Popularity Prediction using Pet Image Classification

Manthan Mehta
Khoury College of Computer Sciences
Northeastern University
Boston, MA, 02215
mehta.manth@northeastern.edu

Riddhi Narayan
Khoury College of Computer Sciences
Northeastern University
Boston, MA, 02215
narayan.ri@northeastern.edu

Yash Bhojwani
Khoury College of Computer Sciences
Northeastern University
Boston, MA, 02215
bhojwani.y@northeastern.edu

Abstract

Since the last decade, adoption shelters have taken to the web to promote photos of their rescue animals. However, many of such rescues do not capture much attention from the people on the internet viewing them. In this project, we intend to find the essential features that contribute towards making an image of the animal popular on the internet. This inference will thereby provide suggestions into posting photos that garner more attention and thereby increase the probability of the rescue animal getting adopted. The original regression problem was converted into a classification problem by binning the 'Pawpularity' scores into four different classes for better interpretability. These four classes signify the speed of adoption. Using either the images or the metadata solely is not enough for a model to accurately predict the class of an image. Thus, we propose an ensemble using both the images and the metadata features to generate better results. In addition, we have added about 14000 images from a separate dataset to enhance the network's performance by working under the assumption that training on more data yields better results.

1 Introduction

Millions of pets require foster homes. However, not all animals get equal attention on the internet. This is because some images and particular breeds are more appealing than others amongst all the posted animal photos online. Let us call the term - 'Pawpularity Score,' which refers to the attention that a particular image has on the internet. What if we had a way of knowing whether a pet was going to go unnoticed before publishing the photo? This knowledge could then be used to post the photo in such a way that includes all the essential features that contribute to a 'popular' photo. The 'Pawpularity Score' is derived from each pet profile's page view statistics at the listing pages, using an algorithm that normalizes the traffic data across different pages, platforms (web mobile), and various other metrics. We are, thus, primarily trying to predict this popularity score.

Firstly, this is an exciting project for the simple fact that we all love pets, and working with such a dataset was an enriching experience for us. Secondly and more importantly, it is still an unsolved and ongoing problem with scope for improving the RMSE score. The project will help drive more attention to photos of unnoticed pets, thereby increasing their chance of being adopted into foster homes. This problem, if scaled, could then have a tremendous social impact.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

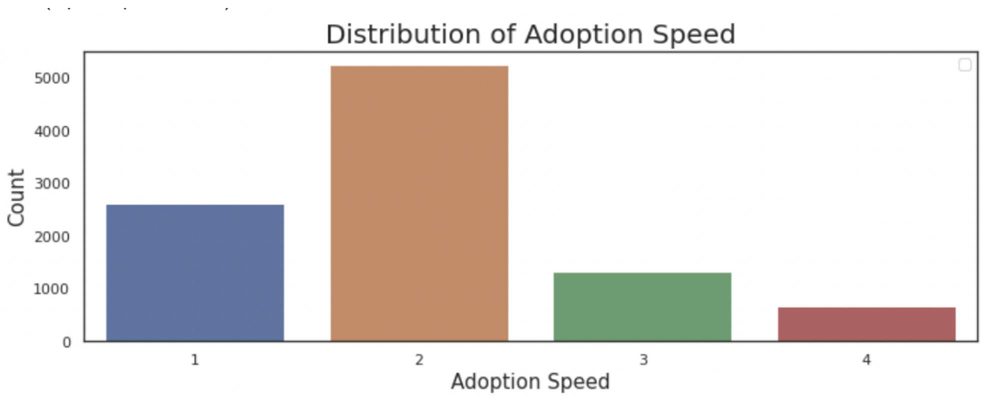


Figure 1: Distribution of ’Pawpularity’/ Adoption Speed

The current method that the organization is using uses a cuteness meter to rank the photos and compare them to other pet profiles. This method fails since there is no interpretation of what is wrong with lower-ranked pet photos. In addition, due to the low availability of data and proposed optimal solutions, we were forced to think out of the box. Lastly, the limited number of resources available for us to refer to (as it is an ongoing Kaggle competition) makes for a challenging task.

The previous methods use Regression on metadata. The methods have failed to consider the image features that can retrieve valuable contextual information that denotes ’cuteness’/’popularity’ of the image.

Augmenting new data is a challenge since we do not measure accurate labels for the new data. Our approach uses a fusion stream between metadata and image data to retrieve contextual information and spatial features. The key emphasis is on interpretability. We are trying to analyze and deduce the most influential features from the metadata and the actual image.

2 Preliminaries

2.1 Dataset and Features

The dataset consists of the following attributes: ’Id’, ’Subject Focus’, ’Eyes’, ’Face’, ’Near’, ’Action’, ’Accessory’, ’Group’, ’Collage’, ’Human’, ’Occlusion’,’Info’, ’Blur’, ’Pawpularity’. The ’Pawpularity’ attribute has been transformed into an attribute called ’Adoption Speed,’ which has values that range from 1-4, and the histogram for this attribute is as seen as in 1. The higher the number, the greater the chance of adoption. This has been done by binning the ’Pawpularity’ scores in the following way: 0-25 (Pawpularity-1), 25-50 (Pawpularity -2), 50-75 (Pawpularity-3), and 75-100 (Pawpularity-4).

