PetFinder.my

1. INTRODUCTION

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.

We came across the Pet Finder dataset on Kaggle competition and decided to use it for our class project. 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. This dataset will help answer the questions which will guide shelters and rescuers around the world on improving their pet profiles appeal, reducing animal suffering and euthanization.

The dataset is about of 2.7 GB of size and has 24 variables with a mix of continuous and categorical. Main dataset contains all important information about pets: age, breed, color, some characteristics and other things. The dataset also includes images and sentiment for each pet. There are few features with missing values in the dataset i.e. images, sentiment, this will be addressed appropriately during project execution. This is a classification problem and the target variable of the dataset is Adoption speed. The value is determined by how quickly, if at all, a pet is adopted, and it ranges from 0-4 where 0 being the fastest and 4 being the slowest adoption rate. The detail about the dataset is shown in the link below:

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

The questions we are focusing on to explore, analyze and predict the target variable are

- Data exploration - Exploring the features and their interactions – specifically, what are some of the most important factors impact the adoption rate.
- Comparing distribution of features in train and test data
- Developing algorithms to predict the adoptability of pets – specifically, how quickly is pet adopted?
- Trying various types of feature engineering
- Trying various models and comparing the accuracy of each model on test data

We used Anaconda Python distribution for preprocessing and data exploration tasks and Google Cloud Platform (GCP) for model training and testing.

2. DATASET

The PetFinder dataset has many different features about the pet and the listing for that pet. This dataset is more interesting and approachable problems available on Kaggle right now for few reasons. The major one being the variety of data given to solve the problem. The given features are of most commonly found types of data: tabular, text and images. The dataset is relatively small which enables to iterate relatively quickly and find new insights which will contribute to the solution. Below are the data fields and their description in the dataset.

2.1 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
- Breed2 - Secondary breed of pet, if pet is of mixed breed
- Gender - Gender of pet (1 = Male, 2 = Female, 3 = Mixed, if profile represents group of pets)
- Color1 - Color 1 of pet
- Color2 - Color 2 of pet
- Color3 - Color 3 of pet
- 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
- RescuerID - Unique hash ID of rescuer
- VideoAmt - Total uploaded videos for this pet
- PhotoAmt - Total uploaded photos for this pet

2.2 Class Variable

The question we are trying to answer using the dataset is how long it will take for a pet to be adopted. The class variable is captured by the AdoptionSpeed column in the dataset. The column has five values 0, 1, 2, 3, 4. The guide to what these numbers correspond to is below.

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).

We are treating this as a 5-class classification problem instead of Regression problem.

2.3 Tabular Data

There are 14,993 instances of information of pets i.e. type, age, breed, gender, color etc. The descriptions were analyzed using Google's Natural Language API providing sentiments and entities. The data is classified as below tabular data and split those further into two categories shown in **Figure 1. Tabular data**: Categorical and Numerical. Numerical features are those that are continuous numbers with their relative values holding meaning. A categorical or discrete variable is one that has two or more categories. There are two types of categorical variable, nominal and ordinal. A nominal variable has no intrinsic ordering to its categories. For example, gender of pet is a categorical variable in the dataset having three categories. 1 = Male, 2 = Female, 3 = Mixed with no intrinsic ordering to the categories. An ordinal variable has a clear ordering. For example, AdoptionSpeed as a variable with 5 orderly categories on how long it took for a pet to be adopted.

Tabular Data	
Categorical	**Numerical**
Type	Age
Breed1	Maturity Size
Breed2	Fur Length
Color1	Vaccinated
Color2	Dewormed
Color3	Sterilized
State	Health
	Quantity
	Fee
	VideoAmt
	PhotoAmt

Figure 1. Tabular data

2.4 Text

The dataset has two text-based fields in the main file and few more available as supplemental files. The two text fields in the main CSV file are Name and Description. The names are likely a special case where you might not want to pull out any of these models though as the input is very simple. The possible feature that could be extracted from Name field is simply do they have a name or not.

2.5 Images

In addition to all the other information in the dataset, we also have access to all the images used in the listings. These are stored as separate JPGs with the petID and then the index of the picture making up the filename. Not all the pets have the images some have no pictures and others have one or many. The first indexed picture is the profile, so it likely has the most importance. We are using the images in our models by doing basic statistics such as image size, width and height and focusing on the quality of the pictures such as dullness, whiteness, pixel width and blurriness.

2.6 Supporting Data

There are additional files for each petId, Image metadata where the images are run through Google's Vision API providing analysis on Face Annotation, Label Annotation, Text Annotation and Image Properties. 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. Sentiment Data where each pet profile is run through Google's Natural Language API, providing analysis on sentiment and key entities. This may be utilized for 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. There are other files for describing the breed labels, color labels and state labels to support the data in the main train file.

2.7 Data Distribution

One thing to note about all this data is that there is a mismatch in distributions. It has been analyzed and shown that the number of cats and dogs and various other features are not perfectly matched distributions.

3. DATA EXPLORATION

In the data exploration part, we focused on exploring features and their interactions and found some interesting and noticeable patterns of the dataset.

