|  |  |
| --- | --- |
| Machine Learning Engineer Nanodegree | Rafael Torres |
| **Capstone Project** | January XX, 2017 |

# Definition

## Project Overview

Animal shelters across the United States end up getting receiving a total of 7-to-8 million new animals each year[[1]](#footnote-1). These shelters are often able to find new homes for their animals; however, about 35% of them end up being euthanized, as these shelters are not able to find new caregivers for them. Utilizing historical data, and Machine Learning (ML) techniques, we could predict animal outcomes based on certain features, which could help these shelters to refocus their budgets and efforts to help the most needed segments of their animal population to find new homes.

This project was inspired by the [Shelter Animal Outcomes project](https://www.kaggle.com/c/shelter-animal-outcomes) featured in Kaggle.

## Problem Statement

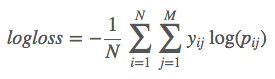
In the US, more than 2.5 million animals per year end up being euthanized across animal shelters, because they are not able to find new homes for those animals. The [Austin Animal Center](http://www.austintexas.gov/department/aac) has made publicly available historical information pertaining dogs and cats they have handled over almost the past 3 years.

The goal of this project is to define a classification model based on data collected by the shelter, which could help the shelter better predict the outcome of their animals. Such information can in turn help them identify segments of their animal population that need extra help in finding new homes.

## Metrics

Kaggle’s evaluation engine is leverage for this project. The platform uses Logarithmic Loss (aka Log Loss) to evaluate the model’s performance. It is asked by the evaluation engine to submit predictions as a probabilistic distribution of the possible outcomes. Given such conditions, Log Loss is a natural metric to be used in this case.

Log Loss is defined as “the logarithm of the likelihood function for a Bernoulli random distribution.”[[2]](#footnote-2) The function is defined by:



*Where:*

* *N = number of examples*
* *M = number of classes*
* *yij = binary variable indicating whether class j was correct for sample i*

In plain terms, Log Loss works by penalizing wrong predictions, and the penalization is even more severe for more “wrongly” confident predictions (i.e., the more the predicted probability diverges from the actual value). All of this means that the lowest (closest to zero) the Log Loss score for a model the better.

# Analysis

## Data Exploration

The data utilized in this project is available here:

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

There are 2 main files included in that link:

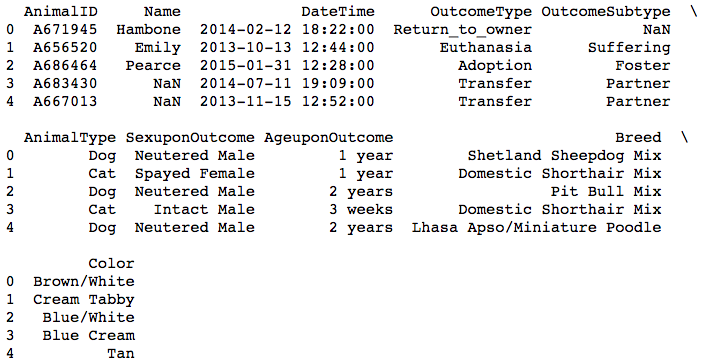
* train.csv
* test.csv

### The train.csv file

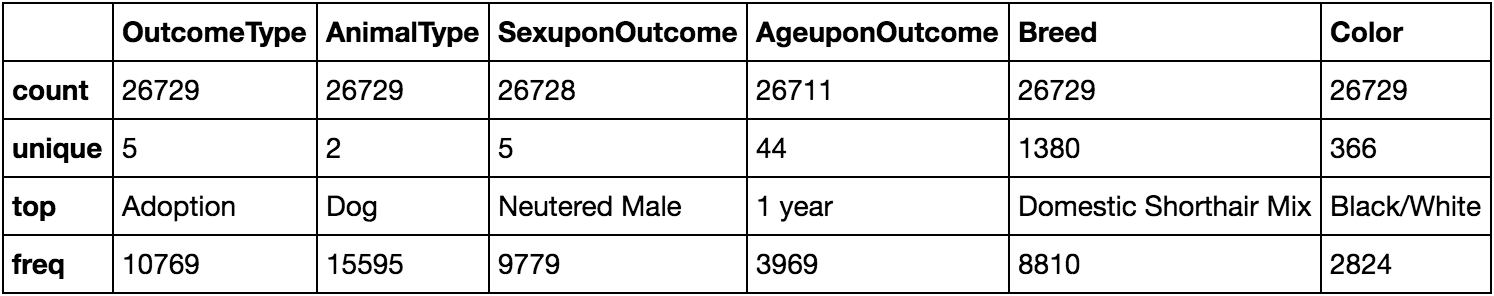
This file contains the dataset used for (a) training the ML algorithms, and for (b) testing the algorithm locally. The data set size is 26,729 x 10 (rows x cols). The columns are labeled as such:

* AnimalID
* Name
* DateTime
* OutcomeType
* OutcomeSubtype
* AnimalType
* SexuponOutcome
* AgeuponOutcome
* Breed
* Color

Here’s a summary of the “head” section of the dataset, as generated by Pandas:



And here’s a broad description of the dataset:



The data represents the shelter’s Dog and Cat population, with their corresponding Outcome, as tracked from October 2012 through March 2016.

The target label for this project is OutcomeType, since we’re mostly interested in minimizing the number of animals that are euthanized, or die in the shelter. For the purposes of this study, the OutcomeSubtype is irrelevant. Similarly, AnimalID, Name and DateTime won’t give us any insight into trying to predict an animal’s outcome – these are data points useful for internal tracking purposes. For that reason, we’ll be ignoring those as well.

In short, the features and target for our ML algorithms are:

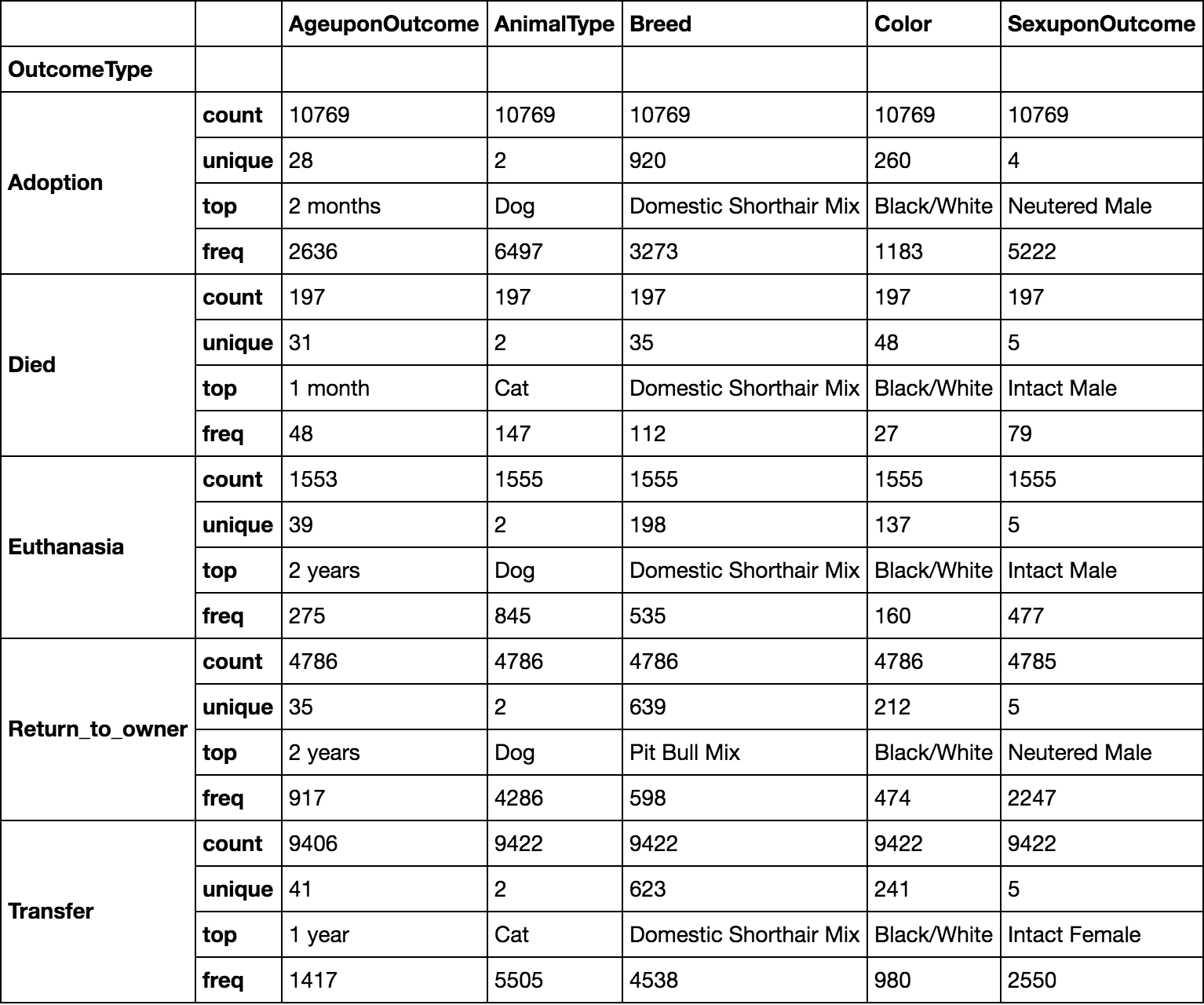
* Features: AnimalType, SexuponOutcome, AgeuponOutcome, Breed, Color.
* Target: OutcomeType.

