CIS520 Project: Animal Adoption Prediction

Haozhi Fan
School of Engineering
and Applied Science
University of Pennsylvania
h3fan@seas.upenn.edu

Guo Cheng
School of Engineering
and Applied Science
University of Pennsylvania
guocheng@seas.upenn.edu

Jiaying Hou
School of Engineering
and Applied Science
University of Pennsylvania
hjiaying@seas.upenn.edu

Abstract

The objective of this project is to develop a binary classification model to predict animal adoption rate based on objective facts and descriptions of the animals. The model was trained on a dataset of cats and dogs, incorporating information such as breed, age, and gender as well as preprocessed sentiment data from text descriptions of each animal using Google's Natural Language API. The training data was labeled with the outcome of the animal's adoption, allowing the model to learn the relationship between the available information and the likelihood of adoption. This model can be used by animal shelters and welfare organizations to help predict which animals are likely to be adopted and provide guidance on how to increase the chances of adoption for animals that has a not adopted prediction. Throughout the process, we experimented with different machine learning models as well as feature engineering techniques. We find that the model using the LightGBM algorithm results in the highest accuracy score, achieving an accuracy of 77%.

1 Motivation

As overpopulation of strayed and abandoned animals becomes more serious, rescuing and providing them with shelters is one of the most humane and ethical ways to control overpopulation. Unfortunately, many animals that entered shelters are euthanized due to not being adopted over time. In this project, we hope to use the information provided by PetFinder.my, a Malaysian animal welfare platform to build a binary classification model that can predict whether or not a pet would be adopted.

This model could be used to provide helpful information for shelters and rescuer organizations to better utilize their resources. It could also provide insight into the factors that are most important in determining whether or not an animal will be adopted, which could help shelters and rescue organizations make more informed decisions about how to care for and promote their animals.

2 Related Work

Since the problem derives from an existing Kaggle challenge, there are many submissions regarding the original problem. Our dataset contains tabular, text, and image data. A common technique is to extract image features from pretrained neural network models. Both Dieter's(Dieter, 2018b) and XHLULU's(XHLULU, 2018) notebook demonstrated how DenseNet121 from Keras could be helpful here. On the other hand, ANDREW LUKYANENKO's(LUKYANENKO, 2018) notebook has a comprehensive walkthrough of data exploration as well as how sentiment analysis from the text data could be handled.

The baseline that most people choose is tree based gradient-boosting method. ERIK BRUI's(BRUI, 2018) notebook used XGB, and WOJTEK ROSINSKI's(ROSINSKI, 2018) work provides a baseline LGBM model upon which winners' models are based.

The prize-winning approaches are mostly a stacking of NN and gradient-boosting models. U++'s team(u++, 2018) stacks one NN, two LGBM, and one XGB model. To create diversity among their models, Takuoko(takuoko, 2018) also used all image and sentiment files but kaeru(kaerururu, 2018) and gege(gege, 2018) used only the 1st one. Benjamin Minixhofer(Minixhofer, 2018) stacks 5 models - three NNs using PyTorch, one LightGBM model, and one FFM with xlearn.

Aside from supervised learning models, there are also people who incorporated unsupervised learning into their approaches. After concatenating features extracted from text and image data as well as categorical features processed with one hot encoding, Dieter uses autoencoder to train the reconstruction of numerical and categorical features and then removes the reconstruction head of the autoencoder to train for classification models. Since autoencoder is unsupervised, he could leverage test and training data. Autoencoder is also better than PCA in that it captures non-linear connections among features. Simply dropping out input features in autoencoder also helps controlling overfitting.

3 Dataset

3.1 Introduction

The dataset[1] used in this project is adapted from the PetFinder.my Adoption Prediction challenge on Kaggle. It consists of six .csv files: train.csv, test.csv, sample_submission.csv, breed_labels.csv, color_labels.csv, and state_labels.csv. The main categorical data is contained in train.csv, which includes 24 columns of tabular and text data and has 14,993 entries. The breed_labels.csv, color_labels.csv, and state_labels.csv files contain the name string of breed,

[1]https://www.kaggle.com/competitions/petfinder-adoption-prediction/data

Figure 1: Relevance between Description Sentiment and Adoption Rate

color, and states. The images for each pet are joined with train.csv via pets' PetID, and the image metadata was generated by running the images through Google's Vision API for analysis of the face, label, text annotation, and image properties. The sentiment data was generated by running the pets' descriptions through Google's Natural Language API to analyze the sentiment and key entities. Both the image metadata and sentiment data are JSON files and are then joined with train.csv using the corresponding Pet ID. The summary of the processed data is as follows

Float	*Fee, SentiMagnitude, SentiScore*
Integer	*Type,Age,Gender,MaturitySize,Furlength, Vaccinated,Dewormed,Sterilized,Health, Quantity,PhotoAmt,VideoAmt,State*
Object	*Description, Breed1, Breed2, Color1, Color2, Color3*

Table 1: Feature Types

3.2 Summary Statistics

Statistical findings showed that approximately 72% of the animals were adopted within 90 days or less, which we labeled as adopted in our analysis. During our initial data exploration, we observed that cat has a 4% higher chance to be adopted than dogs. Additionally, since most pets in the dataset have the age between 1 to 5 months old, we divide the dataset into two parts and found that younger pets (those less than or equal to 5 months old) are 20% more likely to be adopted than older pets (those more than 5 months old).

