# Petfinder Adoption Prediction

DATA SCIENCE LAB TERM PROJECT REPORT

Abhishek Sharma | Prem Potabotti | Srinjoy Ray | Vignesh Pai B H

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. Here we have developed an algorithm to predict the adoptability of pets – to answer the specific question, how quickly is a pet adopted?

## Exploratory Data Analysis

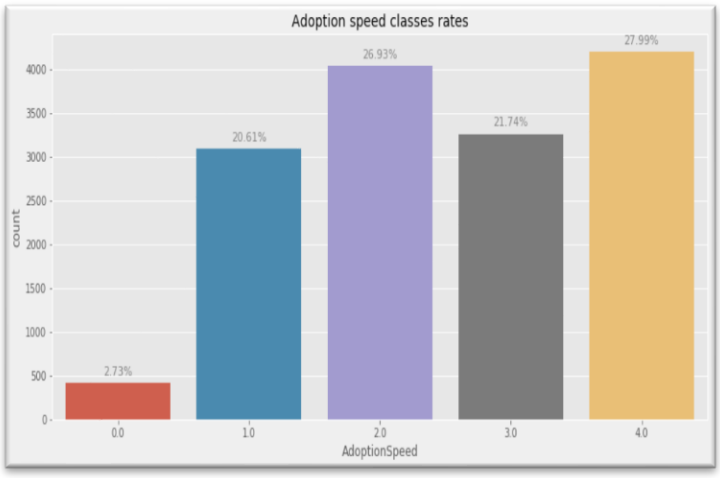
Petfinder data primarily consists of three types:

- Tabular Data (csv files)
- JSON
- Images

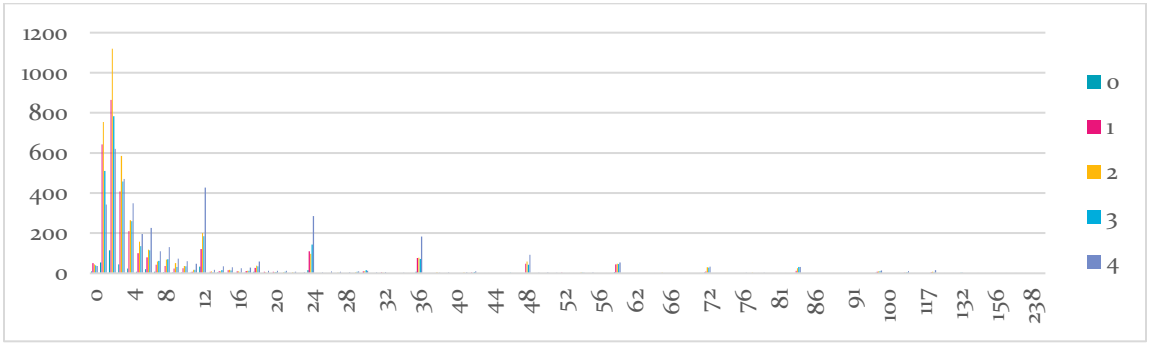
We have data and images for close to 15,000 pets in train data, and around 4,000 pets in test data. We have detailed information regarding each pet, like Animal Type, Name, Age, Breed, Gender, Colour, Appearance, Health Condition, Adoption Fees, Description, and Adoption Speed. We also corresponding image and video files, as well as sentiment data for each pet.

It is observed that about 46% of the pets in training data are cats (minority) whereas they account for 53% of the animals in test data (majority). The reverse is true for dogs. It should be noted here that all the pets in this dataset are either cats or dogs.

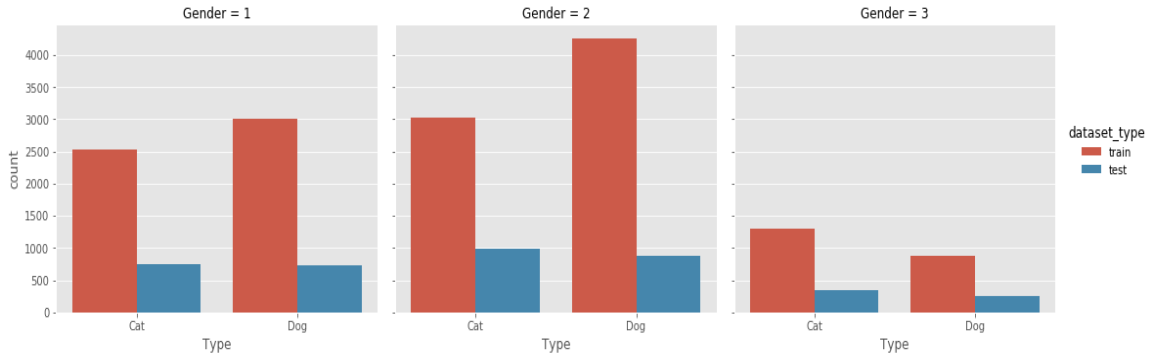
This plot shows us the current adoption speed class against number of pets adopted. Few lucky ones get adopted as soon as they are put up on the site, although it is a small number at 2.73%. However, there are still many which are not adopted at all. Our model and analysis are aimed at helping them all find their forever homes.




Pet names like "Mimi“, “Tiger” and “Lucky" are quite common. Often people simply write "Kitten", "Puppies“, etc. on the listing. Many times, the colour of pet is written in place of Name (“Black”, “Grey”, etc.), sometimes gender (“Boy”, “Female”, “Girl”), and even adjectives in some cases (“Little”, “Tabby”, “Baby”, “Mix”). Less than 10% of pets don't have names, but they have a higher possibility of not being adopted.




Most young pets are adopted, and quite fast. Most listed pets are less than 4 months old, with a huge spike at 2 months. Also, it seems that a lot of people do not input the exact age of a pet and convert the approximate number of years into months.



Male pets are preferred and adopted sooner. Unspecified gender dramatically reduces the chances of getting adopted, for both cats and dogs.

The data suggests that mixed breed pets are adapted faster and in larger numbers, especially in case of cats.