Each feature in the dataset is a binary value: i.e., 0 or 1. 1-signifying presence of the feature in the image and 0 otherwise. The count of each feature in the dataset is as observed in 2:

The correlation of the features with ’Pawpularity’, quantified using Spearman’s p-value, is seen to be minimal- between 0 and 0.25.
To further show the distribution of the features,3 and 4 show the violin and KDE plots of the metadata features ’Eyes’ and ’Face’. The distribution plots of the remaining ten metadata features can be found at : `https://docs.google.com/presentation/d/1eYFzwFEoKgjeaxnK4agENvpkmXAOXzEXMpn-NZl92x0/edit?usp=sharing`

The Violin and KDE plots in 3, 4 and on the link attached above are almost identical within each feature. This shows that the ’Pawpularity’ is not massively dependent on these features. This, in turn, implies that it would be difficult to train any algorithm to predict ’Pawpularity’ based on these

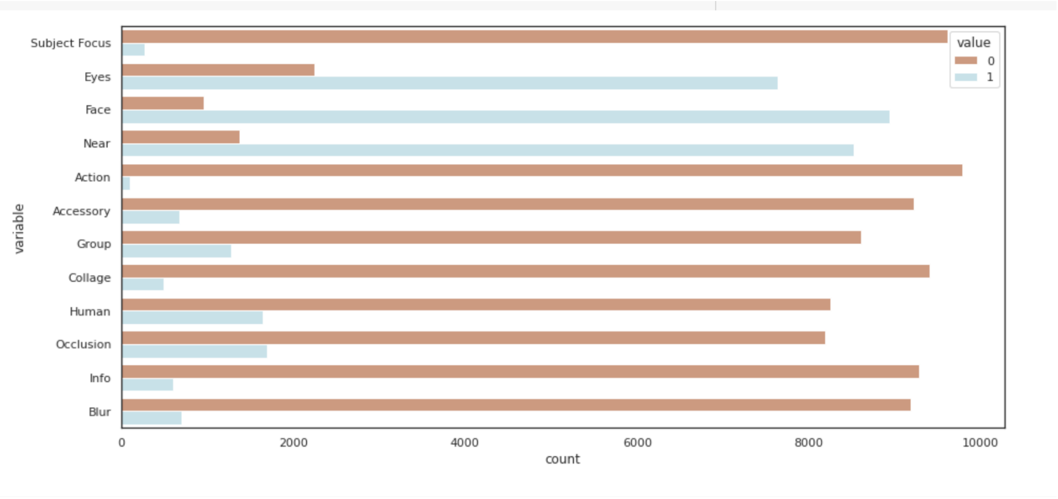


Figure 2: Count of Metadata Features

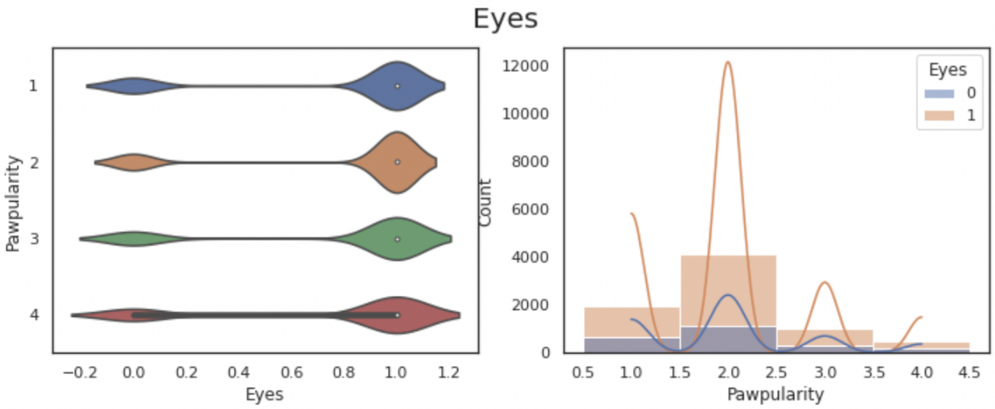


Figure 3: Violin and KDE plots for ’Eyes’

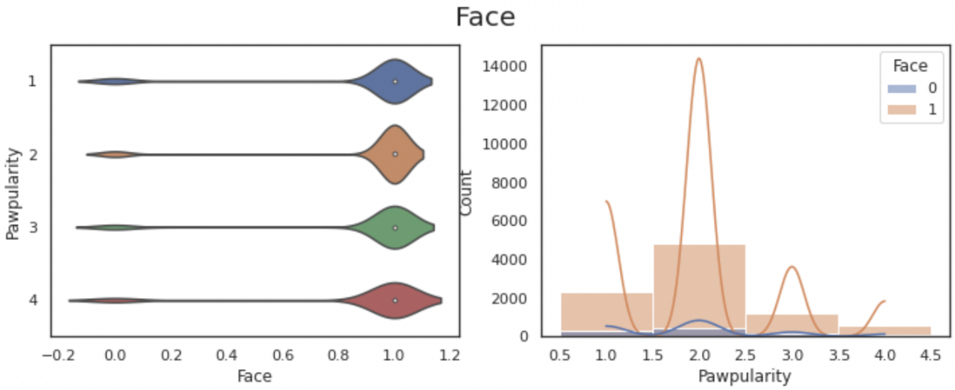


Figure 4: Violin and KDE plots for ’Face’

metadata features alone. Hence, we can deduce that we might also have to add images.

Preliminary preprocessing steps involved:

- The mean value of the 'Pawpularity' score(real-valued) was approximately 20.2, and the best RMSE score on the dataset was approximately 17.3. Therefore, it made sense for us to build a model that predicts the average value at all times. There are images in the dataset that look almost similar but yet, have a different score. To address this randomness in the scores, we converted the 'Pawpularity' score attribute to a categorical variable between the range of 1-4.
- Later, duplicate images were removed using a hash scoring that can be found at [1]. Twenty-seven images were found with different scores, and these were then manually removed.
- Two separate train datasets were formulated with 9885 and 14000 images. These two datasets were then merged to form the master training set.Dataset link
- Approximately 23 non-RGB images were dropped from the master dataset. This was done to nullify the unnecessary noise that such images contributed to during the modeling process.
- The final step was splitting the dataset into train and test set containing 7908 and 1977 images. (We will use the 9885 images only to model our further experiments.)Train data. Test data

2.2 Experimental setup

We aim to demonstrate a sequential study that aims at solving the problem by using the least complex approach. We first begin with modeling solely the 12 metadata features in the image(such as Blur, Eyes, Focus, and so forth). The initial step was to set up a baseline and later investigate the usefulness of the features. We then proceeded with training only the images using pre-trained models. Augmentation techniques like Mixup, Cropping, Rotation, and so forth were implemented at this step. Later, we trained the network with both the binary features, i.e., the meta-data and the image data. To further improve the performance, we augmented the current dataset with 14,000 more images, assuming that deep learning models tend to converge better with more training data. We used the Kaggle kernels and the discovery cluster GPU p100 with 16 GB RAM with one node.

