

Machine Learning Nanodegree Capstone Project

Kaggle Challenge: PetFinder.my Adoption Prediction

Mahesh Babu Gorantla - March 20, 2019

- Machine Learning Nanodegree Capstone Project
- Kaggle Challenge: PetFinder.my Adoption Prediction
- Project Overview
- Problem Statement
- Metrics
 - Choosing the Evaluation Metric
- Data Exploration
 - Dataset Sample
 - Data Fields
 - AdoptionSpeed
 - Images
 - Image Metadata
 - Sentiment Data
 - Dropping Irrelevant Columns
 - Missing Values Per Column
- Exploratory Visualization
 - Descriptive Statistics
 - Dataset Distribution (per class)
 - Understanding the Descriptive Statistics
 - Data Distribution Plot
 - Feature Correlation Heatmap (Dogs and Cats)
- Algorithms and Techniques
 - kNN Classifier Model
 - XGBoost Model
 - Choosing Baseline Model
 - Choosing Solution Model
- Benchmark
- Data Preprocessing
 - 1. Transforming Missing Values to None
 - 2. Imputing the Missing Values
 - Quantitative Columns
 - Categorical Columns
 - 3. Anomalies in Dogs and Cats Data
 - Anomalies in Dogs Data
 - Anomalies in Cats Data
- Implementation
 - Challenges with Data
- Refinement
- Model Evaluation and Validation
- Justification
- Free-Form Visualization
- Reflection
 - Summary of Workflow Process
- Improvement
- References

Project Overview

Each year, approximately 7.6 million companion animals enter the animal shelters nationwide (ASPCA). Of those, approximately 3.9 million are dogs and 3.4 million are cats. About 2.7 million shelter animals are adopted each year (1.4 million dogs and 1.3 million cats). This leaves around two-thirds of pets going unadopted.

This uncertainty of the unadopted pets future is further worsened because many a times a pet shelter's future is typically uncertain and many pet shelters cannot keep the unadopted pets for extended periods of time due to logistical reasons and many of the pet shelters are run voluntarily (mostly dependent on generous donations).

If at all we can find those stray animals a forever home, many precious lives can be given a new life that they truly deserve and in turn we will see more happier families.

I am big fan of pets. My personal motivation to solve this problem explicitly is because doing so will make the PetFinder.my Agency streamline their pet adoption services which in turn help more pets being adopted. Being able to predict how quickly a pet will be adopted will help pet adoption agencies optimize their logistics and operate more efficiently reducing the percentage of unadopted pets.

Problem Statement

PetFinder.my has been Malaysia's leading animal welfare platform since 2008, with a database of more than 150,000 animals. PetFinder collaborates closely with animal lovers, media, corporations, and global organizations to improve animal welfare.

The problem at hand currently is to create a model to predict the adoption speed of a pet given its features (mentioned below).

Using a machine learning algorithm(s) to predict the adoption speed of a pet should help us solve the above mentioned problem.

The Pet Adoption Prediction is a multi-class classification problem. In this problem, I need to develop algorithms to predict the adoptability of pets - specifically, how quickly is a pet adopted based on its features (which have been explained in the section below).

It is a classification problem because a pet's adoption speed is classified into 5 categories.

A relevant potential solution is to use a few classification algorithms such as

- k-Means Clustering (Baseline model)
- DBSCAN
- Random Forests
- XGBoost

And, choose the one with best **cross_validation** score on the training dataset.

Relevant Academic Research has been cited in the **References** section

Metrics

Choosing the Evaluation Metric

Given, the problem is a supervised **multi-class** classification problem.

Here are a list of model evaluation metrics available for us

Source: <https://www.sciencedirect.com/science/article/pii/S0306457309000259>

μ & M indices represent **micro-** and **macro-** averaging

1. **Average Accuracy** - Average per-class effectiveness of a classifier
2. **Error Rate** - The average per-class classification error
3. **Precision $_{\mu}$** - Agreement of the data class labels with those of a classifiers if calculated from sums of per-text decisions
4. **Recall $_{\mu}$** - Effectiveness of a classifier to identify class labels if calculated from sums of per-text decisions
5. **FScore $_{\mu}$** - Relations between data's positive labels and those given by a classifier based on sums of per-text decisions
6. **Precision $_M$** - An average per-class agreement of the data class labels with those of a classifiers
7. **Recall $_M$** - An average per-class effectiveness of a classifier to identify class labels
8. **FScore $_M$** - Relations between data's positive labels and those given by a classifier based on a per-class average

From the above list we have broadly two types of metrics - **micro-average** and **macro-average**

Most commonly used metrics for multi-classes are **F1 score**, **Average Accuracy**, **Log-loss**. There is yet no well-developed **ROC-AUC** score for **multi-class**.