Maturity size does not seem to have a very prominent effect on adoption speed. Medium sized pets are most common, and they are at a slightly higher risk of not being adopted.

Almost all pets are recorded as healthy. Pets with minor injuries are rare, and they aren't adopted well. The number of pets with serious injuries is negligible. People prefer non-vaccinated pets. It could be posited they want to bring their pets to vets themselves. Non-sterilized pets are preferred. When there is no information about health condition, the probability of not being adopted shoots up. The following information are available:

- 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 – (1 = Healthy, 2 = Minor Injury, 3 = Serious Injury, 0 = Not Specified)

The adoption rate decreases with increasing age of the pet. This could be explained by the fact that most older pets are rescued animals, and they require specialized care, which deters many prospective adopters. Also, most people prefer to adopt puppies or kittens.

Most pets can be adopted for free. However, it seems that an adoption fee slightly decreases the chance of adoption. Free cats are adopted faster than free dogs.

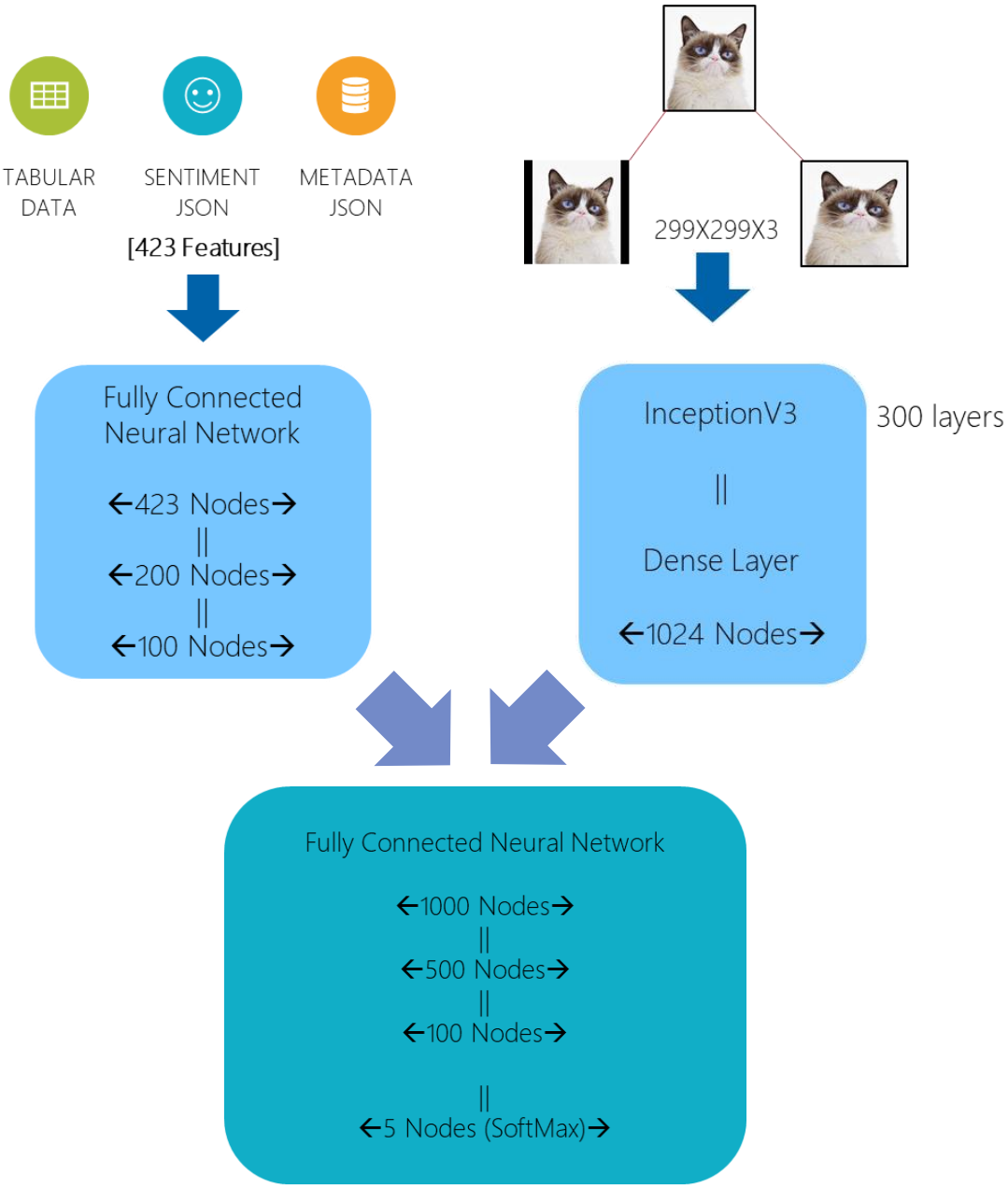
Each pet can have up to 30 photos. However, the number of photos a pet has does not seem to influence its chance of getting adopted soon.

All the plots and charts are available here: http://bit.ly/2UYocjW

## Neural network with Multiple inputs

In the initial approach, the JSON files (sentiment and metadata) are flattened into CSV files, and tabular data is extracted. It is then combined with the train.csv and related data (CSV) provided in the dataset. This results in 423 columns or attributes, including numerical and categorical. This data is fed into a fully connected neural network with 423 input nodes, and hidden layers of 200 and 100 nodes.

In a parallel path, image files are prepared for use by either padding or stretching. Initially, the InceptionV3 image recognition model was used, which takes image inputs of dimension $299 \times 299 \times 3$, and consists of about 300 layers. Since the available images are of varying dimensions, our options are to scale each image to $299 \times 299$ pixels and then pad the smaller side with black, or to stretch the image to $299 \times 299$ without preserving original ratio. We noticed that both approaches give roughly similar accuracy. The output of InceptionV3 is passed through a dense layer of 1024 nodes.