2.3 Problem setup

- Firstly, for training just the metadata, we used different classification algorithms like logistic Regression, KNN, boosting, and so forth. The parameters were selected using 10-fold cross-validation with Grid Search to select the hyperparameters. Feature selection techniques such as PCA, VarianceThreshold, mutual_info_classif, and chi^2 were implemented. The same set of models was also trained with 256 densenet[2] features combined with the 12 metadata features.
- Next, we train the model on only the image dataset to try and understand the relation between the output and the image features. For this purpose, we use the pre-trained swin patch transformer [3] with ten epochs and an image size of 224. The optimizer used was the Adam optimizer. The reason is, for our case, it is the most optimal optimizer as it combines the best of rmsprop[4] and momentum.[5]. Also, Adam works best with lesser memory and is very fast. Since our problem had limited resources and time, we used this optimizer. We used a learning rate of 1e-5 with a cosine annealing warmer scheduler. The batch size was set to 64 to optimize memory usage and balance RAM constraints and training speed.
- Next, we train the model with both the image data and the metadata. We will be using the horizontal and vertical flip, affine transformation and image normalization, and lastly, just the image normalization with validation data. We will also be using the mixup augmentation technique. In addition, we perform training with several other pre-trained models in order to do a comparative study.
- Next, we combine the final output layer of the swin transformer with the 12 metadata features to augment the learning since these features act as correlated labels for our output.

- To further optimize modeling performance, we add 14000 other images with similar model configurations to determine whether more data can help us improve the model performance.

2.4 Design choice for several bins

There were several discussions regarding treating the current problem as a classification problem[6] that involved only two classes(0,1). These discussions used the binary cross-entropy loss function and suggested scaling the probabilities to 100. However, these methods would increase the RMSE score to a value that was more than the RMSE from the original problem.
However, our model was trained on the dataset used in an earlier competition. This earlier dataset has richer features with approximately 14,000 images. It was thus a business decision to use them. Furthermore, we decided to use the categories present with their attached semantic meanings behind each one of them as follows:

- 1: Strongly Unpopular(Needs to be changed)
- 2: Fairly Unpopular (Requires some work in order to be improved)
- 3: Fairly Popular (Can be deployed with minor tweaks)
- 4: Strongly Popular (Does not need many changes and can be deployed with minor changes)

3 Results

3.1 Multiple classification models were trained on the metadata

The KNN model works well when properly labeled data is classified. In our case, the data records have classification labels in the range 1-4 and get their values from the ’Pawpularity’ attribute. KNN also works well with noise-free data, and from our data preprocessing step, the noise component has been removed.
Random Forest is used as our next classifier. The reason is, it averages out the variance present by splitting on a random subset of our 12 different metadata features.
Logistic Regression is used for our classification task as it works well in cases where it assumes no noise in the output variable, in our case, ’Pawpularity.’ In addition, the metadata features are uncorrelated to each other. This further serves the purpose of using Logistic Regression.
The results obtained by fitting all the above models on the test dataset were as follows:

Model	Score
Logistic Regression	0.531108
Random Forest	0.531108
KNN	0.497724

Hyperparameter Tuning
The best ’k’ value for the KNN model was chosen by using the results obtained from the elbow method. 5 From figure 5, the error rate is minimal for our k-value = 6. Hence, this k-value configuration was chosen for the KNN model.
Similarly, we cross-validate our Random Forest algorithm by using GridSearchCV to choose the best estimators for our model. We conclude that the best parameters for random forest are as follows: max depth: 2, no. of estimators: 5.
Gradient boosting is used to evaluate the importance of all features. Subject focus, Eyes, and Face were observed to be the essential features. Occlusion, info, and Blur were observed to be the least important features.

3.2 Working with images only.

Through analysis of results, in our opinion, working solely with the features was useful. However, this is not a feasible option as they are unavailable in production settings. Another solution would be to train a binary classifier that can easily give us these labels. However, that would be an overhead on the modeling and lead to a slower inference. As a result, we performed modeling using only the images. The model was trained on a t40 GPU on Kaggle for ten epochs on the swin tiny patch transformer

