

# PawPal: Predictive Modeling of Pet Adoption Speed

Kay Eugenia Purnama, Poon Zhe Xuan, Tan Yi Long

April 28, 2024

## 1 Introduction

### 1.1 Background

In Singapore, the number of pet abandonment cases have generally been on an upward trend, rising by roughly 25% from 2017 to 2018 [You22]. At the same time, local animal shelters have seen up to a 50% fall in adoption numbers which may be attributed to the rising living cost and post-COVID landscape [Koh23]. Recognizing the urgency and gravity of this situation, our project aims to address and mitigate the low adoption rate by developing a comprehensive pet adoption prediction tool to enhance the efficiency of adoption listings, ultimately increasing the chances of successful pet adoptions.

### 1.2 Description

Using a [dataset](#) [19] containing adoption listings and images from PetFinder.my, Malaysia's leading animal welfare platform, we aim to analyze the data and create machine learning models to predict adoption rates based on the data provided. With these models, we would create an online suggestion tool that accepts adoption listings and images, providing suggestions on how to improve pet adoption rates.

## 2 Input Data

### 2.1 Tabular

This contains information about the pets on the adoption listing as well as details about the listing itself. Some examples of the data fields include the name and gender of the pet, and the adoption fee.

### 2.2 Text

Each adoption listing contains a text description, which we will analyze using Natural Language Processing (NLP) methods.

### 2.3 Image

For pet listings that contain photos, we will use models such as Convolutional Neural Networks (CNNs) and Computer Vision (CV) algorithms to analyze the pet images.

## 3 Prediction Task

The target variable is `AdoptionSpeed`, which is determined by how quickly, if at all, a pet is adopted. The values are determined in the following way:

- `0` : Pet was adopted on the same day as it was listed.
- `1` : Pet was adopted between 1 and 7 days (1st week) after being listed.
- `2` : Pet was adopted between 8 and 30 days (1st month) after being listed.

**3** : Pet was adopted between 31 and 90 days (2nd & 3rd month) after being listed.

**4** : No adoption after 100 days of being listed. (There are no pets in this dataset that waited between 90 and 100 days).

As elaborated in a later section, we decided to combine classes 0 and 1 due to class imbalance. Hence, this is a **4-way classification** task.

## 4 Data Quality Assessment

Before starting with our Exploratory Data Analysis (EDA), it is important to check the given dataset for any quality issues, namely missing values, duplicate rows and class imbalance.

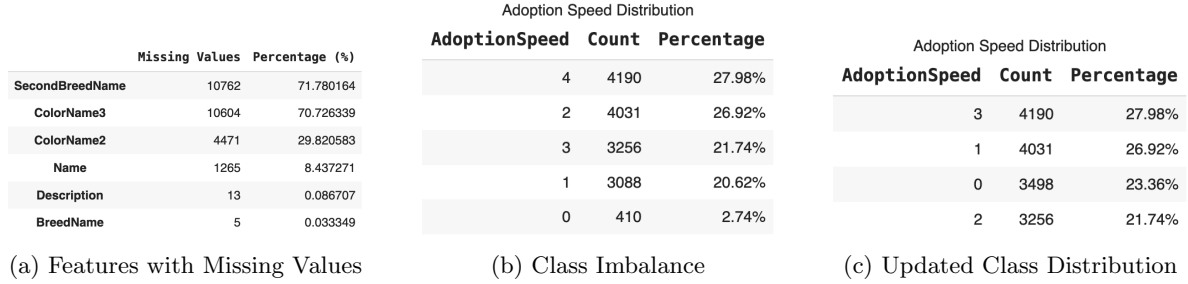


Figure 1: Data Validation Figures

### 4.1 Missing and Duplicate Values

The first step is to check for missing values and impute or deal with them. As seen above in Figure 1(a), there are 6 features detected to have missing values.

The columns **SecondBreedName**, **ColorName2** and **ColorName3** can be imputed with “None”. As **Name** will eventually be dropped before modeling (since names are unique values), there is no need to deal with missing **Name** values.

Given the small number of rows with missing data (17) as compared to the total number of rows (14993), these rows with missing data can be removed as **BreedName** and **Description** are important features that could be related to the pet adoption rates.

### 4.2 Class Imbalance

It is important to ensure that each target column has a balanced representation. This step helps in understanding whether certain classes are overrepresented or underrepresented in the dataset, which can significantly impact model performance and bias the learning process.

A common rule of thumb is that if the minority class in the dataset constitutes less than 10-20% of the total data, it can be considered imbalanced. As seen above in Figure 1(b), the Adoption Speed for class '0' is underrepresented as it only makes up around 2.73% of the total data.

### 4.3 SMOTE

We initially experimented with SMOTE (Synthetic Minority Over-sampling Technique) on the train data to solve this class imbalance issue. However, it caused an artificial increase in the cross-validation F1 scores and it had a big difference with the test F1 scores. This could be due to overlapping classes in the feature space, which meant that there was no clear boundary separating the classes. Hence, SMOTE would just introduce noise into the data, negatively affecting the generalization of the models trained.

## 4.4 Merging Classes

Hence, another solution was to merge classes 0 and 1, renaming that category to "adopted within a week". This was a viable solution given the small differences of Adoption Speed between both classes. As seen in Figure 1(c), the classes are now evenly distributed. Hence, the updated categories of `AdoptionSpeed` are as follows:

- 0 : Pet was adopted on the same week (0 to 7 days) as it was listed.
- 1 : Pet was adopted between 8 and 30 days (1st month) after being listed.
- 2 : Pet was adopted between 31 and 90 days (2nd & 3rd month) after being listed.
- 3 : No adoption after 100 days of being listed.

## 5 Exploratory Data Analysis

We used dynamic visualizations using the *plotly* package to explore several key relationships present in the data. To interact with the figures, you may reference our code notebook.

### 5.1 Adoption Speed by Animal Type

Using the target variable which is `AdoptionSpeed`, we can explore the difference between cats and dogs.

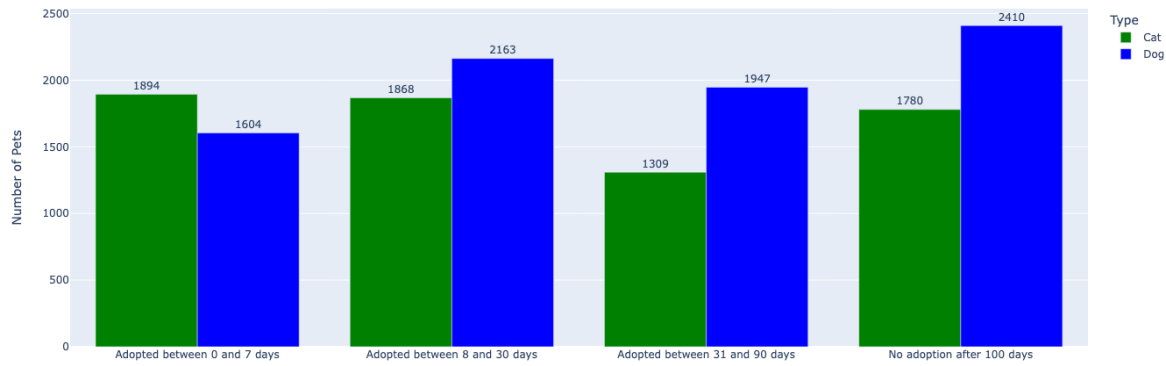


Figure 2: Bar Chart of Adoption Speed by Animal Type

Above in Figure 2, it can be seen that cats generally had faster adoption rates than dogs as a higher number of cats were "Adopted between 0 and 7 days" while dogs had higher adoption rates for the other three categories. Since there were noticeable differences between cats and dogs for each category, the type of animal likely affects adoption rates.

## 5.2 Adoption Rates and Speed by State

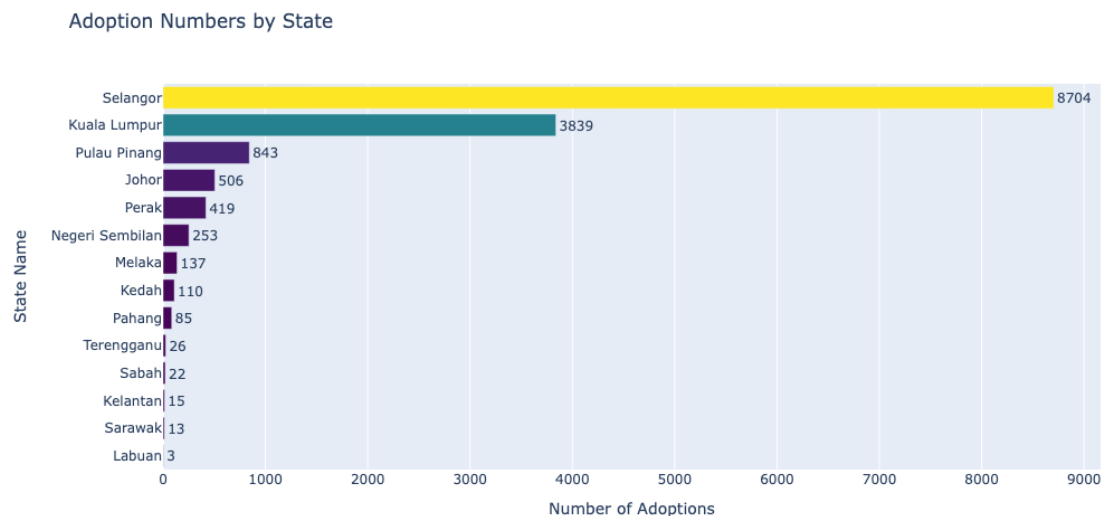


Figure 3: Horizontal Bar Chart of Adoption Numbers by State

Figure 3 shows that there are much more pet adoptions in the states of Selangor and Kuala Lumpur.