The two neural networks mentioned above (fully connected neural network with 423 input nodes, and InceptionV3 model) feed into a fully connected neural network with multiple

hidden layers (1000 nodes, 500 nodes, 100 nodes) and the last layer consisting of 5 nodes, for the 5 classes denoting different adoption speeds. This produced an accuracy of 24-25%.

This model posed a possibility of overfitting due to so many hidden layers. This could have resulted in terrific training accuracy, and inconsistently poor validation accuracy. To avoid this possibility, dropout rates were introduced into the last fully connected neural network hidden layers. Each of the layers were given a dropout rate of 0.4. This increased the accuracy to around 26%.

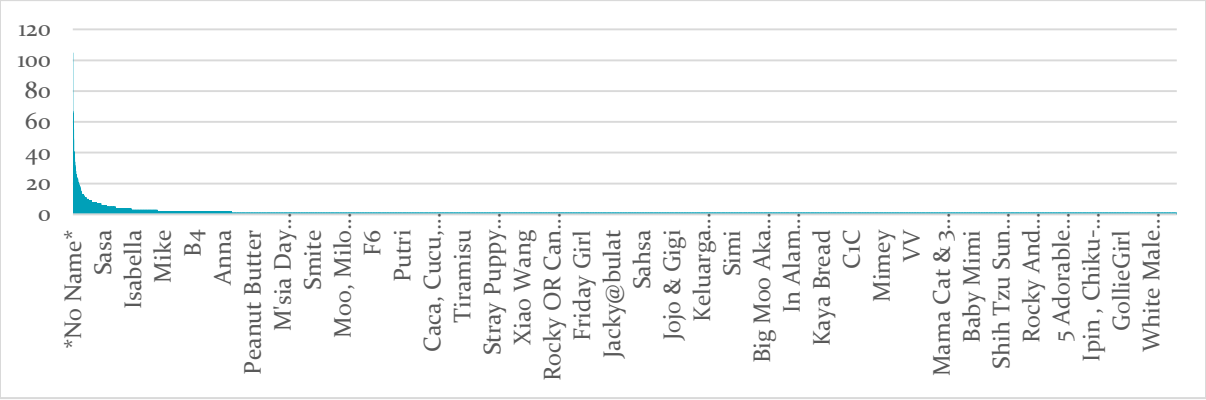
Hidden layers were constrained and modified to tweak the model in an attempt to increase the model accuracy. The combining neural network was configured at 500 nodes with a dropout rate of 0.4, followed by a 5 node SoftMax layer. Accuracy was improved to 30-32%.

# Alternate Approach

This approach consists of dealing with images, tabular data, and text data separately. After applying various algorithms to individual types of data, the results are ensembled.

## Readying the data

This step involved making the tabular data ready for various treatment. Firstly, features that were not relevant for classification were discarded. The features that were discarded were {'Description','Name','RescuerID', 'PetID', 'State'}. Description does not add any value to the predictor since it is a text data. Name is of no use. There are many datapoints where the




name entry is empty. It is but natural since these pets are up for adoption and some of these pets may be stray. It can be seen from the histogram that no valuable information can be extracted from names variable such that it can be used for tabular data analysis.

RescuerID does not add any value to a pet's adoption rate. Thus, we ignore it. We ignore PetID and State features for similar reasons.

After removing these variables, direct application of below mentioned algorithms was done. After checking that the results were not as impressive, feature engineering was done.

Feature engineering was done based on interaction effect using correlation plot, and basic intuition of what a pet owner would want.

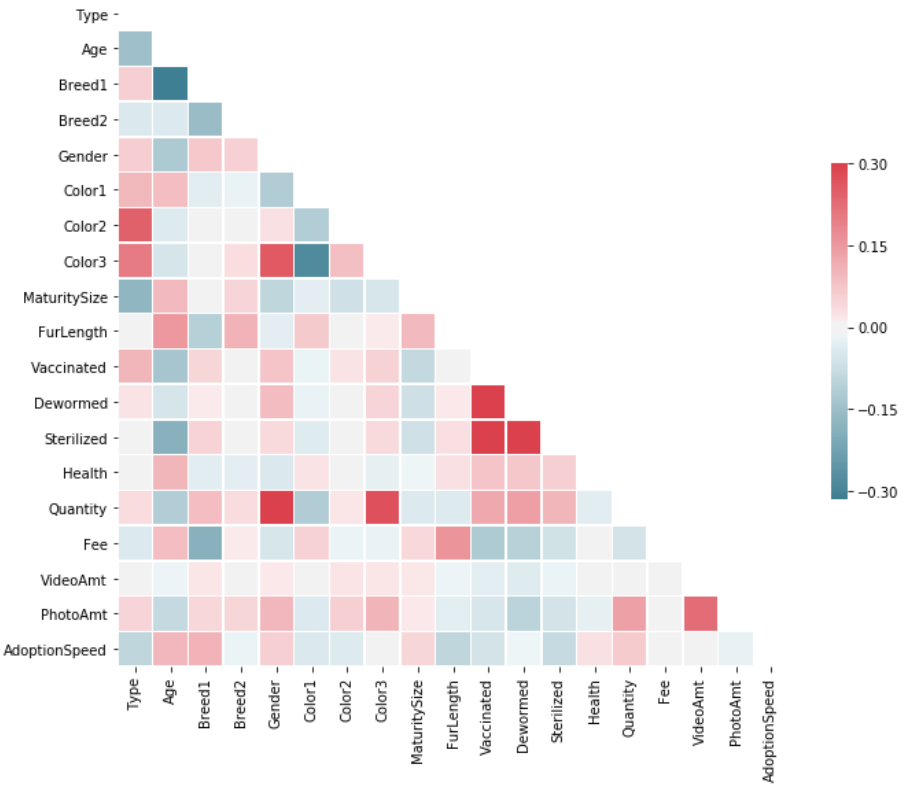
## Feature Engineering

1. Feature engineering using correlation plot:

- Check the pair of variables where the correlation is high. For e.g. Breed1 and Age.
- A new feature, data['Breed1']*data['Age'], is created
- Other new features are data['Fee'] × data['FurLength'], data['FurLength'] × data[Age], data['Breed1'] × data['Fee']