Among the 3 different metrics available to use for the given problem I chose **Log-loss** metric to evaluate my classification model

Log-Loss for multi-class is defined as:

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

Where,

N - Number of rows in Test Set

M - Number of Fault Delivery Classes

y_{ij} - 1 if observation belongs to class j ; else 0

p_{ij} - Predicted Probability that observation belongs to class j

- In **Micro-average** method, you sum up the individual true positives, false positives, and false negatives of the system for different sets and then apply them to get the statistics.
- In **Macro-average**, you take the average of the precision and recall of the system on different sets

Micro-average is preferable because of the class imbalance problem

Using **log_loss** metric in **scikit-learn**

```
from sklearn.metrics import log_loss, make_scorer
```

```
# Multi-class classification Scoring Function
```

```
scoring_function = make_scorer(log_loss, needs_proba=True)
```

I would like to use **log_loss** estimator along with **StratifiedKFold** cross-validation technique to compute the robustness of the classification model. **StratifiedKFold** cross-validation method is a variation of **KFold** that returns stratified folds. The folds are made by preserving the percentage of samples for each class.

The simplest way to use cross-validation is to call the **cross_val_score** helper function on the estimator and the dataset.

The model whose cross validation score and model fit time is lower than the benchmark model is my solution model.

Data Exploration

The dataset to solve the above mentioned problem has been obtained as a part of Kaggle Challenge. In this problem, I will predict the speed at which a pet is adopted, based on the pet's listing on PetFinder. Sometimes a profile represents a group of pets. In this case, the speed of adoption is determined by the speed at which all of the pets are adopted. The data included text, tabular, and image data.

Dataset Sample

Removed **Description**, **PetID** and **RescuerID** Column due to space constraints

Type	Name	Age	Breed1	Breed2	Gender	Color1	Color2	Color3	MaturitySize	FurLength	Vaccinated	Dewormed	Sterilized	Health	Quantity	Fully	Stat	Video	Photo	Adopted	AdoptionSpeed
2	Nibble	3	299	0	1	1	7	0	1	1	2	2	2	1	1	100	41326	1	2		
2	No Name Yet	1	265	0	1	1	2	0	2	2	3	3	3	1	1	0	41401	1	2		
1	Brisco	1	307	0	1	2	7	0	2	2	1	1	2	1	1	0	41326	7	3		
1	Miko	4	307	0	2	1	2	0	2	1	1	1	2	1	1	150	41401	8	2		
1	Hunter	1	307	0	1	1	0	0	2	1	2	2	2	1	1	0	41326	3	2		

Open [*this*](#) to clearly see a sample of the dataset

Data Fields

- **PetID** - Unique hash ID of pet profile
- **AdoptionSpeed** - Categorical speed of adoption. Lower is faster. **This is the value to predict.** See below section for more info.
- **Type** - Type of animal (1 = Dog, 2 = Cat)
- **Name** - Name of pet (Empty if not named)
- **Age** - Age of pet when listed, in months
- **Breed1** - Primary breed of pet (Refer to BreedLabels dictionary)
- **Breed2** - Secondary breed of pet, if pet is of mixed breed (Refer to BreedLabels dictionary)
- **Gender** - Gender of pet (1 = Male, 2 = Female, 3 = Mixed, if profile represents group of pets)
- **Color1** - Color 1 of pet (Refer to ColorLabels dictionary)
- **Color2** - Color 2 of pet (Refer to ColorLabels dictionary)
- **Color3** - Color 3 of pet (Refer to ColorLabels dictionary)
- **MaturitySize** - Size at maturity (1 = Small, 2 = Medium, 3 = Large, 4 = Extra Large, 0 = Not Specified)
- **FurLength** - Fur length (1 = Short, 2 = Medium, 3 = Long, 0 = Not Specified)
- **Vaccinated** - Pet has been vaccinated (1 = Yes, 2 = No, 3 = Not Sure)
- **Dewormed** - Pet has been dewormed (1 = Yes, 2 = No, 3 = Not Sure)
- **Sterilized** - Pet has been spayed / neutered (1 = Yes, 2 = No, 3 = Not Sure)

- **Health** - Health Condition (1 = Healthy, 2 = Minor Injury, 3 = Serious Injury, 0 = Not Specified)
- **Quantity** - Number of pets represented in profile Fee - Adoption fee (0 = Free)
- **State** - State location in Malaysia (Refer to StateLabels dictionary)
- **RescuerID** - Unique hash ID of rescuer
- **VideoAmt** - Total uploaded videos for this pet
- **PhotoAmt** - Total uploaded photos for this pet
- **Description** - Profile write-up for this pet. The primary language used is English, with some in Malay or Chinese.

AdoptionSpeed

Contestants are required to predict this value. The value is determined by how quickly, if at all, a pet is adopted. The values are determined in the following way:

- 0 - Pet was adopted on the same day as it was listed.
- 1 - Pet was adopted between 1 and 7 days (1st week) after being listed.
- 2 - Pet was adopted between 8 and 30 days (1st month) after being listed.
- 3 - Pet was adopted between 31 and 90 days (2nd & 3rd month) after being listed.
- 4 - No adoption after 100 days of being listed. (There are no pets in this dataset that waited between 90 and 100 days)

Images

For pets that have photos, they will be named in the format of *PetID-ImageNumber.jpg*. Image 1 is the profile (**default**) photo set for the pet. For privacy purposes, faces, phone numbers and emails have been masked.

Image Metadata

We have run the images through **Google's Vision API**, providing analysis on **Face Annotation**, **Label Annotation**, **Text Annotation** and **Image Properties**. You may optionally utilize this supplementary information for your image analysis.