3.1 Adoption speed classes rates

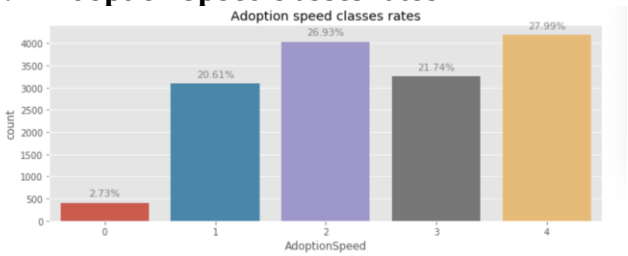

Figure 2. Adoption Speed Class Rates

In **Figure 2. Adoption Speed Class Rates** we can see that some pets were adopted immediately, but these are rare cases: maybe someone wanted to adopt any pet, or the pet was lucky to be seen by person, who wanted a similar pet. A lot of pets aren't adopted at all, which is quite sad, and we hope our models and analysis will help them to find their home. It is nice that 20.61% of pets are adopted within a first week of being listed

3.2 Distribution of Pets' Age (in Months)

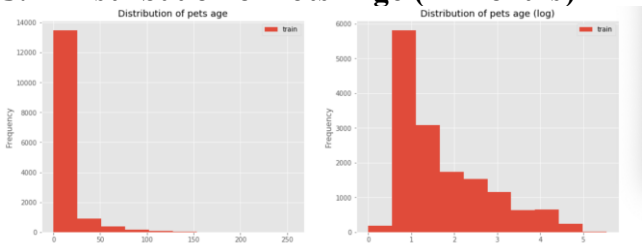

Figure 3. Distribution of Pet's Age

Figure 3. Distribution of Pet's Age shows that most pets are young – less than 25 months old.

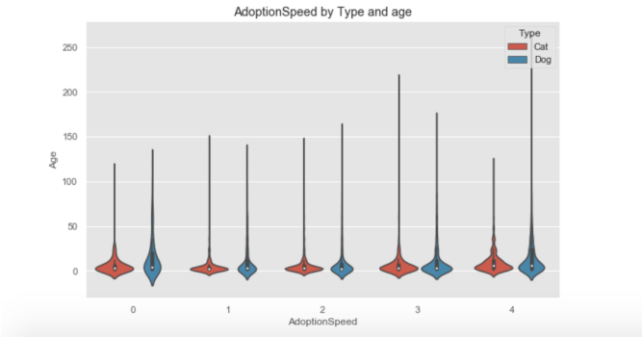

Figure 4. AdoptionSpeed by Type and Age

Figure 4. AdoptionSpeed by Type and Age that young pets are adopted quite fast and most of them are adopted.

3.3 Type

From here on we'll compare not only counts of pets in different categories, but also compare adoption speed rates with base rate. This is an example of how the base rate was calculated:

- As we saw earlier the base rate of pets with Adoption speed 0 is 410 / 14993 = 0.027;
- There are 6861 cats in dfTrain dataset and 240 of them have Adoption Speed 0. So the rate is 240 / 6861 = 0.035;
- 0.035/0.027 = 1.28, so by splitting out the data to cat vs dog, we can see that cats have a 28% increased chance of adoption speed class 0 over the base rate of adoption;

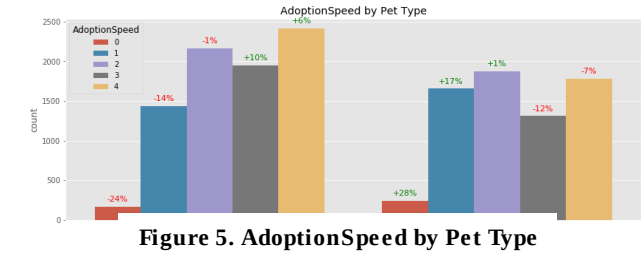

Figure 5. AdoptionSpeed by Pet Type

Figure 5. AdoptionSpeed by Pet Type shows that cats are more likely to be adopted early than dogs and overall the percentage of not adopted cats is lower than that of dogs.

3.4 Breed

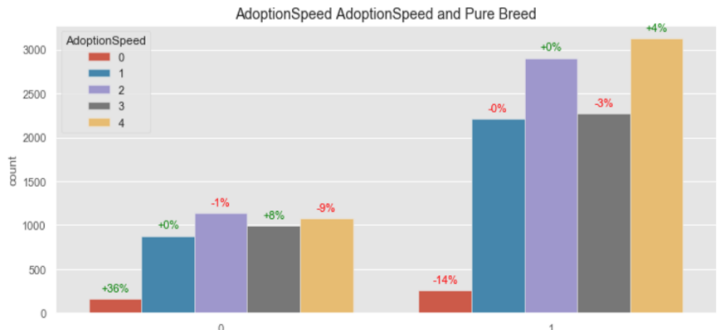

Figure 6. AdoptionSpeed Vs Pure Breed

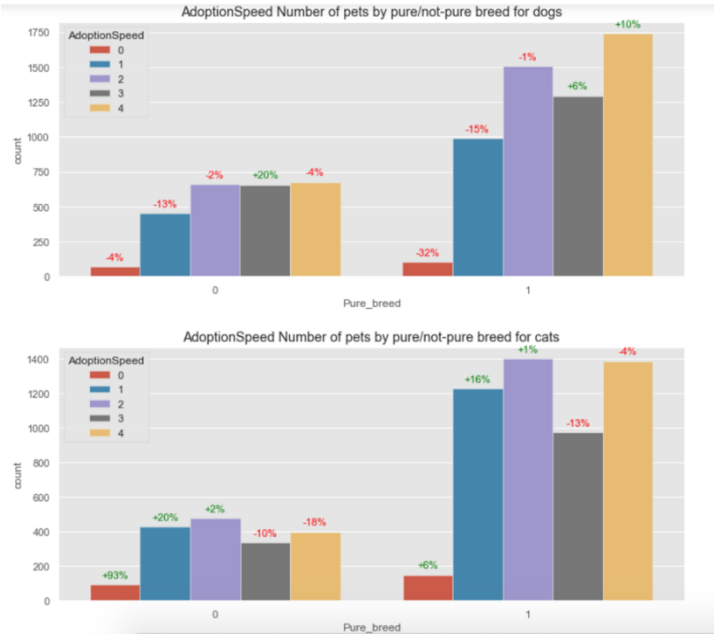

Figure 7. AdoptionSpeed Vs PureBreed by Type

In **Figure 6. AdoptionSpeed Vs Pure Breed** and **Figure 7. AdoptionSpeed Vs PureBreed** by Type seems that non-pure breed pets tend to be adopted more and faster, especially cats.

