Final Project: Animal Rescue Dog Returns Classification

Naïve Project Team
Rachel Kelley and Justin Schulberg

Table of Contents

Breakdown of Roles

For the most part, the tasks in the project were equally-shared between the two team members, Rachel and Justin. Almost all code and accompanying write-up was worked on by both teammates. For a general breakdown of tasks and their respective assignee, we have:

Task	Assignee
Requirements Gathering	Justin (Lead) & Rachel (Support)
Data Preparation	Justin (Lead) & Rachel (Support)
Data Cleaning	Rachel (Lead) & Justin (Support)
Classification Models + Analysis	Rachel (Lead) & Justin (Support)

For a more thorough breakdown of the tasks covered by each team member, feel free to access the *Issues* page of the GitHub project, where various tasks are assigned to the team member who completed them.

Problem Statement & Methodology

For this final project, the Naïve Project Team will be analyzing a dataset from a local Animal Rescue, a nonprofit dedicated to rescuing homeless, neglected, and abandoned animals from euthanasia in kill shelters and getting them adopted into their forever homes. The nonprofit educates the community and all pet parents on responsible pet parenting, including the importance of spay/neuter, obedience training, and good nutrition.

Successful adoptions of dogs depend on a variety of factors, both on the parts of the pet and of the family. The nonprofit helps bridge the gap, building strong relationships between the two. The nonprofit rescues hundreds of animals every year, provides them with loving temporary care, and finds them well-matched, carefully screened forever homes.

One of the struggles that the nonprofit runs into is pets being returned after adoption. Even though they adopt out ~2000 dogs per year, about 10% get returned for a variety of factors. The nonprofit assiduously tracks information on all their adoptions and returns. **In this project, the Naïve Project Team will use various analytical and classification methods learned in Computational Data Analysis to help the nonprofit predict whether or not a dog that they adopt out will be returned**.

To approach this, we followed these general steps:

1. Gather requirements
2. Get access to data
3. Clean data
4. Impute missing values
5. Resample data
6. Scale data
7. Split data into test-train sets
8. Run various classification methods on the train set
9. Evaluate the results against the test set
10. Tune the parameters of the most successful models

Data Source

The project will be overseen by the *Program Manager for Volunteers and Data Integrity* at the nonprofit. She will provide the data needed for this analysis to the Naïve Project Team. Data will be provided across various spreadsheets, detailing the adoptions and returns per year (i.e. Adoptions 2020, Returns 2018, etc.) from the nonprofit, both of which contain records over the past decade, and can be easily linked by a unique ID field for each dog represented in each dataset. The team has worked to programmatically concatenate the various adoption and returns spreadsheets from various years, which unfortunately are not always in the same format and sometimes have different column headers, into two main data sources:

- **Dog List** | A list of every dog that has been adopted out by the nonprofit over the past 10 years. This list includes a variety of features describing each dog, where each row corresponds to a dog being adopted out, and each column represents a different attribute related to that dog. This dataset is composed of 10 spreadsheets, one for every year over the past decade. Each spreadsheet has records of ~2000 adoptions per year. The attributes in this dataset are as follows:

Field	Description
Dog Name	Name of the adopted dog
ID	Unique ID corresponding to that dog
Link	Link to dog's profile on the nonprofit site
Foster/Boarding	Type of adoption for the dog (short vs. long-term)
Sex	Gender of dog
Age	Estimated age (sometimes a range if actual age not known) of dog at time of adoption
Weight	Estimated weight of dog at time of adoption
Breed Mixes	Type of dog
Color	Color of dog
Behavioral Notes	Free text field describing the behavior of the dog. There's some consistency in entries depending on the individual entering this field, along with some key words to describe the dogs behavior around others
Dogs in Home	Yes or No variable indicating if the dog gets along with other dogs or not. Also can show Required if a second dog is recommended.
Cats in Home	Yes or No variable indicating if the dog gets along with cats
Kids	Yes or No variable indicating if the dog is good around kids

BS/W	Indicator for whether or not the dog is part pitbull or other 'bully' breed
Medical Notes	Free text field describing any health conditions of the dog, if any exist at all
Transport Date	Date of adoption

- **Returns List** | A list of every dog that's been returned *after being adopted out* by the nonprofit. Each row corresponds to a returned dog, and each column represents a different attribute of the dog. This dataset is composed of 10 spreadsheets, one for every year over the past decade. Each spreadsheet has records of ~200 returns per year.