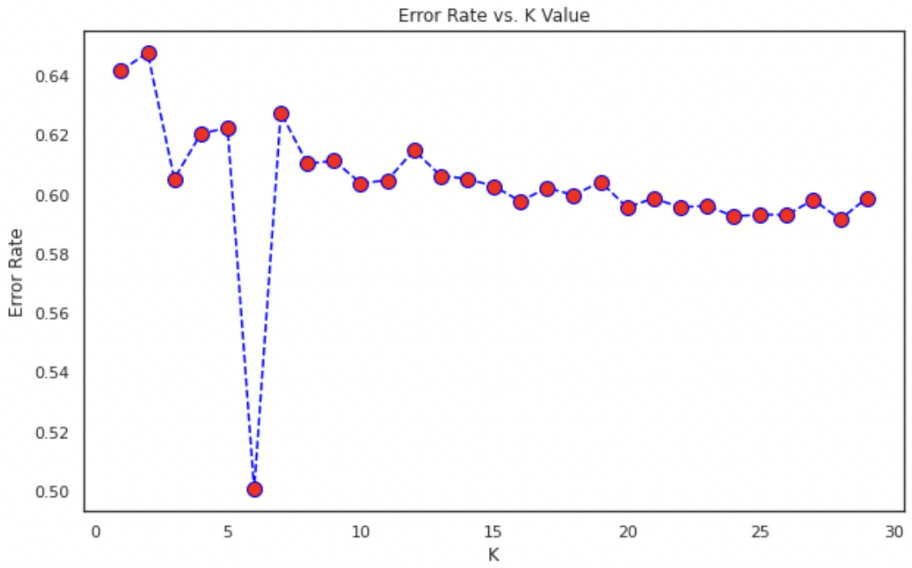
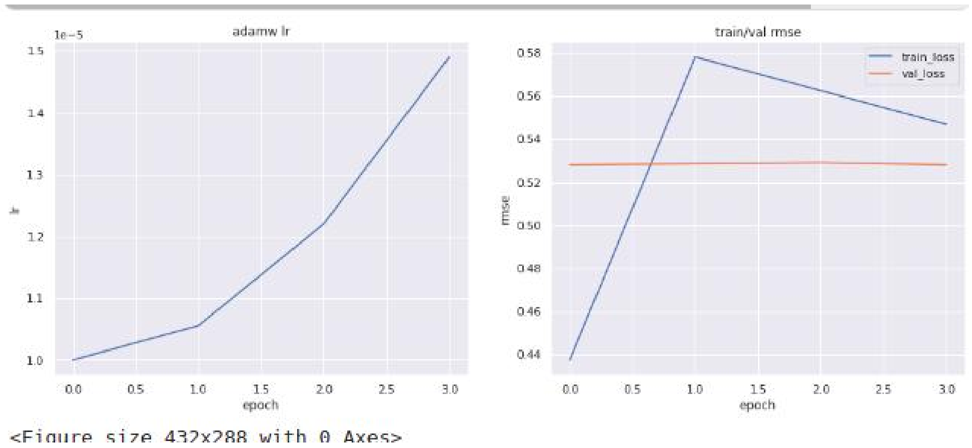


Figure 5: Error rate vs. K-value



```
<Figure size 432x288 with 0 Axes>
```

Figure 6: Learning rate vs epochs on the left and training and validation accuracies on the right

using adam with stratified k folds with 7908 images. The best validation accuracy came to be about 53.2%. Figure 6 shows for only three epochs since we used early stopping using validation loss. Moreover, best validation leads to the least training loss, which, as we can see, does not increase. The results were not as expected since image features are not correlated to the output. The learning rate kept increasing to eventually try and get out from the saddle points. However, it was stuck due to noisy data. We tried the model with three other pre-trained models with similar configurations and data. Following were the results:

Model	accuracy
Inception v4	21.7%
Efficient Net B2	52.44%
vgg19	53.97%

Models were chosen based on their commonality with image classification tasks and their robust architectures.

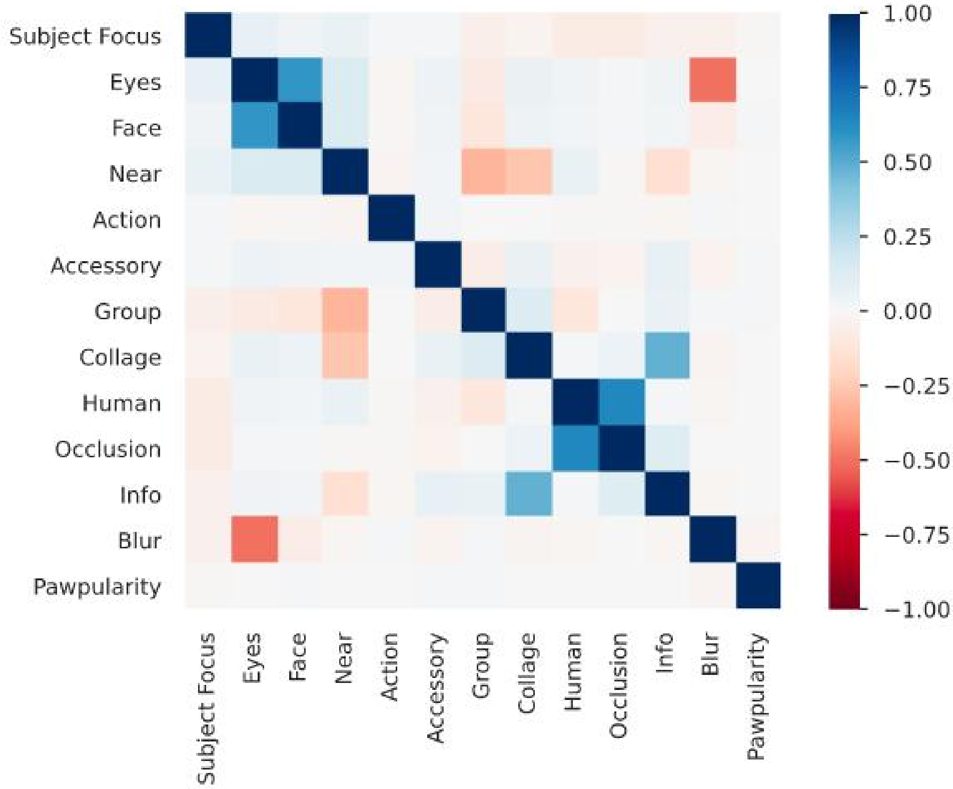


Figure 7: Correlation between metadata features and pawpularity.

3.3 Combining the metadata and the image features.

As discussed earlier, the images had started predicting the signals that were the most common label(class 2). This could mean that there was no signal to correlate the features and the image. An attractive solution could be to explicitly feed in extra information in the form of meta-data, which could serve as valuable information. However, contrary to our belief, the best validation score that we obtained with this approach was 53.8%. This was because the images and the features were very minimally correlated. See Figure7.

3.4 Working with 256 Densenet features and metadata

PCA [7] was used for dimensionality reduction. Figure 8 shows at approximately what number of features the variance of our data stopped getting captured.

Tree-based models such as AdaBoost and Random Forest classifier and a KNN and Logistic Regression model were trained using the first 150 features chosen by PCA.