3.5 Gender

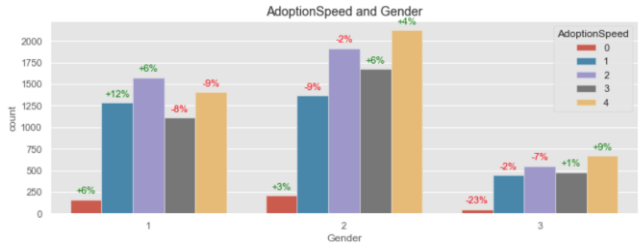

Figure 8. AdoptionSpeed Vs Gender

In **Figure 8. AdoptionSpeed Vs Gender** shows that male pets are adopted faster than female. Having no information about the gender really decreases chances.

3.6 Maturity Size

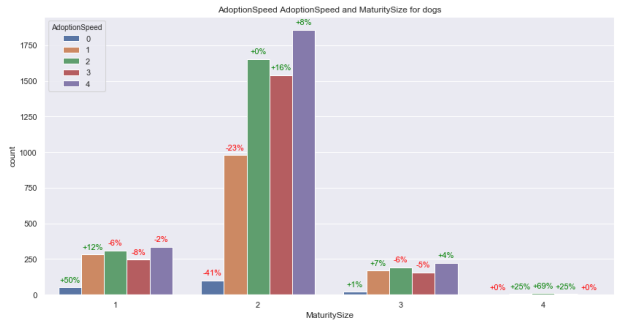

Figure 9. AdoptionSpeed Vs MaturitySize for Dogs

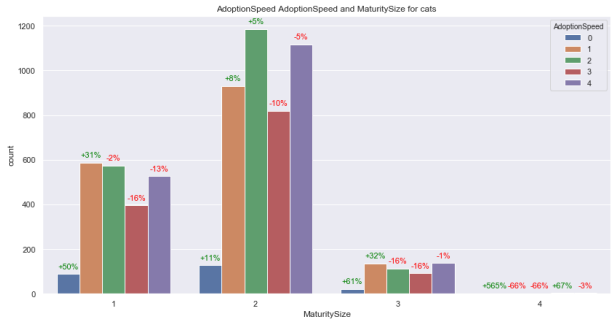

Figure 10. AdoptionSpeed Vs MaturitySize for Cats

In **Figure 9. AdoptionSpeed Vs MaturitySize for Dogs** and **Figure10. AdoptionSpeed Vs MaturitySize for Cats**, we can see that medium sized pets are most common, and they have slightly more chances to be not adopted. There are not many extra-large pets.

3.7 Fur Length

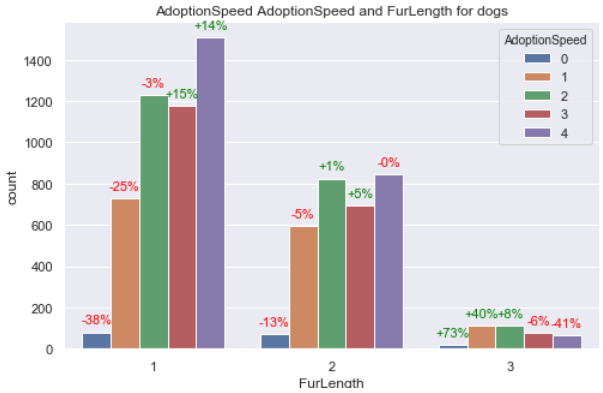

Figure 11. AdoptionSpeed Vs FurLength for Dogs

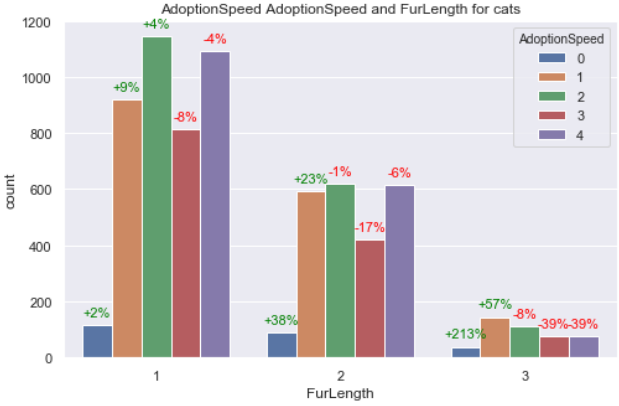

Figure 12. AdoptionSpeed Vs FurLength for Cats

In **Figure 11. AdoptionSpeed Vs FurLength for Dogs and Figure 12. AdoptionSpeed Vs FurLength for Cats**,We can see that most pets have short fur length, and pets with long hair tend to have a higher chance of being adopted. Though it could be because of randomness due to low count.

