Final Project Written Report

CS-UY 4563: Introduction to Machine Learning

05/02/2022

Prof. Linda M. Sellie

Nick Broome & Daniel Be'eri Longman

Introduction

Unfortunately, millions of pets are euthanized worldwide every year in shelters because they have not been adopted fast enough. Is there anything pet adoption platforms can do to increase the rate of pet adoption, thus saving the lives of many pets? While we do not have a definite answer, we utilize a few machine learning algorithms to analyze pet adoption profiles in order to attempt to assess and predict the popularity of these profiles. Being able to predict the popularity of these profiles opens the door to the option of providing adoption profile owners with practical tips to enhance the popularity of their profile, thus increasing the chances of a successful adoption.

Specifically, the Malaysian pet adoption platform, PetFinder.My, provided us with a set of 9912 adoption-profile images of pets along with metadata containing some basic descriptive features of each image.

The metadata features assigned to each image are as follows:

- Focus - Pet stands out against an uncluttered background, not too close / far.
- Eyes - Both eyes are facing front or near-front, with at least 1 eye / pupil decently clear.
- Face - Decently clear face, facing front or near-front.
- Near - Single pet taking up a significant portion of the photo (roughly over 50% of photo width or height).
- Action - Pet in the middle of an action (e.g., jumping).
- Accessory - Accompanying physical or digital accessory / prop (i.e. toy, digital sticker), excluding collar and leash.
- Group - More than 1 pet in the photo.
- Collage - Digitally-retouched photo (i.e. with digital photo frame, combination of multiple photos).
- Human - Human in the photo.
- Occlusion - Specific undesirable objects blocking part of the pet (i.e. human, cage or fence). Note that not all blocking objects are considered occlusion.
- Info - Custom-added text or labels (i.e. pet name, description).
- Blur - Noticeably out of focus or noisy, especially for the pet’s eyes and face. For Blur entries, “Eyes” column is always set to 0.

Along with the image and the above-given features, the training data also contains a target variable, which is the popularity (pawpularity) index of the profile associated with the given image. The pawpularity score is derived by PetFinder.My based on profile page view statistics.

Throughout the rest of this document we provide an analysis and review of the data processing and algorithms we utilized to attempt to predict the images' pawpularity score. Specifically, we implemented logistic regression, support vector machine (SVM), and neural networks (NN) as methods to predict an image's pawpularity. The provided pawpularity score for each image is an integer value between 0-100.

Initial Data Preprocessing for Logistic Regression & SVM

Since we wanted a categorical value and the pawpularity score is a continuous value between 0-100, division into categories was necessary. We measured the mean of the training set's pawpularity score, which turned out to be approximately 38. Accordingly, we labeled the images with a score of 38 and above as popular (belonging to the group labeled as +1), and all images with a lower score as unpopular (belonging to the group labeled as -1). Since all of the metadata consists of boolean features, normalization was not required. We applied this to our Logistic Regression and SVM models.

Logistic Regression

We utilized sklearn's LogisticRegression object to implement our logistic regression. After preprocessing our data we first recorded our training set and test set's accuracy with no

penalty. Once that was completed and we had our initial values, we first tested the data with L1 (Lasso) and L2 (Ridge) regularization while utilizing the C-values [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]. The results can be found below this section. We did not notice substantial differences with regularization added, it was just a small change between 0-1 percent for the various C-values. While accuracy did increase for larger values of C, the difference was very minor so we felt it was negligible.

