | Adoption | 8803 |
| 1 | Died | 16 |
| 2 | Euthanasia | 1 |
| 3 | Return_to_owner | 4786 |
| 4 | Transfer | 6 |

- 8,803 of the Adoption outcome entries have the *outcome subtype* as null. As we already have *outcome subtypes* for barn, foster, and offsite within adoption, we feel confident in assuming that these missing subtypes are simply the animals that were adopted. We will assume this is true and fill these null values with ***adopted***.

- As the return-to-owner outcome type also has no subtype, we will make the outcome subtype ***return to owner***. It seems that, within the dataset, when the outcome subtype would have been redundant with the outcome type, the outcome subtype was not always listed. Logically, once an animal is adopted or returned to their owner, a shelter feels it does not need further classification or follow up on specific outcomes.

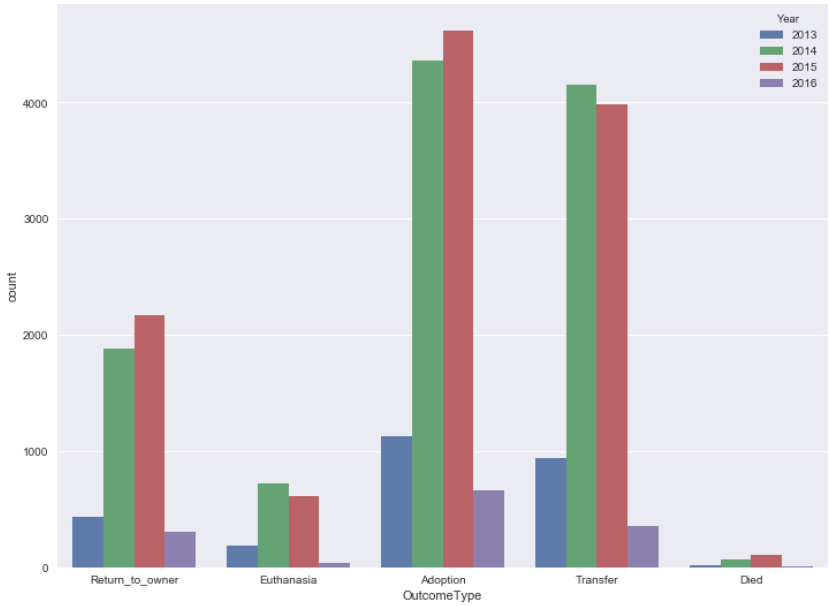
- For the remaining 23 null outcome subtypes, we will label their subtype as ***other.***

## DateTime Variable

We split the date time variable into different variables i.e year, month, date, hour and minute.

### Year

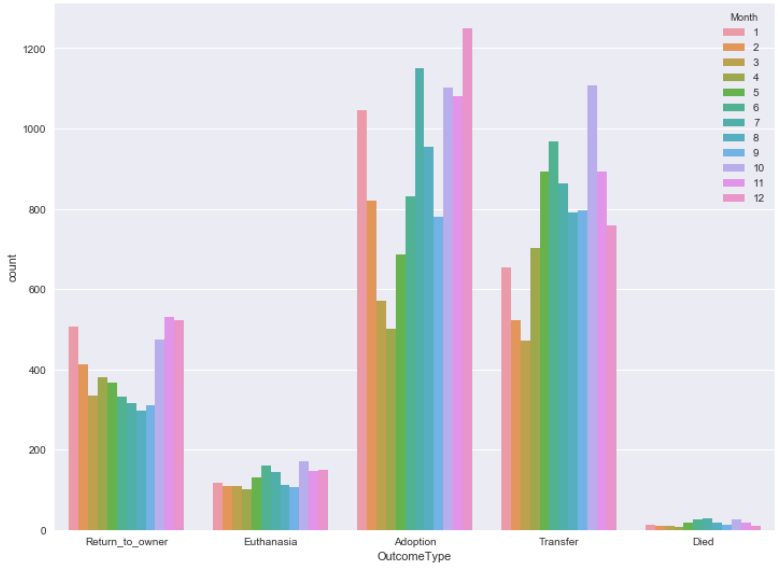
Fig 2 shows the outcome of all categories increased in 2014 and 2015 with drastic increase in "*Adoption*" and "*Transfer*"




Fig

### Month

We can observe that adoptions in the months of July, October, November and December are much more when compared to adoptions in the other months.