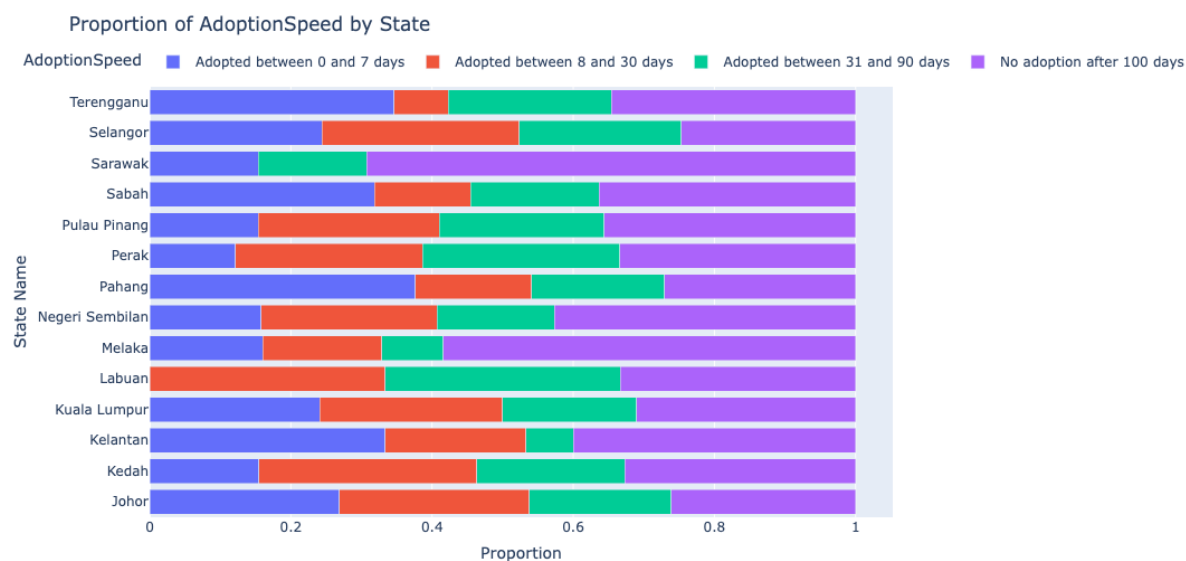


Figure 4: Stacked Bar Chart of Adoption Speed by State

Figure 4 shows that the proportion of each category for **AdoptionSpeed** differs greatly between states. This variability suggests that the state in which a pet is located could significantly influence its chances of being adopted quickly or slowly. Hence, it is reasonable to conclude that the **State** is likely to be a good predictor of **AdoptionSpeed**.

### 5.3 Exploring Age

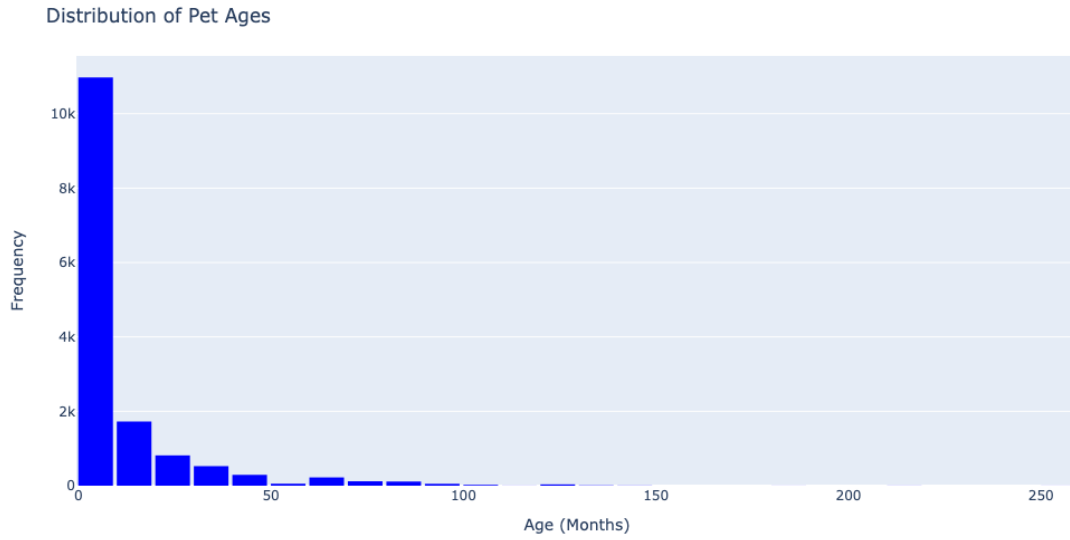


Figure 5: Histogram of Pet Ages

The histogram in Figure 5 shows that most pets adopted are young, which is expected. To reduce noise and simplify the data, we will convert the continuous `Age` to discrete bins.

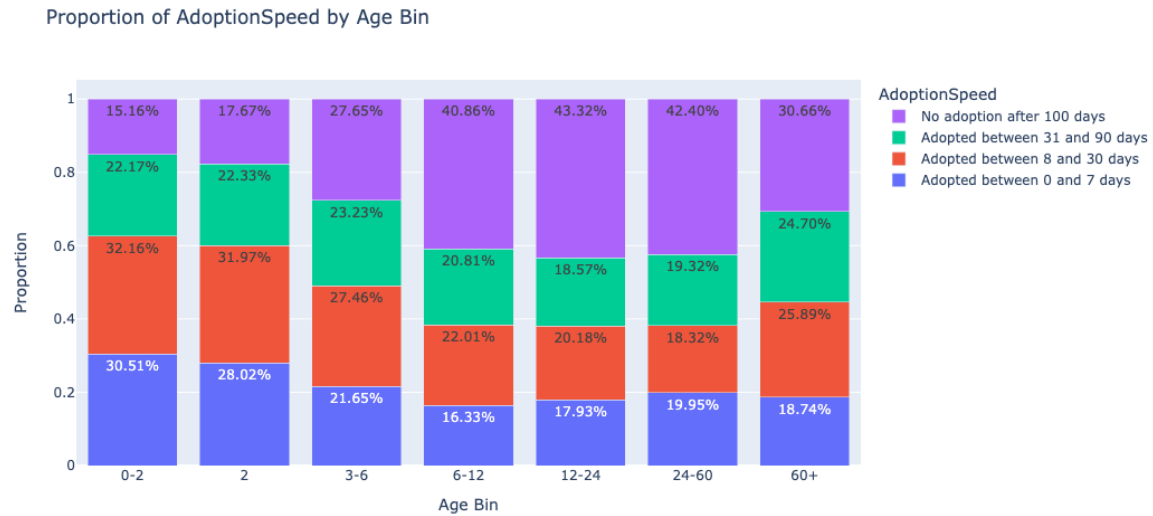


Figure 6: Stacked Bar Plot of AdoptionSpeed by Age Bin

The stacked bar plot in Figure 6 supports the point that adoption rates are higher for younger pets as pets with the ages of 0-2 months had highest proportions for being adopted within the week. A general downward trend can also be observed that younger pets had higher proportions in the first two classes.

### 5.4 Relationship between pet's missing name and Adoption Speed

Creating a new variable `hasName` which indicates whether the `Name` column is not empty can capture the influence of pet names on adoption rates, enhancing the predictive power of the models.

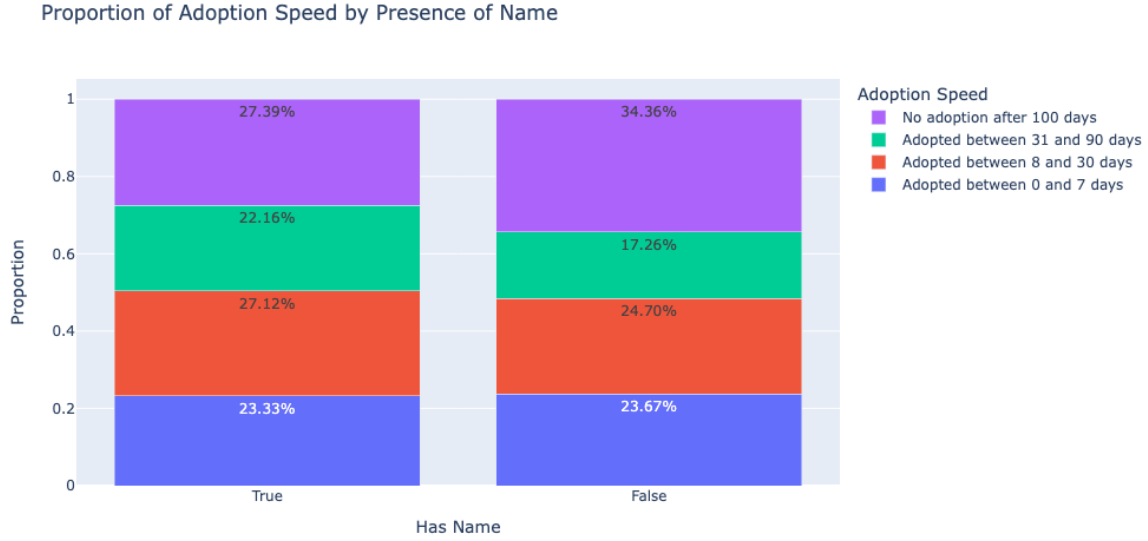


Figure 7: Stacked Bar Plot of AdoptionSpeed by HasName

It can be seen from Figure 7 that there are noticeable differences especially for the slower adoption speed categories. Pets that do not have a name have much lower adoption rates, as we can see that a higher proportion of pets with no name fall in the 'No adoption after 100 days category'.

## 5.5 Relationship between Quantity and Adoption Speed

**Quantity** is the number of pets represented in the listing. As there are many different values for the **Quantity** column, we will create a new grouped variable for the quantity to reduce the complexity. Any quantity value above 7 is grouped together in an "Other" category.

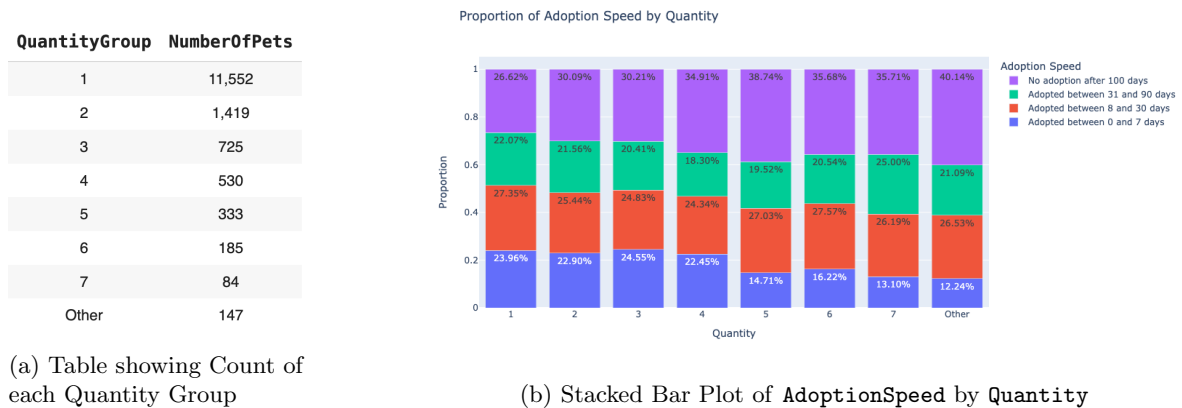


Figure 8: Quantity Grouped Table and Charts

It can be seen above in Figure 8(b) that pets with lower quantities seem to be adopted slightly faster, which can be explained by how most adopters only have the capacity for 1 or 2 pets.