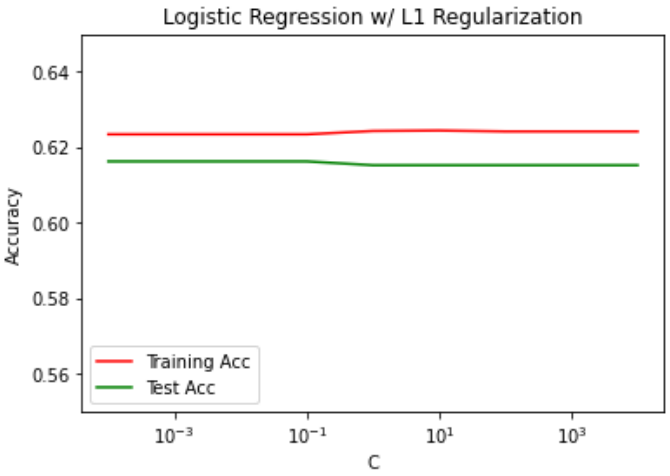

Figure 1: Log. Reg. with L1 Regularization applied

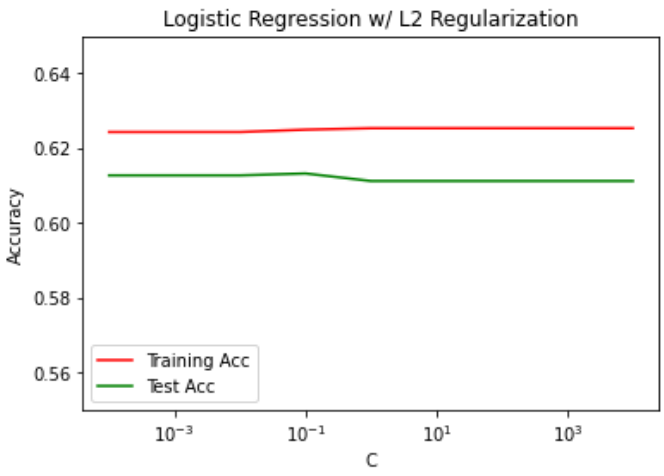

Figure 2: Log. Reg. with L2 Regularization applied.

After completion of these two types of regularization, we tried polynomial feature transformation on the data. Again, there was no large impact on accuracy, with it only rising by about 1% on the training accuracy. However, we did notice the accuracy diverged with larger values of C after implementing polynomial feature transformation. As our C value increased, the Test Accuracy dropped instead of raising alongside the Training Accuracy.

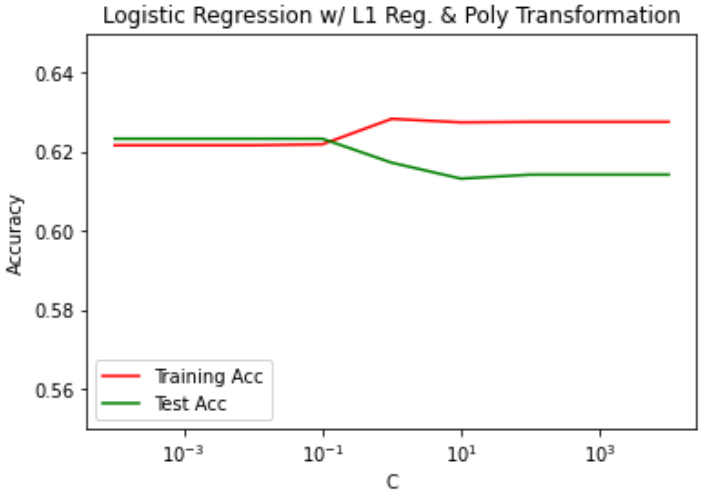

Figure 3: Log. Reg. with L1 Reg and Poly. Transform.

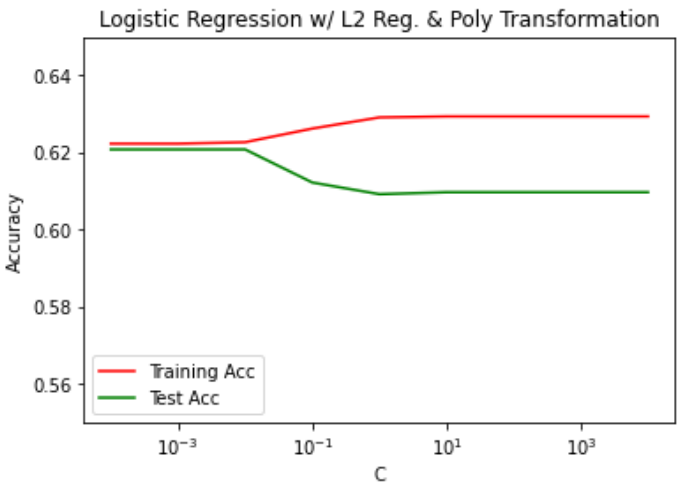

Figure 4: Log. Reg. with L1 Reg and Poly. Transform.