Field	Description
Dog Name	Name of the adopted dog
ID	Unique ID corresponding to that dog
Dog Info	Free text field with general information on the dog
Reason for Return	Free text field describing the reason the dog was returned by the adopters. There's some consistency in this field, but it generally depends on the person entering the information into the spreadsheet
Behavior with Dogs	Free text field describing the dog's general behavior around other dogs
Behavior with Kids	Free text field describing the dog's general behavior around kids
Behavior with Cats	Free text field describing the dog's general behavior around cats
Energy Level	General energy of the dog
Socialization/Daycare	Categorical field for whether the dog was in a daytime program for dogs
Vetting	Categorical field for whether the dog was being taken to the vet
Date of Adoption	Date dog was adopted
Previous Return?	Boolean for whether the dog has been returned in the past
Previous Return Info	Free text field describing why the dog had been returned in the past
Date of Return	Date dog was returned
Type	Boolean for dog vs. puppy

Data Preparation/Cleaning

To begin, we processed, prepared, and cleaned the data for analysis. Currently, all data is kept by the nonprofit in different spreadsheets by year (i.e. adoptions in 2021 are in the 'Dog List 2021'); so we programmatically combined all of our datasets into one master dataset. On top of that, the Dog and Returns Lists are kept as two separate data sources, but share a linking variable, 'ID', for dogs that were returned. We used a Left Join to find dogs that were returned; we flagged any dogs which did not have a match in the join as 'Not Returned'.

There were also some issues with the features provided in the data that needed to be addressed. Some of the work done to properly clean the different features is delineated below:

Columns

Color

We created a new column called *COLOR_FIXED*, which is the cleaned version of the COLOR column. After cleaning up the COLOR column with *COLOR_FIXED*, we also made the following new features for predictive purposes:

- **multi_color** | Variable indicating whether a dog has multiple colors denoted in it. This is calculated by seeing if the number of commas (',') in the COLOR_FIXED column is greater than 1 or whether the color is denoted as 'tri-color'.
- **num_colors** | Continuous variable counting the number of commas (',') in the COLOR_FIXED column.
    - Note: This feature is highly correlated to the feature **multi_color** since any dog which has num_colors = 1 ➔ multi_color = 0; conversely, any dog which has num_colors > 1 ➔ multi_color = 1. Though we believe that **num_colors** will be a better predictor than **multi_color** because it contains more information, we will test each of these features in our models **separately** to see which one ends up being more predictive.
- **contains_black** | Variable indicating whether 'black' appears in the COLOR_FIXED column. This feature, in particular, is being incorporated to test the hypothesis that black dogs are adopted less/returned more frequently than other-colored dogs.
- **contains_white** | Variable indicating whether 'white' appears in the COLOR_FIXED column.
- **contains_yellow** | Variable indicating whether 'tan/yellow/golden' appear in the COLOR_FIXED column.
- **contains_dark** | Variable indicating whether the dog has a darker coat, since the colors can sometimes be guesses. This could include both 'brown' and 'black'.

Gender

From the *GENDER* column, due to issues with the manual entry of values, we had to standardize all values as either 'Female' or 'Male', and then one-hot encoded this; 1 for 'Male', 0 for 'Female.

Age at Adoption

In the original data set, age was described in either days, weeks, months, or years, or just a number with no age measure, and some dogs had a listed date of birth. To get the most accurate age for each dog, we calculated age at adoption in days by subtracting the date of birth from the adoption date. Where this information was not available, age at adoption was calculated via the age column and converted to an estimate in days.