3.8 Health

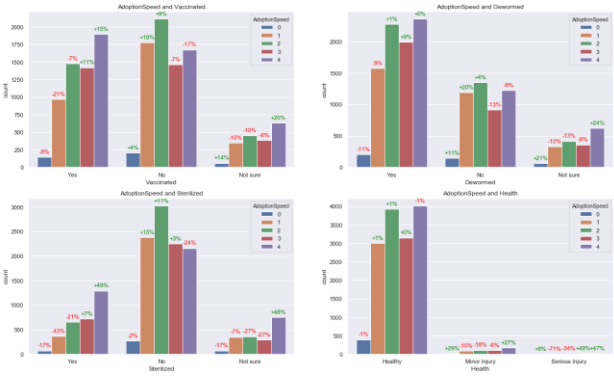

Figure 13. AdoptionSpeed Vs Health

In **Figure 13. AdoptionSpeed Vs Health**, we can interpret:

- Almost all pets are healthy and pets with minor and serious injuries are rare and not adopted well.

- It is interesting that people prefer non-vaccinated pets. Maybe they want to bring pets to vets themselves.
- People also prefer non-sterilized pets, probably because people prefer puppies/ kittens.

3.9 Fee

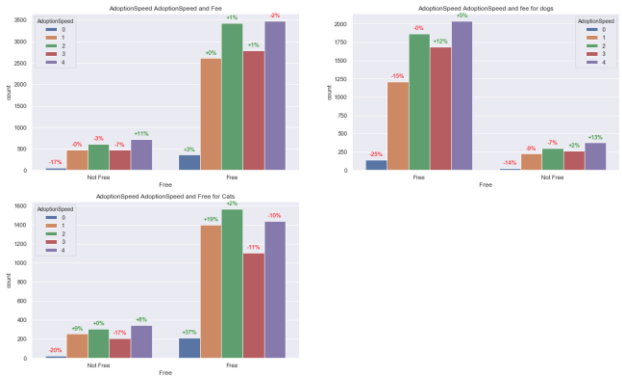

Figure 14. AdoptionSpeed Vs Fee

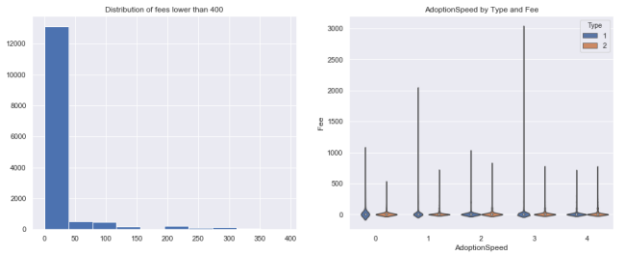

Figure 15. AdoptionSpeed by Type and Fee

In **Figure 14. AdoptionSpeed Vs Fee and Figure 15. AdoptionSpeed by Type and Fee**, we can interpret:

- It is interesting that pets with high fee tend to be adopted quite fast. Maybe people prefer to pay for "better" pets: healthy, trained and so on;
- Most pets are free or lower than $50 to adopt and that the fee may decrease the chance of adoption;
- Cats are adopted faster than dogs;

4. PREPROCESSING

While the number of records or pets in our dataset is relatively small, coming in under 20,000, we have many types of files and features to account for. In this section we will outline how we brought those data sets together, how we created features from JavaScript Object Notation (JSON) and image files (jpeg), cleaning up existing data, and imputation and encoding at the end.

4.1 Bringing in other data

Circling back to the Dataset section, in addition to our primary csv file, we have supporting data for the pet descriptions that were run through Google's Natural Language API and resulting in one file per pet with a filename of 'PetID.json'. These files break the pet description down into individual sentences with a sentiment score and magnitude for each. It also gives us the overall sentiment and magnitude of the full text as well as detail like document type and language.

For our analysis, we wanted to keep the analysis at the pet level, so we did not pull in the sentence details, but instead the overall document sentiment, overall document magnitude, and the text language. The sentiment numbers are normalized between -1 to 1 and magnitude is between 0 and infinity depending on the strength of emotion.

To bring these in, we utilized the os and json modules. Using OS we define our file path directory, which are the train_sentiment and test_sentiment subfolders, and then json.load converts the JSON file into a python dictionary that we can reference. Once it is in that format, we can use the filename (PetID) along with the dictionary values to create a row of data and add that to a data frame which will get joined with our base data set.

In code (**Figure 16. Code Snapshot**), that looks like this:

```
#cycle through JSON files and glean document sentiment and language
for filename in os.listdir(newPath):
    if filename.endswith(".json"):
        with open(os.path.join(newPath, filename), encoding="utf8") as f:
            data = json.load(f)
            print(type(data))

        petID = os.path.splitext(filename)[0]
        row = [petID, data['documentSentiment']['magnitude'], data['documentSentiment']['score'], data['language']]
        tempList.append(row)

labels = ['PetID', 'Desc_Magnitude', 'Desc_Score', 'Desc_Language']
df = pd.DataFrame(tempList, columns=labels)
```

Figure 16. Code Snapshot.

Next, we needed to get some information from the images. The pet listings did not have to have a photo, so some have zero, but they could also have many in which case we would have multiple image files for them. In these cases, they would have a primary photo with a filename "PetID-1.jpg" and each additional photo would be "PetId-2.jpg" and so on. We are most concerned with their primary photo because that is the one you would see on a list view if you are looking to adopt. The additional photos are then viewable if you click into learn more about the pet.

We utilized two methods for gathering image metadata, both of which were outlined by other users on the Kaggle competition discussion page. In the first one, we look across all of the pet's photos, and create independent variables for basic features like the image size, image height, image width, and the min, max, mean, and median for each of these. Even though there could be multiple images per pet, because we are using aggregate functions, we can summarize the image stats into one row.