2. Feature engineering based on buyers' perspective:

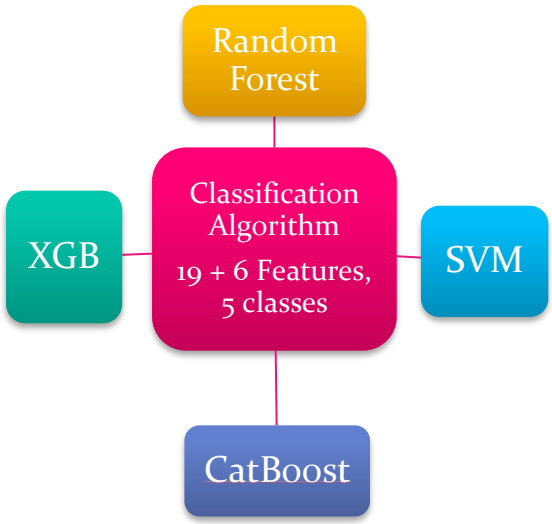
- A prospective owner would want his or her pet to be vaccinated, Dewormed, and Sterilized. If anyone of the parameters is marked negative, then a buyer would probably repel to adopt the pet. So, a new feature, data['Vaccinated'] × data['Dewormed'] × data['Sterilized'] is created.
- Adoption speed of a pet would also be affected by the number of photos and videos that are uploaded. Thus, a new feature, (data['VideoAmt'] + 1) × (data['PhotoAmt'] + 1), is added.




## Approaches used on tabular data

1. Random Forest for Dog and Cat Data
2. XGBoost for Dog and Cat Data
3. CATBoost
4. Support Vector Machine

| | Train | Validation |
|---|---|---|
| Dog_RF | 95% | 44% |
| Cat_RF | 95% | 37% |
| Dog_XGB | 42% | 41% |
| Cat_XGB | 35% | 34% |
| Dog_Catboost | 27% | 22% |
| Cat_Catboost | 19% | 18% |
| Dog_SVM | 38% | 34% |
| Cat_SVM | 35% | 31% |




*Comments on Results of various approaches*

1. Random Forest: Random Forest is most promising among the four but is overfitting. As can be seen from the table above, the model that is overly well on training data but not on test data, because distribution of test data differs from train data. Hyperparameters used for Random Forest are {n_estimators=100, random_state=0, max_depth = 17, criterion = "gini", max_features = "sqrt"}
2. XGBoost: Extreme Gradient Boosting performed better than Random Forest in terms of overfitting. But it cannot be conclusively said that it is performing better than Random Forest on absolute basis. So, hitherto, Random Forest was considered better than XGB. Parameters that were used for XGBoost were default parameters. Parameters were not changed a lot as overfitting was not found. Only a minor increase in variance resulted in poorer results. Thus, it was not opted for.
3. Catboost: Considering the high number of categorical variables among all the variables, Catboost was expected to give better results than the above two algorithms. But it performed very poorly as can be seen from table. Even the baseline accuracy was very poor. Thus, it could not be considered for further analysis.
4. SVM: Support vector machines are one of the most promising classifiers. Nonetheless, for high number of categorical variables, such as in Petfinder data, it did not surpass the performance of XGboost and Random Forest. Parameters that were used for SVM default parameters.

Additionally, Boruta was used to check if important features could be extracted. Boruta's output, out of all the categorical and continuous variables, were only 3 variables. The idea was rejected without trying. The code, however, is not deleted and is only commented out.

## Structure and style of code

Libraries.py (contains code of all the classifiers and encodings, written as functions)

Classifier.py (imports libraries.py, and calls a particular classifier function in libraries.py)

## Approaches used on images

1. Classification (Cost function: Log Loss Function) using pretrained models
2. Regression (MSE) using pretrained models

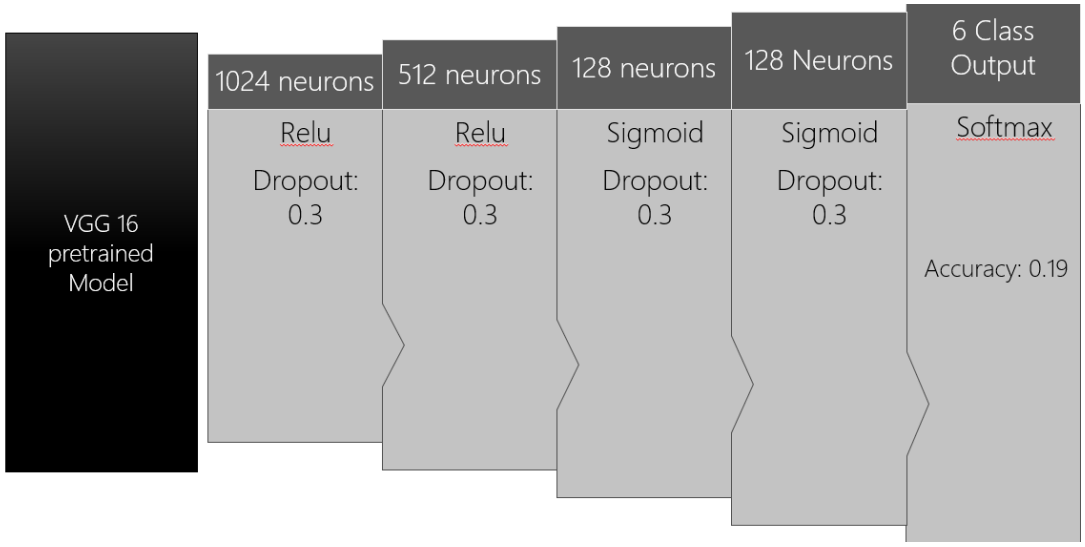
As the number of images was 58,318, training a model from scratch would take massive computation. Thus, it was decided to use pretrained models from Keras.