Breed Mixes

We cleaned the *BREED MIXES* column, which denotes a dog's breed. Because there are so many combinations of different breeds for dogs, we opted to create indicator features for the most popular groupings of dogs (Lab/Retriever, Shepherd, Terrier, Husky, or Other).

Mix

The column, *MIX*, denotes whether a dog is a mixed breed. We cleaned up some of the inconsistencies in the denotations of multiple breeds (i.e. the use of '&', 'and', '/', 'with', etc.). Also, we leveraged the *SECONDARY BREED* column to fill gaps for nulls. Lastly, we looked at the *BREED MIXES* column for a slash '/' or and '&' to determine if a dog is multiple (mixed) breeds. Lastly, we one-hot encoded this feature; 1 for 'Mix', 0 otherwise.

Weight

The column, *WEIGHT*, denotes the weight, in pounds of a dog. Although this column was more consistent than others, some dogs have their weight as just a number (i.e. 60) while others have their weight along with the unit of measurement (i.e. '60lbs' or '60 pounds').

Behavioral Notes

The column, *BEHAVIORAL NOTES*, includes comments about a dog's temperament. There's not too much cleaning that was performed here -- just some general standardization --, but there were a lot of good notes about dogs' temperaments. To pull out this data, we created the following features:

- **num_behav_issues** | Continuous variable denoting the number of behavioral issues/notes a dog has associated with it.
- **puppy_screen** | Variable indicating whether a dog has a behavioral note that it should be screened with puppies.
- **new_this_week** | Variable indicating whether a dog has a behavioral note that it is new in a given week.
    - *Note: We expect this variable to be the least helpful, since a dog being new is not necessarily associated with its adoption date.*
- **needs_play** | Variable indicating whether a dog has a behavioral note that it doesn't walk alone enough and needs extra running or playtime.
- **no_apartments** | Variable indicating whether a dog has a behavioral note that it should not be placed with adopters who live in an apartment, and should instead be placed with adopters who live in a residential home.
- **energetic** | Variable indicating whether a dog has a behavioral note that it exhibits medium-high energy.

- **shyness** | Variable indicating whether a dog has a behavioral note that it exhibits general shyness and should not be placed with kids (who generally expect dogs to be more energetic) and should be committed to a socialization program.
- **needs_training** | Variable indicating whether a dog has a behavioral note that it should be screened with puppies.

Medical Notes

The column, *MEDICAL NOTES*, includes comments about a dog's health conditions. There's not too much cleaning that was performed here -- just some general standardization --, but there were a lot of good notes about the different health conditions that each dog has. We extracted the following features by parsing through the medical notes:

- **has_med_issues** | Variable indicating whether a dog has a medical condition associated with it.
- **diarrhea** | Variable indicating whether a dog has a medical note that contains mention of diarrhea.
- **erlichia** | Variable indicating whether a dog has a medical note that contains mention of Ehrlichia (the disease Ehrlichiosis)/Lyme transmitted by ticks.
- **uri** | Variable indicating whether a dog has a medical note that contains mention of an upper respiratory infection.
- **ear_infection** | Variable indicating whether a dog has a medical note that contains mention of an ear infection.
- **tapeworm** | Variable indicating whether a dog has a medical note that contains mention of Tapeworm.
- **general_infection** | Variable indicating whether a dog has a medical note that contains mention of the word 'infection'.
- **demodex** | Variable indicating whether a dog has a medical note that contains mention of Demodex (Demodectic mange aka mites).
- **car_sick** | Variable indicating whether a dog has a medical note that contains mention of car sickness.
- **dog_park** | Variable indicating whether a dog has a medical note that the dog has not yet been exposed to a dog park, indicating a lack of general immunity to common diseases.
- **leg_issues** | Variable indicating whether a dog has a medical note that contains mention of any leg issues, including amputated, sprained, swollen, lesions, etc.
- **anaplasmosis** | Variable indicating whether a dog has a medical note that contains mention of Anaplasmosis.
- **dental_issues** | Variable indicating whether a dog has a medical note that contains mention of any teeth/dental issues.
- **weight_issues** | Variable indicating whether a dog has a medical note that contains mention of weight issues, including overweight and underweight.
- **hair_loss** | Variable indicating whether a dog has a medical note that contains mention of hair loss.
- **treated_vaccinated** | Variable indicating whether a dog has a medical note that contains mention of treatment or vaccination for the dog.