File name format is *PetID-ImageNumber.json*.

Some properties will not exist in JSON file if not present, i.e. Face Annotation. Text Annotation has been simplified to just 1 entry of the entire text description (instead of the detailed JSON result broken down by individual characters and words). Phone numbers and emails are already anonymized in Text Annotation.

Google Vision API reference: <https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate>

Sentiment Data

We have run each pet profile's description through **Google's Natural Language API**, providing analysis on sentiment and key entities. You may optionally utilize this supplementary information for your pet description analysis. There are some descriptions that the API could not analyze. As such, there are fewer sentiment files than there are rows in the dataset.

File name format is *PetID.json*.

Google Natural Language API reference: <https://cloud.google.com/natural-language/docs/basics>

Dropping Irrelevant Columns

- Name, PetID, RescuerID
- It makes sense to not have Name column because it does not help us predict how faster a pet animal will be adopted as we know that any prospective pet owner will not adopt a pet animal based on its name.
- Likewise PetID, RescuerID also have no relevance to the task at hand
- However, for now to simplify the modeling let's ignore the following columns
- Fee - Might have some implication if a pet will adopted (Need to check the correlation)
- State - Might not influence the adoption speed (Need to check the correlation)
- VideoAmt, PhotoAmt & Description - Will have some impact on the adoption speed. Because a beautiful looking pet might get adopted sooner. But just to make the modeling simpler let's ignore these columns as well for now.

Missing Values Per Column

A common abnormality found in datasets is missing values

Column Name	No. of Missing Values
Type	0
Name	1257
Age	0
Breed1	0
Breed2	0
Gender	0
Color1	0
Color2	0
Color3	0
FurLength	0
Vaccinated	0
Dewormed	0
Sterilized	0
Health	0
Quantity	0
Fee	0
State	0
RescuerID	0
VideoAmt	0
Description	12
PetID	0
PhotoAmt	0
AdoptionSpeed	0

Very often rows with missing values will be dropped as anomalies in the data however, given that the dataset only has 14,993 datapoints removing 1269 rows we are almost throwing away nearly 10% of the data which will not be an ideal operation to perform as a part of data cleaning process.

However as mentioned above, the columns Name and Description are being dropped as irrelevant columns

for the problem at hand. Hence, we are effectively removing anomalies instead of completely deleting the other columns that are needed.

Exploratory Visualization

Descriptive Statistics

Glimpse of Descriptive Statistics of the dataset

	Type	Age	Breed	Breed2	Gender	Color1	Color2	Color3	Maturity	Size	Length	Marcina	Devon	Sterilized	Health	Quantity	Adoption	Speed
count	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993	14993
mean	1.4576	0.8710	265.2734	0.0097	7.7761	2.2341	3.2228	1.8820	1.862	1.4674	81.7312	11.5587	3.9142	31.0366	2.5760	7.5164	4	
std	0.4982	1.5129	80.0568	23.0102	6.8159	7.4522	7.4252	3.9840	9.5479	59.5990	70.6676	49.6958	17.5661	72.1995	35.1724	8.1772	6	
min	1	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	0	
25%	1	0.1666	265	0	1	1	0	0	2	1	1	1	2	1	1	2		
50%	1	0.25	266	0	2	2	2	0	2	1	2	1	2	1	1	2		
75%	2	1	307	179	2	3	6	5	2	2	2	2	2	1	1	4		
max	2	21.25	307	307	3	7	7	7	4	3	3	3	3	3	20	4		

Open *this* to clearly see descriptive statistics

Dataset Distribution (per class)

	Adoption Speed	Data Points Size
0		410
1		3090
2		4037
3		3259
4		4197

From the above observations we can determine that the dataset is almost **imbalanced**.

Understanding the Descriptive Statistics

Just by glancing at the above table, columns **Age**, **Breed1**, **Color 2** & **Color 3** stand out especially.

Because the **min** value in each of the above mentioned columns is 0.

- **Age** of 0 years is an abnormal value
- This indicates that the value for age is missing
- Need to replace the 0 with **None**
- **Breed1** number 0 is an abnormal value.
- **BreedID** for Dogs ranges from 1 - 241
- **BreedID** for Cats ranges from 242 - 307
- **Breed2** value of 0 indicates the pet is pure breed
- **Color2** & **Color3** has color 0