Various pretrained models that were used were VGG16, VGG19, Resnet-50. Densenet and Inception were used but not directly though. Images were combined with the tabular data in the usage of Densenet and Inception.

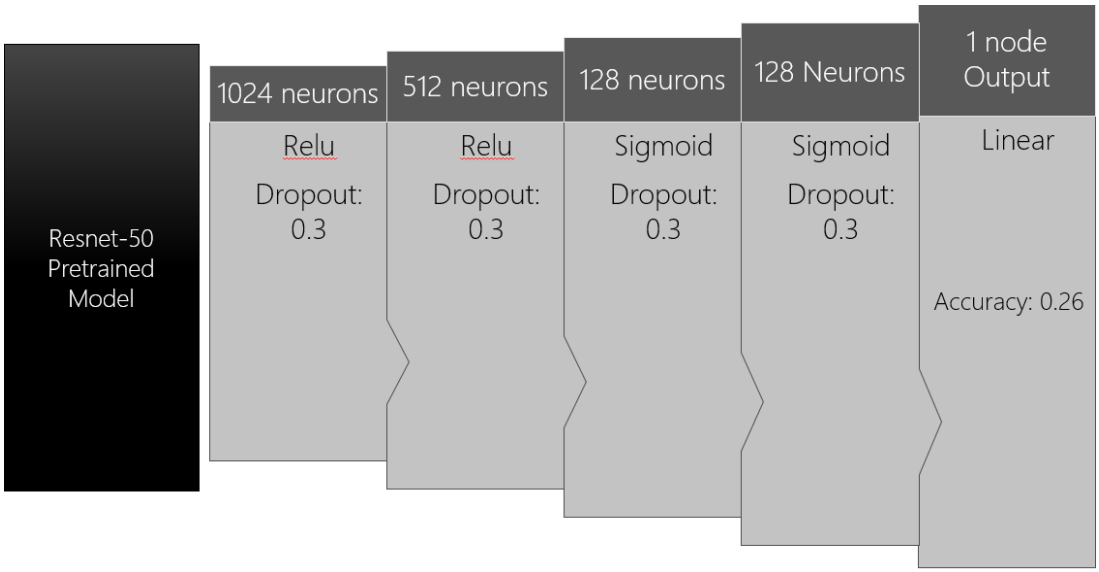
### *VGG16 + Neural Network*

VGG16 was built by Visual Geometry Group, University of Oxford and has 16 layers of convolution, max pooling. This model outputs a Flattened vector of image pixels. After which this is fed as input to multiple hidden layers. As shown in the diagram above, hidden layers have 1024, 512, 128, 128 neurons in 4 layers respectively. And the final output layer has six classes and is a SoftMax layer, i.e. probabilities of being in an individual class.

Accuracy for this model turned out to be 19%. So, VGG19 model was used. VGG19 is also developed by University of Oxford. This model improved accuracy slightly to 21%. This was insufficient as this is the bare minimum accuracy for a 5-class classification. This accuracy is equivalent to randomly selecting a class for an image, which translates to – not having a model is better than having such model.

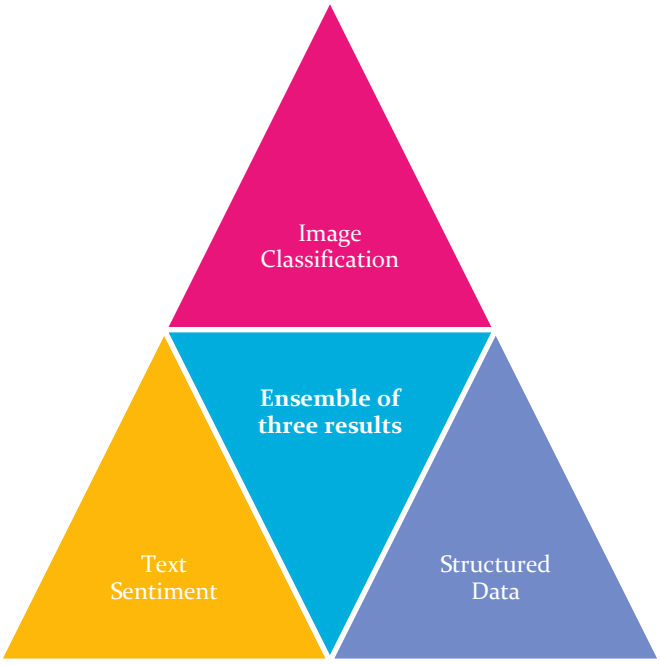


*Resnet50*




Resnet-50 pretrained model has been shown to work well on "cats and dogs" kind of data. But that accuracy was good only for binary image classification of cat vs dog. Thus, it was not very surprising when this model gave an accuracy of 26% accuracy for 5 class classification. This 5-class classification could be viewed as cuteness of the pets. Rest of the Neural Network architecture was same as VGG16 for this as well.

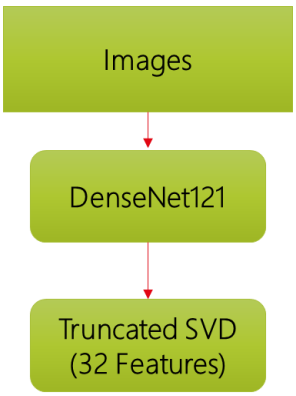
## Summary of this approach



# Another Approach

Since the approaches until now did not provide very fruitful results, as a team we realized that we must change the approach that we follow. Until now, we were feeding all the data available with us to the model and hoping that these models will somehow pick up the important features from the entire available data. This strategy clearly did not work for us. Hence, we decided to carefully select only the important features from the amount of data available to us. We will consider each kind or data available to us and look at the approach that we followed to filter the right features.

