Pet Adoption Prediction

Capstone Project Report

❖ Project Overview

Stray animals are animals that are born on the street or being abandoned by people. Most stray animals live their lives suffering from human abusing, hunger, disease and other health problems. They also may harm humans and other animals through aggressive conflicts and can spread diseases such as rabies.

Kaggle describe stray animals' problem as follows:

"Millions of stray animals suffer on the streets or are euthanized in shelters every day around the world. If homes can be found for them, many precious lives can be saved — and more happy families created.

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.

Animal adoption rates are strongly correlated to the metadata associated with their online profiles, such as descriptive text and photo characteristics. As one example, PetFinder is currently experimenting with a simple AI tool called the Cuteness Meter, which ranks how cute a pet is based on qualities present in their photos."

Literature review

This problem is really important for this reason some good work has been done in this area. This problem is approached by analyzing a related dataset which is the most important step, then, a discriminated model is introduced.

These are some of the work related to this problem:

- https://data.world/rdowns26/austin-animal-shelter
  This research tried 4 different predictive models:
  (KNN, Naïve Bayes, Classification Tree, Random Forests)
- https://www.kaggle.com/c/shelter-animal-outcomes
  This competition was held by kaggle to predict if the animals is adoptable.
  Most of the kernels in the competition introducing supervised learning techniques such as (Random Forests, XGBoost, ensemble methods)

❖ Problem Statement

The aim of this project is to develop an algorithm to predict the adoptability of pets.

If we can predict how quickly a pet is adopted, we would be able to guide shelters and rescuers around the world on improving their pet profiles' appeal, reducing animal suffering and getting more pets to be adopted.

The adoptability of pets can be classified into 5 categories representing how quickly the pet is adopted.

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).

Note: This problems is a competition on kaggle called 'PetFinder.my Adoption Prediction' https://www.kaggle.com/c/petfinder-adoption-prediction

In this project I will use the dataset of the competition on kaggle to solve this problem using neural networks by using two approaches:

- This problem is a classification problem, which classify a pet into 5 categories. We can use neural networks to perform this classification.
- The target result of the project should be one of the 5 categories (0, 1, 2, 3, 4). Note that, increasing the number of the category means decreasing the adoptability of the pet. So I can use regression to predict the level of adoptability of the pet.

❖ Evaluation Metrics

The 5 classes we have in this problem is related to each other where increasing the number of class means decreasing the adoptability of the pet.

Quadratic weighted kappa metric, which is used by kaggle competition to calculate the scores, takes the relation or indexing between the classes into consideration. That's why I think it's the best choice.

This metric is described on kaggle as follows

"This metric typically varies from 0 (random agreement between raters) to 1 (complete agreement between raters). In the event that there is less agreement between the raters than expected by chance, the metric may go below 0. The quadratic weighted kappa is calculated between the scores which are expected/known and the predicted scores.

Results have 5 possible ratings, 0,1,2,3,4. The quadratic weighted kappa is calculated as follows.

First, an N x N histogram matrix O is constructed, such that $O_{i,j}$ corresponds to the number of adoption records that have a rating of i (actual) and received a predicted rating j. An N-by-N matrix of weights, w, is calculated based on the difference between actual and predicted rating scores:

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2}$$

An N-by-N histogram matrix of expected ratings, E, is calculated, assuming that there is no correlation between rating scores. This is calculated as the outer product between the actual rating's histogram vector of ratings and the predicted rating's histogram vector of ratings, normalized such that E and O have the same sum.

From these three matrices, the quadratic weighted kappa is calculated as :"

$$k = 1 - \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}}$$

Link on kaggle: https://www.kaggle.com/c/petfinder-adoption-prediction#evaluation

I used this code to compute the quadratic weighted kappa.

https://github.com/benhamner/Metrics/blob/master/Python/ml_metrics/quadratic_weighted_kappa.py

❖ Data Exploration

In this project, I will use the same dataset of the competition on kaggle. This data is available on kaggle platform

https://www.kaggle.com/c/petfinder-adoption-prediction/data

The data consists of tabular data, images, image metadata and sentiment data.

Tabular data

Tabular/text data contain these fields:

- PetID - Unique hash ID of pet profile
- AdoptionSpeed - Categorical speed of adoption. Lower is faster. This is the value to predict.
- 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.

Color Dictionary is a mapping from ColorID to ColorName.

Breed Dictionary is a mapping from BreedID to BreedName specifying the type of the pet for each breed.

State Dictionary is a mapping from StateID to StateName.

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

It's analysis of images on Face Annotation, Label Annotation, Text Annotation and Image Properties using Google's Vision API.

https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate

Sentiment Data

It's analysis of pet profile's description on sentiment and key entities using Google's Natural Language API.

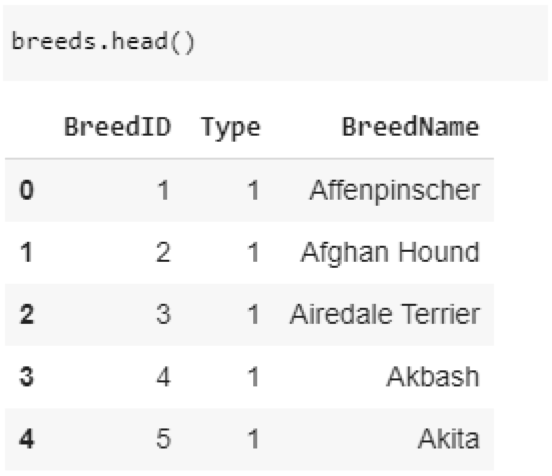
https://cloud.google.com/natural-language/docs/basics#sentiment-request

Document Sentiment contains the overall sentiment of the document, which consists of the following fields:

- Score: of the sentiment ranges between -1.0 (negative) and 1.0 (positive) and corresponds to the overall emotional leaning of the text.

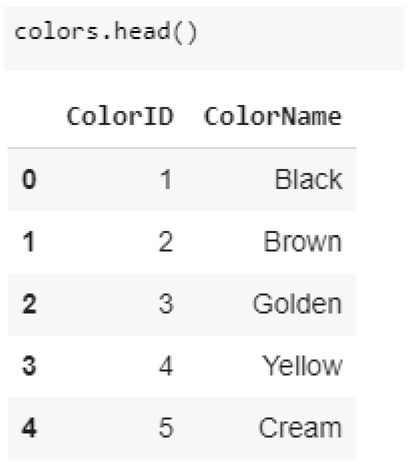
- Magnitude: indicates the overall strength of emotion (both positive and negative) within the given text, between 0.0 and +inf. Unlike score, magnitude is not normalized; each expression of emotion within the text (both positive and negative) contributes to the text's magnitude (so longer text blocks may have greater magnitudes).