Upon exploring the data, it is quickly observed that most of the features have a very high variety of “categorical” values. For example, Breed has values like Abyssinian Mix, Affenpinscher Mix, and many other ones. Here’s a summary of the variation of unique values (the number of unique value) shown per feature:

* AnimalType: 2
* SexuponOutcome: 5
* AgeuponOutcome: 176
* Breed: 5,520
* Color: 1,464

The train dataset is made up of approximately 58% Dogs and 42% Cats. About 6.5% of the population in the dataset was recorded as euthanized or died, which amounts to 1,752 animals.

The following table shows the group-by OutcomeType breakdown of the data:



It’s interesting to note that the majority of the animals euthanized were dogs. Also, the age of the population most euthanized was 2 years old. On the other hand, it appears that the category of animals mostly recorded as died were cats, while the age is usually 1 month.

As for the target, the value that we are interested in predicting, fortunately only 5 unique values are part of it:

Adoption, Died, Euthanasia, Return\_to\_owner, Transfer.

The large variation of values for the data features represents a challenge. Moreover, the amount and value of unique variables between the *train.csv* and the test*.csv* files differs, which makes things a bit more challenging. The technique to deal with these factors is explained in detail in the Data Preprocessing section of this document.

### The test.csv file

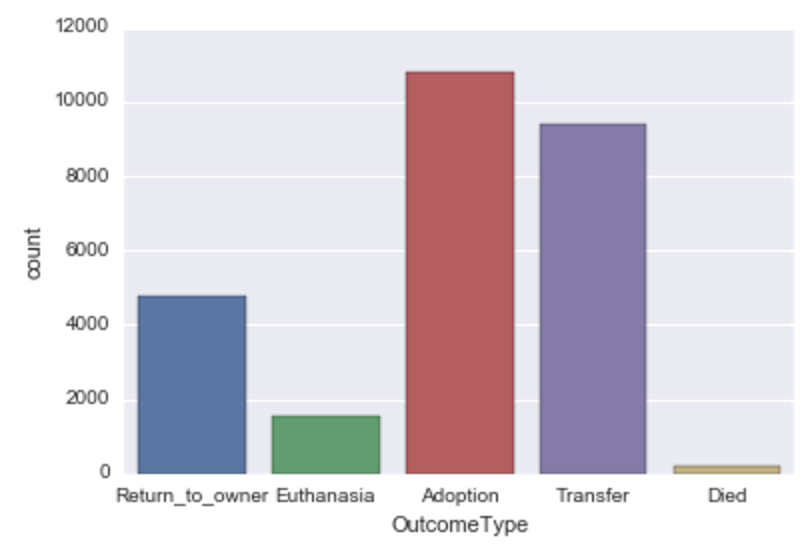
This file contains the dataset used to make the predictions for a formal submission to the [Kaggle evaluation engine](https://www.kaggle.com/c/shelter-animal-outcomes/data). The dataset size is 11,456 x 8 (rows x cols). The columns are labeled as such:

* ID
* Name
* DateTime
* AnimalType
* SexuponOutcome
* AgeuponOutcome
* Breed
* Color

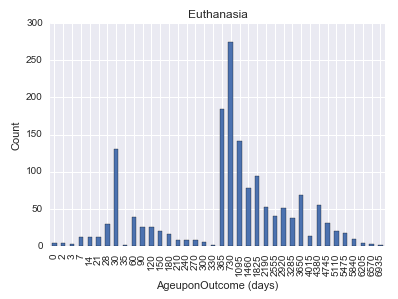
Similarly, with this dataset, we’ll only take into account the same column labels as the one considered for the train dataset.

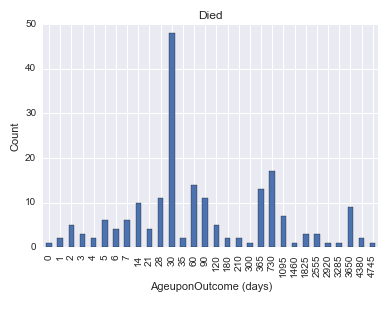
## Exploratory Visualization

The following chart shows the difference between the OutcomeType counts in the dataset. This shows that most of the animals are (fortunately) adopted. On the other hand, Died and Euthanasia are the lowest, but the latter still has a representative level.



Further dissecting the data, while focusing on the least desirable outcomes (Euthanasia and Died), the correlation of Age (in days) and the count of these outcomes shows the following:





As it can be observed in the first bar graph (Euthanasia), the peak of events happens at around the 2-year mark of the animal’s age (730 days) and at the 1-year mark (365 days). The count is still high at the 3-year mark (1095 days), after which point it starts to become lower. Another interesting time mark is at 30 days, in which the count is about 125, almost at the same level as at the 3-year mark. This seems to challenge the intuition that euthanasia would only happen on older animal population. It would be interesting to find out what’s causing such event at an early age.

As for the second bar graph (Died) the peak count happens at 30 days (which concurs with the finding made on the first graph at the same animal age). Similarly, the Died outcome count also goes up around the 2- and 3-year.

This suggest that the first-month mark (30 days) is an important event for the shelter’s animal, as it can has significance on determining possible non-desirable outcomes for them. The same implication could be done towards animals reaching an age of 2-3 years.

## Algorithms and Techniques

The analysis made in this project relies on python/sklearn. Broadly speaking, the approach utilized in this analysis can be described in this fashion:

1. Data exploration: to better understand the data.
2. Data processing: to normalize the data, as many features were originally not quantifiable. Also a split of the train data was obtained (at 80% train vs 50% test data split) for the models’ training/testing, using sklearn.model\_selection.train\_test\_split.
3. ML algorithm training: see below for the list of algorithms selected.
4. ML testing: using sklearn’s metric, the log loss for each model was calculated. Additionally, submissions to Kaggle’s evaluation engine were made, in order to compare models’ outcomes.
5. Model tuning: from the previous model comparison, XGBoost was picked as the most promising. Further parameter-tuning was done to this model, using a manual approach of the Grid Search technique.[[3]](#footnote-3)
6. Re-calculation of log loss: to measure the delta of the tuned model and determine whether any improvement was attained.

Since the end goal is to predict the OutcomeType of the animals, the analysis was focused on different classification algorithms. Three were selected to be used in this project:

* Decision Trees: Due to the simplicity of this algorithm, it is considered a good starting point to obtain some base-line predictions.
* SVM: Based on personal experience, SVM have always delivered a higher level of accuracy, albeit at much slower pace due to the higher complexity of the algorithm. For this reason, this was one of the algorithms chosen. At the end, better performance was obtained with SVM, but its slowness made it practically impossible to fine tune.
* XGBoost: In an attempt to combat the inefficiency of SVMs, with the hope to not only get faster results, but also higher performance, it was decided to give XGBoost a try. This a gradient boosting algorithm that uses advanced optimization techniques to improve efficiency dramatically[[4]](#footnote-4). It turned out that at the end both the accuracy and efficiency of this algorithm were far above the previous 2 for our dataset.