In the next image method, we do more complex analysis at the pixel level of the photos. The goal of these features is to tell us more about the quality and appearance of the photo. Because this is very resource intensive and the primary photo is the most seen, we only do this for image files ending in "-1.jpeg". The first feature returned is the dullness_whiteness which a tuple result giving us both the light percentage and dark percent of the photo. Because these are percentages, they are already normalized, and all values are between 0-100. An example of a photo (**Figure 17. Dullness in Images**) with high dark percentage can be seen below on the left (dullness: 92.005) and on the right the opposite (dullness: 3.16):

Figure 17. Dullness in Images

In addition to the dullness_brightness, we created features for the average pixel width and the blurriness of the photos, again trying to measure the quality.

4.2 Imputing, Encoding, and Clean up

After left joining all these new dataframes to our original data set, we are just about ready for our analysis. Now that we have our datasets together, we were able to drop five columns from our analysis in PetID, Description, Name, RescuerID, and VideoAmt. The pet id has no benefit other than providing us a key to join all of the data together, the description is now represented as sentiment, magnitude, and language, and the name/rescuer id both had cardinality in the thousands and we did not have any automated way to glean value from them.

The final steps required in our preprocessing are making sure that all of our variables are encoded and that null values are accounted for. We know that our dataset has a mix of categorical and numeric. Some of the categorical variables are already label encoded into numeric values such as state, breed, color, etc. Before we begin our imputing and encoding, we want to check the columns to see where we have nulls as well as our cardinality on the categorical columns to get a better feel for what those will look like after one-hot-encoding.

	column	cardinality
0	Type	2
1	Breed1	176
2	Breed2	135
3	Gender	3
4	Color1	7
5	Color2	7
6	Color3	6
7	MaturitySize	4
8	FurLength	3
9	Vaccinated	3
10	Dewormed	3
11	Sterilized	3
12	Health	3
13	State	14
14	Desc_Language	5

Figure 18. Cardinality Check

In **Figure 18. Cardinality Check**, we can see most columns are single-digits, state is a little bigger at 14, and then for breeds we have over 100 which will be tough for one-hot-encoding. We'll talk about how we handled that after we take care of the easy parts. Because not all pets have a description or photo, we have null values for many of our supporting data sets that we just prepared. Most of these columns are numeric columns, so we handled them by imputing with the mean value. The one categorical variable from the sentiment portion is the language, and we imputed the mode for the blanks here.

As mentioned, all of the categorical variables in the base file were already label encoded, so we just need to label encode the language column to make it numeric. We followed that by one-hot-encoding the language, state, health, sterilized, dewormed, vaccinated, fur length, maturity size, color1, color2, color3, gender, and type.

The last step was to figure out what we could do with the breed1 and breed2 columns. The simple answer would have been to toss them out because they are too difficult for our analysis, but we know anecdotally that certain breeds of cats and dogs are more favorable than others, so we do not want to lose that from our model. The method that we turned to instead of one-hot-encoding was binary encoding. This is similar to one-hot, except that instead of having n-1 columns to represent a categorical variable with n elements where they are all zeros and one/none of them have a 1 to indicate, binary converts each integer into binary digits and each binary digit gets a column. Rather than having 175 columns for Breed1 and 134 for Breed2, we now have 18 total and are ready to start running it through our models.

5. ALGORITHMS - MODELS

In machine learning there is no one model/algorithm works best for every problem, and it's especially relevant for supervised learning. For example, you can't say that Logistic Regression are always better than Decision Tree or vice-versa. There are many factors at play, such as the size and structure of the dataset. As a result, we decided to try different algorithms to solve PetFinder problem, while using a hold-out test-set of data to evaluate performance and select the best model. However, there is a common principle that underlies all supervised machine learning algorithms for predictive modeling. Machine learning algorithms are described as learning a target function that best maps input variables (X) to an output variable (y). This is a general learning task where we would like to make predictions in the future (y) given new examples of input variables (X). We don't know what the function looks like or its form. The most common type of machine learning is to learn the mapping y=f(X) to make predictions of y for new X. Our goal is to make the most accurate predictions possible. To make best possible prediction of the AdoptionSpeed. We tried Logistic Regression, Decision Tree, Naïve Bayes, Random Forest, Support Vector Machines (SVM), Kernel Support Vector Machines, K-nearest neighbors (K-NN), XGBoost and Light Gradient Boosting Method (LGBM). Before running through each model, we have performed Dimensionality Reduction on the features. We used Backward Elimination which is part of feature selection and Principal Component Analysis (PCA, Linear Discriminant Analysis (LDA) and Kernel Principal Component Analysis (KPCA) which is part of feature extraction. Below are the different feature methods tried on each model to train the model and capture the optimal performance in predicting the class variable more accurately. Our findings and results on each model are presented in its section.

Different Methods
1. Applying PCA on all features
2. Applying LDA on all features
3. Applying Kernel PCA on all features
4. Applying PCA on features (with backward elimination)
5. Applying LDA on features (with backward elimination)
6. Applying Kernel PCA on features (with backward elimination)

Figure 19. Different Methods used

5.1 Accuracy Metrics

Classification problems are perhaps the most common type of machine learning problem and as there are a myriad of metrics that can evaluate predictions. For our models we are using Classification Accuracy to evaluate the performance of the model. Classification accuracy is the number of correct predictions made as a ratio of all predictions made. We are using the score function on each model which returns the mean accuracy on the given test data and labels. We are also using the confusion matrix to check the distribution on the target variable which might help to see if there are any anomalies.