Support Vector Machine

Data Preprocessing & Discussion of Results

In addition, in order to attempt to increase the accuracy rate of the results, we took a few different approaches when working with the data.

First, we ran some statistical analysis on the data in order to see which features explained the variation in the pawpularity results better than others. We did that by calculating the p-value of each feature, and set a threshold of 0.05 for the p-value. Any feature with a p-value higher than 0.05 was removed, as it is not likely to tell much about the behavior of the dependent variable. The only features remaining after the p-value filtering were eyes, near, and blur. Looking through some of the Kaggle competition's posted solutions showed that the metadata wasn't helpful for the competitors either, as they were approaching this problem.

Thus, we decided to take the images themselves as inputs and extract a new set of features from them based on the Histogram of Oriented Gradients (HOG) algorithm. This algorithm basically extracts thousands of features from every image that have to do with the orientation of the objects in it, whether there are faces in the image, where they are located respectively, etc'. The advantage of using HOG over the human-derived metadata is that it provides a much higher resolution of features that we may not be able to explain with words, but that are brain notices when we observe an image. Extracted HOG features are normalized in the feature extraction process.

Using the HOG algorithm did indeed increase the accuracy of the training set by almost 10% (to 69.29% with C=10,000), however - it significantly reduced the testing accuracy to below 50%.