Dealing with NAs

For various reasons, a number of the columns mentioned above were missing values. It could be because no medical notes were reported, thus making it impossible to know whether a dog actually has a given medical issue; or because the information was unable to be discerned. To deal with these null values, we had the following options:

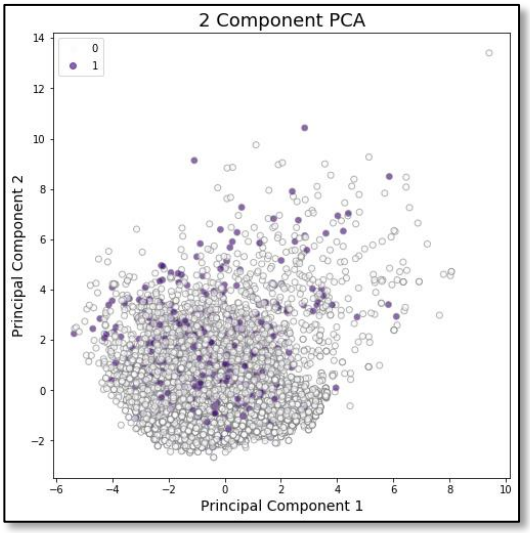
- Impute using mean of column
- Impute using median of column
- Impute using clustering algorithm (KNN, K-Means, etc.)

With guidance from the program manager at the nonprofit, we decided to combine a mixture of business logic and KNN imputation to impute NAs. In some cases, the program manager advised us that some of the features, if not noted, would be 0 – for example, *shyness* and *needs_play* are two features that are generally always noted for dogs with those features. Thus, for the rest of the dogs we can safely fill missing values with 0. For other columns, such as *weight*, *number of colors*, and *no_apartments*, we used KNN imputation in order to fill those values. KNN imputation allows us to impute the values of missing values with values that are seen in dogs with similar features. For example, the missing weight of a dog that *is_husky* would be imputed by looking at dogs of a similar breed and age, along with the other features that *are* reported.

It is worth noting that imputing missing values with KNN adds a degree of error to our results *before* we apply any classification models on the data.

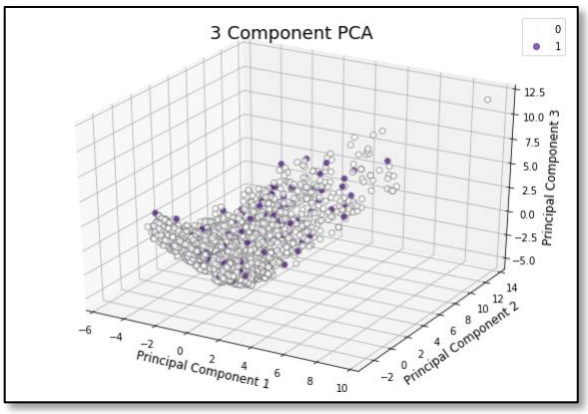
Dimensionality Reduction

After all the above data cleaning steps were completed, we found ourselves with 50 features! Many of these features were binary variables, while some were continuous variables. To tackle this, we applied PCA on our dataset. On the right, we can see what the first two principal components look like, shaded by whether a dog was returned (1) or not (0).

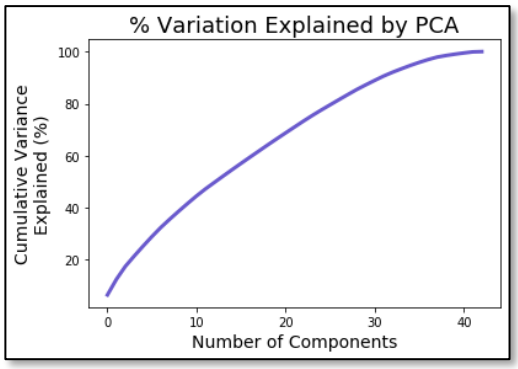


There's not much separating out the dogs that were returned (1) from the dogs that stayed adopted (0). Expanding this out to the first three principal dimensions, gives us the following 3-dimensional representation of our principal components, featured in the image below.