Hyperparameter Tuning
The elbow method was employed again to select the ’k’ value for our KNN model.
The graph in 9 is obtained, and from this, we can deduce that the best value for k at which the lowest error rate is obtained is equal to 29.
The best parameters for our Random Forest classifier were found using GridSearchCV, and the model was accordingly configured.
Logistic Regression and AdaBoost Classifier were also used, and the following were the accuracy scores as seen in table 3.4 were generated:

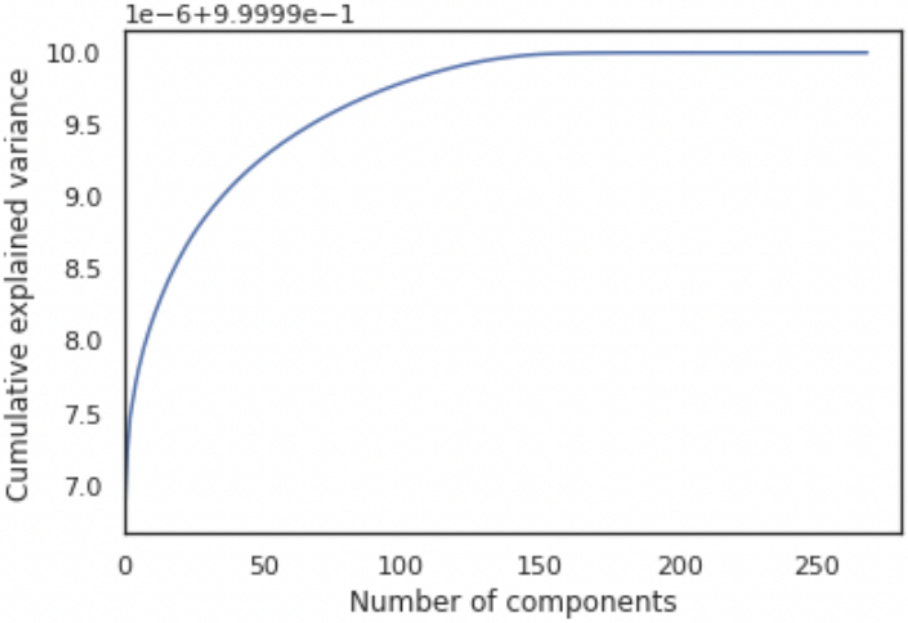


Figure 8: PCA: Variance explained vs. Number of components

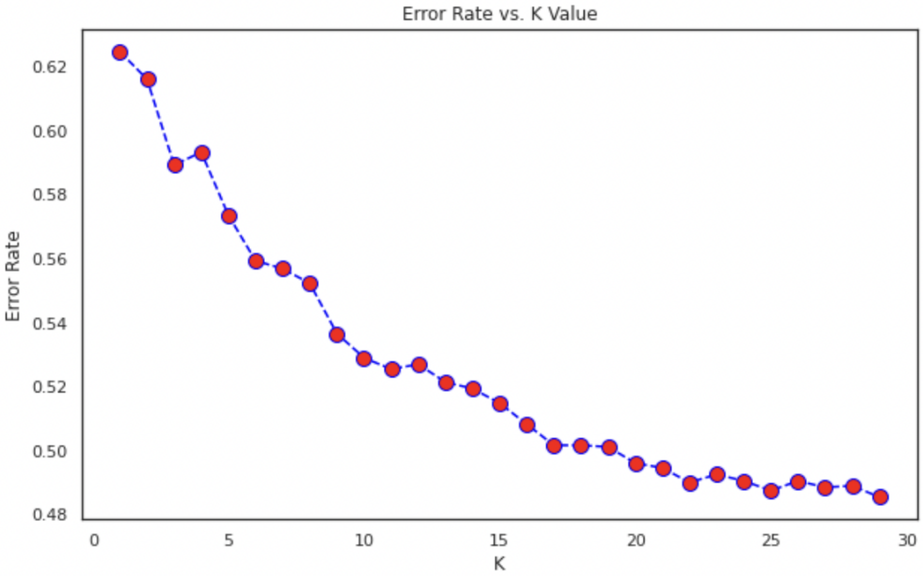


Figure 9: PCA: Variance explained vs. Number of Components

Model	Score
Logistic Regression	0.5311077
Random Forest	0.531108
KNN	0.4405
AdaBoost	0.375

3.5 Our last attempt at improving the performance: More data

As studied in one of the lectures, ultimately, neural networks tend to perform better if you feed more data. We used the swin tiny transformer [3] to perform training. We had approximately 24,529 images for ten epochs. The results were even worse. The best validation accuracy came to be 40.7%. This was because the added data was from a different dataset. The labels in that dataset were actually 'Adoption Speed'. The label here however,was dependent on more factors apart from the ones present in the photo itself. Similar is the situation for our data. However, the factors are different. And hence our best guess wois that there was a covariate shift in our dataset, leading to model learning wrong patterns.