## Images



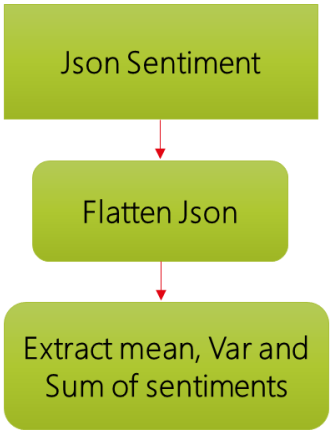

Images are very high dimensional data; in our case we have images of various size. Even an image with 224X224 pixel has 150528 features. This is extremely high dimensional data. In order to reduce the dimension of this data, we decided that we can feed the image to a pre-trained neural network. Any well-trained neural network would have already identified the important features, therefore removing the final SoftMax layer would mean all these important image features would be available for us to feed to other models. For this reason, we selected the DenseNet121 network that was trained on ImageNet Data. Without the final SoftMax layer this network provides 1000 different features as output.

Although a reduction from 150528 to 1000 is a large reduction, it still is not a sufficiently small number of features for us as we have only 14000 odd pets. Therefore, next we looked for ways to reduce feature size further. A common way this we usually reduce features is through SVD. Therefore, we applied a truncated SVD to extract 32 most representative features from these 1000 features.

## Sentiment Json Files

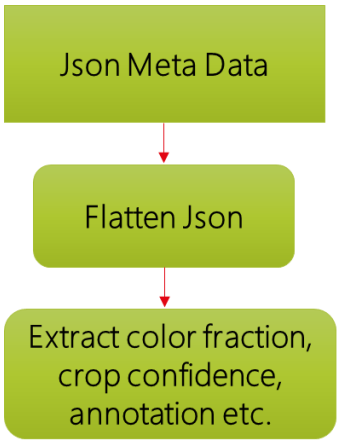
In our earlier approach we Fattened the entire json file and then we converted it to a CSV and fed all these features to the various models. This obviously added noise to our data set, this time we decided to feed in only handpicked features from these sentiment files to our model.

At first just like our standard methodology, we flattened the Json file. Once the json file was flattened, we aggregated values. Each pet can have 1 to 5 sentences in its description. We calculated the mean, the sum and the variance of sentence sentiment scores for each of the pets. This ensured that we have a constant number of fields for each pet instead of having varying number of fields depending on the number of sentences in its description.




## Metadata Json Files

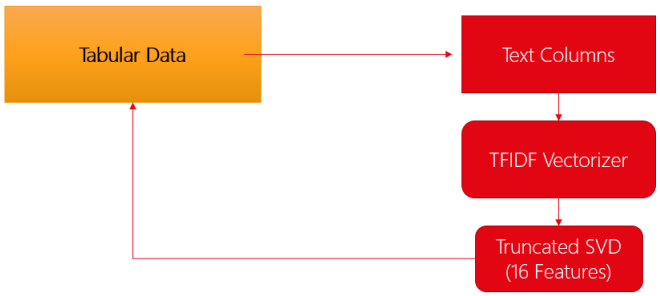
Just like the sentiment scores, we had json files of the metadata as well. However, features were not as clear as the features in the Sentiment files. Each image of a pet had an accompanying Metadata file. Each metadata file provided features like highest density colour. Pixel density of each colour, annotation details if any, crop confidence etc. Again, aggregate vales were taken from the flattened json files and these aggregates were fed to the subsequent models.




## Tabular Data

The data obtained in the previous steps after processing were merged with the tabular data that was already provided. However, before we could proceed to the final step of applying various models to the data, we had to extract some useful features from the text data that was available to us within the tabular data that we have.

One good feature that is used by most basic IR systems that are available is the TF-IDF. TF-IDF stands for term frequency inverse document frequency. In this approach, the number of time log of the frequency of every word is calculated and it is divided by the log of the number of documents that contain this word. Therefore, a greater number of times a word is repeated in a document the higher the value of the entry in that dimension. Additionally, a rare word, i.e. a word that is gets a very high value as well.




## XG Boost Regression

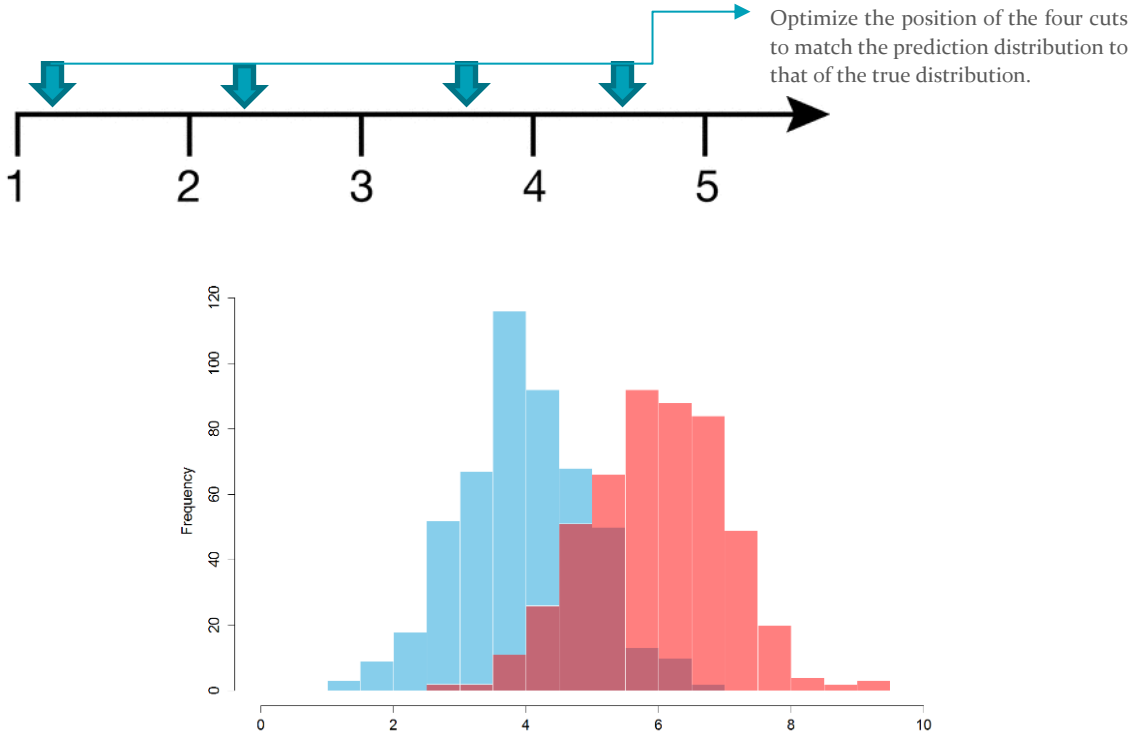
We came across one interesting method applied on this data set in one of the kernels, it strangely used XG Boost regression instead of using XG Boost classification. This was confusing to us at first, however on careful study of the methodology we understood the advantages of this method.