- ColorID ranges from 1 - 7

Color	ColorID
Black	1
Brown	2
Golden	3
Yellow	4
Cream	5
Gray	6
White	7

Data Distribution Plot

some features have been removed in the pairplots due to space constraints

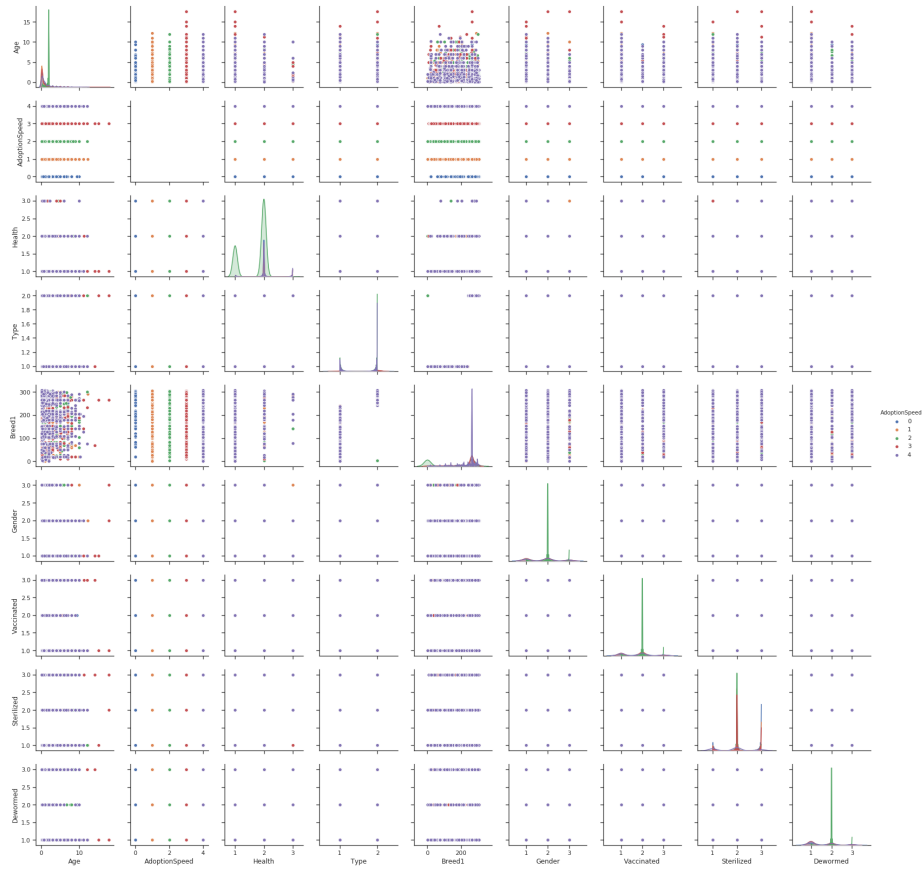


Figure 1: Features Pair Plot

From the above pairplot of the features we can infer that **Age** feature has some impact on how fast a pet will be adopted. The lower the **Age** the faster a pet will get adopted.

Feature Correlation Heatmap (Dogs and Cats)

From the above two feature correlation heatmaps, we can determine that there are clearly no features that clearly draw the classification boundaries to determine **AdoptionSpeed** of a given pet. This characteristic of

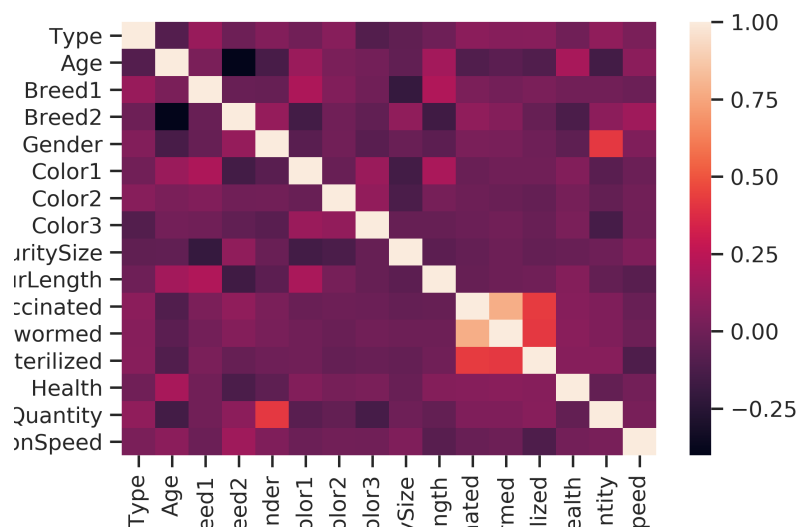


Figure 2: Dog Features Correlation Heatmap

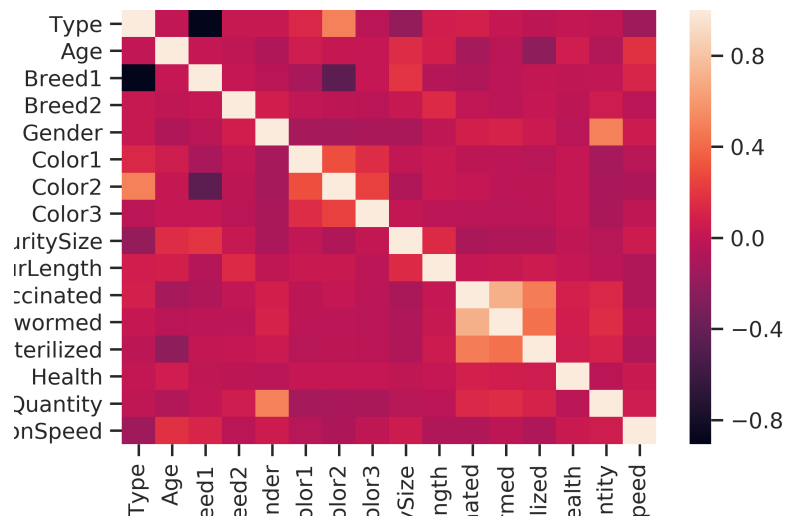


Figure 3: Cat Feature Correlation Heatmap

the dataset necessitates us to perform **Feature Transformation** to establish more explicit boundaries to predict the adoption speed of a pet animal.