4 Error Analysis

4.1 Understanding the wrong predictions

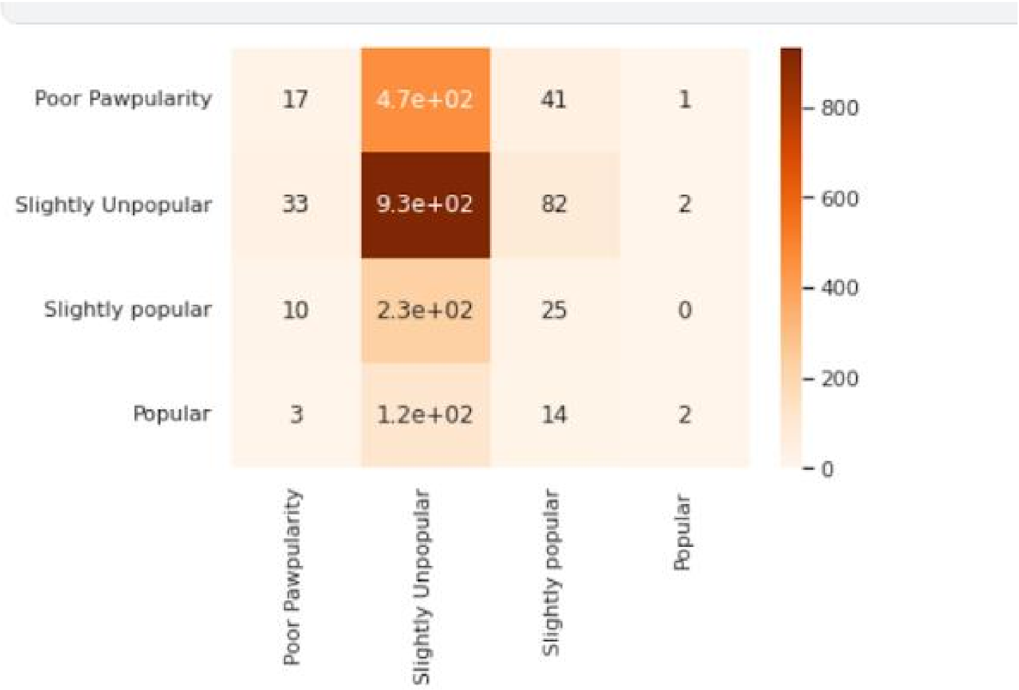
From the confusion matrix, in 10, we can deduce that the model is constantly predicting the majority class since the dataset is imbalanced, and no matter what algorithm is employed, the algorithm will predict class 2. The worst prediction was for the popular class, where from the visual analysis, we can see that the images contained both the eyes and did not have anything else. However, it is not conclusive to say that these will be 'popular' images. We can also observe that images are not being predicted as popular, suggesting that they do not have any typical pattern; but are 'popular' for some random reasons. Another quick observation is that- a lot of poor and popular images are being classified as slightly popular.

4.2 Metadata analysis

Figure 11 shows the relative percentage difference between the percentage of each label present in the original data set and the percentage of each feature in the incorrect predictions. We can see that the feature 'Near' is the only feature that is relatively more in the incorrect predictions. Hence, we can conclude that the images with 'Near' photos are hard for the model to classify since these images are evenly distributed(20,40,20,20) within the classes. This explains why such photos do not give us any useful information. In short, images with 'Occlusion,' 'Eyes,' 'Human,' and 'Collage' will be easier to predict.

Figure 12 shows the comparative study between the presence of features in the original test set versus its presence in the incorrect predictions. We can see that features like 'Accessory,' 'Collage,' 'Info,' and 'Blur' have fewer values in incorrect predictions, thereby demonstrating that these features are easier to classify. This makes sense since images with such features would be less popular and easier to predict.
Other quick observations:

- Dogs were more challenging to classify than cats.
- When trying to predict using some common models, the model quickly converges to the most common class. This shows that the signal in the data is minimal.13
- On performing some visual analysis of some images, some very similar images had very different labels.14



	Poor Pawpularity	Slightly Unpopular	Slightly popular	Popular
Poor Pawpularity	17	4.7e+02	41	1
Slightly Unpopular	33	9.3e+02	82	2
Slightly popular	10	2.3e+02	25	0
Popular	3	1.2e+02	14	2

Figure 10: Confusion Matrix for our classes.

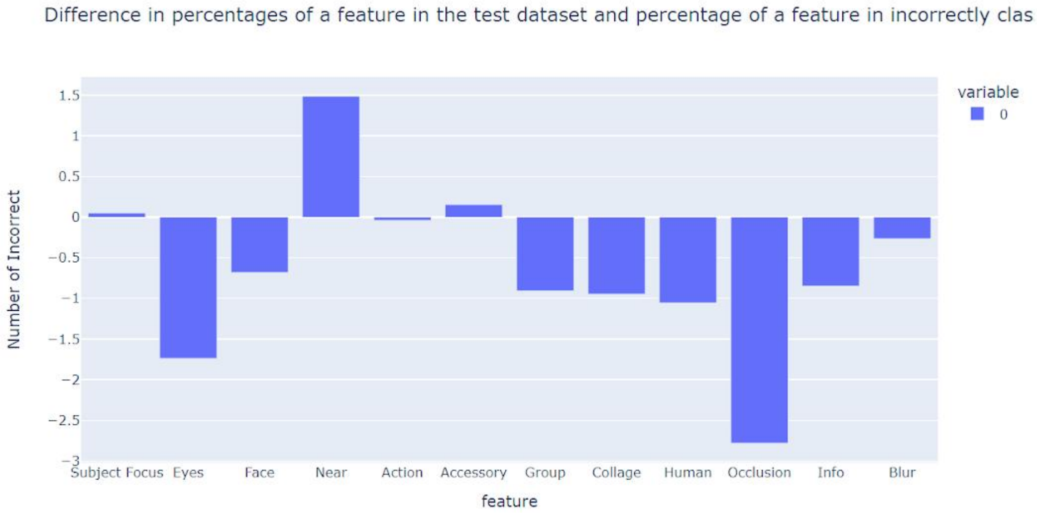


Figure 11: Relative difference of features in the original test set and incorrect predictions.