Next we explore the correlation between features and plotted a correlation matrix as shown in figure 2.

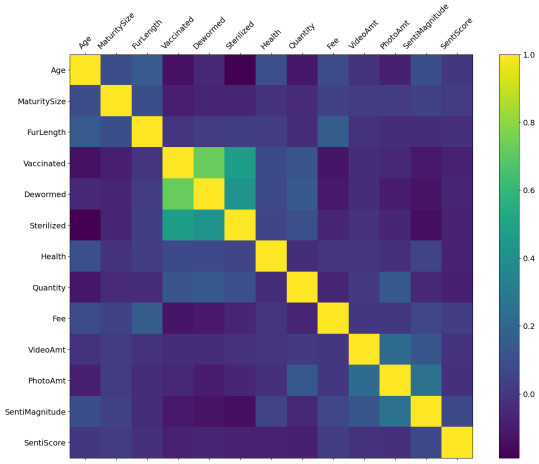

Figure 2: Feature Correlation Matrix

The correlation matrix is obtained by dropping categorical features that have no meaning in numerical value (Breed, Color, Gender, etc.), and computing the pairwise correlation between the remaining features. Only a few features have high correlation, and from the correlation matrix, we can see that pets that had been vaccinated are very likely to be dewormed and sterilized, as intuition may suggest. Also, there is a small correlation between maturity size and fur length, as larger pets are more likely to have long fur. Additionally, the photo amount, sentiment magnitude, and sentiment score have small correlation in the sense that if the pets' description is emotionally appealing, then there is likely more photos available.

To explore further on the relationship between sentiment analysis data and adoption rate, we plotted the sentiment analysis result along with the adoption rate as shown in figure 1, where the sentiment score indicates the overall emotion of a

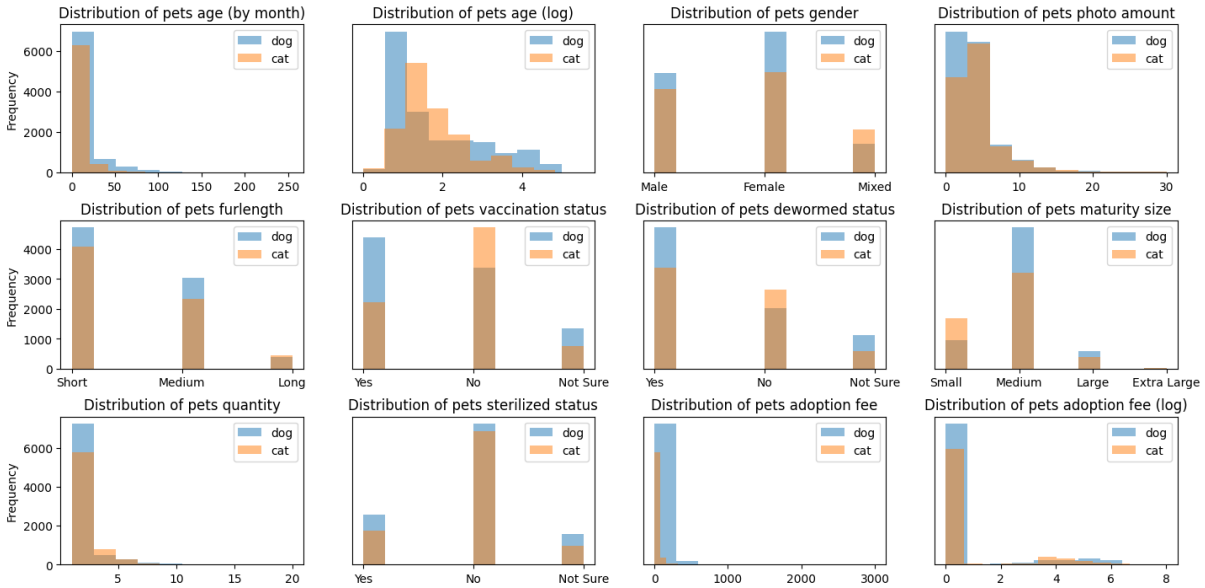

Figure 3: Feature Distribution

document (negative sentiment score means negative emotion and vice versa), and the magnitude of a document's sentiment indicates how much emotional content is present within the document[2]. There is a clear trend that the stronger the description's emotion is the more likely the pets are going to be adopted, regardless of the polarity of the emotion. The adoption statistics around where the description's emotion is neutral (score = 0 and magnitude = 0) is due to data imbalance.