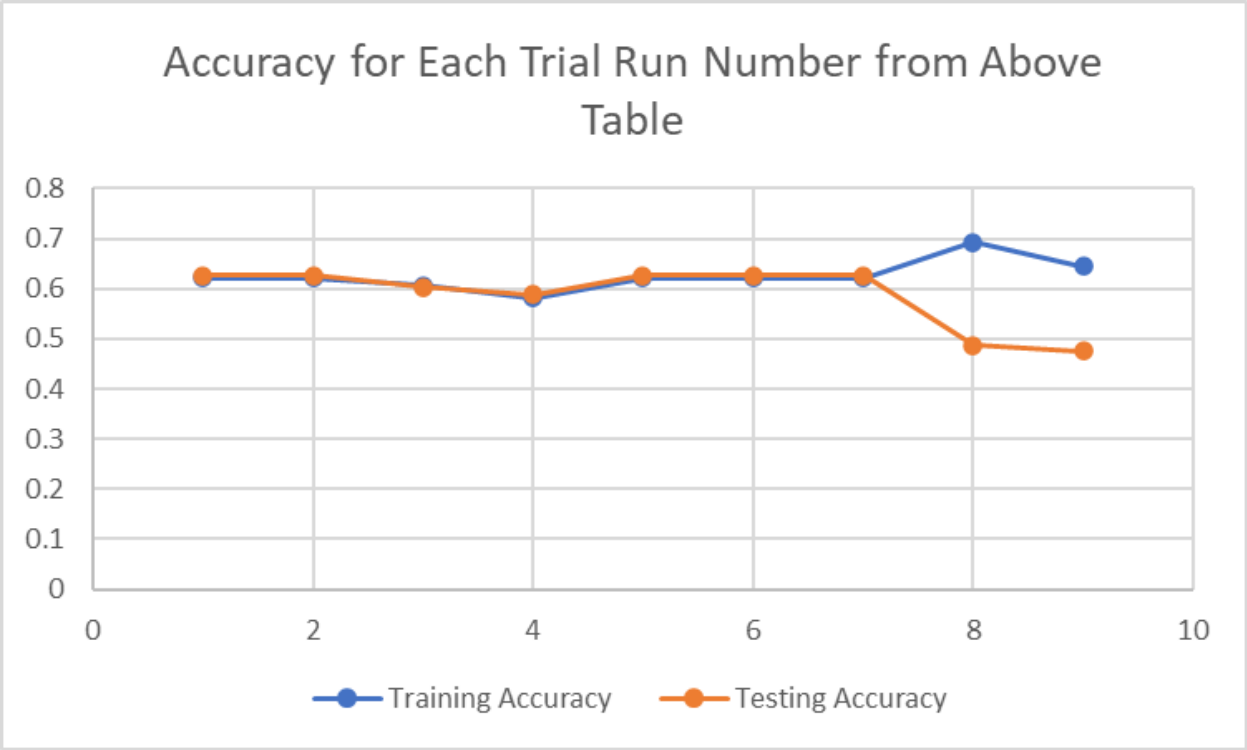

Figure 5: SVM Accuracy scores for each case number as detailed in figure 8.

Neural Networks

Data Preprocessing & Discussion of Results

The data processing for neural networks is very similar to the one we used for SVM, except we use the pawpularity score as provided by the dataset, without dividing it into categories of pawpularity. We used a model with 3 layers total, consisting of 2 hidden-layers with 2 or 5 nodes in each hidden layer. The third layer consists of one output, which is used as the predictor of the image's pawpularity score.

Attempting to use the sigmoid activation function yielded unclear results which caused an overflow after very few iterations. For that reason, we mostly stuck to the RELU activation function and used no regularization. Using the same statistical analysis of p-values for the features from SVM didn't decrease the root mean square error (RMSE) in this case either. The learning rate that yielded the best accuracy (lowest RMSE) is of value 0.0001. In addition, we

transformed a couple of the statistically significant boolean features (eyes and near) to have values of 0 and 3, or 0 and 5, instead of 0 and 1. However, they yielded very similar results hence were not included in this table for redundancy reasons (but are included in the code).

Extremely large RMSE values were left out of the graph, in order to maintain a reasonable scale.