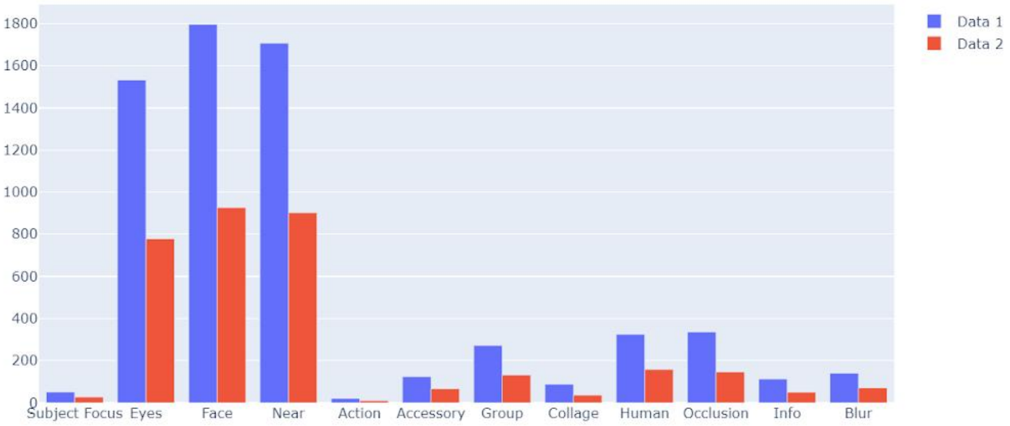


Figure 12: Relative presence of each of the features in the original test set and in the incorrect predictions.

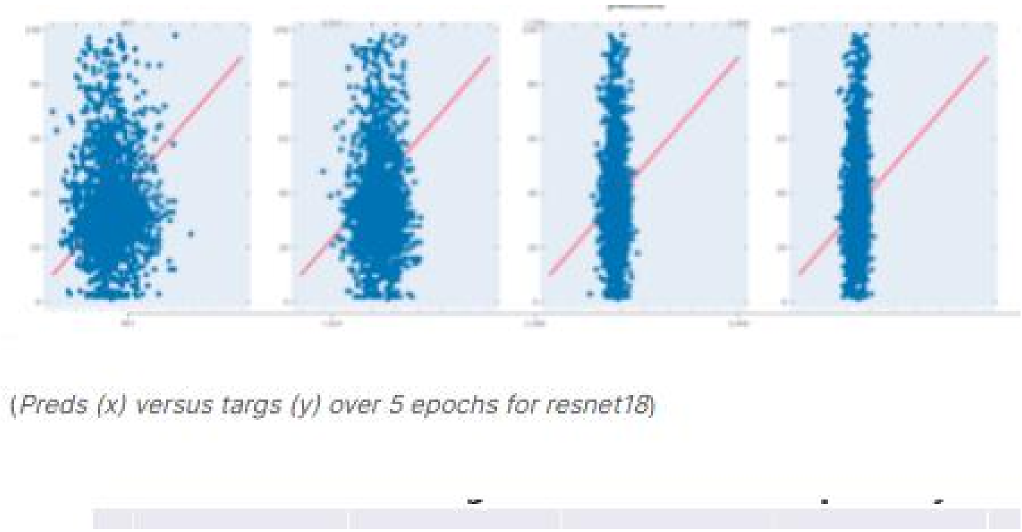


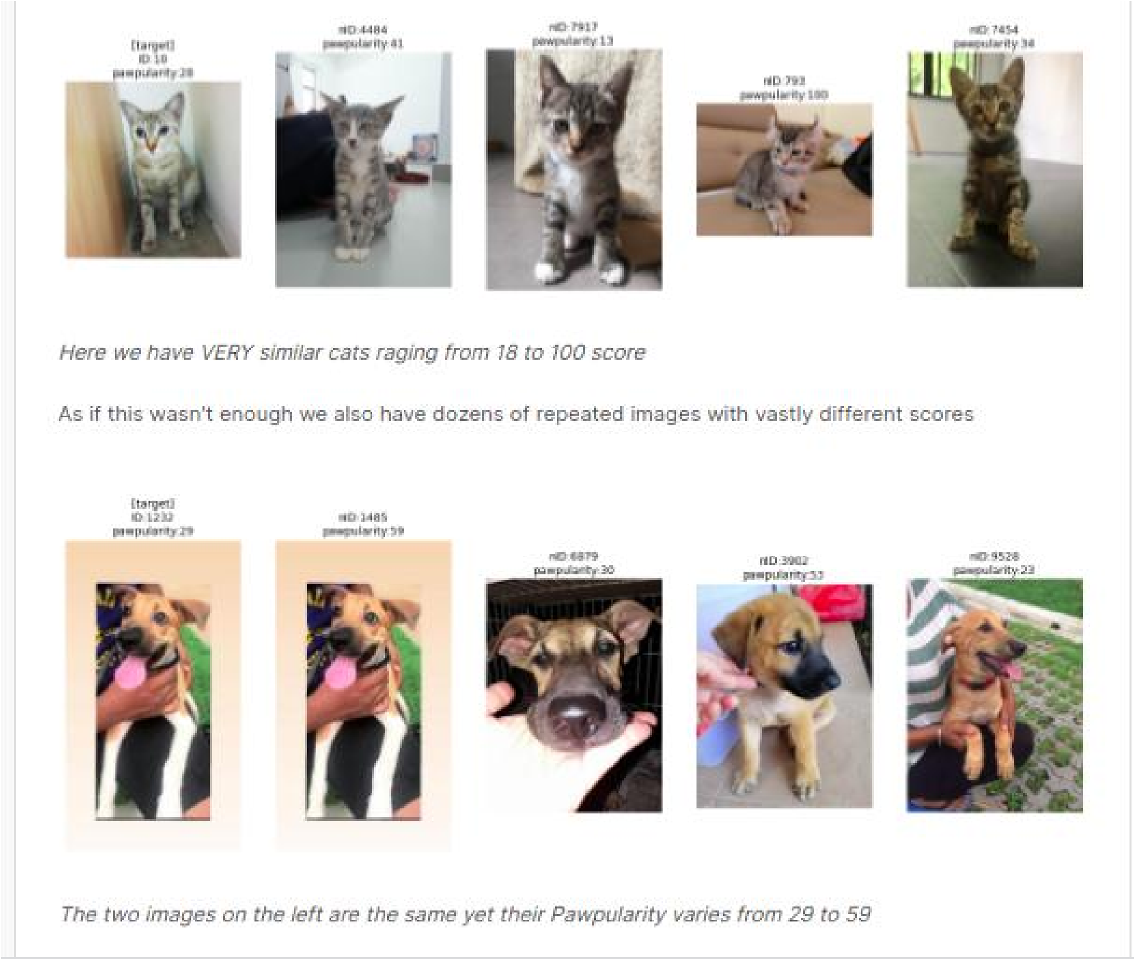
Figure 13: Predictions(y) vs x on resnet

4.3 GRAD CAM

Figure 15 shows that most of the images were very focused on the face of the animal. We can infer that there should be distinctions in the faces of different classes that can help us distinguish between the pets of different classes. However, this was not the case. Images from all the classes had similar faces. We thus believe that this was one of the most significant factors that resulted in the model being confused about it's predictions.