Data Samples



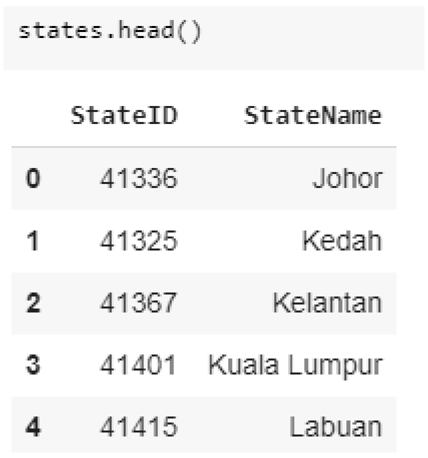
```
breeds.head()
```

	BreedID	Type	BreedName
0	1	1	Affenpinscher
1	2	1	Afghan Hound
2	3	1	Airedale Terrier
3	4	1	Akbash
4	5	1	Akita



```
colors.head()
```

	ColorID	ColorName
0	1	Black
1	2	Brown
2	3	Golden
3	4	Yellow
4	5	Cream



```
states.head()
```

	StateID	StateName
0	41336	Johor
1	41325	Kedah
2	41367	Kelantan
3	41401	Kuala Lumpur
4	41415	Labuan

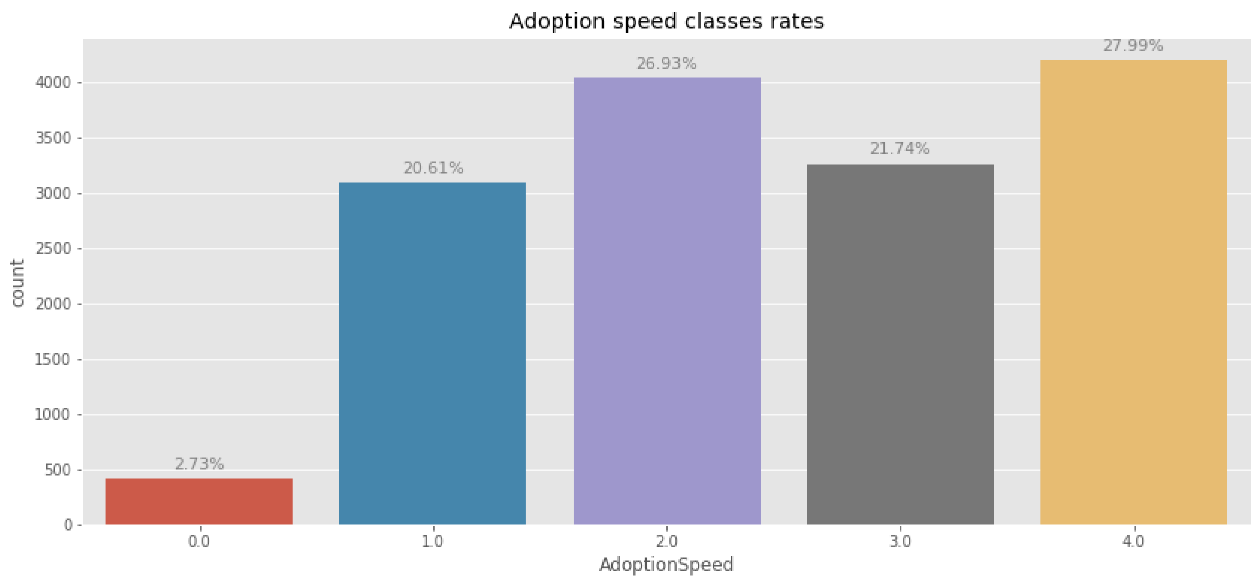
```
sentiment.head()
```

```
   doc_sent_mag  doc_sent_score
0           2.4             0.3
1           0.7            -0.2
2           3.7             0.2
3           0.9             0.9
4           3.7             0.6
```

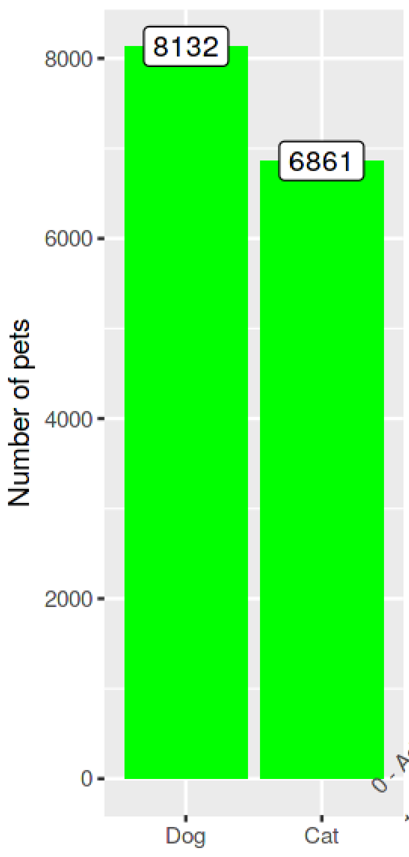
Statistics about the dataset

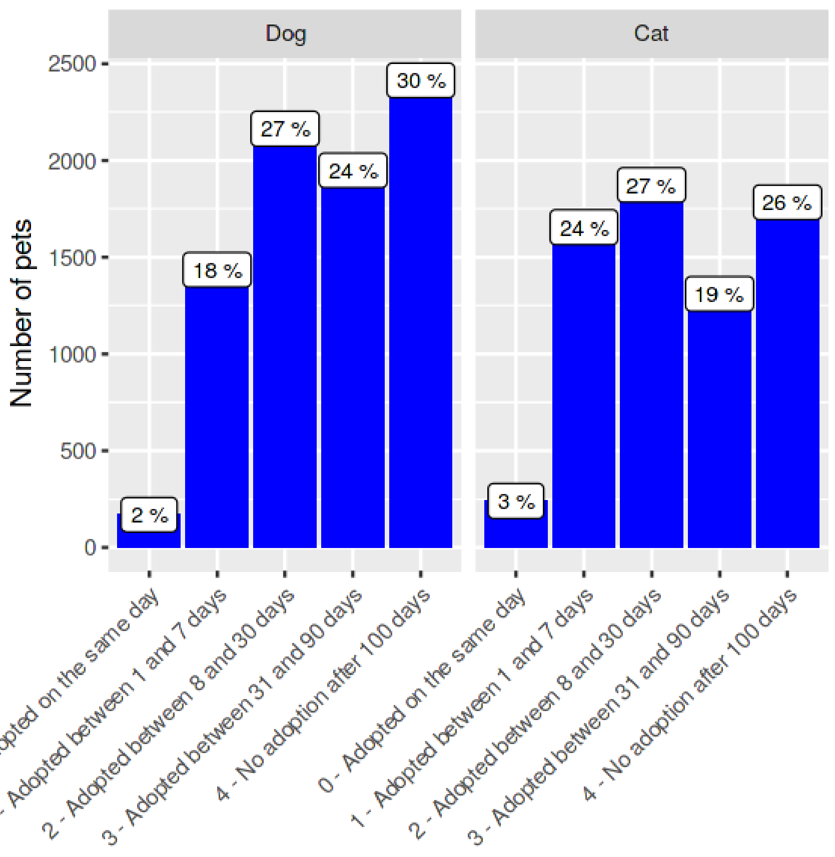
According to these kernels which are very helpful in exploring the data