Again, it's hard to see much linear structure from these components regarding the indicator variable. Looking at how much variance in our data is explained by each successive principal component, we get the image to the bottom-left.



Looking at this graph, we see that the variation in the data is rather evenly described by each of the principal components. We can see this in the almost linear shape of the graph. That is, for each successive principal component included, about the same amount of variance gets explained. This generally makes sense; most of our features are binary variables. While we are theoretically able to transform the linear space of binary variables, the result is not particularly interpretable since their values are only 0 or 1. The results of this PCA lend this notion credence; going forward, our analysis will be applied directly on the data at hand and not a linear combination of the data.



It is worth noting that we also attempted to use Multiple Correspondence Analysis (MCA), a method similar to PCA, but functions better with data that contains *both* numerical and categorical variables. Unfortunately, when running our final classification models afterwards, we found that applying either PCA or MCA usually resulted in accuracy scores of up to 10% less than the scores when not using these methods.

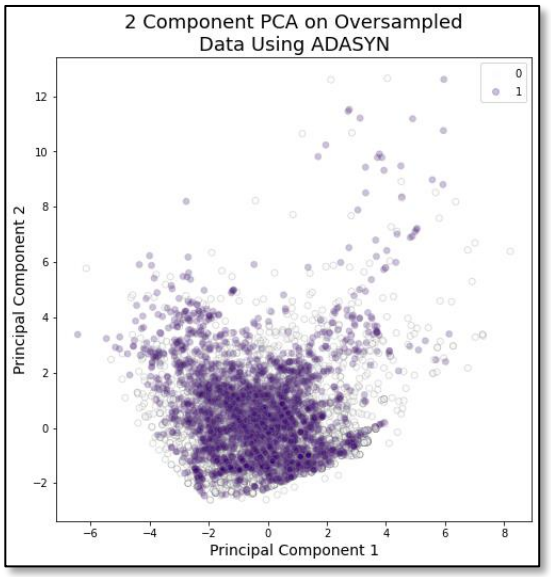
Balancing the Distribution of Data

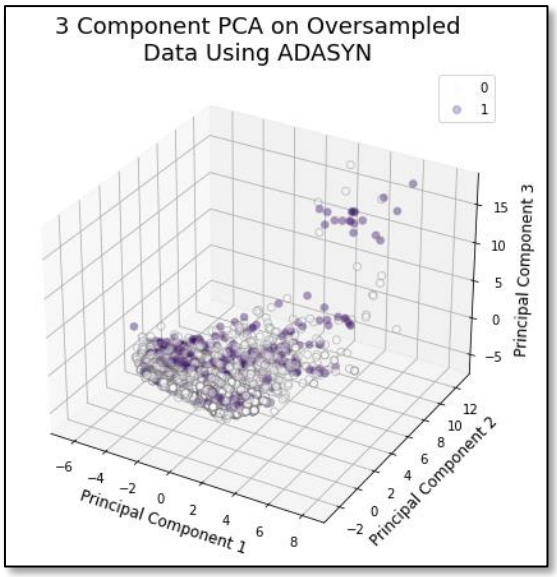
When we started our analysis, 7.8%, or 824 dogs had been 'returned' after adoption. That left our majority class of 'not returned' dogs at an overweighted distribution of 92.2%, or 9664 dogs. We sought to balance these two classes using resampling techniques before running the classification models.