## 5.6 Relationship between Treatment Effects and Adoption Speed

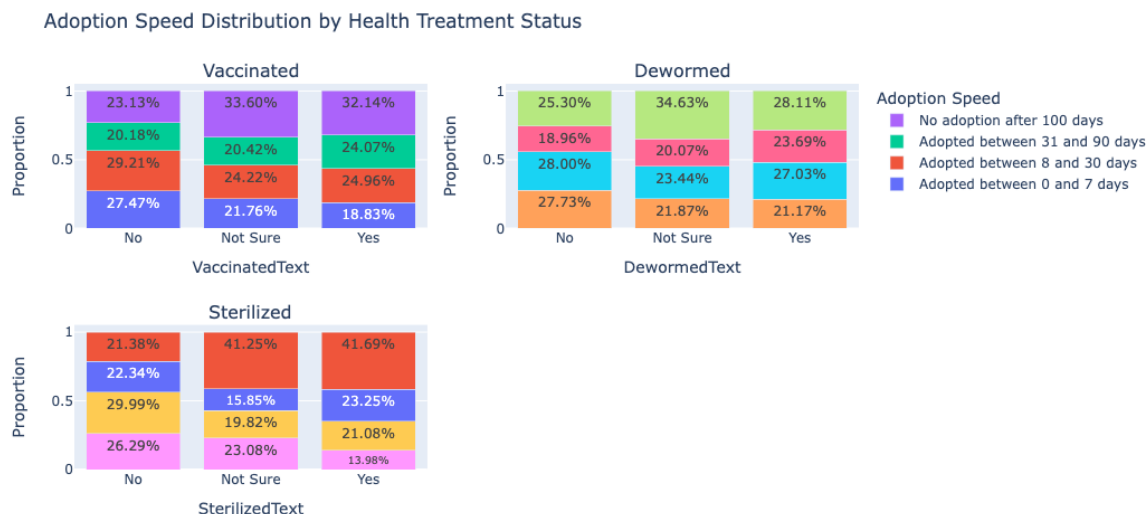


Figure 9: Stacked Bar Plot of Adoption Speed by Different Treatment Statuses

Above in Figure 9, we explore the distribution of `AdoptionSpeed` categories for each of the three health treatment statuses provided (`Vaccinated`, `Dewormed` and `Sterilized`). Surprisingly, the adoption rates were lower for pets that were vaccinated, dewormed and sterilized. Now, we will define a new feature called `Treated` which indicates whether the pet has been vaccinated, dewormed and sterilized.

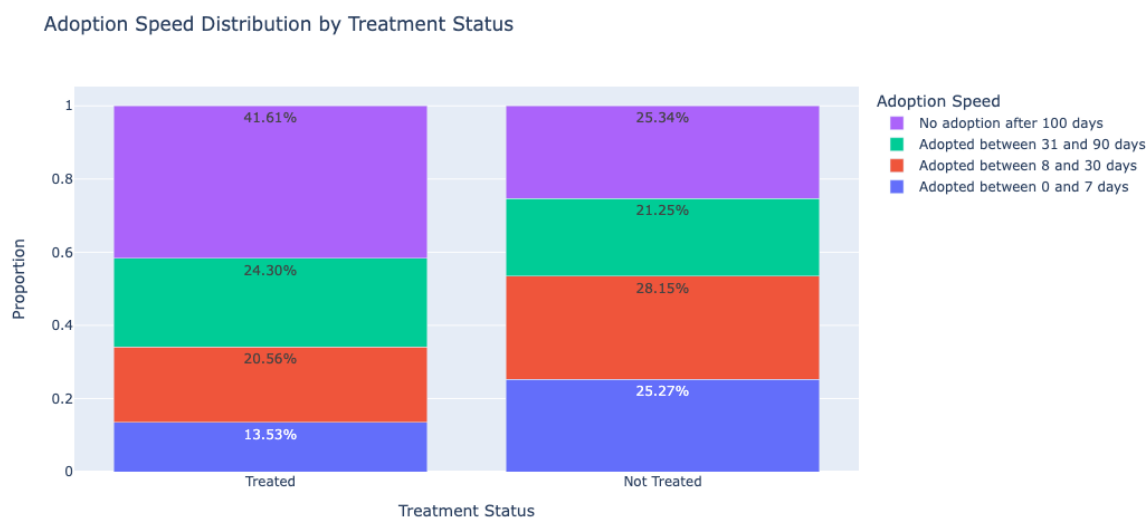


Figure 10: Stacked Bar Chart of Adoption Speed by Treated

Similar to the trend we observed above, there is a significantly lower adoption rate for pets classified as "Treated".

## 5.7 Relationship between Health Condition and Adoption Speed

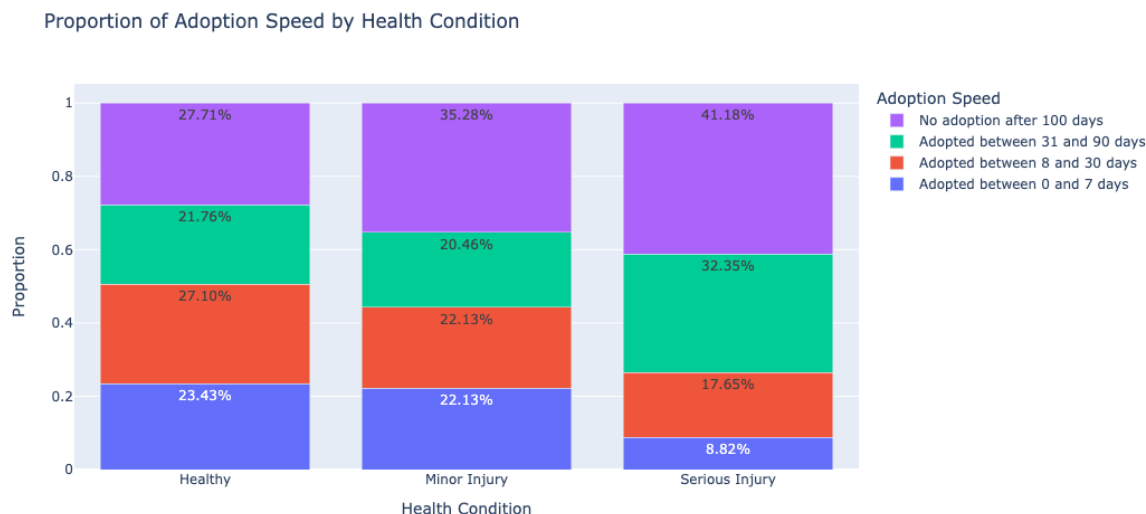


Figure 11: Stacked Bar Chart of Adoption Speed by Health

The above Figure 11 shows that the more healthy the pet is, the faster the adoption speed and the higher the adoption rate.

## 5.8 Relationship between Fee and Adoption Speed

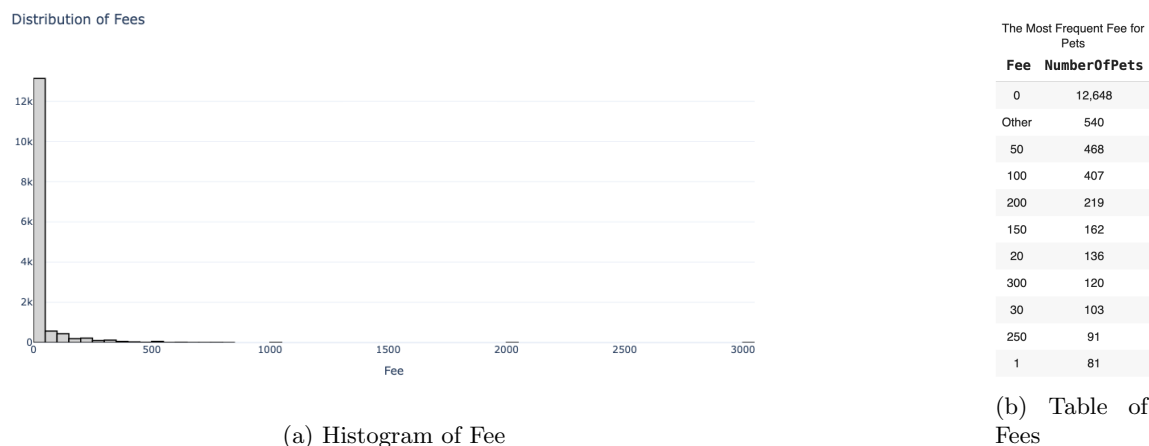


Figure 12: Fee Histogram and Most Frequent Values

We can see from Figure 12(a) that **Fee** is right skewed, the table in Figure 12(b) also shows that most pets do not have an adoption fee and are free of charge. We have binned the **Fee** values into the Top 10 most frequent values, grouping all the other values into the category “Other”.



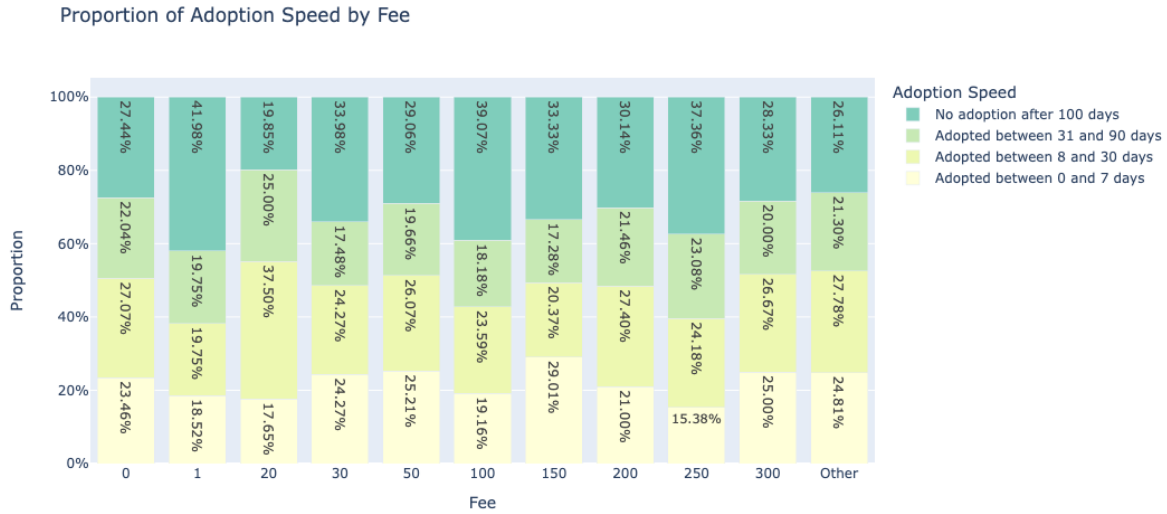


Figure 13: Stacked Bar Chart of Adoption Speed by Fee

Across the different fees, there are differing distributions of `AdoptionSpeed`. One interesting point was that a lower fee did not strongly correlate to a faster adoption speed.