Algorithms and Techniques

Predicting **AdoptionSpeed** of a pet animal in the PetFinder's Animal Database is a supervised multi-class classification problem. To predict **AdoptionSpeed** based on the training features of a pet, I would like to experiment with classification algorithms like

1. Classification and Regression Trees (CART)
2. kNN Classifier (**Baseline Model**)
3. Support Vector Machines (SVM)
4. Random Forests
5. XGBoost (**Solution Model**)

to predict the adaptability of a pet.

I chose to use the below mentioned two classification algorithms to predict pet adoption speed

kNN Classifier Model

The k-Nearest-Neighbors algorithm is a supervised classification algorithm. The algorithm takes in a set of labelled datapoints (**Training Set**) and uses the patterns learned from the dataset to label other points. To label new datapoints, it identifies the labelled datapoints that are closer (nearest neighbors) to the new datapoint and lets the neighbors vote. Whichever label most of the neighbors have will be used to label the new datapoint (the **k** is the number of neighbors the algorithm checks). Unlike most of the other methods of classification, kNN uses **no explicit training phase before classification**. Any attempts to generalize the data is made upon classification.

Advantages

- It is **non-parametric** - algorithm makes no assumptions about the input dataset
- Works best on smaller datasets with not many features
- High Accuracy (relatively) - it is pretty high but not as competitive

Disadvantages

- The model keeps the entire dataset in the memory. Hence, bad for large datasets
- Performing classifications can be computationally expensive because the algorithm parses the whole dataset for each classification
- Suffers tremendously from the curse of dimensionality of the dataset
- Classification speed is slow

XGBoost Model

XGBoost (**eXtreme Gradient Boosting**) is a scalable and accurate implementation of gradient boosting machines and it has proven to push limits of computing power for boosted trees algorithms. It is built with the sole purpose of model performance and computational speed.

XGBoost is an ensemble method that seeks to create a strong classifier (model) based on **weak** classifiers. Here the **weak** and **strong** refer to a measure of how correlated are the learners to the actual target variable. This algorithm sequentially adds predictors and corrects previous models. Instead of assigning different weights to the classifiers after every iteration, this method fits the new model to new residuals of the previous prediction and then minimizes the loss when adding the latest prediction.

Advantages

- Lower prediction time over other supervised classification models
- Prevents overfitting using both L1 and L2 regularization
- Incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data
- Uses a distributed wighted quantile sketch algorithm to effectively handle weighted data
- Block Structure for parallel computing
- Optimizes the available disk space and maximizes its usage when handling huge datasets

Disadvantages

- A very complex model to understand and explain
- Longer model training time than other classification models

Choosing Baseline Model

A very crude baseline model can be just an educated guess based on the `multi-class` prediction variable distribution. Using the dataset distribution as mentioned above, just by guessing the `AdoptionSpeed` as 4 will get us an accuracy of 27.99%.

Any of the classification models (from the above) that will beat the aforementioned crude prediction model (based on educated guessing) even by a narrow margin will be my baseline classification model.

I chose `kNN` classifier algorithm as my baseline. Because, given the number of data points and features of a pet, `kNN` classification algorithm should produce a less complex and easy to understand prediction model than `XGBoost` algorithm.

Choosing Solution Model

`XGBoost` is the right fit to create classification model weighing in the **advantages** and **disadvantages** of the algorithm.

Benchmark

As we know that the above mentioned prediction task is a supervised classification problem, we should use tree based classification models which typically outperform other classification models. Hence, I would like to use `kNN Classifier` algorithm as a benchmark and beat the benchmark performance both in terms of time to fit the model as well as increase in the prediction accuracy.

Currently, the baseline prediction accuracy (without feature transformation) is as follows

Classification Model	Model Fit Time (in mins)	Log Loss Score
KNeighborsClassifier	0.68	5.52

Data Preprocessing

1. Transforming Missing Values to None

Lets revisit the abnormalities detected while examining the descriptive statistics again. The aforementioned abornamilities are possibly present in the data due to some human error or it is missing data.

If no documentation is available, some common values used instead of missing values are:

- 0 (for numerical values)

- **unknown** or **Unknown** (for categorical variables)
- **?** (for categorical variables)

Number of missing values after transforming the missing values (abnormal) to **None** are

Column	Missing Values Count
Type	0
Age	179
Breed1	5
Breed2	0
Gender	0
Color1	0
Color2	4471
Color3	10604
MaturitySize	0
FurLength	0
Vaccinated	0
Dewormed	0
Sterilized	0
Health	0
Quantity	0
AdoptionSpeed	0

The maximum number of rows that will be missing if we chose to drop the rows with missing values in a column is 10604 (i.e. we will lose around **70.73%** data).

Losing **70.73%** data is not ideal to solve the problem at hand. We need to impute the missing data (columns) using custom **Quantitative** and **Categorical** imputing methods.