To do so, we opted to leverage the Adaptive Synthetic (ADASYN) oversampling technique on the 'returned' minority class; and a Random undersampling technique on the 'not returned' majority class. We found ADASYN, because of its ability to put synthetic points in low distributed areas of the data and then use K-Nearest Neighbors to classify them, to give us the best representation of the minority class. However, we opted not to use ADASYN to get the minority class to the exact same size (9664) as the majority class; instead opting to increase the size by 3x (2871 'returned' dogs). To get the majority class down in size, we just used a random undersampler to randomly select data points from the majority class ('not returned') to keep, ultimately ending with about half as many points (5742 'not returned' dogs).

Class	# of Original Data Points	Resampled Data
'Returned'	824	2871
'Not Returned'	9664	5742

Even though this is still not perfectly balanced (i.e. a 1:1 ratio), we contend that this prepares the data in an accurate and unbiased manner for classification, as we expect less returned dogs than those that remain adopted. As we tried different techniques and ratios of data points to under/oversample, we applied PCA after the resampling to observe the distribution. Here are the 2-D and 3-D representations of the top three principal components of the resampled data:





Overall, this distribution looks much more reasonable. It's worth noting that resampling the data amounts to creating new (synthetic) dogs based on the features of old (actual) dogs. Doing this adds a degree of error to our results *before* we apply any classification models on the data.

Analysis, Evaluation, and Final Results

After preparing the data, we analyzed the data using the various classification models that we have learned so far in class including, but not limited to, Naïve Bayes, K Nearest Neighbors, SVM, Logistic Regression, Random Forest, Adaboost, and Neural Networks. We split the data into training and test sets and compared their performances. By testing each of these methods, we hope to identify which one is most successful in classifying the adopted dogs as returned vs. not-returned.

A Note on Measurements of Model Efficacy

As we went through running the various classification models, we best compared them using confusion matrices. While maximizing the overall accuracy of the model is an important metric to achieve our goal, we had a long conversation about which errors (Type I vs. Type II) would be more tolerable. In discussion with the *Program Manager for Volunteers and Data Integrity* at the animal rescue, we agreed that it was generally better that a model predicts a dog would be returned (1) even if it stayed adopted (0) than to have a dog be predicted as staying adopted (0) when in actuality it was returned (1). That is, it is better for us to focus on minimizing our false negatives (Type II Error). The best metric to actualize that is to look at the **Recall/Sensitivity score**:

$$true\ returns / all\ actual\ returns$$

Where '*all actual returns*' includes our predicted false negatives, which are actual positives. Recall tells us how many dogs we correctly predicted to be returned out of all dogs that were actually returned. Since we are focusing on the dogs that we believe will be returned, we are **seeking to maximize the Recall score**, or in other words, the correct classification of true positives/returns. In general, the higher the Recall score, the better chances we have of identifying dogs at risk of return.

However, we do not want to have an overly high Recall score at the expense of accuracy, as this would cause strain on the nonprofit to spend extra resources on a large number of dogs mistakenly marked as return. So, in addition to Recall, we looked at the accompanying accuracy, misclassification, precision, specificity, and F1 scores of each model. These scores show the degree to which the different classifiers correctly sort the data points. In more detail:[1]

Metric	Description
Accuracy	General measure of how many data points are correctly classified out of the total number of data points, so the percent of true positives and negatives out of the total.

[1] **Understanding Confusion Matrix, Precision-Recall, and F1-Score** by Pratheesh Shivaprasad; October 19, 2020

Misclassification	Opposite of accuracy and tells us how many data points were incorrectly classified, so the percent of false positives and negatives out of the total.
Precision	Positive predictive rate; that is the number of true positives out of predicted positives.
Specificity	True negative rate; true negatives out of all actual negatives
F1 Score	Statistical measure of accuracy for machine learning models, which combines the precision and recall score.