## 5.9 Relationship between Number of Photos and Adoption Speed

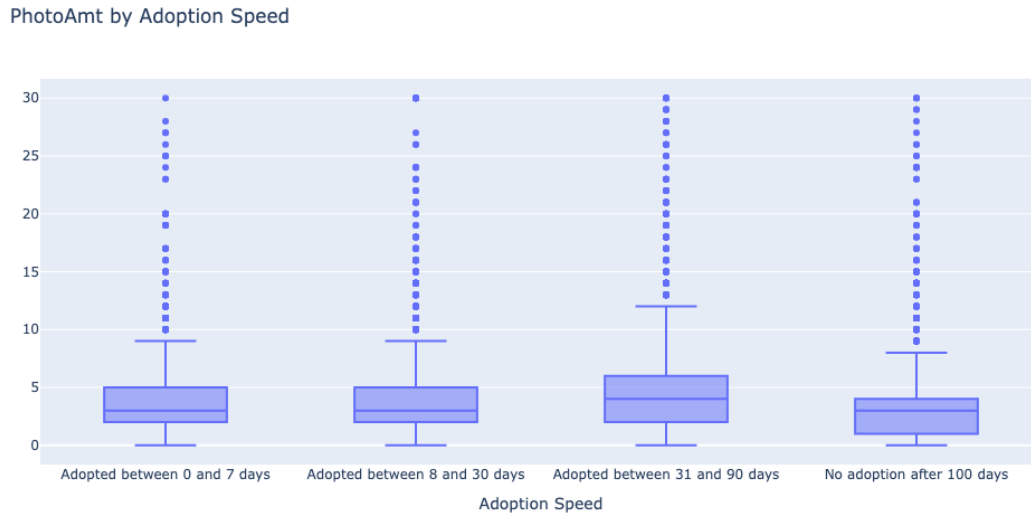


Figure 14: Boxplot of PhotoAmt by Adoption Speed

As seen in Figure 14, there is little relationship between the number of photos and the `AdoptionSpeed`, as the median for most categories of `AdoptionSpeed` is 3.

## 5.10 Relationship between Length of Description and Adoption Speed

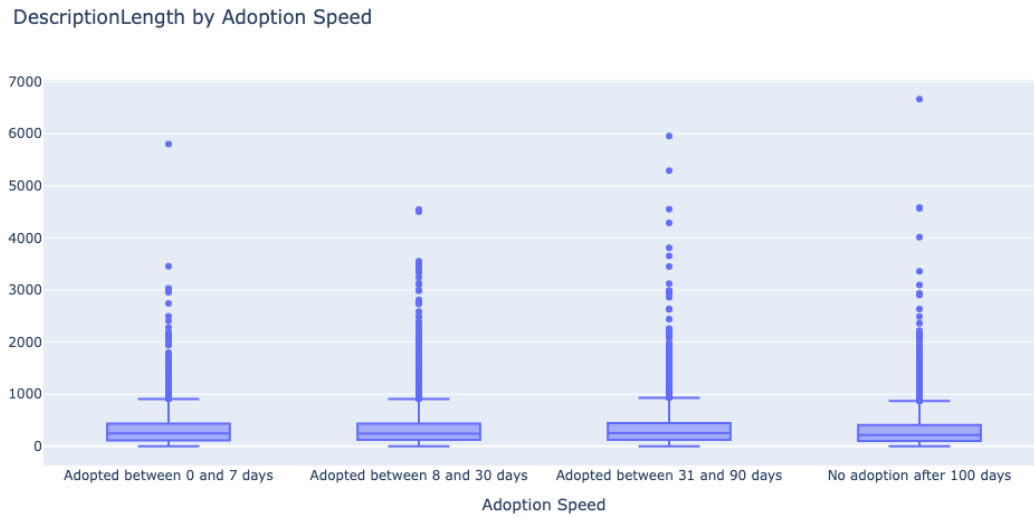


Figure 15: Boxplot of Description Length by Adoption Speed

Figure 15 shows that for the category of pets that have not been adopted, they have the lowest description text lengths with the median of 217, as compared to the other categories with close to 250 characters. We hypothesize that this could be because potential adopters value longer descriptions that contain more information about the pet or pets.

## 6 Feature Engineering

In the above sections, we have already added a few new features:

1. **AgeBinned**: Binning of Numerical Age into smaller categories.
2. **hasName**: Boolean indicating whether the pet was given a name.
3. **QuantityModified**: Binned quantities higher than 7 into one category 'Other'.
4. **Treated**: *True* if the pet is Vaccinated, Dewormed, and Sterilized.
5. **DescriptionLength**: Length of description text.
6. **LumpedFee**: Binned fees into the top 10 most frequent categories and other values into 'Other' category.

Now, let us explore some further feature engineering techniques in order to create new features and improve the performance of our models.

### 6.1 Other Feature Engineering for Tabular Data

1. **SizeAgeInteraction**: The combination of size and age could be important, given that potential adopters could have preferences for younger large dogs or older small dogs.
2. **RescuerActivity**: We counted the number of pets listed by the same rescuer and decided to categorize them into low, medium and high activity. This is because active rescuers might have more experience in getting pets adopted quickly.

## 6.2 Extracting Features from Pet Images

Apart from tabular data, images from each pet listing were provided. As each pet can have none or multiple images of different resolutions and aspect ratios, the following pre-processing steps are made:

- Pad to square aspect ratio
- Resize to 256 pixels

We used a DenseNet121 CNN model to extract 256 numerical features from the images. After which, we manually implemented recursive feature elimination (RFE) to reduce the feature set to the most relevant ones. This process involves iteratively training the model, evaluating the importance of each feature, and removing the least important feature at each step. The goal is to identify a subset of features that contribute most significantly to the model's predictive performance. We also added two additional features with regards to the pet adoption images:

1. **Blurriness:** Pet listings with images that are more blurry are more likely to see lower pet adoption rates. The function we used estimates the sharpness of an image using a Laplace filter and picks the maximum value.
2. **ObjectDetectionBox:** We decided to use a Yolo v3 model (a type of CNN for object detection) to detect the pet in the pet listing image and obtain the bounding box of the pet as 4 additional numerical features to be added into image data.

## 6.3 Using Grad-CAM to visualize DenseNet Model

To better understand how the DenseNet121 model above is extracting features from the images, we decided to use Grad-CAM (Gradient-weighted Class Activation Mapping). Grad-CAM is a technique to visualize the regions of an image that are important for a CNN when making a decision about a specific class. The heatmap generated by Grad-CAM is based on the gradients of a specific class score with respect to the feature maps of a convolutional layer, with bright areas indicating regions where changes to pixel values would most affect the class score, implying there are important features. Hence, these areas contribute most to the features extracted.

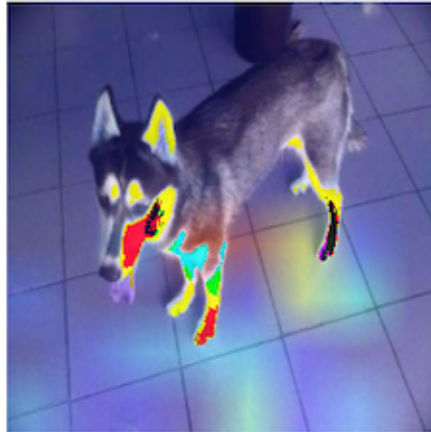


Figure 16: Grad-CAM Heatmap

As seen above, the brighter regions highlight parts of the pet that contribute the most to the features, notably the face, ears and legs.

## 6.4 Extracting Features from Description Text

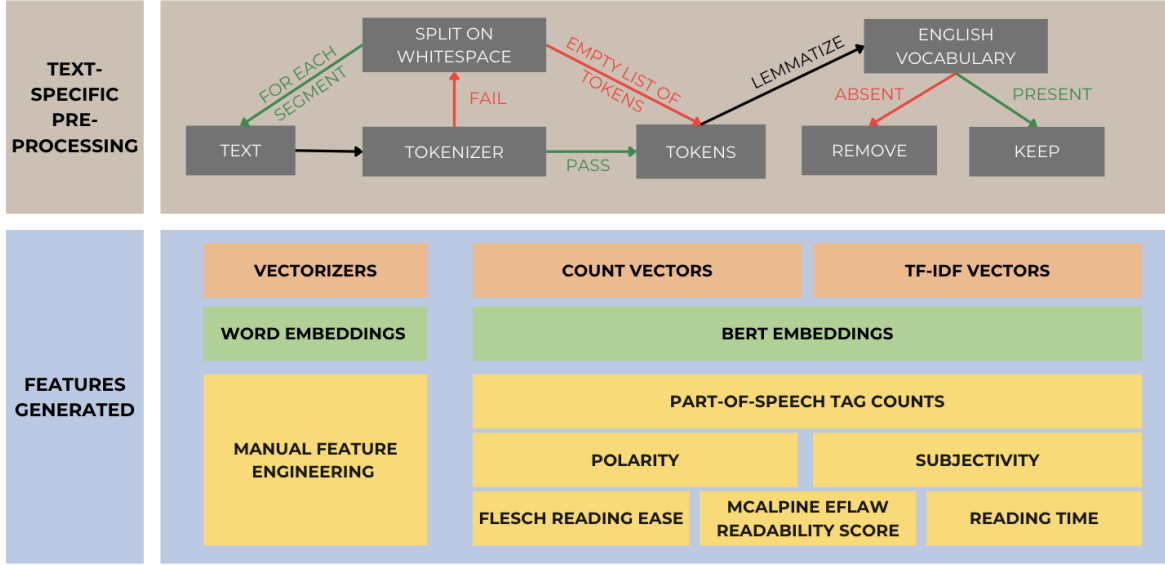


Figure 17: Text Processes Overview

The dataset is pet listings from Malaysia, where multiple languages are spoken, predominantly English, Mandarin and Malay. This implies the presence of non-English text in the descriptions of the pet listings. The first task in our preprocessing for the text is to identify these non-English portions and remove this. We do this by recursively attempting to tokenize a given text. Unsuccessful tokenization implies the existence of unsupported languages, which includes Mandarin. For these unsuccessful tokenization portions, we split the text on the whitespaces and apply the tokenization recursively on each portion. Once the Mandarin text has been filtered out, we then check if each remaining token’s lemmatized form is in the English vocabulary. After this process, we are left with only English texts. We found this method more successful than using libraries with language identification methods, however, it may filter out proper nouns which represent a pet’s name and may not be in the standard English vocabulary.