2. Imputing the Missing Values

Quantitative Columns

To impute the **Quantitative** columns (**Age**), I have used respective **mean** values per column to impute the missing values.

Categorical Columns

To impute the **Categorical** columns (**Breed1**, **Color1**, **Color2**), I have used the most frequently occurring (**mode**) category for respective categorical columns.

3. Anomalies in Dogs and Cats Data

Anomalies in Dogs Data

In 4th row, 3rd column, **Type** column should be **1** for all the dog breeds. However, we can see that some rows of data with **dog** breeds have been marked as **cats**.

There are 36 dogs that are mislabelled as cats

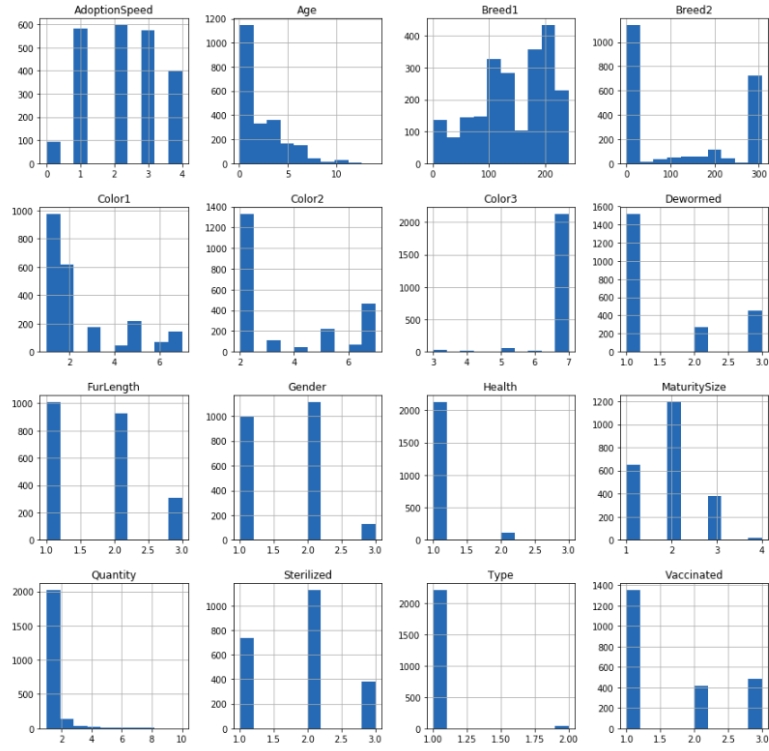


Figure 4: Dogs Data Distribution

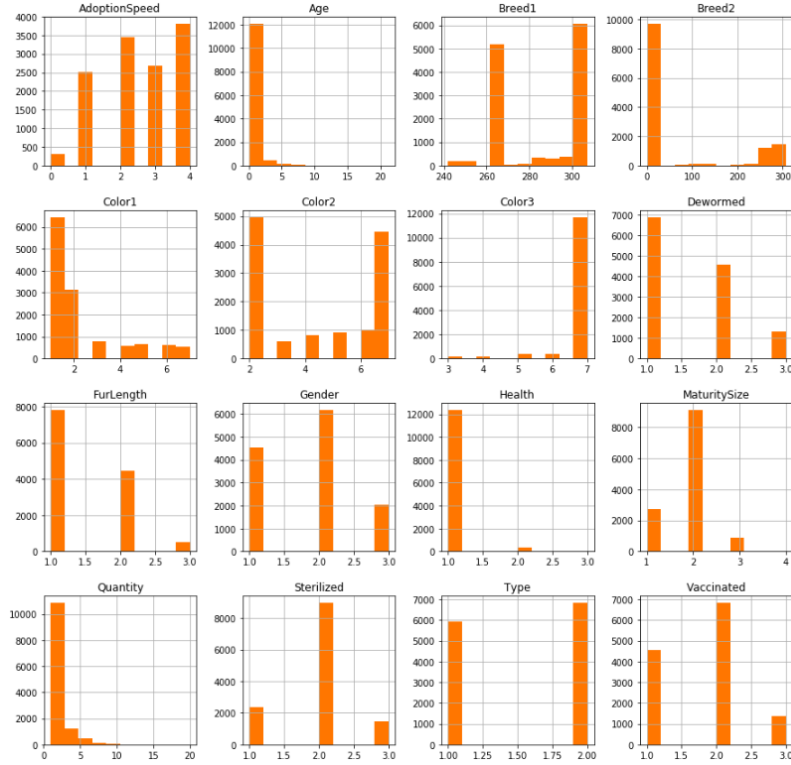


Figure 5: Cats Data Distribution

Anomalies in Cats Data

In 4th row, 3rd column, **Type** column should be 2 for all the cat breeds. However, we can see that almost 50% of data rows with **cat** breeds have been marked as **dogs**.

The above mentioned anomaly can assumed to be because of human errors often seen in manual data collection/entry process

There are 5927 cats that are mislabelled as dogs.