The distribution of other features are shown in figure 3, where we distinguish between cats and dogs based on the assumption that it could be a determining factor when choosing whether to adopt a pet or not.

3.3 Data Pre-processing and Feature Engineering

We need to process the raw dataset to make it more usable by importing sentiment and labels dataset, doing feature selection and extraction, and modifying and relabeling categorical features.

3.3.1 Feature Selection

The sentiment data was generated by running the pets' descriptions through Google's Natural Language API and for each description, we extract the sentiment score and magnitude as new features and put them alongside the corresponding pets. Categorical features like breed, color, and state are converted to their corresponding strings and then encoded using one-hot encoding. This resulted in a total of 363 features and we used Principal Analysis technique to reduce that down to 47 features. The majority of the description is in English with only a minor fraction is in other languages, so we decided to not include the language as a new feature.

Some categorical features contain unknown or missing information, for example: the pet's vaccination status is indicated by numerical value as:

$$Vaccinated = \begin{cases} 1 & \text{Yes} \\ 2 & \text{No} \\ 3 & \text{Not Sure} \end{cases}$$

It may appear beneficial to encode this feature, but a better approach we adopted is to switch the label so the numeric value 1=Yes, 2=Not Sure, and 3=No, based on the intuition that 'yes' should be further away from 'No' but closer to 'Not sure'. This modification is done on several features based on feature importance and it increased the accuracy by 1-3% for different models.

As we can see from figure 3, the numerical features like pets' age, photo amount, and adoption fee center around smaller values, and has large variance and contain outliers. In order to improve the performance of distance-based algorithms such as SVM, We standardized these numerical features.

3.3.2 Feature Extraction

In addition to generating sentiment analysis data from pets' descriptions, we also extracted the number of words and average word length within each description as new features. These two features may help capture the variation in objective descriptions. For example, a description involving the location where the pet was found may have a longer average word length, while a description written by an official organization may have a larger number of words. A more effective approach would be to use the TF-IDF score and cosine similarity as new features. This is explored in the later section.

For the name label of the dataset, inspired by the Kaggle kernel(LUKYANENKO, 2018), we defined a NoName label which includes pets without names as well as pets with a name of fewer than 3 characters. About 10% of the pets in the training dataset are in the No_Name group and is about 4% less

[2]Google-NLP-API-Basics:https://cloud.google.com/natural-language/docs/basics

likely to be adopted than pets that have names. For description, we used the Latent Dirichlet Allocation model to group the entries into 5 topics, and the proportion of each topic is shown in figure 4.

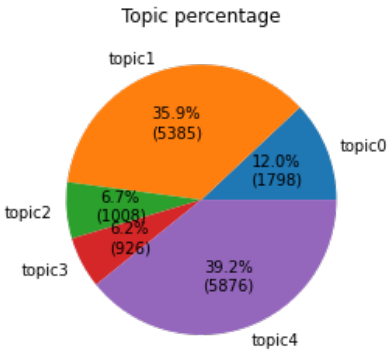

Figure 4: LDA Topics Distribution

Adoption rates came out similar among topics 0 to 3, which is 74.5% on average. However, we noticed that pets in topic 4 had a much lower adoption rate of 68.1%. More comparisons between cats and dogs for different categorical data are as follows:

4 Problem Formulation

Our problem is a supervised machine learning task and we have converted it into a binary classification problem that given information about a pet, we would like to predict whether it will be adopted or not. The raw dataset contains a column of 'AdoptionSpeed' specifying the speed of a pet being adopted after it is kept in the shelter. We generate a new target column of 'Adopted' by defining that if a pet is not adopted within the first ten month, it is considered not being adopted.

We have a total of 14993 samples and 47 features after feature engineering, and the loss function we are using is the binary log-loss function

$$\mathcal{L} = -[y \log(p) + (1-y) log(1-p)]$$

where $y \in \{0, 1\}$ indicates whether the pets will be adopted or not with 1 being yes and 0 being no, and p is the probability of the pet being adopted.

The training data is obtained by sampling randomly from the imported data set using sklearn's train_test_split module with a fraction of 70% being the training set and 30% being the testing set. We use K-fold cross validation to tune the hyperparameters for each model and then train the model on the training set.