This method took advantage of the fact that the output although was categorical, it was ordinal in nature. The model is trained on the entire data set by treating the output as a numeric rather than treating it as a categorical variable. Therefore, when a data is fed to this trained model for prediction, it predicts a continuous value between 1 to 5.

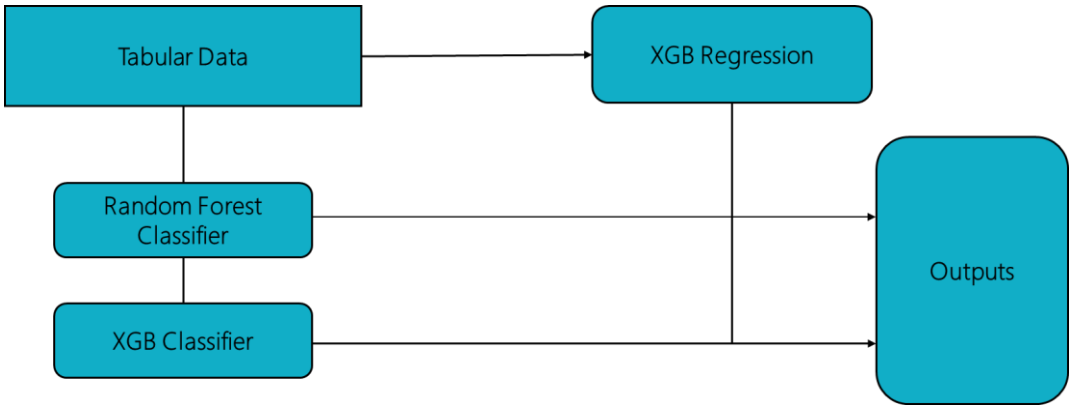
We have an existing distribution of classes available for training set. Prediction also has a distribution of its own. However, for the prediction to be the best, the distribution of the prediction and the actual distribution should be the same. In this method, we make 4 cuts in the continuous range of prediction of the model. A cost function is written which shows the squared distance between the heights of the histogram for every value.

Next an optimization algorithm is applied to make these two distributions match thereby getting the best points to get the cut.

This method is called the weighted kappa method where which is commonly used to bring to scale the ratings given by different reviewers. Here we can think of the output as a rating provided to every pet. Therefore, there is a rating that is 'absolute truth' then we also have a rating that is by the model. The idea behind this kappa function is to make these two distributions match to get the best results.




## Ensemble approach 1




In this method, we applied two classifiers, Random forest classifier and an XGBoost classifier on the feature set that was extracted earlier. Now that we have two prediction which match to a certain extent, we had to identify a way to solve conflicts. We at first applied the XGB regression technique described earlier and this would give us a prediction. Next, we gave scores to each prediction of Random forest and XGB classifier. These scores to each prediction was nothing but the distance of the prediction from the prediction of XGB regression.

The accuracy went up by a significant margin after applying this technique. However, it was not good enough. Therefore, we moved on to the next approach.

## Ensemble approach 2

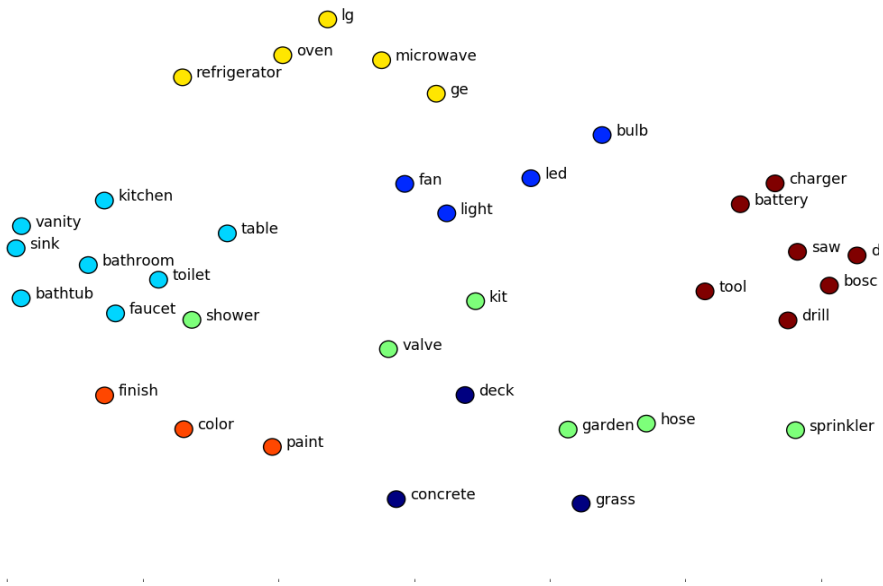

In this method we trained 5 separate classifiers which were specialists in identifying one specific class each. Finally, we ensembled the outputs from these five predictions using the same approach of weighting the vote by the inverse of its distance from the XGB regression prediction. This significantly improved the accuracy of our predictions.

## Word Embeddings

Every word used in a language can be represented by a set of real numbers (a vector). Word embeddings are N-dimensional vectors that try to capture word-meaning and context in their values. Any set of numbers is a valid word vector, but to be useful, a set of word vectors for a vocabulary should capture the meaning of words, the relationship between words, and the context of different words as they are used naturally.