5 Conclusion and future work

In this project, we ran through many techniques varying from using only the metadata features to using images to using both. We found marginal improvements, and the model converged to the most frequent class. Modeling the problem as a classification problem helped us interpret the model better. We were able to leverage helpful information from the metadata features. However, through analysis of results, they turned out to be insignificant. We want the organizers to add more semantic information that could be impacting the popularity of the image, like website on which the photo was posted, breed information, and type of adoption agency. Predicting the popularity only from the



Here we have VERY similar cats raging from 18 to 100 score

As if this wasn't enough we also have dozens of repeated images with vastly different scores

The two images on the left are the same yet their Pawpularity varies from 29 to 59

Figure 14: Similar images with different scores.

images is very random and thus should be augmented by supporting semantic features. We want to propose the following future work:

- Change the target variable to something which is a true measure and not a derived measure. We also propose that images should be filtered before training since there is noise in the data, which is confusing to our ML model.
- We could not try out some heavier models like the heavy transformer,cnn+transformer due to time constraints. We think that the results would not improve much, but would still be worth a try.
- Dataset imbalance was one of the major reasons for wrong classifications. We believe that we can get more images with time, which would ultimately resolve data imbalance and thus help our model perform better.

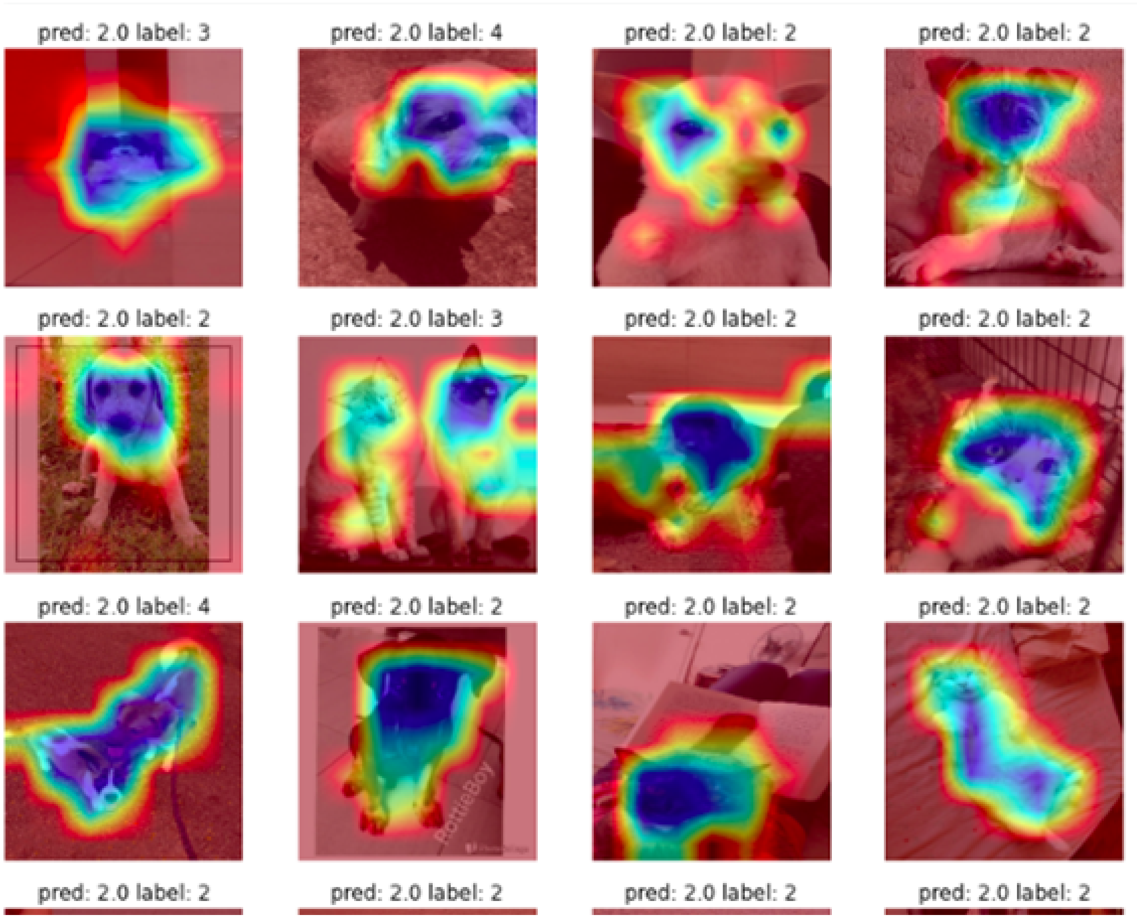


Figure 15: GRAD CAM rendering on some images.

References

[1] Hash scoring. https://www.kaggle.com/c/petfinder-pawpularity-score/discussion/278497. Accessed: 2021-12-10.

[2] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[3] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021.

[4] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[5] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[6] Phalanx Kaggle baseline solution, https://www.kaggle.com/c/petfinder-pawpularity-score/discussion/275094, Accessed: 2021-12-10.

[7] PCA for dimension reduction. https://machinelearningmastery.com/principal-components-analysis-for-dimensionality-reduction-in-python/. Accessed: 2021-12-10.