With these metrics in mind, here are the final results of the models we tested, with each model's performance being discussed in further depth below:

	Classifier	Accuracy	Misclassification	Precision	Recall	Specificity	F1
0	Random Forest	90.92%	9.08%	96.7%	76.76%	98.58%	85.58%
0	Adaboost	89.43%	10.57%	96.92%	72.18%	98.76%	82.74%
0	KNN	86.67%	13.33%	76.58%	89.36%	85.21%	82.48%
0	SVM	85.4%	14.6%	97.35%	60.07%	99.11%	74.29%
0	Decision Tree	84.94%	15.06%	78.37%	78.89%	88.22%	78.63%
0	Neural Networks	84.25%	15.75%	81.97%	70.7%	91.59%	75.92%
0	Kernel SVM	83.85%	16.15%	97.41%	55.48%	99.2%	70.7%
0	Naive Bayes	73.74%	26.26%	67.04%	49.59%	86.8%	57.01%
0	Logistic Regression	71.61%	28.39%	69.83%	33.72%	92.12%	45.47%

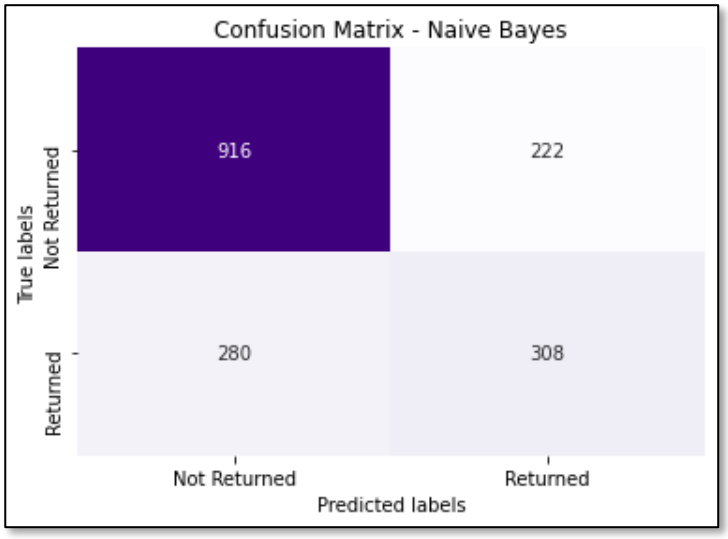
To evaluate our models, we looked at their classification/misclassification rates using a confusion matrix, as well as all the scores mentioned above. These metrics are essential in judging the outcome of a classification model, which is why we will use it to evaluate ours. We compare the results of our test set in order to determine which model is the most accurate.

If our models are accurate, they can be integrated into the nonprofit's systems to classify new dogs that come into the nonprofit. If a dog is classified into the return group, the nonprofit can ensure that the dog is well matched with its new adopter and reach out to the adopter to provide additional support to prevent the dog from being returned. We plan to construct this project so that the nonprofit's data team can continue to use it, helping to place more rescued dogs in loving homes and keep them there.

In the next section, we go through the results of each classifier in more depth.

Naïve Bayes

The Naïve Bayes classifier was overall the worst classifier tested. Without any tuning, its accuracy score of 48.8% was mostly brought down by its abysmal recall score of 26.7%. That is, the classifier predicted that too many dogs in the test dataset would be returned, even though in actuality only about 10% of dogs will be returned.

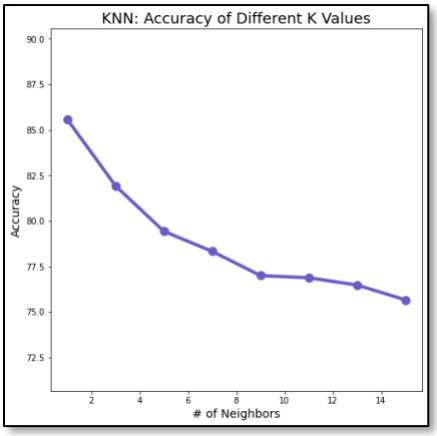


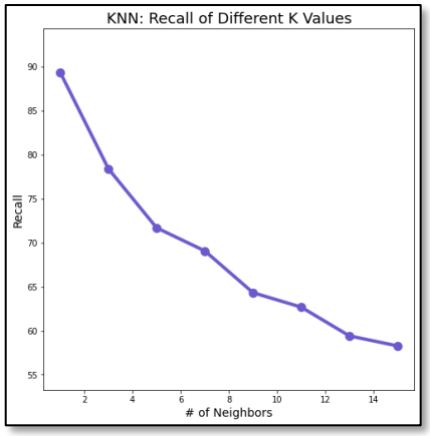
After tuning using *GridSearchCV* on the *var_smoothing* parameter, we were able to increase the accuracy up to 73.74% and recall up to 49.6%; however, this still included a lot of key misclassifications in our returned set.