The first method of generating features from text is to use the simple counts and term frequency-inverse document frequency (tf-idf) of each token. The simple counts count the frequency of a given token in a text, while the tf-idf aims to capture how important each token is to each label based on its frequency. As each dimension of these vectors represent one token, the vectors produced for each sample is very sparse. Therefore, we limited the total number of tokens captured by the simple counts and tf-idf to the top 100 tokens each.

The second group of features we obtained is the Bidirectional Encoder Representations from Transformers (BERT) embeddings. BERT is a pretrained encoder model which can be finetuned for other types of Natural Language Processing (NLP) tasks. For our project, we obtained the BERT embeddings by passing our text input to the model and retrieving the output from its last hidden layer. The last hidden layer has an embedding dimension of 768, providing us 768 additional ‘features’. Although the BERT embeddings are useful, one drawback is that they are less explainable as compared to manually engineered features.

Our last group of features is the manually engineered features. First, we tag each token using the 36 Part-of-Speech (POS) tags from the Penn Treebank project, and 8 punctuation tags. Next, we generated features such as subjectivity, polarity, Flesch reading ease, McAlpine EFLAW readability score and reading time, using libraries TextStat and TextBlob. Subjectivity and polarity are characteristics of a text that can induce emotions in the reader, while features related to how easy a text is to read could impact the retention of the reader’s attention.

## 6.5 Recursive Feature Elimination

As there were many image features generated from DenseNet and text features generated from BERT, we manually implemented Recursive Feature Elimination (RFE) using a `RandomForestClassifier` to identify the most relevant features for predicting the adoption speed of pets based on their data. RFE is a feature selection method that recursively removes the least important features based on the model's feature importance scores, training the model with the remaining features at each iteration. In our manual implementation, we removed 10% of the least important features after each iteration, repeating this process until no features remain.

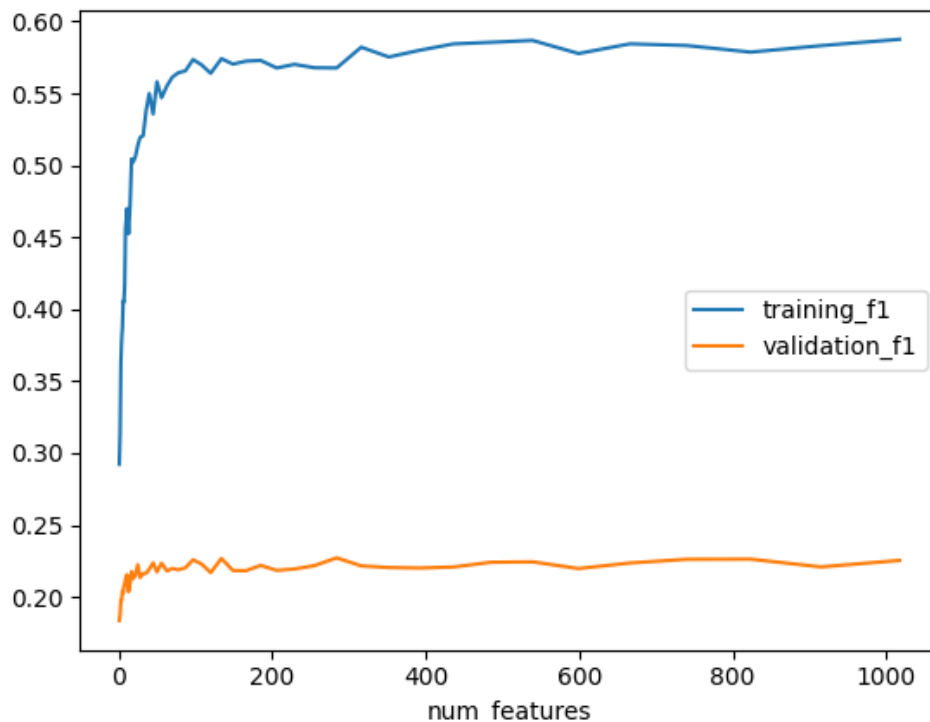


Figure 18: Recursive Feature Elimination Plot

The above figure shows the elbow plot, which plots the number of features against the F1 scores. The "elbow" indicates a point beyond which removing more features leads to a significant drop in performance. This point is considered optimal as it represents a good trade-off between model simplicity and model performance. As seen in the above plot for image features, we preserved the top 15 most important features from the DenseNet model. We did the same manual RFE process for the text features as well, which selected 19 BERT Embedding dimensions and the 3 features related to reading ease.

## 7 Individual Modeling

### 7.1 Encoding Categorical Features

Before building the model, we will use One-Hot Encoding and `LabelEncoders` to deal with categorical variables in our data. Categorical features which are ordinal such as `MaturitySize` (small, medium, large) would be label encoded to preserve their ordering while the other categorical features would be one hot encoded.

### 7.2 Scaling Features

We utilize `MinMaxScaler` from `scikit-learn`, which scales and translates each feature individually such that it is in the given range on the training set, typically between zero and one. The transformation

is given by the formula:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

where  $X_{\max}$  and  $X_{\min}$  are the maximum and minimum values of the feature respectively. This would ensure that the model is not biased towards features with a larger scale.

### 7.3 Evaluation Metric: Quadratic Weighted Kappa and F1 Score

Our models will be evaluated with **quadratic weighted kappa**, which measures the amount of agreement between an algorithm's predictions and true labels. This metric varies from 0 (random agreement) to 1 (complete agreement). When there is less agreement than expected by chance, the metric may go below 0. This metric is suitable since the category being predicted is ordinal and is to penalize predictions that are off by multiple categories. Our notebook implements the calculation of this metric manually. You may refer to this [notebook](#) for more information regarding the calculation of this metric.

We also used F1 score as another metric to evaluate our models, as it provides a balance of both precision and recall.

### 7.4 Baseline Model: Linear Regression

A baseline linear regression model obtained a score of 0.344 for the quadratic weighted kappa and 0.379 for the macro-average F1 score.

### 7.5 Hyperparameter Tuning: Bayesian Optimization

Instead of using the traditional Grid Search or Random Search for hyperparameter tuning, we used Bayesian Optimization. It utilizes a probabilistic model to predict the performance of the models based on the hyperparameters, and then it chooses new hyperparameters to test by balancing exploration of new areas with exploitation of known good areas. Bayesian Optimization is well-suited for ML models like Random Forests and XGBoost, where the hyperparameter space can be vast and the relationship between hyperparameters and model performance is complex and non-linear.

iter	target	max_depth	min_sa...	min_sa...	n_esti...
1	0.3717	47.64	1.906	9.727	456.9
2	0.3737	39.39	1.011	6.647	914.2
3	0.3696	25.52	4.927	13.28	986.7
4	0.3607	46.55	2.215	13.0	165.2
5	0.3786	40.35	2.002	3.248	946.3
6	0.3716	42.17	3.073	13.36	599.5
7	0.3688	25.25	3.721	7.453	658.4
8	0.3716	32.39	1.422	14.56	770.8
9	0.3771	35.58	2.317	2.344	813.5
10	0.3629	48.78	2.132	10.44	302.3
11	0.3641	13.45	2.676	10.66	541.5
12	0.3693	43.45	1.684	14.72	336.2
13	0.3713	15.46	4.167	13.49	987.9
14	0.3197	5.723	1.818	10.39	535.1
15	0.3734	14.46	2.236	2.229	662.6

Figure 19: Bayesian Optimization

Figure 19 shows a snippet from our code notebook of Bayesian Optimization. It experiments with different hyperparameters and tracks the combination with the best target metric, quadratic weighted kappa in this scenario. We used the `bayes_opt` package for the Random Forest and XGBoost models and the `optuna` package for the Pytorch NN.

## 7.6 5-Fold Cross Validation

We decided to use cross validation instead of a simple train-validation-test split due to the limited amount of rows our dataset contains (14993 rows). Cross validation is also more effective than validating on one specific validation set.

For the Random Forest and XGBoost models, we used the `cross_val_score` function and for the Pytorch NN, we manually implemented cross validation by splitting the data using `StratifiedKFold` and taking the average validation score calculated on each fold.

## 7.7 Random Forest Classifier

The first model chosen is the Random Forest Classifier, which is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. We chose this model as it is known for its robustness and ability to handle non-linear data effectively.

<code>max_depth</code>	25.0
<code>min_samples_leaf</code>	1.0
<code>min_samples_split</code>	2.0
<code>n_estimators</code>	683

Table 1: Random Forest Classifier Hyperparameters

Based on the optimized hyperparameters, a `max_depth` of 25.0 was selected for the decision trees. This depth is moderate enough to capture complex patterns in the data while still being restrained to help prevent overfitting. This balance is crucial for maintaining generalizability of the model to new, unseen data. The optimal `n_estimators` determined was 683, which is the number of trees built before taking the maximum voting of predictions.

Our random forest model achieved a Quadratic Weighted Kappa score of 0.386 and a F1 score of 0.423 on the heldout test set.

## 7.8 XGBoost Classifier

The next model is XGBoost, which is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. We chose this model due to its performance and speed, which has made it a popular model in machine learning competitions such as on Kaggle.

<code>colsample_bytree</code>	0.848
<code>learning_rate</code>	0.130
<code>max_depth</code>	25
<code>n_estimators</code>	288
<code>reg_alpha</code>	9.60
<code>reg_lambda</code>	94.3
<code>subsample</code>	0.828

Table 2: XGBoost Classifier Hyperparameters

As seen in the optimized hyperparameters above, some interesting values would be the high `reg_alpha` and `reg_lambda` values, which indicate a strong emphasis on both L1 and L2 regularization. This suggests that there is a significant penalty on model complexity to reduce overfitting. This is supported by the `colsample_bytree` value of 0.848, which means that each tree in the ensemble uses approximately 84.8% of the features, ensuring that the model is robust to noise.