To correct such anomalies we need to make sure that all the rows whose **Breed1** values belong to cats breeds need to be marked with 2 and likewise mark dog breeds with 1

Implementation

Theoretical Workflow for implementing the Solution Model

1. Import the Training Dataset

- Load the necessary libraries such as **numpy**, **pandas**
- Load the **data/train/train.csv** dataset
- Drop irrelevant columns

2. Data Exploration

- Find missing rows in the each column using **df.isnull().sum()**
- Obtain and Understand Descriptive Statistics

3. Feature Engineering

- Decipher more missing values from descriptive statistics (if any)
- Impute missing values using Custom Quantitative/Categorical Imputers
- Visualize per column distribution using histograms
- Find any anomalous data distributions from the plots
- Dummify Categorical Columns
- Scale all the columns either by using **z-scaling** (**StandardScaler**) or **min-max scaling** (**MinMaxScaler**) technique

4. Evaluate Classification Models

- Choose a few classification models
- Split the dataset into train/split test using **StratifiedKFold** cross-validation technique
- Score each classification model using **cross_val_score** and **log_loss** scoring function
- Choose the classification model with best cross validation score
 - The lower the **log_loss** score the better the classification model is

5. Feature Transformation

- Reduce the dimensionality of features
 - Using **LDA** - Linear Discriminant Analysis (Best for multi-class classification)

6. Tune the Classification Model

- Use **GridSearchCV** to improve the accuracy

7. Save the best model

- with best prediction time and accuracy

Challenges with Data

- I have initially performed just very nominal data cleaning by just removing all the rows where any column is empty. However, after close introspection of the data distribution just by glancing at the **Descriptive Statistics** of the data, I found glaring anomalies in a few features such as **Age**.
- Make sure to clearly interpret the descriptive statistics and correlate the results with respect to each individual feature to decipher the anomalies/outliers
- Imputing the missing columns in each row
- Ensure the data imputation techniques are valid by **re-examining** the descriptive statistics after filling in missing values
- Determining abnormal data values (other than missing values)
- Clearly examine each feature distribution using histograms to easily point out the abnormal data distributions

Implementing appropriate methods to impute missing values and correct out of place values for each feature significantly improved the classification model's performance

Refinement

Steps used to improve the prediction accuracy of the classification model.

1. Preprocess the Data (as mentioned in the Feature Engineering)
2. Transform the input features to reduce the dimensionality of the input data using LDA - Linear Discriminant Analysis (Best for multi-class classification)
3. Used `n_components = 5` to transform the data using LDA
4. Obtain the prediction accuracy of classification using kNN Classifier (baseline model)
5. Use `GridSearchCV` method to best parameter set for **XGBoost Classifier** algorithm that will result in a better prediction accuracy than **kNN Classifier**

Final Results after Feature Transformation

Classification Model	Model Fit Time (in mins)	Log Loss Score
XGBoost	0.07	0.92

Model Evaluation and Validation

GridSearch Cross-Validation technique is utilized to evaluate the **XGBoost** model parameters and pick the best parameters that will give a lower **log-loss** score of **0.92** for the classification model.

Here are the parameters that give us the best cross validation score

XGBClassifier Parameter	Value
<code>objective</code>	<code>multi:softmax</code>
<code>subsample</code>	<code>0.7000000000000001</code>
<code>max_depth</code>	<code>5</code>
<code>colsample_bytree</code>	<code>0.5</code>
<code>scale_pos_weight</code>	<code>0.5</code>

The classification model can be called robust if it has a higher precision and low recall in my scenario. It is

very highly required for the pet adoption agencies to precisely predict the adoption speed of a pet animal given its features to optimize their logistics. It is not okay to have higher recall because that will lead to a bad logistics management as well as it will be a bad experience for the pet owners

The model with a higher **F1 Score** will be an ideal robust model because it is higher precision and lower recall

Currently the model has an **F1 Score** of **68.09%** which is an indication of higher precision.

Justification

A perfect model will have a log loss of 0. My solution model has a way better **log loss** score and lesser model fit time than benchmark model

Classification Model	Model Fit Time (in mins)	Log Loss Score
KNeighborsClassifier	0.68	5.52
XGBoost	0.07	0.92

As we can observe that the solution model has much better log loss score than just educated guessing or benchmark model. Hence, the solution model should be able to provide significant help to the pet adoption agency in predicting a pet animals adoption speed.

Free-Form Visualization

A visualization has been provided that emphasizes an important quality about the project with thorough discussion. Visual cues are clearly defined.