- https://www.kaggle.com/erikbruin/petfinder-my-detailed-eda-and-xgboost-baseline
- https://www.kaggle.com/artgor/exploration-of-data-step-by-step
- https://www.kaggle.com/risntforpirates/petfinder-simple-lgbm



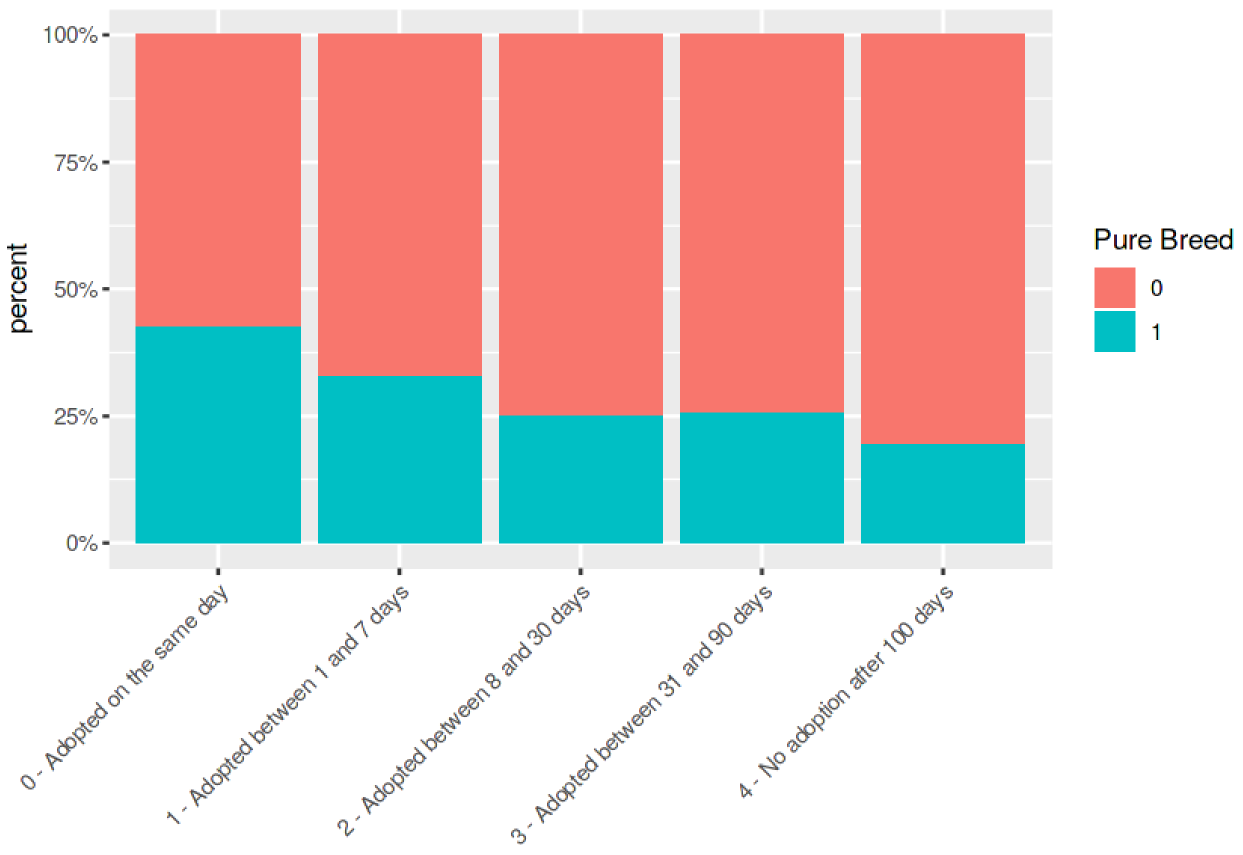
This graph shows that very small number of pets is adopted immediately. Most of the pets aren't adopted at all. The second most of the pets are adopted between 8 and 30 days (1st month) after being listed.





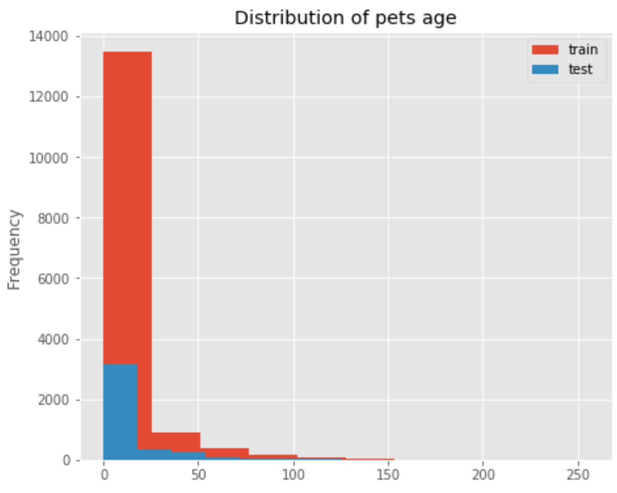
Dogs versus Cats

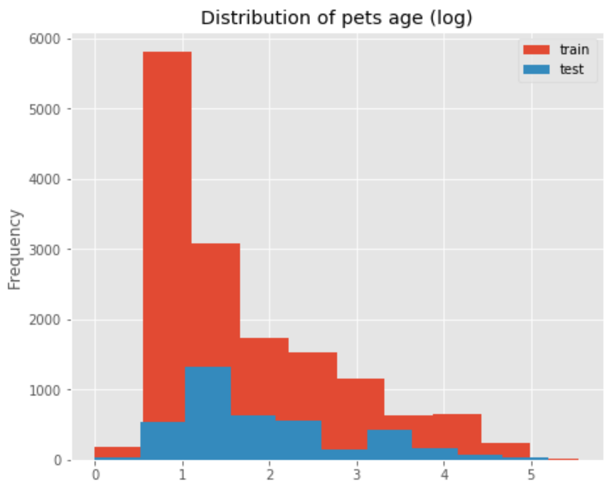
Number of dogs is greater than number of cats in the dataset. Cats are adopted a bit faster than dogs.