## Benchmark

Since this project is inspired by the [Shelter Animal Outcomes](https://www.kaggle.com/c/shelter-animal-outcomes) competition in Kaggle, originally [its public leaderboard](https://www.kaggle.com/c/shelter-animal-outcomes/leaderboard) was used to get an idea of the realistic benchmark numbers. However, as it turns out there’s a bug in the competition that, when exploited, it allows submitters to get near perfect (or even perfect) Log Loss scores[[5]](#footnote-5).

With that in mind, looking at submissions the numbers seem to vary from all the way up at >34 points to near 0 points. The sentiment by browsing through the [competition’s forums](https://www.kaggle.com/c/shelter-animal-outcomes/forums) is that scores below 1.0 are attainable without exploiting the bug.

Our benchmark for this project is to get below the 1.0 score. At first a base-line score is established by using a Decision Tree classifier, and consequently look for improvements by using the other 2 models along with tuning techniques.

# Methodology

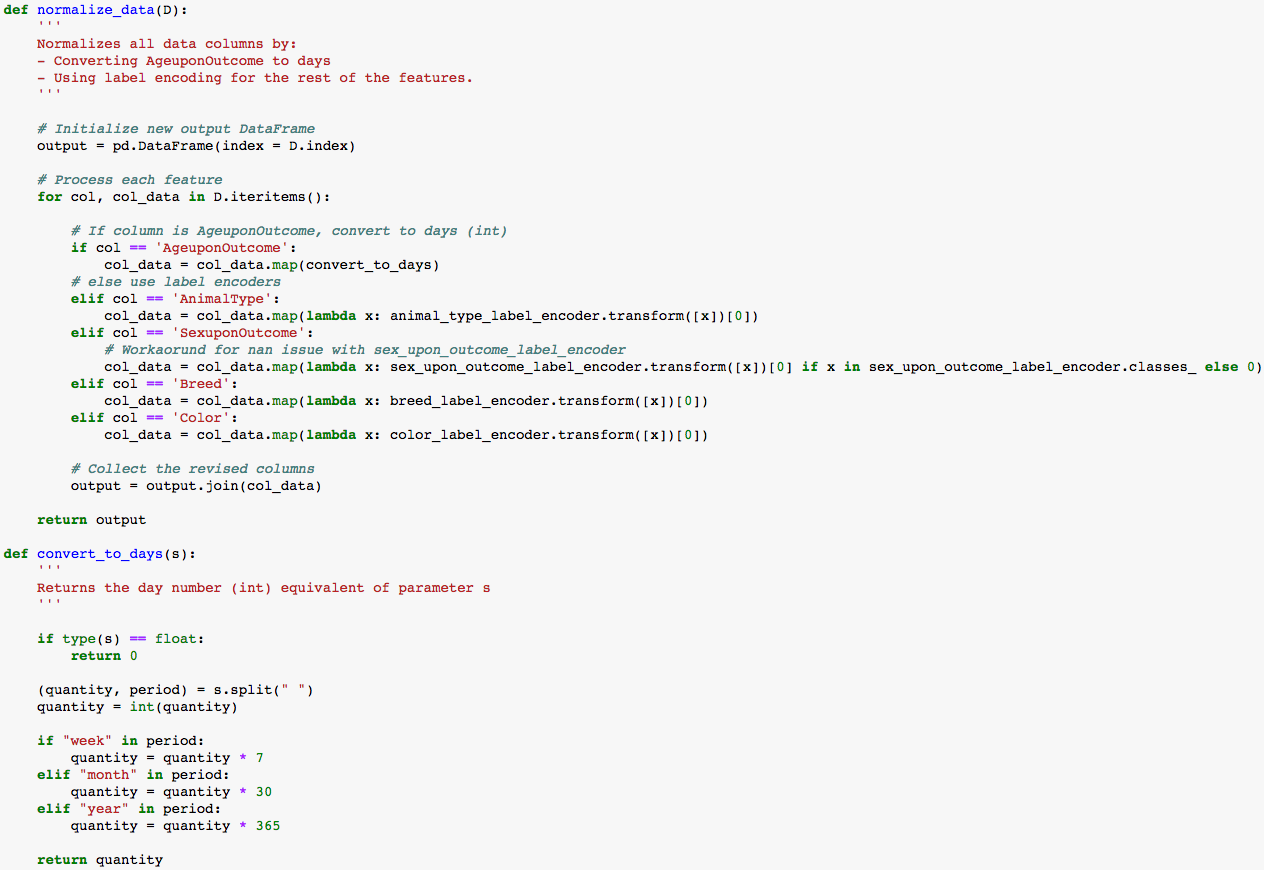
## Data Preprocessing

As previously explained in the Data Exploration section of this document, both the train and the test datasets contain non-quantifiable (string-value based) features that needed to be processed before being able to be consumed by the ML algorithm.

For this, initially the one-hot-encoding approach (aka dummy variables) was used; this, however, ended up generating close to 2000 columns/features in the post-processed DataFrame. This made it a bit cumbersome to consume. Alternatively, a better approach was to use [sklearn.preprocessing.LabelEncoder](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html), which would result in the same amount of columns/features as in the original datasets, except for the AgeuponOutcome, which required a distinct approach (as explained below).

The following features were processed, using the normalize\_data function documented below:

* AgeuponOutcome: The original dataset contains string entries on this feature as such: “1 day”, “2 weeks”, “3 months”, “4 years”. These data values were converted to an int, representing “days” (the lower common denominator).
* AnimalType: 2 unique values.
* SexUponOutcome: 5 unique values.
* Breed: 1,678 unique values.
* Color: 411 unique values.
* OutcomeType: 5 unique values.



This preprocessing step was necessary for both datasets (the one in *train.csv* as well as the one in *test.csv*). Given that some values under some features were unique to one of the datasets, the label encoders for each feature were created based on the combination of possible values for the same feature in both datasets. This would allow us to us the same label encoding values across the board, which was necessary to maintain the model’s integrity through training and testing