Our XGBoost model performed slightly better than the Random Forest model, achieving a Quadratic Weighted Kappa score of 0.429 and a F1 score of 0.446 on the heldout test set.

## 7.9 Neural Network Classifier

Our last model is a Neural Network (NN) implemented using Pytorch. It consists of the following layers:

- **Input Layer** that matches the dimensions of the feature space
- **Dense Layers**, with each dense layer followed by batch normalization and dropout for regularization to prevent overfitting
- **ReLU** (Rectified Linear Unit) activation functions are used throughout the model to introduce non-linearities
- The **Output Layer** consists of 4 neurons with a 'softmax' activation function to cater to the multi-class classification, representing the probability distribution over the 4 classes
- Training process includes **early stopping** to prevent overfitting by monitoring the validation loss and stopping the training if the validation loss does not improve for a predefined number of epochs

Similar to the other models, we used Bayesian Optimization to tune the hyperparameters such as activation function, optimizer and number of dense layers.

optimizer	RMSprop
activation_function	relu
num_layers	3
hidden_units_layer_0	128
hidden_units_layer_1	16
hidden_units_layer_2	128
dropout_rate	0.420
lr	0.000975
batch_size	64

Table 3: Neural Network Classifier Hyperparameters

It can be seen in Table 3 that the optimal number of layers is 3, which starts with a high number of units (128), has a significantly smaller number (16) in the middle layer, and then expands these refined features back to a higher-dimensional space (128 units) for the final classification task. The “**bottleneck**” layer can help the network to compress or refine the features extracted by the first layer, potentially improving the feature representations and aiding in noise reduction. A dropout rate of 42% is quite substantial, which would prevent overfitting. RMSprop was selected as the optimizer, it is an adaptive learning rate method which allows it to converge faster on training data by adjusting the learning rate for each weight of the NN individually.

The performance of the NN on the test set was worse than the other 2 models, obtaining a Quadratic Weighted Kappa score of 0.392 and a F1 score of 0.386.

## 7.10 Model Performance Comparison

Model	Quadratic Weighted Kappa	F1 Score
Linear Regression	0.344	0.379
Random Forest	0.386	0.423
XGBoost	<b>0.429</b>	<b>0.446</b>
Neural Network	0.392	0.386

Table 4: Test Metrics



XGBoost shows the highest performance, with a QWK score of 0.429 and a F1 Score of 0.446. This can be attributed to XGBoost's ability to handle different data types efficiently, making it versatile for complex datasets. The L1 and L2 regularization prevent overfitting more effectively compared to basic ensemble models like Random Forest.

Although the Neural Network ranks second in terms of QWK, it has the lowest F1 score. As NNs tend to require large amounts of data to capture complex patterns effectively, the amount of training data given (around 11,000 rows) might have been too little for the model. Additionally, NNs perform much better than other models when dealing with datasets with a huge amount of features (examples being images), since we did feature selection and filtered out the more important features, the value of NNs would decrease.

Random Forests have the lowest QWK score but a moderate F1 score. As Random Forests prioritize generalization over precision, this would impact its performance for more complex decision boundaries.

All three models significantly outperform the baseline linear regression as expected, which is unable to capture complex relationships present in the data.

### 7.11 Ensemble: Weighted Averaging

Combining the three models above would mitigate the individual weaknesses and variances of each, potentially leading to a more accurate and stable prediction on unseen data. The ensembling method we used is **Weighted Averaging**.

Instead of averaging the predictions of each model equally, the models are combined in a weighted manner where the weights are derived from their QWK scores obtained from cross validation.

$$\text{weight}_i = \frac{\text{kappa\_score}_i}{\sum \text{kappa\_scores}}$$

Above is the formula used to calculate the weights for each individual model. This method ensures that models which perform better (i.e. have higher QWK validation scores) have a greater influence on the final ensemble prediction.

Model	Quadratic Weighted Kappa	F1 Score
Ensemble	0.428	0.435

Table 5: Ensemble Performance on Test Dataset

As seen above, while the performance of the model ensemble is slightly worse than the XGBoost model, this might be attributed to the specifics of this particular test set where XGBoost is particularly suited for. By ensembling the models, this could lead to better generalization to unseen data in general, mitigating the risk of overfitting and improving the stability of predictions.

## 7.12 Model Architecture

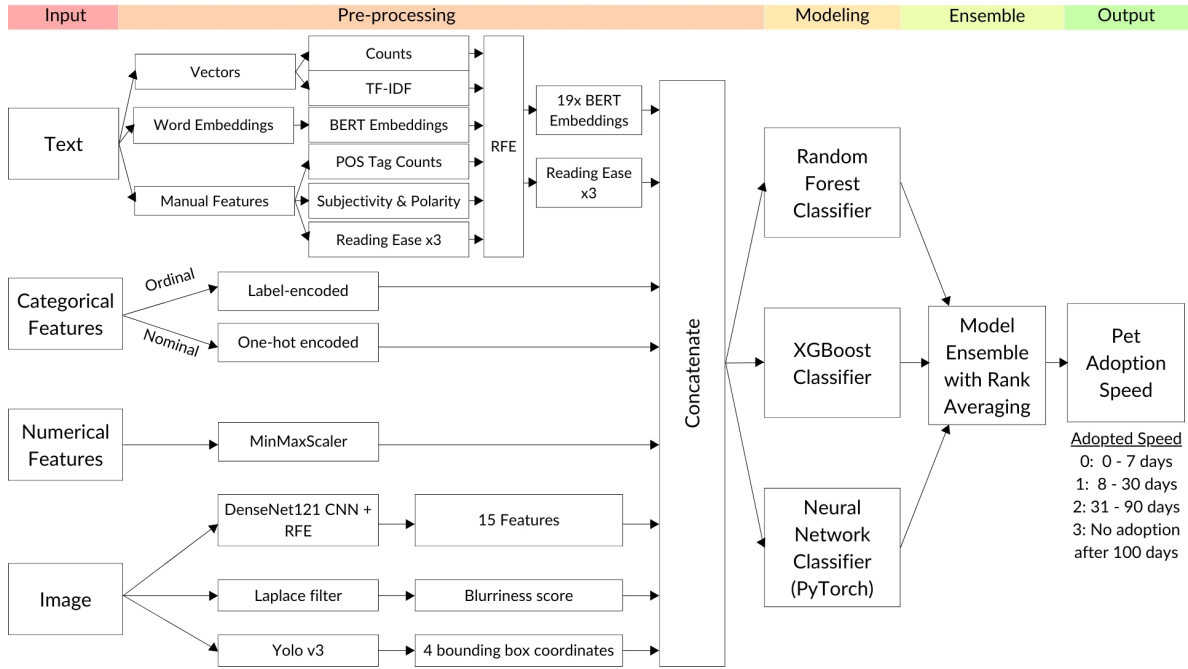


Figure 20: Model Architecture

## 8 Evaluation of Model

To evaluate the performance of our model, we will be comparing its predictions against GPT 4 Vision API as well as the entries from the related Kaggle competition.

### 8.1 GPT-4 Vision API

This evaluation can be found at the bottom of our Python Notebook. We first filtered out 20 random rows in our original training dataset along with the text description as well as adoption listing photos. Using the GPT-4-Vision OpenAI API, we first provided the 4-way classification task context and the different categories to predict. We also provided a description of each feature in the original tabular data, before prompting the API to respond with the classification prediction.

Model	Quadratic Weighted Kappa	F1 Score
GPT-4 Vision API	0.212	0.125

Table 6: GPT-4 Vision API Performance

After which, we evaluated GPT-4 Vision's predictions using Quadratic Weighted Kappa and F1 score metrics. As seen above, it performs significantly worse than our model ensemble scores of 0.428 (QWK) and 0.435 (F1).

This evaluation demonstrates that while GPT-4 Vision is a powerful tool, particularly for tasks involving natural language understanding and generation, it may not always excel in specialized tasks like 4-way classification based on pet adoption data given. This highlights how specialized models such as our model can offer superior performance by leveraging domain-specific knowledge and targeted data processing techniques.

## 8.2 Kaggle Leaderboard

The pet adoption listing dataset and images obtained were used in a Kaggle competition which can be seen [here](#). The competition metric used was Quadratic Weighted Kappa as well.






















PetFinder.my Adoption Prediction							Late Submission	...
Overview	Data	Code	Models	Discussion	Leaderboard	Rules	Team	Submissions
91	rookzeno			0.42788	2	5y		
92	tj_bh			0.42788	2	5y		
93	111			0.42787	2	5y		
94	BrownBear			0.42778	2	5y		
95	Costas Voglis			0.42778	2	5y		
96	Dull Wolf			0.42777	2	5y		
97	Katsunori Nakai			0.42774	2	5y		
98	Thomas SELECK			0.42771	2	5y		
99	sridhark8			0.42768	2	5y		
100	Lars L			0.42757	2	5y		

Figure 21: Kaggle Leaderboard

Comparing our QWK score on our test set (0.42772) to the leaderboard, our score would have placed 98th out of 1788 total teams, which would have earned us a silver medal (awarded to the top 5% of teams). While this is not entirely accurate given that the test dataset used in the competition is different from our test dataset, this provides a good indicator of how well our model performs on unseen data and attests to the strength of our feature engineering techniques as well as model ensemble.

## 9 Model Explainability: SHAP Values

In this section, we aim to provide an explanation as to how a model derives its prediction by using SHAP. SHAP values are defined with respect to a label. A positive SHAP value implies that the feature contributes positively towards a prediction for the label, and vice versa. Using SHAP values, we focus on important features that the model identifies and analyse the results. For this analysis, we will be focusing on the label representing adoption within a week, and the model used is the XGBClassifier model.