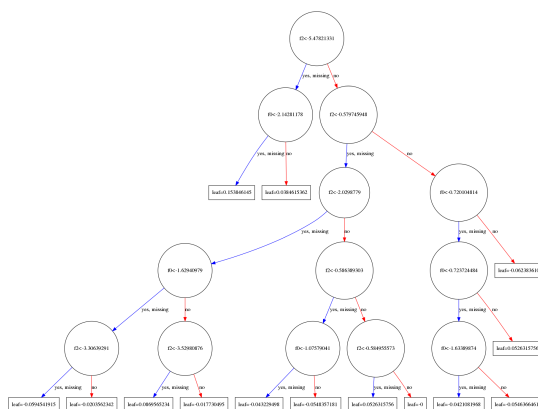


Figure 6: XGBoost Graph

From the above XGBoost Graph Tree Visualization we can observe that there are two derived features **f0** and **f2** that influence a significant portion of the classification accuracy. Here is an another plot that corroborates the above evidence

According the research articles referenced below, what makes a pet more adoptable is its **cuteness**, **health** condition and if it is not **neutered**. Because I have discarded the **cuteness** (a derived feature from pet

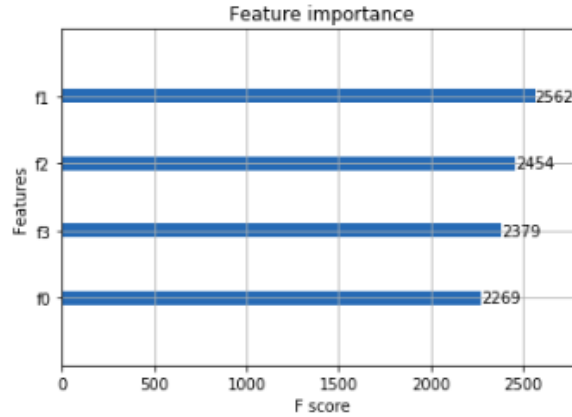


Figure 7: Feature Importance Plot

images) to make the problem simpler to solve. It is very possible that those 2 features (f0 and f2) that significantly influence the adoption speed should be a pet's **Health** and **Sterilized** features (or derived features using both the primitive features).

Reflection

Summary of Workflow Process

1. Initially, reviewed prior literature using research journals to gain a superficial understanding of the pet adoption problem
2. Download the public dataset from Kaggle that helps to solve the problem
3. Determine and resolve anomalies/missing data if any (PreProcessing step)
4. Create a Baseline model to prove that the problem is solvable
5. Use a more complex model than baseline model to improve the prediction accuracy of the classification model

Most of the project was very exciting to solve. However to make a giant leap in improving the prediction accuracy I had to investigate for anomalies in the dataset more closely which was the difficult and exciting part of the problem. Rest of the problem followed a typical procedure used to create a good classification model.

Improvement

Initially, to simplify the problem at hand I have ignored a few columns as irrelevant. Among those irrelevant columns are **photos** of a pet and **description** of a pet. From the literature review of the previously published articles on pet adoption it is found that a cuter pet animal has a higher possibility of getting adopted faster and also pets that are not **neutered** are in high-demand for adoption.

From photos, we can compute the cuteness factor of a pet animal and inject those findings as a new feature into our existing feature set as well as perform sentiment analysis of the pet description to extract the sentiment score and use this newly derived feature to help improve the prediction accuracy. The aforementioned derived features when augmented with the original dataset should significantly improve the prediction accuracy.

References

1. Weiss, E., Miller, K., Mohan-Gibbons, H., & Vela, C. (2012). Why Did You Choose This Pet?: Adopters and Pet Selection Preferences in Five Animal Shelters in the United States. *Animals*, 2(2), 144-159. doi:10.3390/ani2020144
2. Weiss, E., Dolan, E., Garrison, L., Hong, J., & Slater, M. (2013). Should Dogs and Cats be Given as Gifts? *Animals*, 3(4), 995-1001. doi:10.3390/ani3040995
3. Garrison, L., & Weiss, E. (2014). What Do People Want? Factors People Consider When Acquiring Dogs, the Complexity of the Choices They Make, and Implications for Nonhuman Animal Relocation Programs. *Journal of Applied Animal Welfare Science*, 18(1), 57-73. doi:10.1080/10888705.2014.943836
4. PetFinder.my Adoption Prediction. (n.d.). Retrieved from <https://www.kaggle.com/c/petfinder-adoption-prediction>
5. Raschka, S. (2014, August 03). Linear Discriminant Analysis. Retrieved from <https://sebastianraschka.com/Articles/2014>
6. Cross-validation: Evaluating estimator performance¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/cross_validation.html
7. Sklearn.metrics.log_loss¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_
8. Sklearn.model_selection.cross_val_score¶. (n.d.). Retrieved from <https://scikit-learn.org/stable/modules/generated/skle>
9. Sklearn.model_selection.StratifiedKFold¶. (n.d.). Retrieved from <https://scikit-learn.org/stable/modules/generated/skle>
10. Sklearn.base.TransformerMixin¶. (n.d.). Retrieved from <https://scikit-learn.org/stable/modules/generated/sklearn.base.>
11. Ozdemir, S., & Susarla, D. (2018). Feature engineering made easy: Identify unique features from your dataset in order to build powerful machine learning systems. Birmingham, UK: Packt Publishing
12. Swalin, A., & Swalin, A. (2018, May 02). Choosing the Right Metric for Evaluating Machine Learning Models - Part 2. Retrieved from <https://medium.com/usf-msds/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428>
13. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427-437. doi:10.1016/j.ipm.2009.03.002
14. Synced. (2017, October 22). Tree Boosting With XGBoost - Why Does XGBoost Win “Every” Machine Learning Competition? Retrieved from <https://medium.com/syncedreview/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition-ca8034c0b283>
15. KDnuggets. (n.d.). Retrieved from <https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html>