5 Methodologies

5.1 Feature Engineering

We use OneHotEncoder to handle categorical features (color, breed, state), generating 360 new columns. Encoding is necessary for these features because they have many variation. For example, there are around 300 hundreds different breeds within the dataset.

sklearn.preprocessing.OneHotEncoder

We use PCA for dimensionality reduction, resulting in 28 number of components enough to explain 90% of the encoded features.

sklearn.decomposition.PCA

5.2 Models

5.2.1 Logistic regression - Baseline

We choose Logistic Regression to be our baseline model for easy implementation and interpretation. It is easy to interpret how important certain feature is using coefficients. However, logistic regression works best for linearly separable datasets and requires no co-linearity among independent variables, and that is not the case with our dataset.

sklearn.linear model.LogisticRegression

5.2.2 Decision Tree

We run our decision tree classifier using the sklearn DecisionTreeClassifier and tune parameters by iteration. Decision trees are non-parametric classification algorithms that split and branch out greedily on features that maximize information gain. Then, we train another decision tree model using pruning, with the leaf and depth counts generated by the default decision tree as the maximum constraints.

sklearn.tree.DecisionTreeClassifier

5.2.3 Random Forest Classifier

Random forest is an ensemble learning method that consists of several decision trees and takes their majority vote for the classification task. It utilizes bagging and feature randomness to reduce correlation among trees to mitigate overfitting, and thus usually gives higher accuracy than Decision Tree.

sklearn.ensemble.RandomForestClassifier

5.2.4 Gradient Boosting

We use GradientBoostingClassifier to perform our gradient boosting method. Gradient boosting is an ensemble technique that integrates multiple weak learners to generate a final prediction. It keeps fitting the next model to the residual of the current one and then adds the new model to the previous model with a negative sign to reduce errors of previous models. The weak learner in Gradient Boosting is typically a decision tree.

sklearn.ensemble.GradientBoostingClassifier

5.2.5 AdaBoost

Similar to gradient boosting, Adaboost trains weak learners in a sequential manner. However, it assigns a higher weight to the samples that were misclassified by previous weak learners. The final prediction is made by combining the predictions of all weak learners, with more weight given to the stronger learner.

sklearn.ensemble.AdaBoostClassifier

5.2.6 XGBoost

XGBoost is an optimized distributed gradient boosting that implements gradient boosting with high efficiency and

flexibility. We used this library to efficiently train the model with regularization, and tuned the parameters by iteration.

XGBoost

5.2.7 Kernal SVM

SVM tries to find the best classification by maximizing the margins between support vectors. It's possible to use different kernel function other than linear function to handle complex non-linearity inherent in data. We tuned our SVM model by iteration for linear, polynomial, radial basis function and sigmoid function kernels.

sklearn.svm.SVC

5.2.8 AutoML

AutoML in sklearn performs automated model selection and hyperparameter tuning. This can also serve as a baseline model for our more complex machine-learning algorithm.

autosklearn.classification

5.3 Feature Importance and Cross Validation

We used sklearn's StratifiedKFold package to split data into training and validation set, and then trained a 5-fold cross-validated Gradient Boosting Decision Tree model in order to rank feature importance.

sklearn.model_selection.StratifiedKFold LightGBM

5.4 Evaluation Metric

We use a combination of accuracy and AUC to evaluate our model. The accuracy score is well suited for balanced datasets, since our data is slightly skewed as 72% of data falls in the "adopted" category, we take AUC into consideration when accuracy is not enough. High accuracy score means that the model makes few incorrect predictions. While it is useful, it does not consider the business cost behind those incorrect predictions. For our project, the True Positive rate means that we successfully indicate an animal that will be adopted to be adopted, and the False Positive rate represents the rate that we classify an animal that will not be adopted to be adopted. We might want low FP even at cost of TP because an animal that will be adopted will be adopted anyway but marking an animal that will not be adopted as adopted might cause the staff at shelters and rescuing agencies to allocate less time to improving that animal's profile, for example adding pictures, writing descriptions, naming, etc, to help increase the likelihood of it being adopted. Accuracy does not take into account such a tradeoff, whereas AUC does.

sklearn.metrics

6 Experiments and Results

6.1 Text Data

we tried to extract a no_name feature from the names of animals. To be more specific, we define no_name as animals with names "No Name Yet", "No Name", "No_name" or names with fewer than three characters. However, as we train the models with this new feature added, it did not improve the accuracy. This is likely due to the fact that there is only about 10% of the entries are marked as no name while the adoption rate does not differ significantly between the two groups. We also tried using CountVectorizer to count the frequency of appearance of each word in text and LatentDirichletAllocation as an approach for dimensionality reduction to map text data into probabilities over 5 topics. We also try to add a few more features regarding characteristics of the text data such as average word length and description length. Along with features extracted from sentiment analysis, there were 10 features from text data added to the training dataset. Yet, this did not improve the performance of the model.

sklearn.feature_extraction.text.CountVectorizer

sklearn.decomposition.LatentDirichletAllocation

6.1.1 Image Data

Another feature engineering technique we tried is to pass image data through a pre-trained neural network model(Dieter, 2018a), and then perform a Singular value decomposition to the outputs of the neural net to reduce the dimension. Then we use the corresponding PetIDs to connect with the main training dataset. However, this does not improve the accuracy of our current models as well.