Similarly, another preprocessing step that was necessary before being able to submit our predictions to Kaggle’s evaluation engine, was to convert the predictions to a [predefined csv format](https://www.kaggle.com/c/shelter-animal-outcomes/details/evaluation) which would include the animal IDs to the predicted records. This processing was done via a custom python function that leveraged the power of numpy to manipulate data arrays and export them to a csv file.

## Implementation

To train the models, the dataset in train.csv (after being pre-processed via the method described above) was split in 80/20 (train/test data).

Using the 80% train split, the 3 selected classifiers were trained with their corresponding default hyper-parameters:

* [sklearn.tree.DecisionTreeClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)
* [sklearn.svm.SVC](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html)
* [xgboost.XGBClassifier](https://xgboost.readthedocs.io/en/latest/python/python_api.html)

With the each classifier model trained, the 20% test split data was then used to evaluate and compare each model’s performance using [sklearn.metric.log\_loss](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html).

In the end, XGBoost performed significantly better out-of-the-box, which made it an ideal candidate for refinement / fine-tuning[[6]](#footnote-6).

## Refinement

For the refinement stage, the XGBoost model was selected[[7]](#footnote-7). Also, a manual approach similar to sklearn’s [GridSearchCV](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) was followed. The process considered the following hyper-parameters[[8]](#footnote-8):

* learning\_rate
* max\_depth
* min­\_child\_weight
* gamma
* subsample
* colsample\_bytree
* nthread

The process to fine-tune the classifier’s hyper-parameters consisted of tweaking certain hyper-parameter values, training a new model with the tweaked hyper-parameters, and testing the new Log Loss score for the updated model. The order in which these hyper-parameters were tuned was the following:[[9]](#footnote-9)

1. Tweak learning\_rate
2. Tweak nthread
3. Tweak max\_depth
4. Tweak subsample and colsample\_bytree
5. Tweak min\_child\_weight
6. Tweak gamma

In some cases (for some hyper-parameters) the new values would actually generate worse Log Loss scores. In those cases, preference was given to the corresponding default hyper-parameter value.