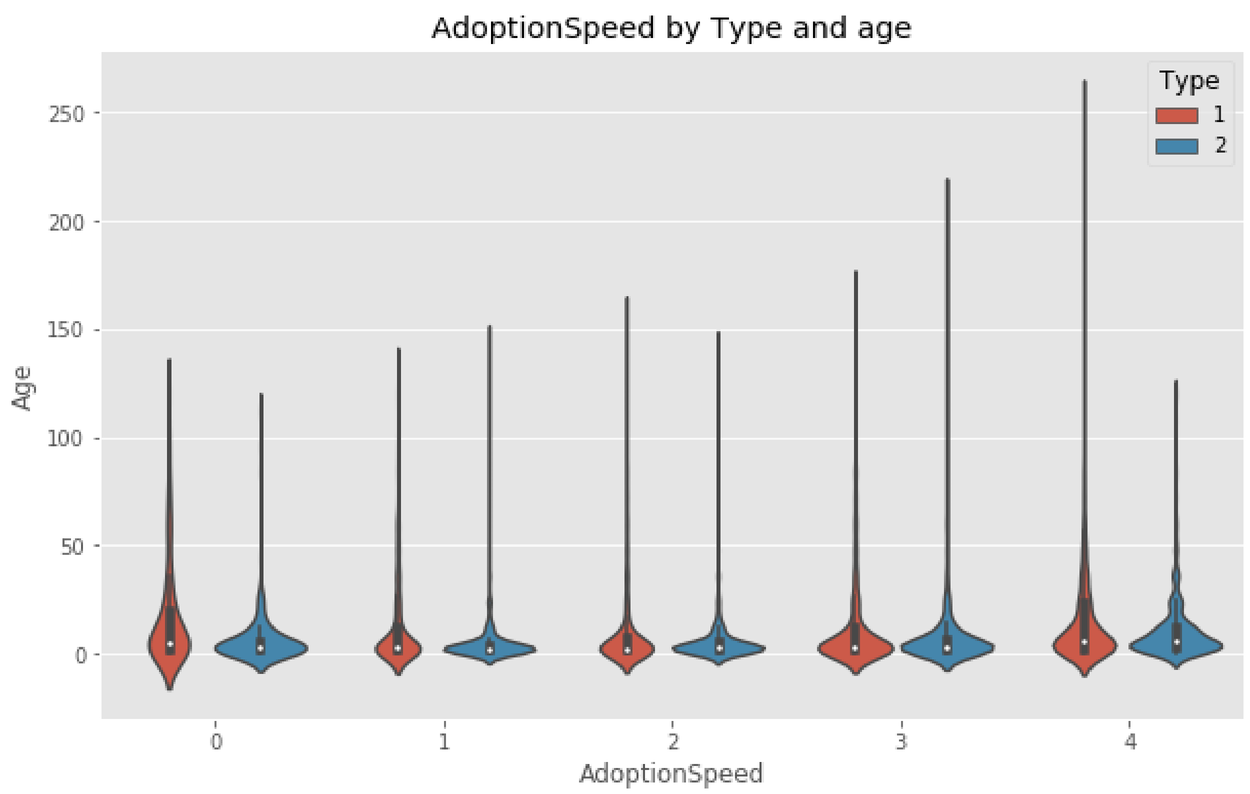


Pure Breed

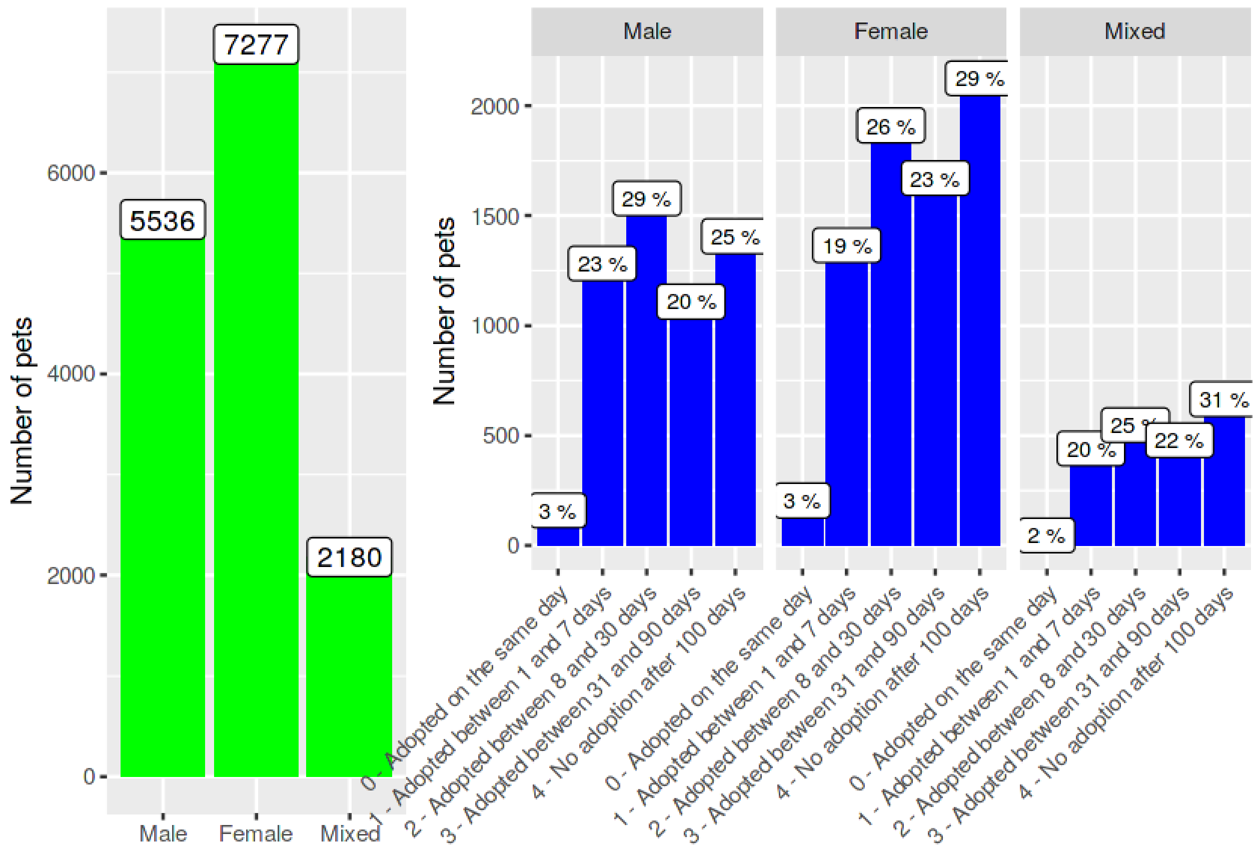
This graph shows that pure breed increase the adoptability of the pets.