6.2 Logistic regression

Our baseline model gives a test accuracy of 73%, which would be the same as accuracy from a dumb classifier that predicts all data to be "adopted" since 72% of our data falls into the "adopted" category.

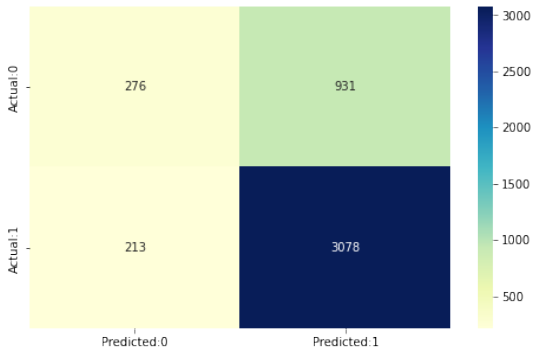

Figure 5: Confusion Matrix for Logistic Regression

6.3 Decision Tree

We started with the default classifier where min_samples_leafint = 1 and max Depth = None, which clearly overfitted. We then tried decision trees with pruning for the following parameters, which helped prevent overfitting and result in a higher test accuracy than the default decision tree classifier.

```
Obtain max_depth and max_leaf from the default
decision tree model:
   max_depth = dt_clf.get_depth()
   max_leaf = dt_clf.get_n_leaves()
Max depths and nodes for pruning method:
   depth=np.arange(10, max_depth, 5)
   nodes=np.arange(10, max_leaf, 2)
```

6.4 Ensemble Learning Model

Examining different ensemble learning models is the next phase in our experiment since ensembling multiple models often outperform the capability of individual model(Verma and Mehta, 2017).

6.4.1 Random Forest Classifier

We started with random forest since it is a good classification algorithm that can also help prevent overfitting. It results in a slight increase of the test accuracy. We tried the following values for parameter tuning, and observed that n_estimators = 500 gives the best testing accuracy of 76.033%.

```
n_estimators = [1, 5, 10, 50, 100, 500]
```

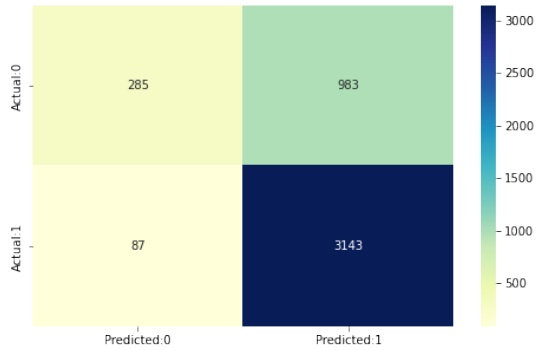

Figure 6: Confusion Matrix for Random Forest

6.4.2 Gradient Boosting

In gradient boosting, we tried out our parameters as follows, and a model with 50 estimators and max depth of 3 yield a test score of 76.011% with the following parameters:

```
num_estimators = [1,5,10,50,100,150,200]
learning_rate = 0.1
random_state = 20
max_depth = [1,3,5]
```

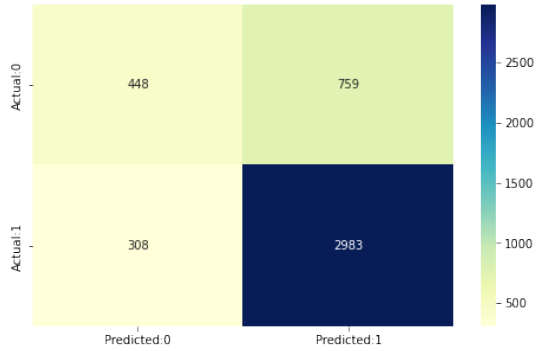

Figure 7: Confusion Matrix for Gradient Boosting

Even though the accuracy score of Gradient Boosting is approximately the same as that of a Random Forest Classifier and the False Negative rate has increased, Gradient Boosting is still preferred over Random Forest because there is a decrease in False Positive rate, meaning lest mis-clasification for animals who will not be adopted to be marked "will be adopted".

6.4.3 AdaBoost

We tund our model for the folloing parameters:

```
num_estimators = [1,5,10,50,100,150]
learning_rate = 0.1
random_state = 20
max_depth = [1,3,5]
```

Num_estimators = 100 ad max_depth = 3 leads to the best model and gives an accuracy of 76.256%, which is slightly better than the Gradient Boosting approach.