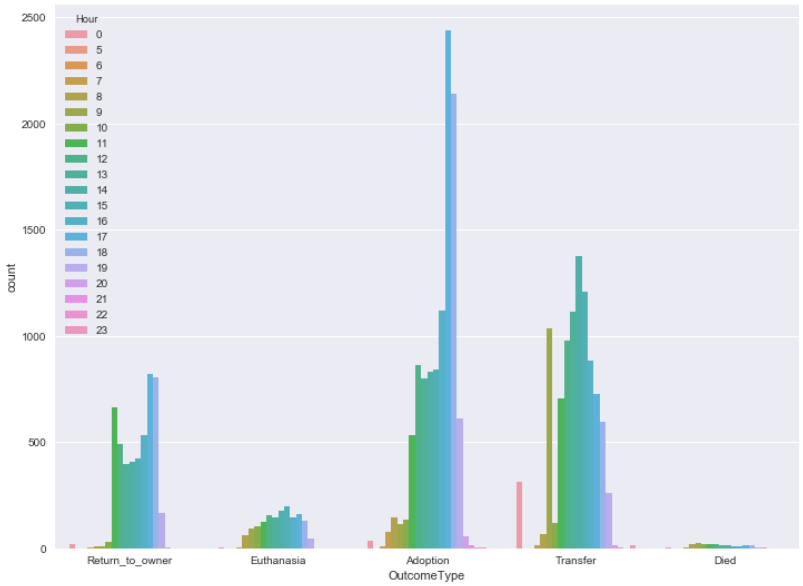



Fig

### Hour

We can observe that no dataset is available from 12 am to 5 am which may mean that the shelters are closed during this period.

There is an increase in *Transfer* of animal between the hours 10 AM to 3 PM while it decreases after 3:00 PM. *Adoption outcome* is at the peak from 5 PM to 8 PM. *Return to owner* decreases from 8:00 AM to 1:00 PM and then increases.

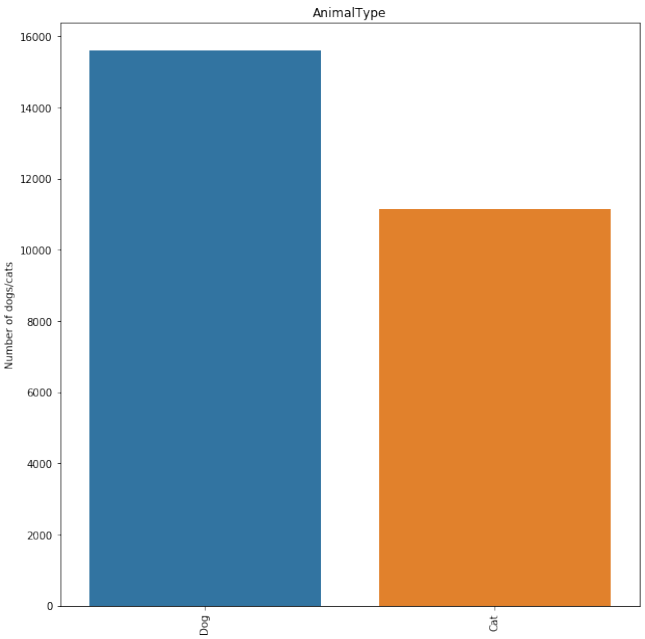



Fig

**Other Features**

**Animal Type**

A breakdown of Animal Type shows that there are only dogs and cats in the dataset, with dogs accounting for about 60% of cases.