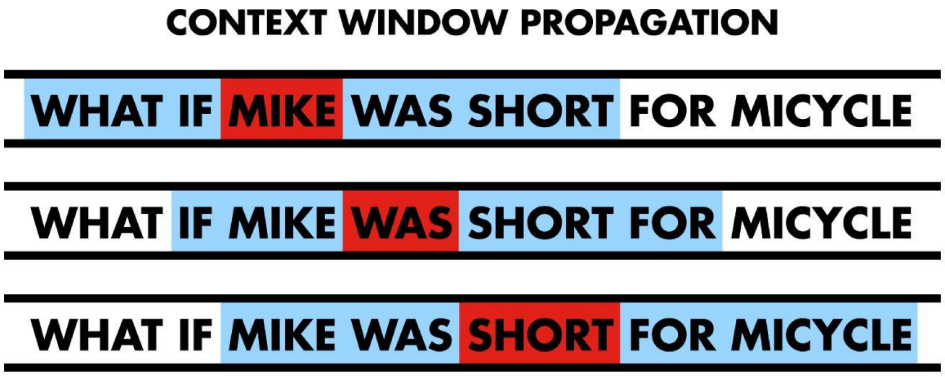
### Global Vectors (GloVe)

In NLP, global matrix factorization is the process of using matrix factorization methods from linear algebra to reduce large term frequency matrices. These matrices usually represent the occurrence or absence of words in a document. Global matrix factorizations when applied to term frequency matrices are called Latent Semantic Analysis (LSA). Local context window methods are CBOW and Skip–Gram.




## fastText

It is another word embedding method that is an extension of the word2vec model. Instead of learning vectors for words directly, fastText represents each word as an n-gram of characters. So, for example, take the word, "artificial" with n=3, the fastText representation of this word is <ar, art, rti, tif, ifi, fic, ici, ial, al>, where the angular brackets indicate the beginning and end of the word.




This helps capture the meaning of shorter words and allows the embeddings to understand suffixes and prefixes. Once the word has been represented using character n-grams, a skip-gram model is trained to learn the embeddings. This model is considered to be a bag of words model with a sliding window over a word because no internal structure of the word is taken into account. As long as the characters are within this window, the order of the n-grams doesn't matter.

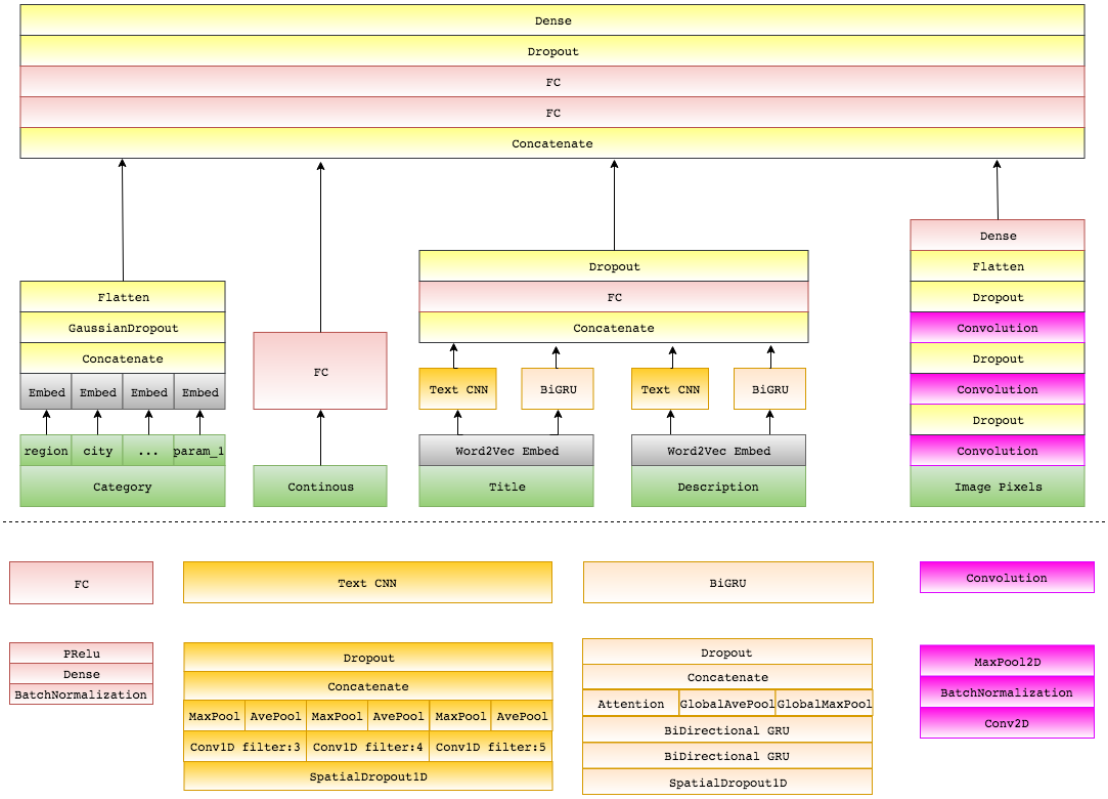
Three methods have been applied.

Training NN1 is done with MSE loss using Regular image features and a TF-IDF text representation.

Training NN2 is done with MSE loss using Regular image features and a non-trainable 300d crawl embedding for text.

Training NN3 is done with SmoothL1 loss using Regular image features and a trainable 200d GloVe embedding for text.

### Architecture for Word Embeddings:




The word vectors obtained after applying word embeddings on description and Title fields were concatenated with the extracted image features and tabular data into the neural network. The final network is the composite of text image and tabular data.

## Future Scope

- Detecting Cuteness (for Cat and Dog) using Fisher Vector and Gaussian Mixture Model (https://dl.acm.org/citation.cfm?id=2655046)
- Very focused capturing of pixels of a particular section rather than uniform max pooling.
- Better handling of categorical variables apart from applying CatBoost
- Bayesian Approach