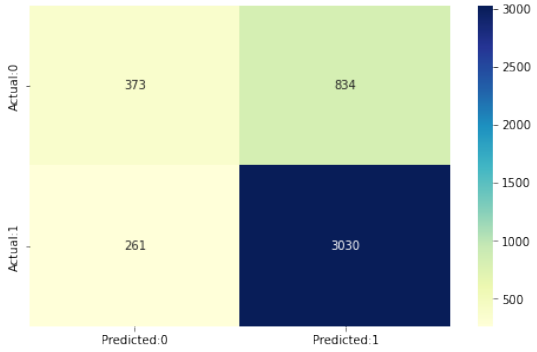

Figure 8: Confusion Matrix for AdaBoost

6.4.4 LightGBM

We used the following parameters for our model:

```
Params = {'num_leaves': 512,
          'objective': 'binary',
          'metric': 'binary_logloss',
          'max_depth': -1,
          'learning_rate': 0.01,
          "boosting": "gbdt",
          "feature_fraction": 0.9,
          "bagging_freq": 3,
          "bagging_fraction": 0.9,
          "bagging_seed": 11,
          "random_state": 42,
          "verbosity": -1}
```

LightGBM is the model that gives the best testing accuracy of 77.17%. Feature importance generated from this model will be discussed in following sections.

6.5 Kernel SVM

We tuned the model parameters with the following possible values and plotted the heatmap. According to our result, SVM with an rbf kernel with the regularization parameter equals 50.0 gives the best testing accuracy of 72%. In addition, non-linear kernel functions as well as large sample sizes as in our case caused the training to be extremely slow for kernel SVM, and thus kernel SVM is not ideal for us regarding runtime.

```
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
reg_params = [0.01, 0.1, 1.0, 10.0, 50.0]
```

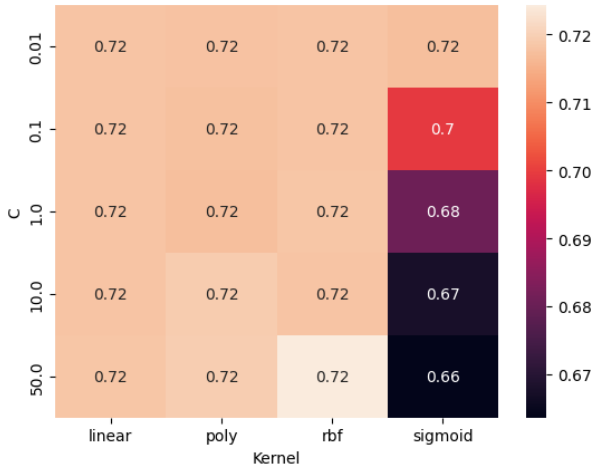

Figure 9: HeatMap for SVM

6.6 AutoML

For our AutoML model, we set the parameter as follows, and from the outputted leaderboard, we learn that out of all models that AutoML ran, gradient boosting has the best performance, and the test accuracy comes out to be 76.078%.

```
time_left_for_this_task=1200
```

6.7 Summary of Model Performance

Model	Train Accuracy	Test Accuracy
Logistic Regression	73.806%	73.855%
Default Decision Tree	99.999%	69.497%
Decision Tree with Pruning(best test score)	77.941%	74.900%
Random Forest(best test score)	99.980%	76.033%
Gradient Boost	86.850%	76.011%
AdaBoost	79.190%	76.256%
LightGBM	N/A	77.17%
Kernel SVM	72.939%	71.587%
AutoML	N/A	76.078%

Table 2: Train vs. Test Accuracy Summary

7 Conclusion and Discussion

7.1 Model Comparison

In this project, we conducted experiments with several different machine learning models. Our results, shown in the Train vs. Test Accuracy Summary(Table 2), indicate that on average, ensemble models performed the best. Of the models we tested, LightGBM produced the highest test accuracy for our binary classification problem, while Gradient Boosting achieved the highest AUC. While accuracy represents the rate of making correct predictions, an AUC score of 0.77 means that there is a 77% chance that our model will be able to distinguish between adopted and non-adopted classes. In general, an AUC of 0.5 should indicate no discrimination of the model, and thus our model should have fair discriminating ability. Moreover, the plot of the ROC curve of LightGBM, Gradient Boosting, and AdaBoost are almost indistinguishable from each other for high TP rate. It makes sense to see LightGBM and Gradient Boosting give similiar results since their major difference lies in efficacy and memory usage. They do, however, give lower FP rate than AdaBoost model when TP rate is low. In conclusion, these results suggest that ensemble models are well-suited for this type of problem, and that LightGBM and Gradient Boosting are particularly effective at predicting the adoption outcome.