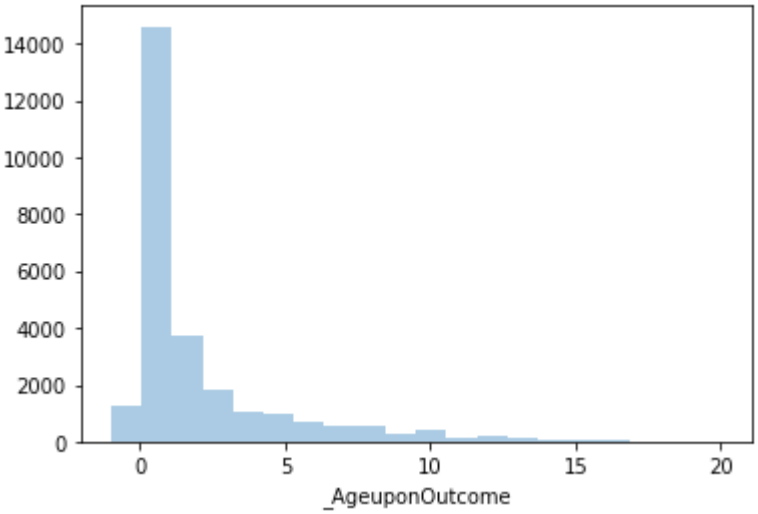



Fig

**Age upon Outcome**

We believed that it is important to make the ages variable continuous, as opposed to categorical, as the relationship of age is continuous in nature. The most logical unit to standardize ages is weeks.

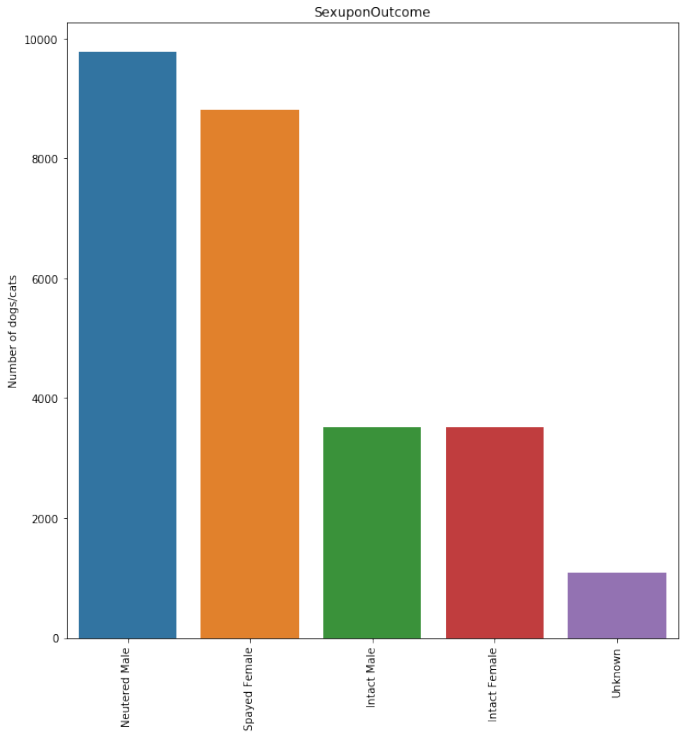
Category of animals was further divided into three categories young, young adult, adult and old. We classified “Baby” in range (0 to 3),”Young” in range (3 to 5),”Adult” in range (6 to 10) and rest ages belongs to old.




Fig

**Sex upon Outcome**

Sex Upon Outcome shows us whether the animal is male or female and whether they were spayed or neutered (fixed) prior to the outcome. There is a roughly even split between male and female animals, an overwhelming majority of which were spayed or neutered upon outcome. There are also a small, but notable, quantity of unknowns.