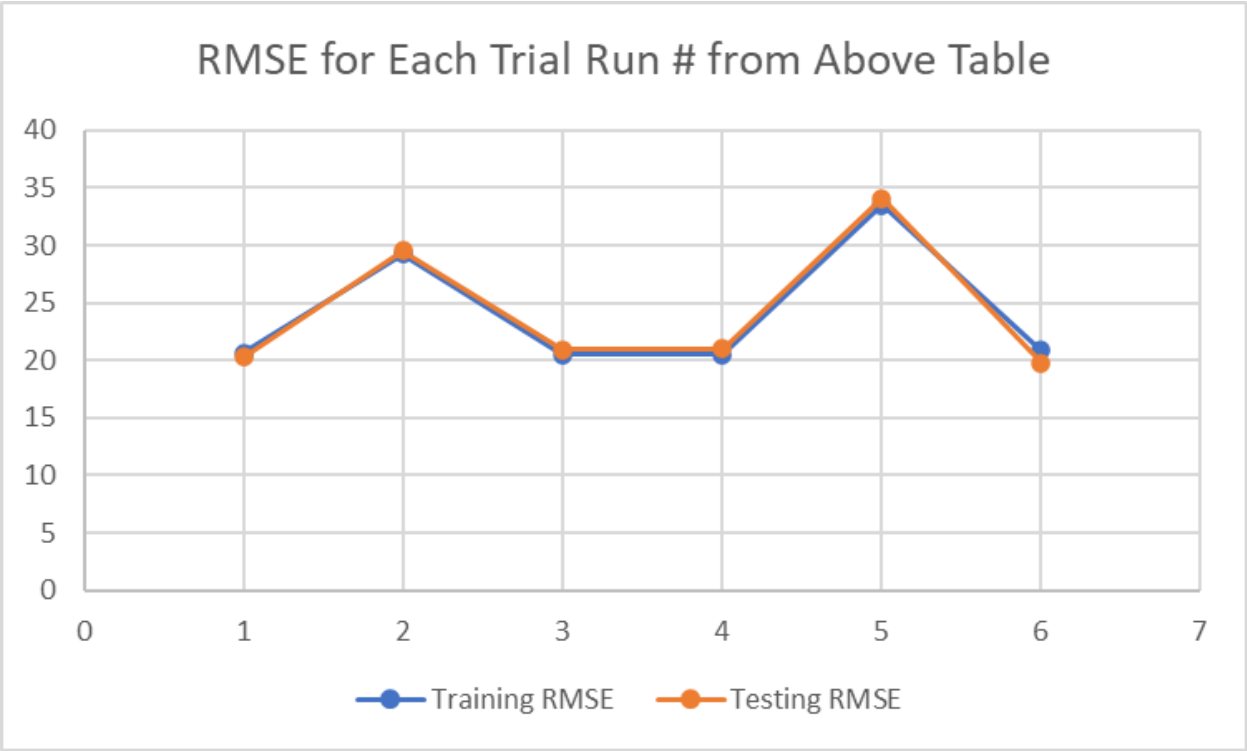

Figure 6: Root mean squared error for each case number as described in figure 9.

Table of Results

	0.0001	0.001	0.01	0.1	1	10	100	1000	10000
None, None, Train	0.62417	0.62417	0.62417	0.62417	0.62417	0.62417	0.62417	0.62417	0.62417
None, None, Test	0.6123	0.6123	0.6123	0.6123	0.6123	0.6123	0.6123	0.6123	0.6123
L1, None, Train	0.62341	0.62341	0.62341	0.62341	0.62429	0.62442	0.62416	0.62416	0.62416
L1,	0.61624	0.61624	0.61624	0.61624	0.61523	0.61523	0.61523	0.61523	0.61523

None, Test									
L2, None, Train	0.62429	0.62429	0.62429	0.62429	0.6253	0.6253	0.6253	0.6253	0.6253
L2, None, Test	0.61271	0.61271	0.61271	0.61321	0.6112	0.6112	0.6112	0.6112	0.6112
L1, Poly, Train	0.62164	0.62164	0.62164	0.62189	0.62833	0.62744	0.62757	0.62757	0.62757
L1, Poly, Test	0.6233	0.6233	0.6233	0.6233	0.61725	0.61321	0.61422	0.61422	0.61422
L2, Poly, Train	0.62227	0.62227	0.62265	0.62618	0.62908	0.62934	0.62934	0.62934	0.62934
L2, Poly, Test	0.62078	0.62078	0.62078	0.61223	0.60918	0.60968	0.60968	0.60968	0.60968

Figure 7: Table of Results from Logistic Regression. Green rows are accuracy from training data, blue rows are accuracy from test data, and then yellow cells indicate highest accuracy for test data. The top row is our C-Values we used for Log Regression, and the left most column is what was being tested.

Learning rate = 0.000001=10^-7	**Training Accuracy**	**Testing Accuracy**
1. No Regularization, Only Metadata	0.6210	0.6258
2. L2, C=10,000, Only Metadata	0.6210	0.6258
3. L2, C=100,000, Only Metadata	0.6074	0.6026
4. L2, C=1,000,000, Only Metadata	0.5815	0.5885

5. No Regularization, Only statistically significant metadata (eyes, near, blur)	0.6210	0.6258
6. L2, C=10,000, Only statistically significant metadata (eyes, near, blur)	0.6210	0.6258
7. No Regularization, Image-extracted features	0.6210	0.6258
8. L2, C=10,000, Image-extracted features	0.6929	0.4876
9. L2, C=100,000, Image-extracted features	0.6446	0.4750

Figure 8: Accuracy for our SVM model tests. The left most column indicates what is being tested, the middle/green column is accuracy for the training data, and the right/blue column is our accuracy for test data. Yellow cells indicate highest accuracy for test data.