5.2 Logistic Regression

Logistic Regression is a popular technique in machine learning which was borrowed from the field of statistics. It is a go-to method for binary classification problems. In Logistic Regression the goal is to find the values for the coefficients that weight each input variable. The prediction for the output is transformed using a non-linear function called the logistic function or sigmoid function. The logistic function looks like a big S and will transform any value into the range of 0 to 1. This is useful because we can apply a rule to the output of the logistic function to and values to 0 to 1 e.g. If less than 0.5 then output 0. Because of the way that the model is learned, the predictions made by logistic regression can also be used as the probability of a given data instance belonging to class 0 or class 1. This can be useful for problems where you need to give rationale for a prediction. Logistic regression does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. For that reason, we are using backward elimination, PCA, LDA and KPCA in predicting our class variable. It's a fast model to learn and effective on binary classification.

Just to visualize on how the training set results, the below three **Figures 20, 21, 22** show us the two features considered to plot the boundaries and the predicted values. We can see there is a high misclassification rate that is because the model was using only 2 features to predict the class variable.

As discussed earlier on using different methods to train the model and predict the class variable, the logistic regression results are captured in the **Figure 23. Logistic Regression Accuracy results** on how Logistic Regression performed.

Figure 20. Training Set results using PCA

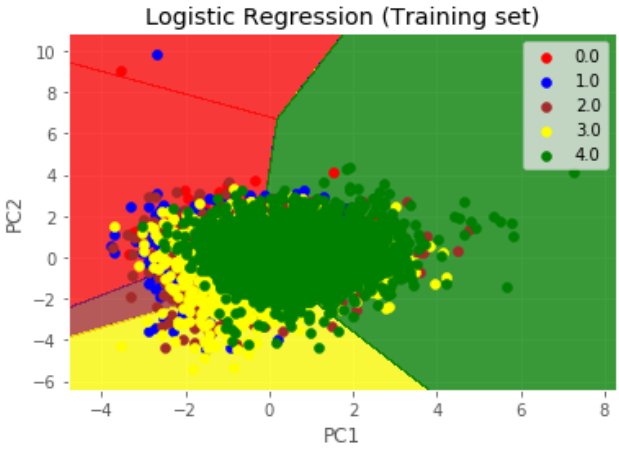

Figure 21. Training Set results using LDA

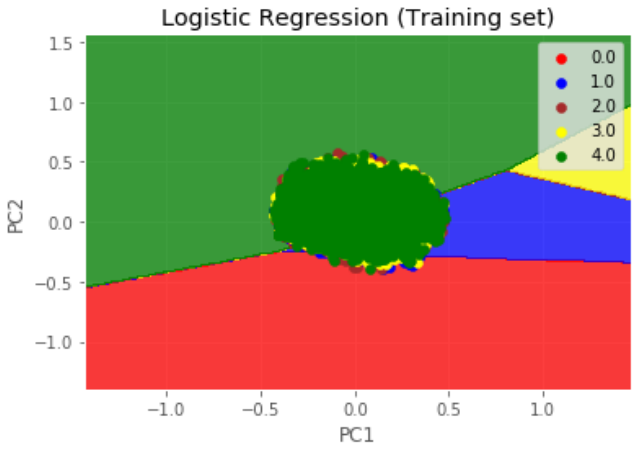

Figure 22. Training Set results using KPCA

Method	Logistic Regression Accuracy
1. Using PCA with all features	0.369 or 36.9%
2. Using LDA with all features	0.368 or 36.8%
3. Using Kernel PCA with all features	0.373 or 37.3%
4. Using PCA with backward elimination features	0.368 or 36.8%
5. Using LDA with backward elimination features	0.363 or 36.3%
6. Using Kernel PCA with backward elimination features	0.378 or 37.8%

Figure 23. Logistic Regression Accuracy results

The best method predicting higher accuracy is using Kernel PCA with backward eliminated features with 37.8% accuracy and the method with least accuracy is using LDA with backward eliminated features with 36.3% accuracy. **Figure 24** shows the confusion matrix for Kernel PCA with backward elimination features method.

```
[[  0,  26,  21,   7,  22],
 [  0, 186, 249,  45, 162],
 [  0, 120, 313, 102, 263],
 [  0,  97, 223, 119, 217],
 [  0,  81, 166,  64, 516]]
```

Figure 24. Confusion Matrix for Logistic Regression

5.3 Decision Tree

A Decision Tree is a series of nodes, a directional graph that starts at the base with a single node and extends to the many leaf nodes that represent the categories that the tree can classify. Another way to think of a decision tree is as a flow chart, where the flow starts at the root node and ends with a decision made at the leaves. Trees are fast to learn and very fast for making predictions. They are also often accurate for a broad range of problems.

Using Decision Tree and different methods to train the model and predict the class variable, the table in **Figure 25. Decision Tree Accuracy results** shows the accuracy on how well the Decision Tree predicted the AdoptionRate of pets.

Method	Decision Tree Accuracy
1. Using PCA with all features	0.29 or 29%
2. Using LDA with all features	0.28 or 28%
3. Using Kernel PCA with all features	0.27 or 27%
4. Using PCA with backward elimination features	0.30 or 30%
5. Using LDA with backward elimination features	0.29 or 29%
6. Using Kernel PCA with backward elimination features	0.26 or 26%

Figure 25. Decision Tree Accuracy results

The best method predicting higher accuracy is using PCA with all features and using PCA with backward eliminated features with 29% accuracy and the method with least accuracy is using Kernel PCA with 26% accuracy. **Figure 26** shows the corresponding confusion matrix for method PCA with all features.

```
[  6  21  19  12  18]
[ 16 163 174 137 152]
[ 19 151 266 176 186]
[ 14 138 193 161 150]
[ 23 133 205 172 294]
```

Figure 26. Confusion Matrix for Decision Tree

5.4 Naïve Bayes

Naïve Bayes is a simple but surprisingly powerful algorithm for predictive modeling. The model is comprised of two types of probabilities that can be calculated directly from training data: 1) The probability of each class; and 2) The conditional probability for each class given each x value. Once calculated, the probability model can be used to make predictions for new data using Bayes Theorem. When your data is real-valued it is common to assume a Gaussian distributed (bell curve) so that you can easily estimate these probabilities. Naïve Bayes is called naïve because it assumes that each input variable is independent. This is a strong assumption and unrealistic for real data, nevertheless, the technique is very effective on a large range of complex problems.