Fig

**Breed**

There were a total of 1380 types of breeds. Although the breed of an animal may play a vital role in deciding its outcome to decrease the complexity, we made the differentiation by changing the feature into pure breed or mix breed.

## Feature Extraction

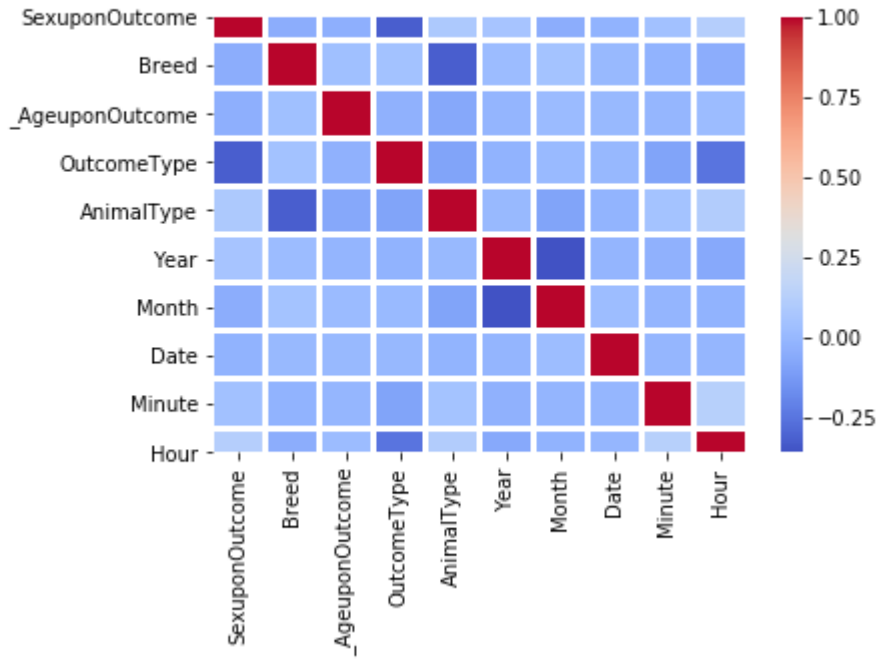
Correlation Matrix

| | SexuponOutcome | Breed | _AgeuponOutcome | OutcomeType | AnimalType | Year | Month | Da |
|---|---|---|---|---|---|---|---|---|
| SexuponOutcome | 1.000000 | -0.043571 | -0.035781 | -0.316375 | 0.092184 | 0.064750 | -0.040418 | -0. |
| Breed | -0.043571 | 1.000000 | 0.034226 | 0.048857 | -0.319317 | 0.022467 | 0.050743 | 0.0 |
| _AgeuponOutcome | -0.035781 | 0.034226 | 1.000000 | -0.025823 | -0.070797 | -0.012944 | 0.012951 | 0.0 |
| OutcomeType | -0.316375 | 0.048857 | -0.025823 | 1.000000 | -0.086064 | -0.020392 | 0.007657 | -0. |
| AnimalType | 0.092184 | -0.319317 | -0.070797 | -0.086064 | 1.000000 | 0.007345 | -0.086570 | -0. |
| Year | 0.064750 | 0.022467 | -0.012944 | -0.020392 | 0.007345 | 1.000000 | -0.358674 | -0. |
| Month | -0.040418 | 0.050743 | 0.012951 | 0.007657 | -0.086570 | -0.358674 | 1.000000 | 0.0 |
| Date | -0.019887 | 0.004810 | 0.005532 | -0.002716 | -0.016706 | -0.011404 | 0.027120 | 1.0 |
| Minute | 0.042274 | -0.019058 | -0.005235 | -0.084680 | 0.052667 | -0.029248 | -0.011629 | -0. |
| Hour | 0.121099 | -0.040409 | 0.019601 | -0.252149 | 0.109835 | -0.065426 | -0.022732 | -0. |

Heat Map

```
sns.heatmap(corre,linewidths=2, cmap="coolwarm")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f47c5e26f28>
```



**Label Encoding**

Many column values had repeating entries, e.g. Breed- Labrador, German Shepard, etc. Dataframe can't handle such values and such values can't be worked upon by algorithms. So, we encode such entries with integer type values which are easier to work upon and can be input to algorithms.