Learning rate = 0.000001=10^-7	**Training RMSE**	**Testing RMSE**
No Regularization, Only metadata, 2 hidden layers (5 nodes each), RELU activation, 5 iterations, learning_rate=0.001	5.8345*10^24	5.8345*10^24
1. No Regularization, Only metadata, 2 hidden layers (5 nodes each), RELU activation, 10 iterations, learning_rate=0.0001	20.6758	20.2525
2. No Regularization, Only metadata, 2 hidden layers (5 nodes each), RELU activation, 10 iterations, learning_rate=0.00001	29.2259	29.5044
3. No Regularization, Only metadata, 2 hidden layers (2 nodes each), RELU activation, 10 iterations,	20.5021	20.9426

learning_rate=0.0001		
No Regularization, Only [Eyes, near, blur], 2 hidden layers (5 nodes each), RELU activation, 10 iterations, learning_rate=0.00001	3.5779*10^21	3.5779*10^21
4. No Regularization, Only [Eyes, near, blur], 2 hidden layers (5 nodes each), RELU activation, 10 iterations, learning_rate=0.0001	20.4846	21.0112
5. No Regularization, Only [Eyes, near, blur], 2 hidden layers (5 nodes each), RELU activation, 10 iterations, learning_rate=0.001	33.5247	34.0555
6. No Regularization, Only metadata, 2 hidden layers (5 nodes each), Sigmoid activation, 4 iterations, learning_rate=0.0001	20.8382	19.8128

Figure 9: Table of Results for neural network, middle/green column is based off of training data, right/blue column is based off of test data.

Conclusion

For logistic regression, our results were consistent, but accuracy wasn't the highest. Our highest accuracy was achieved for low C-values and implementing an L1 penalty and polynomial feature transformation. For the initial logistic regression test and comparing it to the results from testing with L1 and L2 penalty, the accuracies were very close with only about a 1% difference. To us, this indicates that these models fit fairly well and do not suffer from overfitting or underfitting. Our testing data scores lower accuracy than our training data, but that was expected prior to anything being done. After applying polynomial transformation to our data, we did notice that accuracy was slightly higher at lower C-values for both L1 and L2 penalty. However as the C-Value increased, the training and testing accuracy diverged significantly indicating that with polynomial transformation it exposed a problem with bias or variance within the data and model. As such, for low C-values it may be permissible to utilize polynomial

transformation with an L1 penalty for the highest accuracy possible, but for the most general consistent model for all C-values no transformation should be applied, only an L1 penalty. Perhaps a different feature transformation would yield better results, but apart from that increasing sample size is another solution to possibly combat these issues.

For the SVM implementation, an overfitting started occurring only when extracting the HOG features from the images themselves. This means that the HOG features may have explained the data a bit "too" well, and taken irrelevant data (noise) from the image into account. The regularization parameter, C (1/lambda), of value 10,000 presented the best training and testing accuracy results over all others experimented with. There seems to be underfitting as well, as results of around 60% accuracy for a binary decision problem don't display an accuracy much higher than when running random selection between categories on the data.

It appears that yielding results with higher accuracy would require utilization of other regularization techniques, as well as more advanced image processing techniques, potentially with convolutional neural networks.

The results of the neural network implementation show that it is not the best fitting model for this dataset, and the nature of the cost function not being convex yielded a significant amount of highly varying results, in which the cost didn't necessarily decrease between one iteration and another.

In addition, no significant overfitting occurs, as most training RMSE scores are very close in value to the corresponding testing RMSE scores. However, there appears to be an overall high underfitting as RMSE scores remain relatively high. A potentially better approach would include L2 regularization along with image processing features extractions. Overall, human-made characterizations of images in the form of manually entered metadata seem to present a significantly lower flexibility than that required for a more accurate pawpularity score prediction.

Works Cited

1. https://towardsdatascience.com/an-introduction-to-neural-networks-with-implementation-from-scratch-using-python-da4b6a45c05b
2. https://towardsdatascience.com/svm-implementation-from-scratch-python-2db2fc52e5c2
3. https://www.kaggle.com/competitions/petfinder-pawpularity-score