# Results

## Model Evaluation and Validation

With the trained models, each model’s Log Loss was measured using the 20% test split data. The following were the scores obtained at this stage:

|  |  |
| --- | --- |
| Classifier | Log Loss Score |
| Decision Tree | 11.5973 |
| SVM | 1.1074 |
| XGBoost | 0.8940 |

As it can be observed, the Decision Tree classifier performed poorly as compared with the other 2. The difference is significant, which deemed that model/algorithm unusable for our predictions.

On the other hand, both models generated by SVM and XGBoost showed low Log Loss scores out-of-the-box, which is good. One big disadvantage between the 2 was that the SVM training process very slow (as expected based on the classifier’s complexity). This implied that any kind of refinement via GridSearchCV of that model would turn out to be extremely slow[[10]](#footnote-10). In addition, XGBoost’s score was almost 20% better than SVM’s. The combination of both factors led to the conclusion that the only model worthy of any refinement attempt was the one produced by XGBoost.

After several cycles of trial-and-error, the refinement process generated the following results:

|  |  |
| --- | --- |
| Item | Value |
| learning\_rate | 0.1 |
| max\_depth | 5 |
| min\_child\_weight | 1 |
| gamma | 0 |
| subsample | 0.905 |
| colsample\_bytree | 0.8 |
| nthread | 4 |
| Log Loss Before Refinement | 0.8941 |
| Log Loss After Refinement | 0.8862 |

## Justification

The scores generated pre- and post-refinement by XGBoost were both better than our Benchmark. Moreover, while refining the model, several submissions were done to Kaggle’s evaluation engine. At best, the score generated by the engine classified the model toward the top third of its [public board](https://www.kaggle.com/c/shelter-animal-outcomes/leaderboard). Give the data used to make these submissions were entirely new, this proved that our model has the capacity to generalize its predictions to classify unseen data.

# Conclusion

## Free-form Visualization

## Reflection

## Improvement

**Machine Learning Engineer Nanodegree**

**Capstone Project**

Joe Udacity  
December 31st, 2050

**I. Definition**

*(approx. 1-2 pages)*

**Project Overview**

In this section, look to provide a high-level overview of the project in layman’s terms. Questions to ask yourself when writing this section:

* *Has an overview of the project been provided, such as the problem domain, project origin, and related datasets or input data?*
* *Has enough background information been given so that an uninformed reader would understand the problem domain and following problem statement?*

**Problem Statement**

In this section, you will want to clearly define the problem that you are trying to solve, including the strategy (outline of tasks) you will use to achieve the desired solution. You should also thoroughly discuss what the intended solution will be for this problem. Questions to ask yourself when writing this section:

* *Is the problem statement clearly defined? Will the reader understand what you are expecting to solve?*
* *Have you thoroughly discussed how you will attempt to solve the problem?*
* *Is an anticipated solution clearly defined? Will the reader understand what results you are looking for?*

**Metrics**

In this section, you will need to clearly define the metrics or calculations you will use to measure performance of a model or result in your project. These calculations and metrics should be justified based on the characteristics of the problem and problem domain. Questions to ask yourself when writing this section:

* *Are the metrics you’ve chosen to measure the performance of your models clearly discussed and defined?*
* *Have you provided reasonable justification for the metrics chosen based on the problem and solution?*

**II. Analysis**

*(approx. 2-4 pages)*

**Data Exploration**

In this section, you will be expected to analyze the data you are using for the problem. This data can either be in the form of a dataset (or datasets), input data (or input files), or even an environment. The type of data should be thoroughly described and, if possible, have basic statistics and information presented (such as discussion of input features or defining characteristics about the input or environment). Any abnormalities or interesting qualities about the data that may need to be addressed have been identified (such as features that need to be transformed or the possibility of outliers). Questions to ask yourself when writing this section:

* *If a dataset is present for this problem, have you thoroughly discussed certain features about the dataset? Has a data sample been provided to the reader?*
* *If a dataset is present for this problem, are statistics about the dataset calculated and reported? Have any relevant results from this calculation been discussed?*
* *If a dataset is****not****present for this problem, has discussion been made about the input space or input data for your problem?*
* *Are there any abnormalities or characteristics about the input space or dataset that need to be addressed? (categorical variables, missing values, outliers, etc.)*

**Exploratory Visualization**

In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section:

* *Have you visualized a relevant characteristic or feature about the dataset or input data?*
* *Is the visualization thoroughly analyzed and discussed?*
* *If a plot is provided, are the axes, title, and datum clearly defined?*

**Algorithms and Techniques**

In this section, you will need to discuss the algorithms and techniques you intend to use for solving the problem. You should justify the use of each one based on the characteristics of the problem and the problem domain. Questions to ask yourself when writing this section:

* *Are the algorithms you will use, including any default variables/parameters in the project clearly defined?*
* *Are the techniques to be used thoroughly discussed and justified?*
* *Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?*

**Benchmark**

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section:

* *Has some result or value been provided that acts as a benchmark for measuring performance?*
* *Is it clear how this result or value was obtained (whether by data or by hypothesis)?*

**III. Methodology**

*(approx. 3-5 pages)*

**Data Preprocessing**

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section:

* *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented?*
* *Based on the****Data Exploration****section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected?*
* *If no preprocessing is needed, has it been made clear why?*

**Implementation**

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

* *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
* *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
* *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

**Refinement**

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

* *Has an initial solution been found and clearly reported?*
* *Is the process of improvement clearly documented, such as what techniques were used?*
* *Are intermediate and final solutions clearly reported as the process is improved?*

**IV. Results**

*(approx. 2-3 pages)*

**Model Evaluation and Validation**

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

* *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
* *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
* *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
* *Can results found from the model be trusted?*

**Justification**

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

* *Are the final results found stronger than the benchmark result reported earlier?*
* *Have you thoroughly analyzed and discussed the final solution?*
* *Is the final solution significant enough to have solved the problem?*

**V. Conclusion**

*(approx. 1-2 pages)*

**Free-Form Visualization**

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

* *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
* *Is the visualization thoroughly analyzed and discussed?*
* *If a plot is provided, are the axes, title, and datum clearly defined?*

**Reflection**

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

* *Have you thoroughly summarized the entire process you used for this project?*
* *Were there any interesting aspects of the project?*
* *Were there any difficult aspects of the project?*
* *Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

**Improvement**

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

* *Are there further improvements that could be made on the algorithms or techniques you used in this project?*
* *Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?*
* *If you used your final solution as the new benchmark, do you think an even better solution exists?*

**Before submitting, ask yourself. . .**

* Does the project report you’ve written follow a well-organized structure similar to that of the project template?
* Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
* Would the intended audience of your project be able to understand your analysis, methods, and results?
* Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
* Are all the resources used for this project correctly cited and referenced?
* Is the code that implements your solution easily readable and properly commented?
* Does the code execute without error and produce results similar to those reported?

1. Source: <https://www.kaggle.com/c/shelter-animal-outcomes> [↑](#footnote-ref-1)
2. Source: <https://www.kaggle.com/wiki/LogarithmicLoss> [↑](#footnote-ref-2)
3. The XGBoost library used for this analysis is not part of sklearn’s core library, so it had to be installed as a plugin. Unfortunately, the version used is missing some API integrations with the rest of sklearn, including the ability to use sklearn’s GridSearchCV functionality. [↑](#footnote-ref-3)
4. <https://en.wikipedia.org/wiki/Gradient_boosting> [↑](#footnote-ref-4)
5. An interesting post to this matter is available in the competition’s forum: <https://www.kaggle.com/c/shelter-animal-outcomes/forums/t/22119/cheating-your-way-to-the-top-of-the-lb-remove-the-lb> [↑](#footnote-ref-5)
6. The results of the comparison are detailed in the Model Evaluation and Validation section of this document. [↑](#footnote-ref-6)
7. Refer to the Model Evaluation and Validation section of this document for the reasoning behind this selection. [↑](#footnote-ref-7)
8. Official documentation on XGBoost parameters and their meaning: <https://github.com/dmlc/xgboost/blob/master/doc/parameter.md> [↑](#footnote-ref-8)
9. Inspired by: <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/> [↑](#footnote-ref-9)
10. Several attempts to perform GridSeachCV optimizations on the SVM model were made, but all were stopped after having the process run several hours without any completion. In one particular case, the process was left running for 24 hours before stopping it. [↑](#footnote-ref-10)