Using Naïve Bayes and different methods to train the model and predict the class variable, the table in **Figure 27. Naïve Bayes Accuracy results** shows the accuracy on how well the Naïve Bayes predicted the AdoptionRate of pets.

Method	Naïve Bayes Accuracy
1. Using PCA with all features	0.273 or 27.3%
2. Using LDA with all features	0.364 or 36.4%
3. Using Kernel PCA with all features	0.22 or 22%
4. Using PCA with backward elimination features	0.278 or 27.8%
5. Using LDA with backward elimination features	0.362 or 36.2%
6. Using Kernel PCA with backward elimination features	0.238 or 23.8%

Figure 27. Naïve Bayes Accuracy results

The best method predicting higher accuracy is using LDA with all features with 36.4% accuracy and the method with least accuracy is using Kernel PCA with all features with 22% accuracy. **Figure 28** shows the confusion matrix for LDA with all feature's method.

```
[[  3,  19,  35,   2,  17],
 [ 11, 164, 327,  29, 111],
 [ 14, 103, 405,  65, 211],
 [  8,  82, 287, 105, 174],
 [  8,  76, 269,  57, 417]]
```

Figure 28. Confusion Matrix for Naïve Bayes

5.5 Random Forest

Random Forest is an ensemble learning. It is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because its simplicity and the fact that it can be used for both classification and regression tasks. The random forest is a model made up of many decision trees into a single ensemble model. An ensemble method or ensemble learning algorithm consists of aggregating multiple outputs made by a diverse set of predictors to obtain better results. Formally, based on a set of weak learners we are trying to use a strong learner for or model. Therefore, the purpose of using ensemble methods is to average the outcome of individual predictions by diversifying the set of predictors, this lowering the variance, to arrive at a powerful prediction model that reduces overfitting on our training set.

Using Random Forest and different methods to train the model and predict the class variable, the table in **Figure 29. Random Forest Accuracy results** shows the accuracy on how well the

Random Forest predicted the AdoptionRate of pets. Number of trees n_estimators = 25 as hyper parameter in the model and criterion = entropy.

Method	Random Forest Accuracy
1. Using PCA with all features	0.34 Or 34%
2. Using LDA with all features	0.32 or 32%
3. Using Kernel PCA with all features	0.25 or 25%
4. Using PCA with backward elimination features	0.345 or 34.5%
5. Using LDA with backward elimination features	0.342 or 34.2%
6. Using Kernel PCA with backward elimination features	0.27 or 27%

Figure 29. Random Forest Accuracy results

The best method predicting higher accuracy is using PCA with backward eliminated features with 34% accuracy and the method with least accuracy is using Kernel PCA with backward eliminated features with 26% accuracy. **Figure 30** shows the confusion matrix for PCA with backward elimination features method.

```
[[  4,  16,  26,   9,  21],
 [  3, 207, 214,  94, 124],
 [  1, 184, 288, 121, 204],
 [  1, 130, 232, 129, 164],
 [  1,  91, 209, 105, 421]]
```

Figure 30. Confusion Matrix for Random Forest

5.6 Support Vector Machine (SVM)

Support Vector Machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1. In two-dimensions, you can visualize this as a line and let's assume that all our input points can be completely separated by this line. The SVM learning algorithm finds the coefficients that results in the best separation of the classes by the hyperplane. The distance between the hyperplane and the closest data points is referred to as the margin. The best or optimal hyperplane that can separate the two classes is the line that has the largest margin. Only these points are relevant in defining the hyperplane and in the construction of the classifier. These points are called the support vectors.

Using SVM and different methods to train the model and predict the class variable, the table in **Figure 31. SVM Accuracy results** shows the accuracy on how well the SVM predicted the AdoptionRate of pets.

Method	SVM Accuracy
1. Using PCA with all features	0.365 or 36.5%
2. Using LDA with all features	0.364 or 36.4%
3. Using Kernel PCA with all features	0.38 or 38%
4. Using PCA with backward elimination features	0.355 or 35.5%
5. Using LDA with backward elimination features	0.364 or 36.4%
6. Using Kernel PCA with backward elimination features	0.376 or 37.6%

Figure 31. SVM Accuracy results

The best method predicting higher accuracy is using Kernel PCA with all features with 38% accuracy and the method with least accuracy is using LDA with and without backward eliminated features with 36.4% accuracy. **Figure 32** shows the confusion matrix for Kernel PCA with all feature's method.

```
[  1  25  31   4  15]
[  0 171 291  33 147]
[  0 124 362  83 229]
[  0  86 252 105 213]
[  0  70 213  42 502]]
```

Figure 32. Confusion Matrix for Random Forest

5.7 Kernel Support Vector Machine

A kernel is a function that represents the similarity between two observations in a desired dimension. The similarity between two observations with the same coordinates is 1 and tends to 0 as the Euclidean distance between them increases. Kernel methods, or the kernel trick is to map data sometimes nonlinear data from a low dimensional space to a high dimensional space. In a higher dimension, you can solve a linear problem that's nonlinear in lower-dimensional space.