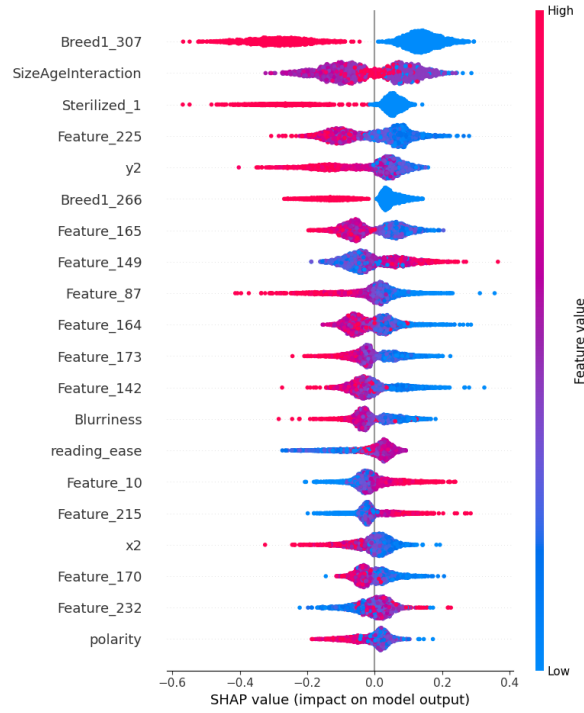


Figure 22: SHAP Summary Plot for Label 'Adopted within a Week'

From the above SHAP summary, we notice 11 features are features generated from image feature engineering with DenseNet121, while there are no features that were generated by BERT embeddings. This suggests that for a fast adoption, the content of the image plays a much more significant role than the content of the text description. Related to images is the feature y2, which represents the upper y-coordinate of the bounding box. When this feature has a low value, it also has a low SHAP value, however, when the feature value is high, the SHAP value tends to be negative. This suggests that if the pet in the image is not in frame vertically, it hinders the potential of a fast adoption. Another interesting feature to analyze is the binary feature Breed1\_307, which is the one-hot encoded feature for the mixed breeds of dogs. The SHAP values suggests that when the dog is a mixed breed, it has a negative impact on a fast adoption, suggesting adopters prefer pure breed dogs. One feature related to text in the top 20 features identified by the SHAP framework is reading ease. When the reading ease is high, there is no significant impact in terms of SHAP value. However, when the reading ease is low, it tends to have negative SHAP values, suggesting that long-winded descriptions should be avoided by posters.

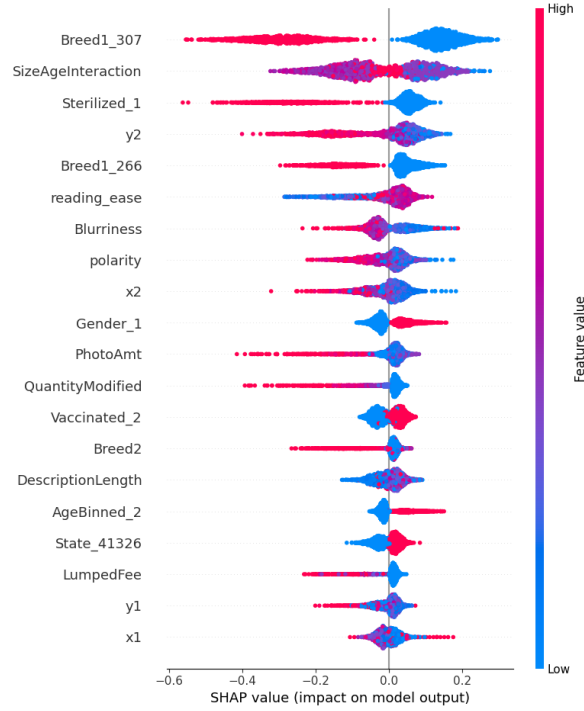


Figure 23: SHAP Summary Plot for Label 'Adopted within a Week' with Explainable Features Only

When we remove the categories that are less explainable and observe the top 20 features, we see some interesting relationships. The first is the feature **QuantityModified** where 1 represents that there are more than 8 pets in a listing. When this feature is activated, it seems to impact the chance of adoption severely. This suggests that posters who are posting a litter of kittens for example, should instead consider posting them individually rather than as a group for better success. At the same time, we observe some curious features reported, such as **Sterilized\_1**, where a 1 instance implies it is sterilized, and **Vaccinated\_2**, where a 1 instance implies it is not vaccinated. The SHAP values reported imply that a unsterilized and unvaccinated pet is preferred. However, we should not take this at surface level and explore a little deeper.

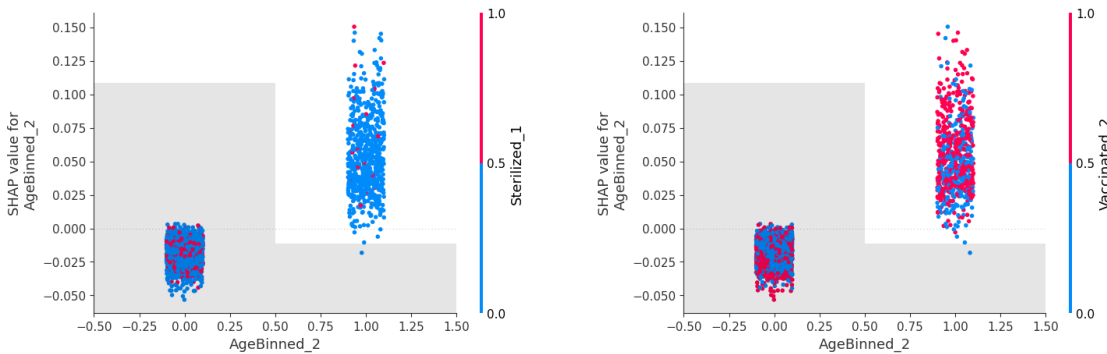
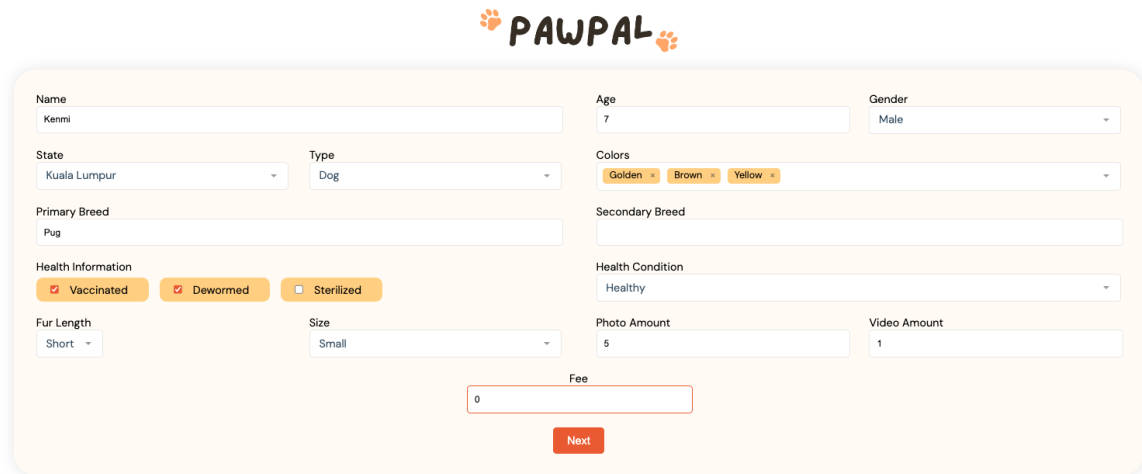


Figure 24: Interaction between Age (2 Months) and Sterilized, not Vaccinated

From the above figures, we observe that 2 month old pets (denoted by a 1-instance for **AgeBinned\_2**) are likely to not be sterilized and not vaccinated. The positive SHAP values assigned to the two features discussed could be due to the correlation between unvaccinated, unsterilized pets to younger pets. A similar graph can be plotted for the **AgeBinned\_0-2** category with similar results.

## 10 Pet Adoption Tool

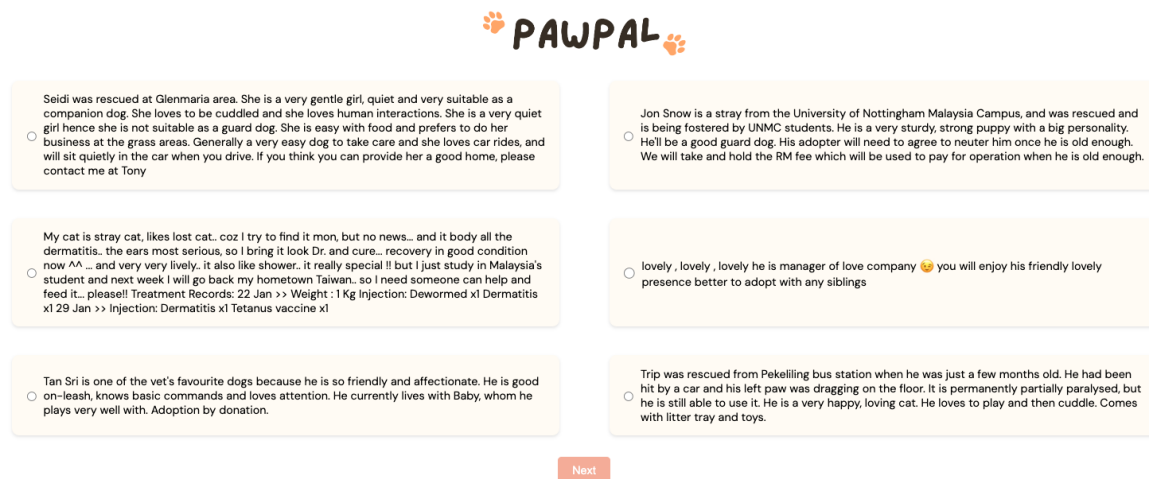
For our initiative, we built a full-stack web application entitled PawPal deployed here (<https://pet-adoption-tool-ml.web.app/>). It was built using Vue.js and Flask. The code repository can be found [here](#). The target audience of this pet listing adoption tool is pet owners or adoption centres. The goal of the website is to provide an evaluation of a specified pet adoption listing and image, as well as provide suggestions on how they can improve the predicted adoption speed of their pet.



The PawPal Text Input Page is a form for entering pet details. It features the PawPal logo at the top. The form is divided into several sections: Name (Kenmi), Age (7), Gender (Male), State (Kuala Lumpur), Type (Dog), Colors (Golden, Brown, Yellow), Primary Breed (Pug), Secondary Breed, Health Information (Vaccinated, Dewormed, Sterilized), Fur Length (Short), Size (Small), Photo Amount (5), Video Amount (1), and Fee (0). A 'Next' button is at the bottom.

Figure 25: PawPal Text Input Page

Figure 25 shows the text input page for PawPal, where users can key in information about their pet such as the name, gender and breeds. Drop-down menus allow users to select fields such as color and users can also input listing details, such as the number of photos and the adoption fee.