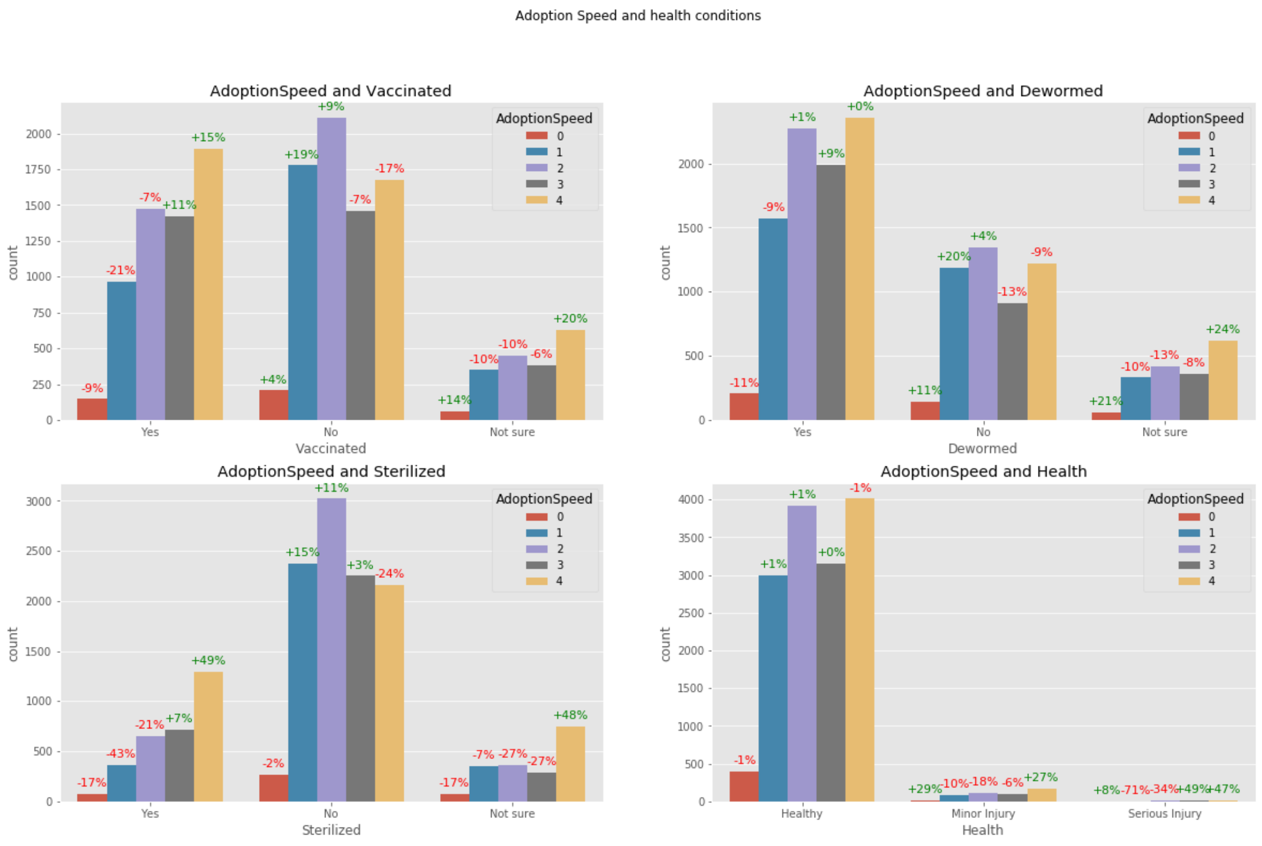




Most of the pets in the dataset are young. Young pets have larger chance to get adopted.



Most pets in the dataset are females. Male pets have better chances to be adopted.



There are four features showing health of the pets:

- 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)

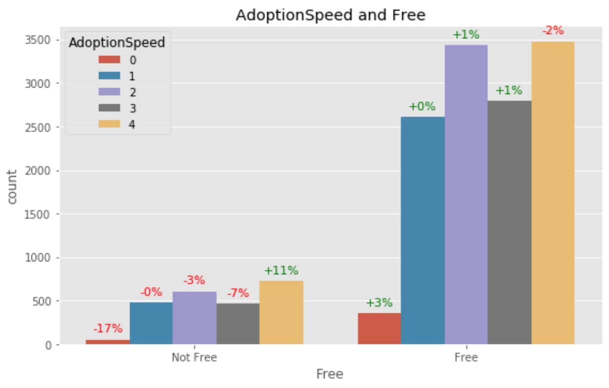
The above graph shows that:

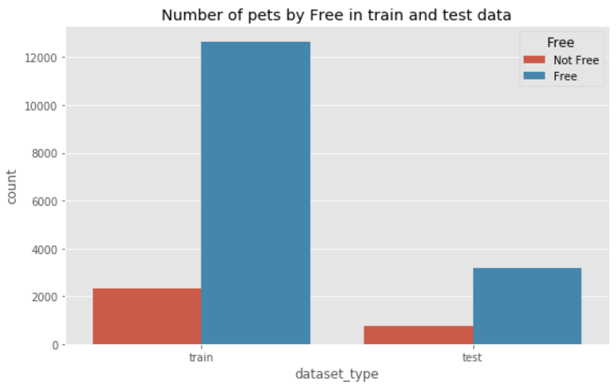
Most pets in the dataset are healthy.

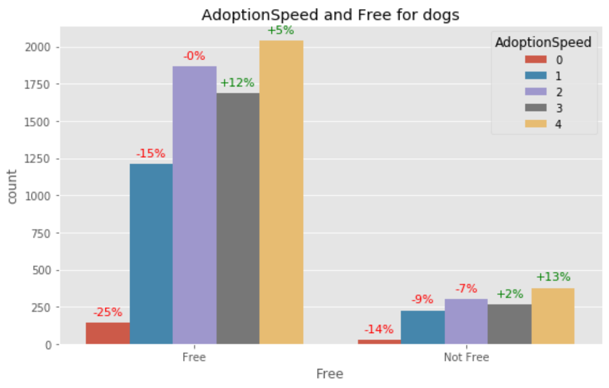
Unhealthy pets are rare in the dataset and they have small chance to be adopted.

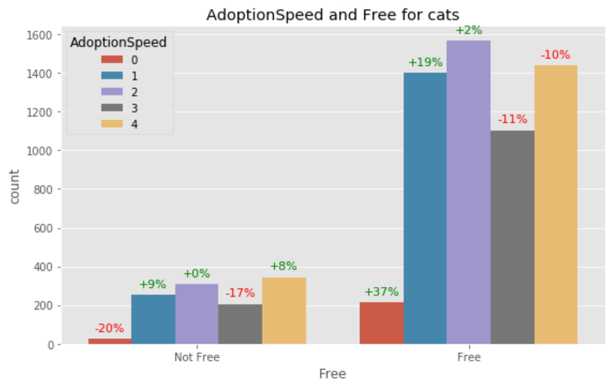
It is interesting that people prefer non-vaccinated and non-sterilized pets.

When there is no information about health condition, the probability of not being adopted is much higher.