Label encoding has been applied to 4 categories which we previously worked upon:

(a.) AnimalType has 2 categories: Dog and Cat which has been labeled as 0 and 1 respectively.
(b.) AgeUponOutcome has 4 categories: Baby, Young, Adult & Old which has been encoded as 0, 1, 2 & 3 respectively.
(c.) SexUponOutcome has 4 categories: Intact Male, Intact Female, Spayed Female, Neutered Male which have been encoded 0, 1, 2 & 3 respectively
(d.) Breed has 2 categories: Purebred and Mix-bred which are encoded 0 & 1.

| | SexuponOutcome | Breed | _AgeuponOutcome | AnimalType | Year | Month | Date | Minute | Hour |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 1 | 2 | 1 | 2014 | 0 | 12 | 22 | 18 |
| **1** | 3 | 1 | 2 | 0 | 2013 | 2 | 13 | 44 | 12 |
| **2** | 2 | 1 | 2 | 1 | 2015 | 0 | 31 | 28 | 12 |
| **3** | 1 | 1 | 2 | 0 | 2014 | 1 | 11 | 9 | 19 |
| **4** | 2 | 0 | 2 | 1 | 2013 | 2 | 15 | 52 | 12 |

**Column Dropping**

Animal Id, Name, Day, Seconds, Outcome Subtype, Color have been dropped since these show low correlation with OutcomeType while some of them are not even present in the test dataset.

**Model Decision**

We used 5 models for training our dataset- Logistic Regression, XGBoost, LGBM, RandomForest and Naïve Bayes.

**Logistic Regression model** is used to model the probability of a certain class or event. It uses sigmoid function to build the regression model.

Advantage is that it is pretty fast and the output can be interpreted as a probability.

Disadvantage is that it is only fit for linear features and tends to perform poorly when features don't have linear relation with the label.

**Random Forests model** trains each tree independently, using a random sample of data. This randomness helps to make the model more robust than a single decision tree. It generally doesn't overfits the data in training model so the results are fairly accurate.

**XGBoost model** build trees one at a time, where each new tree helps to correct errors made by previously trained tree. There are typically three parameters - number of trees, depth of trees and learning rate which can be changed as per the requirement.

These models are better learners than Random forest models but require significantly more time to train owing to the fact that the trees are built sequentially.

**Naïve Bayes model** is a simple probabilistic classifier based on the assumption that features are conditionally independent given the label. It is simple and converge very fast. On the contrary they perform badly when features are not independent of each other.

**LightGBM** is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with faster training speed and higher efficiency along with lower memory usage and good accuracy.

## Train -Test Split and Evaluation Criteria:

To evaluate performance of our prediction models, we split the data into two parts. The first part accounts for 67% of the whole dataset and was used as the training set. The second part accounts for 33% of the whole dataset and was used as the validation set. Prediction results are then evaluated using the multi-class logarithmic loss as defined in the following formula:

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

where N is the number of cases in the validation set, M is the number of class labels, log is the nature logarithm, yij is 1 if observation i is in class j and 0 otherwise, and pij is the predicted probability that observation i belongs to class j.