The PawPal Text Selection page displays six pre-selected description texts for adoption listings. Each text is preceded by a radio button. The texts are: 1. Seidi was rescued at Glenmaria area. She is a very gentle girl, quiet and very suitable as a companion dog. 2. Jon Snow is a stray from the University of Nottingham Malaysia Campus, and was rescued and is being fostered by UNMC students. 3. My cat is stray cat, likes lost cat. 4. Tan Sri is one of the vet's favourite dogs because he is so friendly and affectionate. 5. Trip was rescued from Pekelling bus station when he was just a few months old. 6. lovely, lovely, lovely he is manager of love company. A 'Next' button is at the bottom.

Figure 26: PawPal Text Selection

Users can then select from a few pre-selected description texts from a test dataset (separate dataset from the data the model was trained on) to include in their adoption listing.



Figure 27: PawPal Image Selection

Similarly, users will then be prompted to pick an image for their adoption listing. While we did consider allowing users to key in their own description text and upload their own images to fully enhance the interactiveness of PawPal, our backend BERT models, CNNs and Object Detection models would take considerable time to run and generate features, hence we decided on provided pre-selected text and images which we have already extracted the numerical features from.

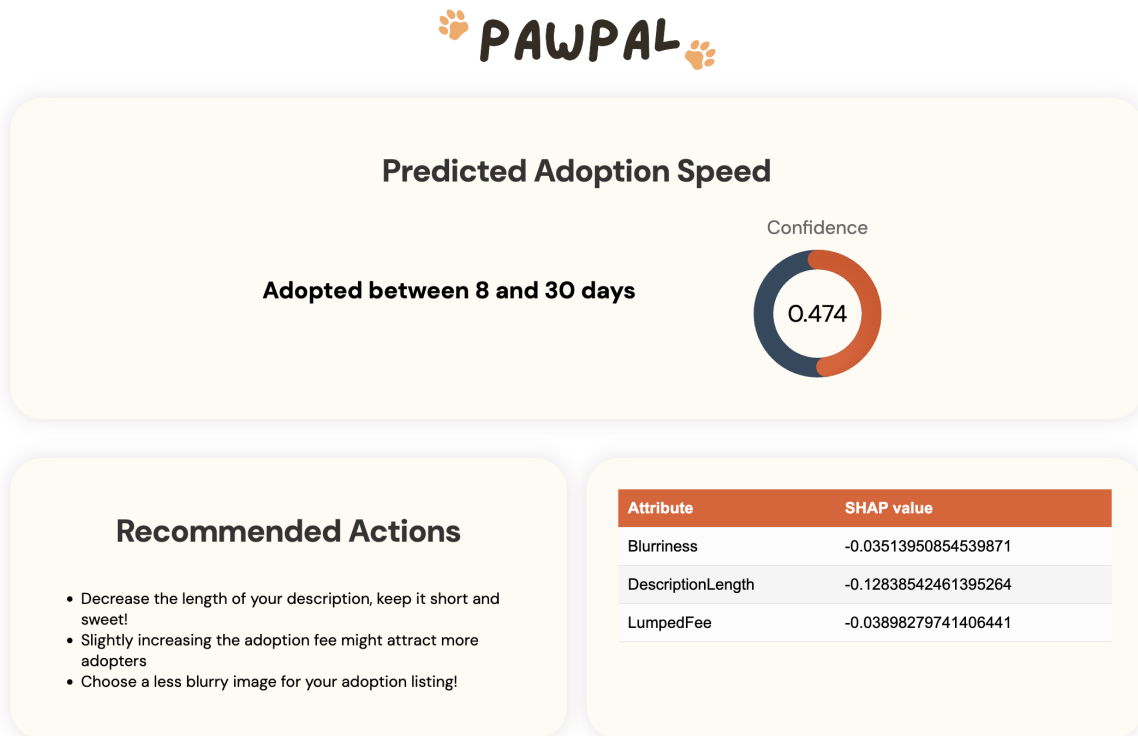


Figure 28: Output Predictions and Recommendations

Above is our model's prediction on the user inputted pet adoption listing. It gives a predicted class as to how long the pet would take to be adopted and also a confidence level of the model in the prediction, which is obtained by the probability of the predicted class from the softmax function.

Below, we provide some recommended actions should the predicted classification of the pet not fall

in the category of “Adopted within a week”. These recommendations are based on the SHAP values of Class 0, which is the ideal category with the fastest adoption speed. We filtered out the top few negative SHAP values of categories that are modifiable as these features are “contributing” the most to the predicted class not being 0. After which, we convert these to recommendations by taking into account the pet adoption listing. As seen in Figure 28, one of the recommendations was to decrease the length of the description text, as the text selected was too long and “DescriptionLength” has one of the most negative SHAP values.

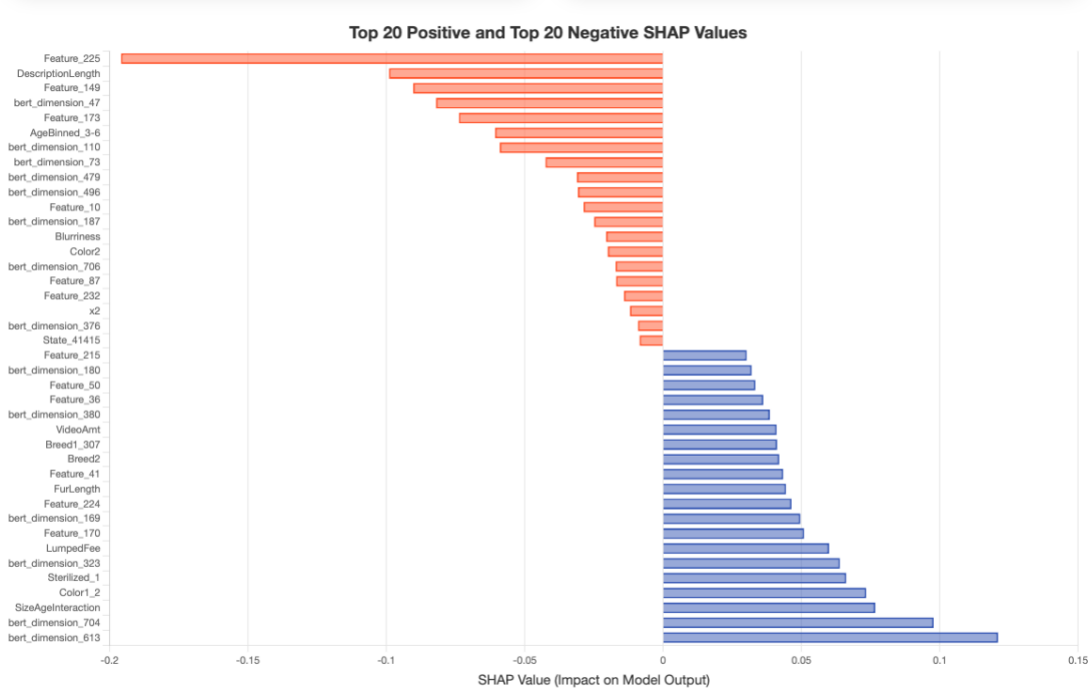


Figure 29: Output SHAP Waterfall Plot

We also provide a waterfall plot containing the top 20 positive and top 20 negative SHAP values for the predicted class as seen above. To interpret this, the features that have a positive SHAP value contributed the most to the model’s prediction for that class while the converse is true for the negative SHAP values.

## 11 Future Development

While the current project focuses on predicting adoption rates and providing suggestions for improving listings, there are several avenues for future development to further enhance the effectiveness of the adoption process. This includes the **generation of pet description** based on the image and tabular data using transformer-based language models, such as GPT-4 or BERT. We could then fine-tune these models on the specific domain of pet adoptions, allowing them to learn the vocabulary, writing style, and nuances of pet descriptions.

Once the fine-tuning process is complete, our pet adoption tool would be able to generate personalized descriptions by taking in user-provided data, such as pet images and tabular information (breed, age, size, etc.). This input data would be preprocessed and converted into the appropriate format for the fine-tuned language model. Ultimately, the generated descriptions would aim to highlight the unique characteristics and personalities of each pet, making them more attractive to potential adopters.



## 12 Ethical Concerns and Conclusion

We ensured that our recommendations only included features that could be changed, such as increasing the number of photos in the listing or changing the fee. While SHAP values also show how the characteristics of the pet would affect the classification (e.g. certain breeds may have a negative SHAP value for class 0), recommending based on these characteristics that cannot be changed could lead to unethical results. If adoption centres were to “prioritize” pets of certain breeds or gender, this would mean unethical misuse of our tool, which is what we are trying to avoid.

Hence, PawPal’s main goal is to guide users and adoption centres on better pet profile and adoption listings creation. By using our tool, we as a group hope that more pets can find their homes and that all pets, regardless of their breed or other immutable characteristics, are given an equal chance for adoption. By carefully considering the ethical implications of our recommendations and focusing on adjustable aspects of pet profiles, we ensure that our technology promotes fairness and equality. Our commitment is to enhance the effectiveness of pet adoptions through thoughtful and ethical use of data, helping every pet find a loving home.

## 13 Team Reflections and Acknowledgements

Through this ML project, our team learnt how to apply many of the concepts taught in class and use ML to solve practical real-life problems. We also learnt a lot in other areas such as Feature Engineering, Extracting Text Features, Hyperparameter Tuning, just to name a few. Our team is very proud of our above report, the strong performance of our predictive models as well as the full-stack adoption tool that we built over the course of a few weeks.

We would like to thank Prof. Xavier Bresson as well as our Project TA Zhiyuan Hu for their kind guidance and help throughout the module and the duration of the project. Their insights and feedback were invaluable in helping us enhance our project and improve our skills. Moving forward, we are excited to apply the lessons learned in this project to future endeavours and continue developing ML models that make a positive impact.

## References

- [19] *PetFinder.my Adoption Prediction* — Kaggle. <https://www.kaggle.com/competitions/petfinder-adoption-prediction/data>. Apr. 2019.
- [You22] L. Youjin. “Pet abandonment generally rose in past five years, half of cases involving cats: MND”. In: *TODAY* (Mar. 2022). <https://www.todayonline.com/singapore/pet-abandonment-generally-rose-past-five-years-half-cases-involving-cats-mnd-1831671>.
- [Koh23] S. Koh. “Animal shelters see up to 50% fall in pet adoptions as more people return to office”. In: *The Straits Times* (Sept. 2023). <https://www.straitstimes.com/singapore/drop-in-pet-adoption-rates-as-more-people-return-to-office-animal-shelters>.