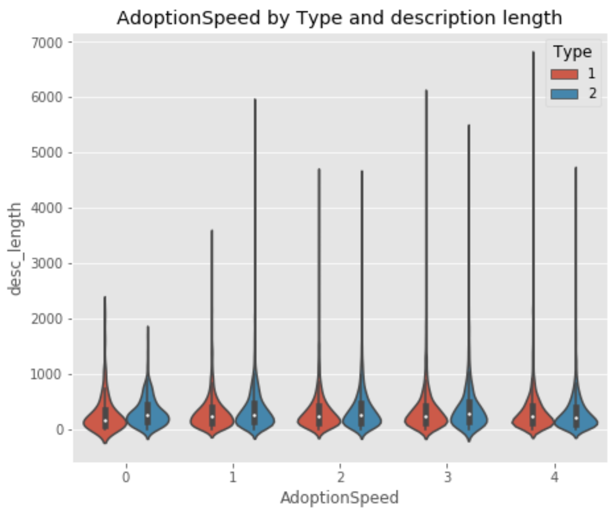


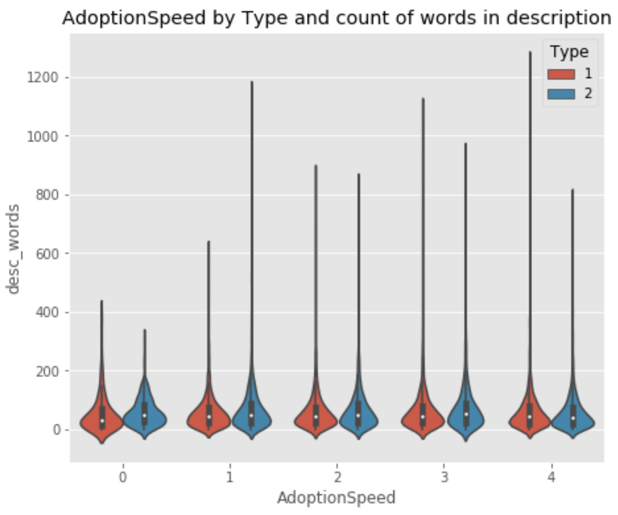




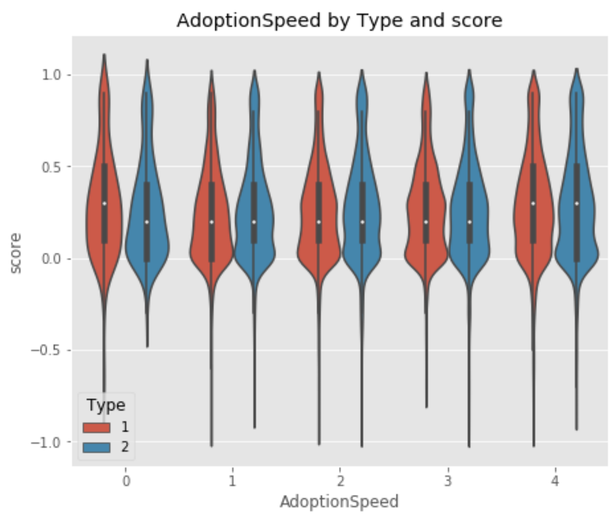


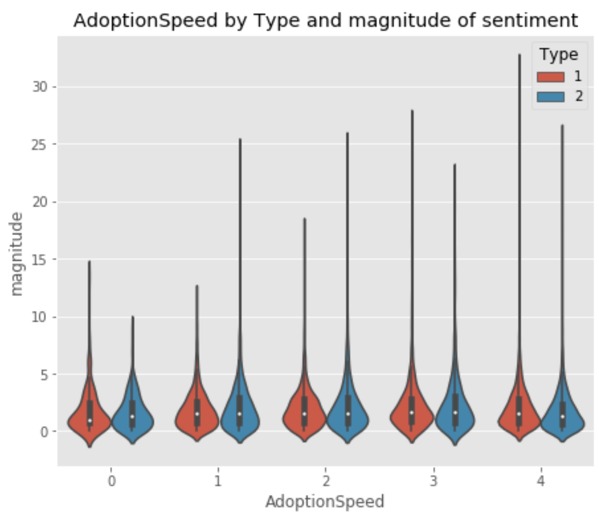
Most pets are free and it seems that asking for a fee slightly decreases the chance of adoption. Also free cats are adopted faster than free dogs.





This graph show that the longer the description of the pet, the worst its adoptability.





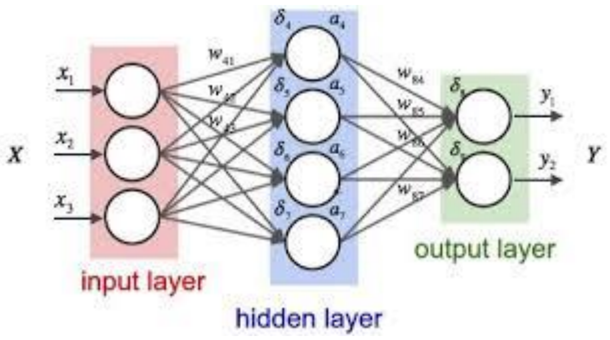
The above graph shows the sentiment analysis of the description of the pet and its effect on adoptability.

It seems that the lower is the magnitude of score, the faster pets are adopted.

❖ Algorithms and Techniques

I am going to use neural network with fully connected layers to perform both the classification and regression task.

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. For the purpose of classification, neural nets are fed with numerical features and labels and its goal is to learn how to map from the input features to the output labels.



From the above figure, neural network consists of inputs, weights, hidden layers and outputs. Figuring out the mapping from inputs to outputs is done by getting the optimal weights that activates the right neurons such that we get the right predicted labels.

So the problem is converted into an optimization problem, where we are trying to figure out the weights that minimize a loss function.

Let's say we have a loss function L(W) where W is the weights of the network. Then we need to find w that gives Min L(W).

We can solve this optimization problem analytically, but this approach is very expensive if the loss function is complicated. The other solution is to solve it iteratively by using optimization algorithm. Gradient descent is one of the most useful optimization algorithms for neural network.

Gradient descent
The main idea of the gradient descent is to iteratively update the weights in a way that decreases the loss function.

$$W \; <== W - \frac{dL(W)}{dW}$$

The loss function is a function of the weights but may not be in an explicit way. To find the gradient of the loss with respect to the weights, gradient descent make a use of the chine rule to calculate the gradient.

$$\frac{dy}{dz} = \frac{dy}{dx} \cdot \frac{dx}{dz}$$

To sum up the approach, neural network works as follows:

- Initialize random weights
- Propagate the input features through the network to calculate its loss
  This step is called 'Feed Forward'
- Update the weight to get better loss
  This step is called 'Back Propagation'
- Repeat till we have acceptable loss

Features in the dataset are numeric or categorical which is suitable to be fed in neural network with some preprocessing.

❖ Benchmark Model

I am going to use a naïve classifier, as a benchmark model, that always classifies the pet to the middle category (Category 2: Pet was adopted between 8 and 30 days (1st month) after being listed)

The results of benchmark model after performing k-Fold Cross-Validation:

- Accuracy: 0.269
- Quadratic weighted kappa: 0.0

The neural network should give results better than these.