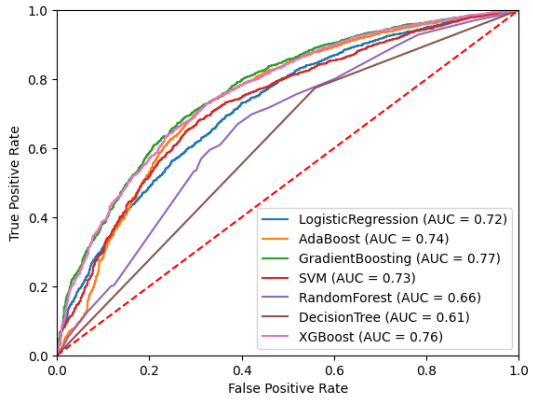

Figure 10: AUC Comparison over all models

7.2 Feature Importance

Using gradient boosting with 5-fold cross-validation, we are able to obtain the feature importance as shown in figure 11.

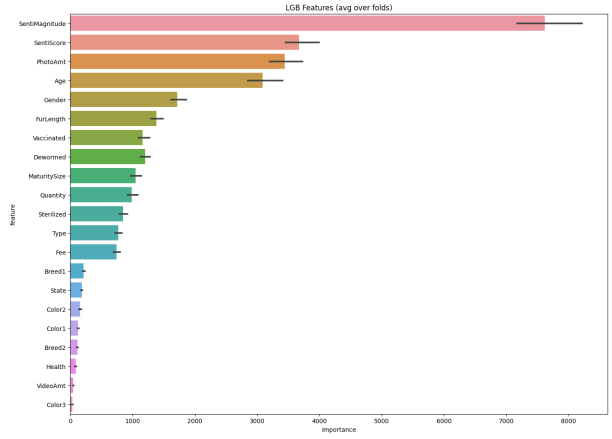

Figure 11: Feature Importance

From the graph, it's clear that the magnitude of the sentiment in the descriptions of each animal is the biggest factor in the final predicted adoption outcome. This makes sense, as the magnitude reflects the amount of emotional content in the descriptions, which is likely to impact people's decision to adopt an animal. Interestingly, the second most important feature is the sentiment score, which shows that a higher sentiment score is correlated with a higher adoption rate. This suggests that positive descriptions are more likely to result in an animal being adopted. Additionally, the total number of

photos uploaded for an animal also ranks high in importance, indicating that the number of images may be a deciding factor for potential adopters. The figure also shows that categorical data such as color, and breed have little significance to the predicted adoption outcome.

Overall, the results of this analysis highlight the importance of providing detailed, positive descriptions and numerous photos in order to increase the chances of an animal being adopted.

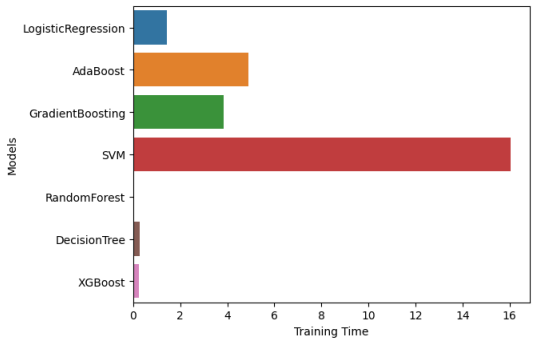

Figure 12: Model Training Time

8 Improvement

One improvement we can make is to create an ensemble method by combining the previous models, possibly by soft-voting to obtain the final prediction, or assign weighting to each model and learn the weights using linear regression. However, we decided not to go on this track because, as we have seen from the previous discussion, the AUC for each model and the accuracy score of their prediction are very close. Instead of devoting on building an ensemble method, extracting more features from the available data seems like a more promising approach.

Our first approach is to balance the data using imblearn's under-sampling and over-sampling implementation. The raw dataset is moderately imbalanced that it contains 70% of one target class and 30% of the other. We first use near-miss approach proposed by Jianping Zhang and Inderjeet Mani in their 2003 paper: it select samples from the majority class that have the smallest average Euclidean distance to the three closest samples. Then we use the SMOTE proposed by Nitesh Chawla, et al. to oversample the minority class of our samples. SMOTE select samples that are close by drawing a line between the samples and create new sample on that line. However, undersampling, oversampling or combined did not improve our prediction. The baseline method generate an accuracy score 10% lower. The reason might be because that the adoption rate is inherently higher than we thought. In other words, the majority of pets are expected to be adopted, and oversampling the not-adopted population or undersampling the adopted population would violate this fact, and thus polluting the data.