**Model Optimization and predictions:**

Following features are worked upon in these models:

- SexUponOutcome
- Breed
- _AgeuponOutcome
- AnimalType
- Year
- Month
- Date
- Minute
- Hour

a. Logistic Regression model:  following are the preferences

## Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
classifier = LogisticRegression(penalty='l2',random_state = 0,class_weight='balanced'
                                solver='lbfgs',n_jobs=-1)
selector = RFE(classifier, 5, step=1)
selector = selector.fit(X, y)
selector.support_
selector.ranking_
```

```
array([1, 1, 1, 1, 5, 2, 3, 4, 1])
```

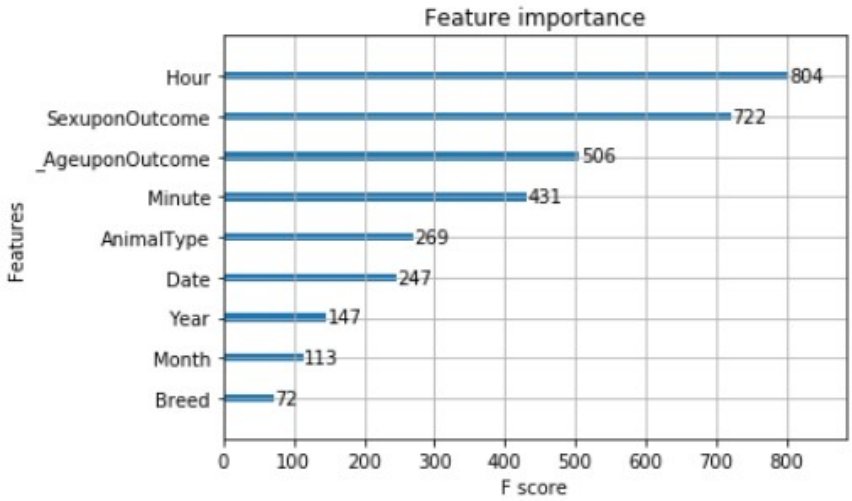
b. XGB model: following are the preferences

## XGBOOST

```
from xgboost import XGBClassifier
import xgboost as xgb
from sklearn.feature_selection import RFE
model = xgb.XGBClassifier(objective='multi:softprob')
selector = RFE(model, 5, step=1)
selector = selector.fit(X, y)
selector.support_
selector.ranking_
```

```
array([1, 4, 1, 1, 2, 3, 5, 1, 1])
```

Importance of features for XGB




c. LightGBM model: Following are the preferences

## LightGBM

```
from lightgbm import LGBMClassifier
from sklearn.feature_selection import RFE
lgbm = LGBMClassifier(objective='multiclass', random_state=5)
selector = RFE(lgbm, 5, step=1)
selector=selector.fit(X, y)
selector.support_
selector.ranking_
```

```
array([1, 5, 2, 4, 1, 3, 1, 1, 1])
```

d. RandomForests Model: Following are the preferences

## Random Forest

```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=4,n_estimators =20,criterion = 'entropy',
selector = RFE(clf, 5, step=1)
selector = selector.fit(X, y)
selector.support_
selector.ranking_
```

```
array([1, 4, 1, 1, 2, 3, 5, 1, 1])
```

The above discussed models have been applied on test dataset and following are the log loss scores obtained:

| MODEL | LogLoss Score |
|---|---|
| Logistic Regression | 1.4594995225944294 |
| Random Forests | 0.8705385613374071 |
| XGBoost | 0.8691580911528276 |
| LightGBM | 0.8537600402307091 |
| Naive Bayes | 1.0993156117806615 |

# CONCLUSION

All the variables we analysed in the dataset exploration section are effective features in classifying Outcome.

The most important feature that came out is Hours, followed by the" SexUponOutcome" in XGboost, LightGBM, Naïve Bayes, Logistic Regression models and Random Forest.

In terms of speed Naïve Bayes is the fastest,obviously due to its simplicity, it only needed less than 2 seconds to converge. Random Forests is rather efficient due to its high accuracy. It took ~5sec to converge.

Tuning the parameter has played a major role in some of the models like logistic regression and random forest and has shown some improvement in terms of log loss.

XGBoost and LGBM models took their time but performed very well with the data gaining minimal logloss values.

Logistic Regression did not seem to be much helpful in this scene probably due to non-linearity in dataset and performed the worst.

Our best score is 0.85376 on LightGBM Model