❖ Data Preprocessing

Data Normalization

The goal of data normalization is to ensure that the statistical distribution of values for each net input and output is roughly uniform. In addition, the values should be scaled to match the range of the input neurons.

This means that along with any other transformations performed on network inputs, each input should be normalized as well.

For this reasons, I will use Min-Max normalization to normalize numerical features

One Hot Encoding

The dataset contains categorical data that is represented by unique number for each category as

Vaccinated - Pet has been vaccinated (1 = Yes, 2 = No, 3 = Not Sure)

This type of encoding allows the model to assume a natural ordering between categories may result in poor performance or unexpected results.

To solve this problem I will use one hot encoding to encode the features where no ordinal relationship exists.

Missing Data

Dataset has some missing information in sentiment data and image metadata.

The missing features are numerical, so I will replace the missing values with the mean value of that feature.

❖ Implementation

Step 1: Process Main Features

- Encode categorical features with one hot encoding
  ['Type', 'Gender', 'Vaccinated', 'Dewormed', 'RescuerID', 'Sterilized', 'State']
- There is 3 features for color
  ['Color1', 'Color2', 'Color3']
- After encoding them I created new feature from there intersection of their one hot encoding, so that the number of features is not too large
- I have done the same thing with ['Breed1', 'Breed2']

- For the sake of not having too many features which they are not important after one hot encoding step. I removed the columns of the categories that are not repeated much in the dataset.

- I use min-max normalization to normalize numerical data.
  ['Age', 'Fee', 'VideoAmt', 'PhotoAmt', 'MaturitySize', 'FurLength', 'Health', 'Quantity']

Step 2: Process Sentiment Data

Sentiment data has two numerical features with some missing rows (contains -1 instead)
[doc_sent_mag', 'doc_sent_score]

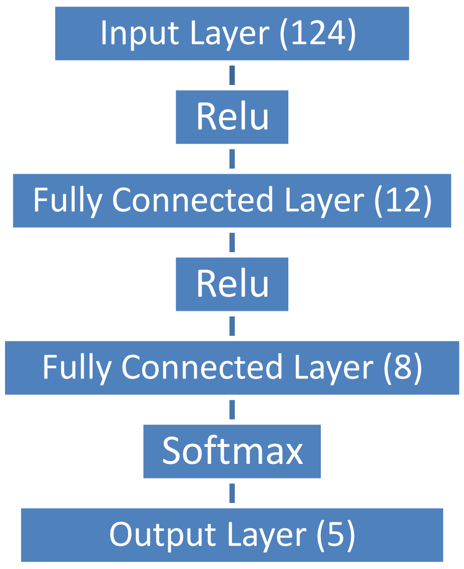
- I filled the missing cells with the mean value of this feature for the same class of the sample.
- I use min-max normalization to normalize these two features.

Step 3: Process Image Metadata

- I encoded 'label_description' using one hot encoded then removed the categories that are not very common.
- I filled the missing cells with the mean value of this feature for the same class of the sample.
- I used min-max normalization to normalize the rest of the features.

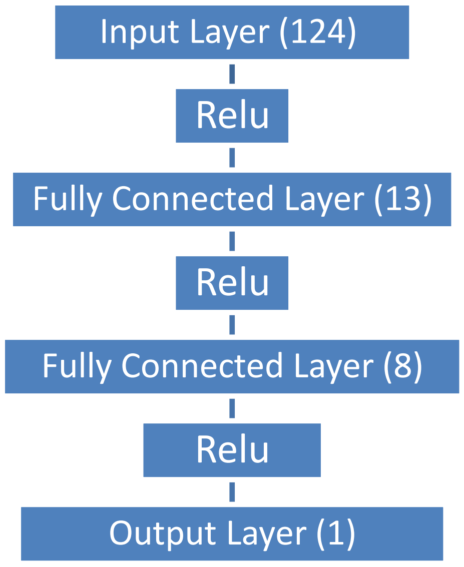
After preprocessing the data, they are combined and fed to two types of neural network (classifier and regressor)

Step 4: Classifier



Optimizer: Adam
Loss: Categorical Cross Entropy

Step 5: Regressor



Optimizer: Adam
Loss: mean squared error

Coding complications
The coding complication I faced in the implementation is that the preprocessing step takes long time. So, it is better to make use of vectorization to help speed up this process.

❖ Refinement

The initial solution I tried is to use all the features, processed as I explained earlier, and fed them to the network.

But, in this case the data has very high dimensionality which can really affect the solution.

An improved approach is to select the most important features to work with. This will decrease the dimensionality of the data focusing on features that we can learn best from them.

Feature Selection
Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Feature Importance
Feature importance is one of the techniques of feature selection, it gives you a score for each feature of your data, the higher the score more important or relevant is the feature towards your output variable.

Feature importance is an inbuilt class that comes with Tree Based Classifiers. So we can build any tree based model with the dataset and extract the feature importance.

Fortunately, Kaggle kernels provide good estimate feature importance which I will use.
Here is the link: https://www.kaggle.com/domcastro/let-s-annoy-abhishek

Using the feature importance estimate on this kernel, I will remove these features from the dataset as they are not important and the model will be better without them.

[' bounding_confidence’, 'VideoAmt', 'Health', ' Type', ' label_description', ' bounding_importance' , ' Vaccinated', ' Dewormed']

❖ Model Evaluation and Validation

Instead of splitting the data to training and testing sets, I decided to use K-Fold Cross-Validation technique to evaluate the model because the data provided is limited.

As mentioned in this blog
https://machinelearningmastery.com/k-fold-cross-validation/

"Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

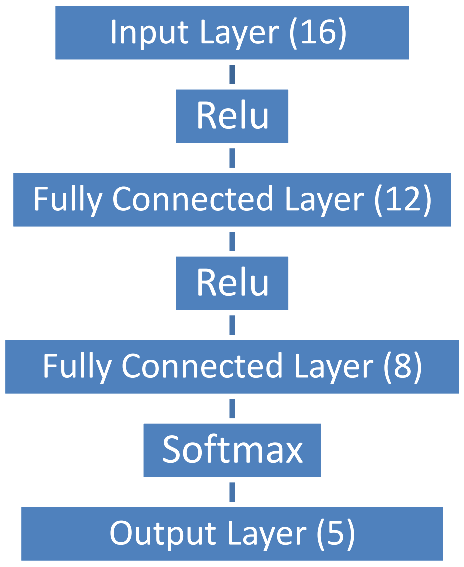
It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split."

For these reasons, K-Fold Cross-Validation is used with number of folds = 5.
Quadratic weighted kappa is calculated, for each iteration, and then the average score is taken.

For the robustness of the model, we need to try the model on unseen data (test data). For this reason I submitted the result of the test data on kaggle to see its performance which gives an indication about how much the model can generalize and it prevent getting fooled with over fitting score.