Our second approach is to incorporate the TF-IDF score obtained from the description as new feature. TF-IDF (term frequency-inverse document frequency) score is based on the intuition that whenever a words is appearing frequenly in the document, we should give it a higher score, but when a word appears across many documents within the sample space, we should down-weight its score (words like this include 'this', 'she', 'they' etc.). Our approach is to compute the TF-IDF score for all the words appearing in all the description of pets in the dataset, compute the sum of the scores for each document and divide that by the number of words contained in the description plus 1. The result is then used as the score for the description. With this new feature, we have a slightly improved prediction and ROC curve:

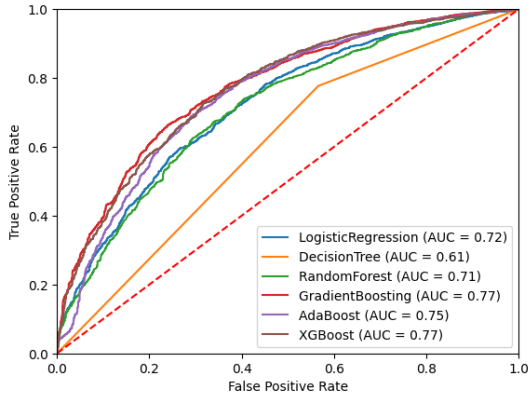

Figure 13: Improved ROC Curve (No SVM)

Due to time constraint, we ignored SVM training. The AUC improved very slightly, specifically, the AUC for random forest improved by 0.05, and for XGBoost improved by 0.01. On the other hand, if we look the feature importance plot again for new models:

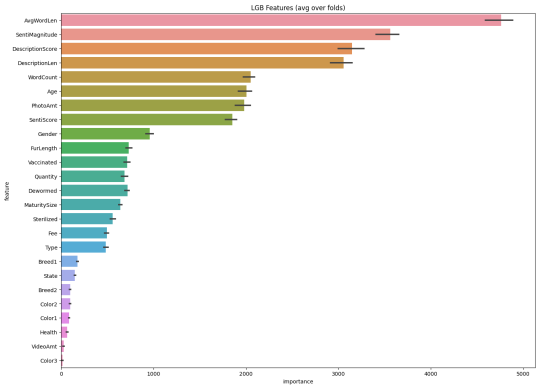

Figure 14: Feature Importance after Adding TF-IDF Feature

We can see that the description score is an importance feature (3rd) for prediction.

9 Acknowledgement

The authors are truly grateful to the University of Pennsylvania CIS 5200 instructor Dr. Ungar, TA Siming He, and all other TAs who helped with this project. They have given us valuable insight, guidance and support.

References

[BRUI2018] ERIK BRUI. 2018. PetFinder.my: detailed EDA and XGBoost baseline. https://www.kaggle.com/code/erikbruin/petfinder-my-detailed-eda-and-xgboost-baseline#2.-Data-exploration. Accessed: 2022-12-13.

[Dieter2018a] Dieter. 2018a. Extract Image features from pretrained NN. https://www.kaggle.com/code/christofhenkel/extract-image-features-from-pretrained-nn/notebook. Accessed: 2022-12-13.

[Dieter2018b] Dieter. 2018b. part of 9th place (denoising auto-encoder NN). https://www.kaggle.com/competitions/petfinder-adoption-prediction/discussion/88740. Accessed: 2022-12-13.

[gege2018] gege. 2018. g_features + XGB (part of 2nd place solution). https://www.kaggle.com/c/petfinder-adoption-prediction/discussion/88963. Accessed: 2022-12-13.

[kaerururu2018] kaerururu. 2018. 2nd Place Solution about k_features and LGBM2. https://www.kaggle.com/c/petfinder-adoption-prediction/discussion/88812#latest-514778. Accessed: 2022-12-13.

[LUKYANENKO2018] ANDREW LUKYANENKO. 2018. Exploration of data step by step. https://www.kaggle.com/code/artgor/exploration-of-data-step-by-step. Accessed: 2022-12-13.

[Minixhofer2018] Benjamin Minixhofer. 2018. 5th Place Solution Summary. https://www.kaggle.com/competitions/petfinder-adoption-prediction/discussion/88690. Accessed: 2022-12-13.

[ROSINSKI2018] WOJTEK ROSINSKI. 2018. BaselineModeling. https://www.kaggle.com/code/wrosinski/baselinemodeling/notebook. Accessed: 2022-12-13.

[takuoko2018] takuoko. 2018. 2nd solution tyu part. https://www.kaggle.com/c/petfinder-adoption-prediction/discussion/89042#latest-516326. Accessed: 2022-12-13.

[u++2018] u++. 2018. 1st Place Solution Summary. https://www.kaggle.com/c/petfinder-adoption-prediction/discussion/88773#latest-515044. Accessed: 2022-12-13.

[Verma and Mehta2017] Aayushi Verma and Shikha Mehta. 2017. A comparative study of ensemble learning methods for classification in bioinformatics. In *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, pages 155–158.

[XHLULU2018] XHLULU. 2018. Keras CNN Starter - PetFinder. https://www.kaggle.com/code/xhlulu/keras-cnn-starter-petfinder. Accessed: 2022-12-13.