Using Kernel SVM and different methods to train the model and predict the class variable, the table in **Figure 33. Kernel SVM Accuracy results** shows the accuracy on how well the Kernel SVM predicted the AdoptionRate of pets.

Method	Kernel SVM Accuracy
1. Using PCA with all features	0.38 or 38%
2. Using LDA with all features	0.37 or 37%
3. Using Kernel PCA with all features	0.275 or 27.5%
4. Using PCA with backward elimination features	0.376 or 37.6%
5. Using LDA with backward elimination features	0.365 or 36.5%
6. Using Kernel PCA with backward elimination features	0.275 or 27.5%

Figure 33. Kernel SVM Accuracy results

The best method predicting higher accuracy is using PCA with all features with 38% accuracy and the method with least accuracy is using Kernel PCA with and without backward eliminated features with 27.5% accuracy. **Figure 34** shows the confusion matrix for PCA with all feature's method.

```
[  1  25  31   4  15]
[  0 171 291  33 147]
[  0 124 362  83 229]
[  0  86 252 105 213]
[  0  70 213  42 502]
```

Figure 34. Confusion Matrix for Kernel SVM

5.8 K-Nearest Neighbors (KNN)

The KNN algorithm is very simple and very effective. Predictions are made for a new data point by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances. For regression problems, this might be the mean output variable, for classification problems this might be the mode class value. The trick is in how to determine the similarity between the data instances. The simplest technique if your attributes are all the same scale is to use the Euclidean distance, a number you can

calculate directly based on the differences between each input variable. KNN can require a lot of memory or space to store all the data, but only performs a calculation when a prediction is needed, just in time. The idea of distance or closeness can break down in very high dimensions which can negatively affect the performance of the algorithm. This is the curse of dimensionality, it suggests that to use only those input variables that are most relevant to predicting the output variable.

Using KNN and different methods to train the model and predict the class variable, the table in **Figure 35. KNN Accuracy results** shows the accuracy on how well the KNN predicted the AdoptionRate of pets. n_neighbours = 10 as hyper parameter in the model.

Method	KNN Accuracy
1. Using PCA with all features	0.321 or 32.1%
2. Using LDA with all features	0.332 or 33.2%
3. Using Kernel PCA with all features	0.321 or 32.1%
4. Using PCA with backward elimination features	0.32 or 32%
5. Using LDA with backward elimination features	0.332 or 33.2%
6. Using Kernel PCA with backward elimination features	0.32 or 32%

Figure 35. KNN Accuracy results

The best method predicting higher accuracy is using PCA with backward eliminated features with % accuracy and the method with least accuracy is using Kernel PCA with backward eliminated features with % accuracy. **Figure 36** shows the confusion matrix for LDA with all feature's method.

```
[  6,  23,  22,   8,  17]
[ 11, 246, 197,  75, 113]
[ 10, 206, 272, 129, 181]
[  6, 149, 203, 133, 165]
[  3, 133, 216, 124, 351]
```

Figure 36. Confusion Matrix for KNN

6. CONCLUSION

After trying all the models **Figure 37. Model Accuracies**, we conclude that the SVM turned out to be best algorithms for Petfinder dataset with 38% prediction accuracy. SVM works well with clear margin of separation and is effective in high dimensional spaces. But all the models were close enough with SVM accuracy except for Decision Tree model. One of the reasons why low accuracies (i.e. around 33%-38%) for all the models we tried is because of the variety within the features in the dataset and dataset has more noise i.e. target classes are overlapping.

As part of future work, we would like to focus more on feature engineering to fine tune the models and try out boosting algorithms like XGBoost and Light GBM to see how they perform on pet finder dataset. We tried XGBoost and the accuracy improved to 40%, code is included in the submission.

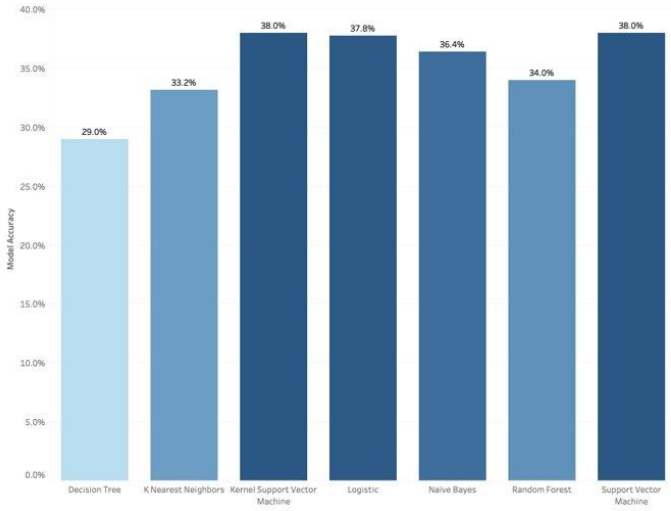

Figure 37. Model Accuracies

7. KNOWLEDGMENTS

Our thanks to Manjeet Rege for assisting on this project.

8. REFERENCES

[1] https://www.kaggle.com/c/petfinder-adoption-prediction/overview

[2] https://www.kaggle.com/teemingyi/image-statistics-for-petfinder

[3] https://www.kaggle.com/kaerunantoka/extract-image-features

[4] https://towardsdatascience.com/a-tour-of-the-top-10-algorithms-for-machine-learning-newbies-dde4edffae11