Final Model

Subset of the features is used from the data set
['Age', 'label_score', 'vertex_y', 'dominant_score', 'dominant_pixel_frac', 'vertex_x', 'PhotoAmt', 'Quantity', 'Sterilized_2', 'Fee', 'MaturitySize', 'FurLength', 'Type_1', 'Sterilized_1', 'Gender_1', 'Gender_2']
These features is preprocessed as I explained earlier, then they are fed to this classifier neural network.



This network is validated using K-Fold Technique and estimated Quadratic weighted kappa is calculated.
Model parameters are:

- Optimizer: Adam
- Training length: 100 epochs
- K (in K-Fold CV) = 5
- Loss = Categorical Cross Entropy

❖ Justification

Results of different solution I tries are as follow:

1. Training with the full dataset
   Here tabular, sentiment data, image metadata is used.
2. Training with the tabular dataset
   Here only tabular data is used.
3. Training with important data only (The Final Model)
   I dropped these features

[' bounding_confidence’, 'VideoAmt', 'Health', ' Type', ' label_description', ' bounding_importance' , ' Vaccinated', ' Dewormed']

	Classification	**Regression**	**Kaggle (cls)**
Training with full data set	QWK_score= 0.0 Accuracy= 0.28	QWK _score= 0.0 Accuracy= 0.21	0.271
Training with the tabular data	QWK _score= 0.34 Accuracy= 0.42	QWK _score= 0.0 , Accuracy= 0.21	0.293
Training with important data only	QWK _score= 0.0 Accuracy= 0.28	QWK _score= 0.0 Accuracy= 0.21	0.299
Benchmark	QWK _score= 0.0 , Accuracy= 0.27		0.0

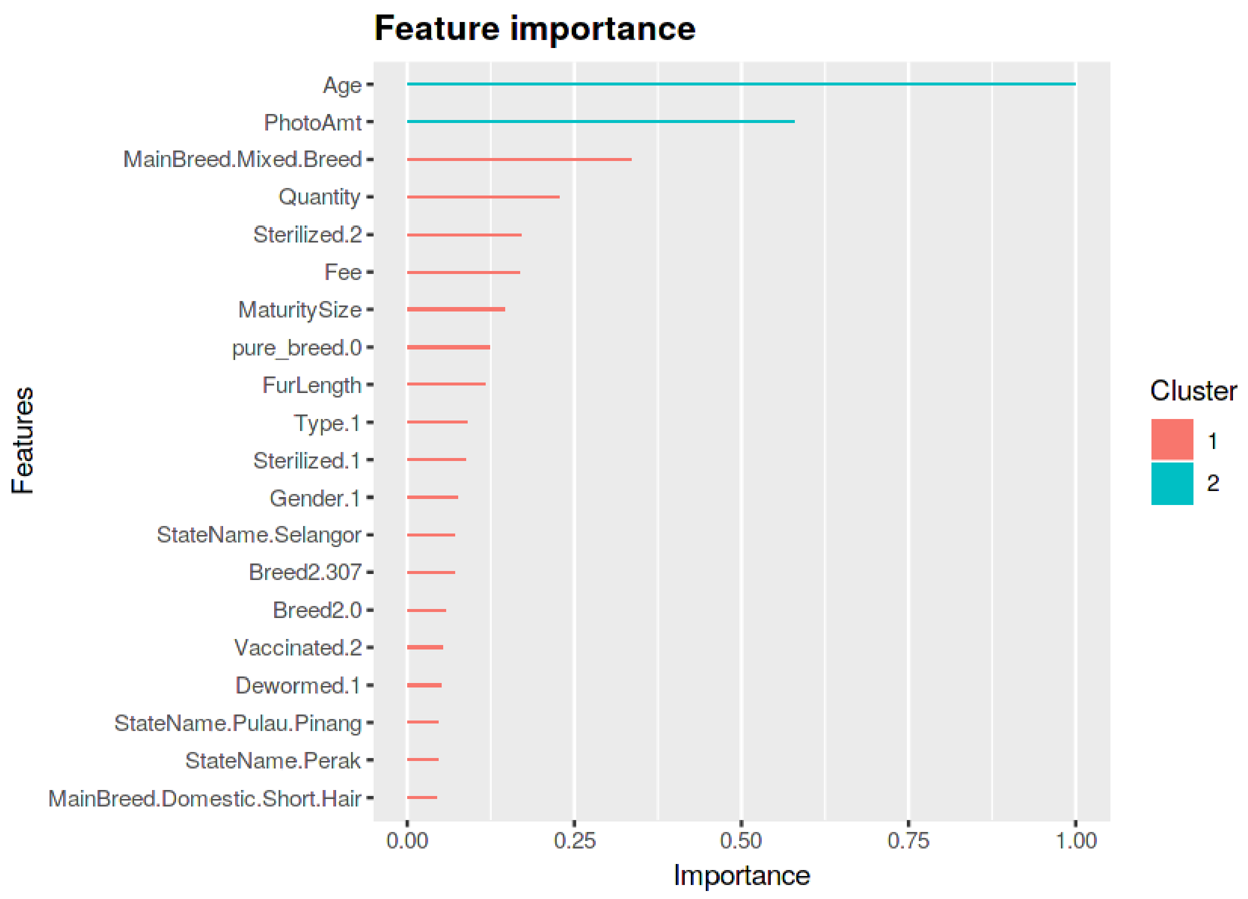
This shows that:

- Classification is really better that regression in this problem.
- There are many features in the dataset that are not important.
- Choosing the features that are used for the training data is critical to have a good performance.
- The final model results are way better that benchmark model.

❖ Free-Form Visualization

According to this article

https://www.kaggle.com/erikbruin/petfinder-my-detailed-eda-and-xgboost-baseline



We can see that some of the features have large importance in classification and some of them have low importance while the rest of the features have really neglected importance.

The data is large and full of data and this figure shows the importance of analyzing the data very well before trying to fit a model.

❖ Reflection

The goal of the project is to predict the adoptability of the pets using their profile information.
This problem can be solved easily by using classification or regression neural network.

The steps I took as follows:

- Loading and analyzing the data
- Preprocess the data
- Build the classifier and regressor networks
- Apply K-Fold Cross Validation to the model
    - Using all features
    - Using main tabular features
- Figuring out the most important features

- Apply K-Fold Cross Validation to the model with only the important features
- Test the models on kaggle

The really important and challenging part is to analyze the data very well. The data contains a lot of information that may or may not be useful. Choosing the right features is the key to solve the problem with good performance.
Fortunately, kaggle community provides very good analysis for the data which was very helpful.

Another thing I found interesting is the evaluation metric. The evaluation metric is chosen to measure the performance in a really good way such that the benchmark model which is random have a not bad accuracy on the other hand it has 'zero' score for Quadratic weighted kappa.

❖ Improvement

The dataset is full of information, so I think the model can give better performance by analyzing the data more and figuring out which features will add a good value to the current model.

Also, further processing for images and text description can be made using different techniques to extract helpful information from this data.