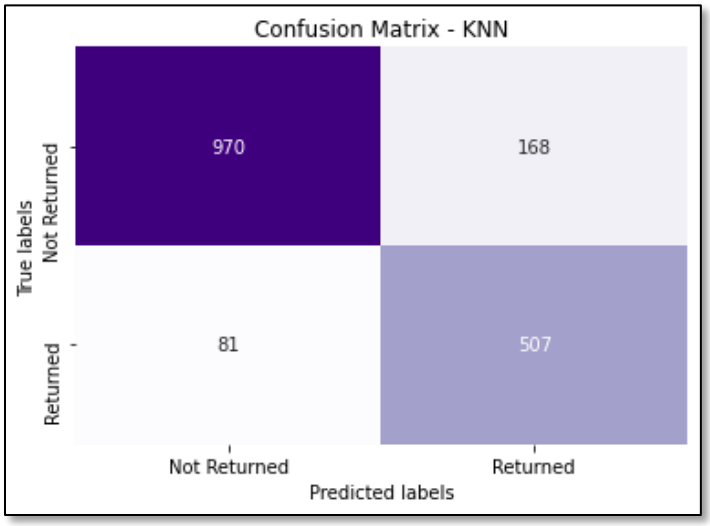
KNN

For K-Nearest Neighbors, we tested all values of 'k' for the parameter specifying how many neighboring points to look at to make a classification determination. As found in the graph below, we achieved a maximum accuracy and recall when we used a 'k' value of 1. That is, the KNN classifier was most accurate when a given test point was only compared to the point nearest to it. This likely has to do with the distribution of our data, which is difficult to parse in a general feature space because of how many of our features are categorical and not continuous, thus making them hard for an algorithm like KNN to properly map and relate different test points together.

After applying a 'k' value of 1 on our entire test dataset, we found a much higher overall accuracy for our dataset:

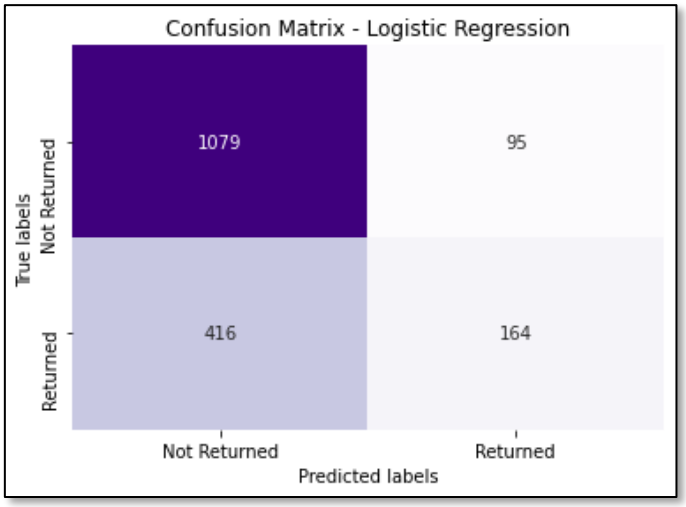






Logistic Regression

Logistic Regression appeared to be an enticing classifier. If it proved to be accurate, we could leverage not just the prediction values, but also the associated probabilities (calculated from the log-odds ratios) of return. During our testing of the Logistic Regression classifier, we looked at various solvers and values of the inverse of regularization strength (C) to find the most fitting value.



After using our optimal values, we still only managed to achieve an accuracy score of 71.61%, with many of our misclassified dogs being the ones we want to avoid (False Negatives).

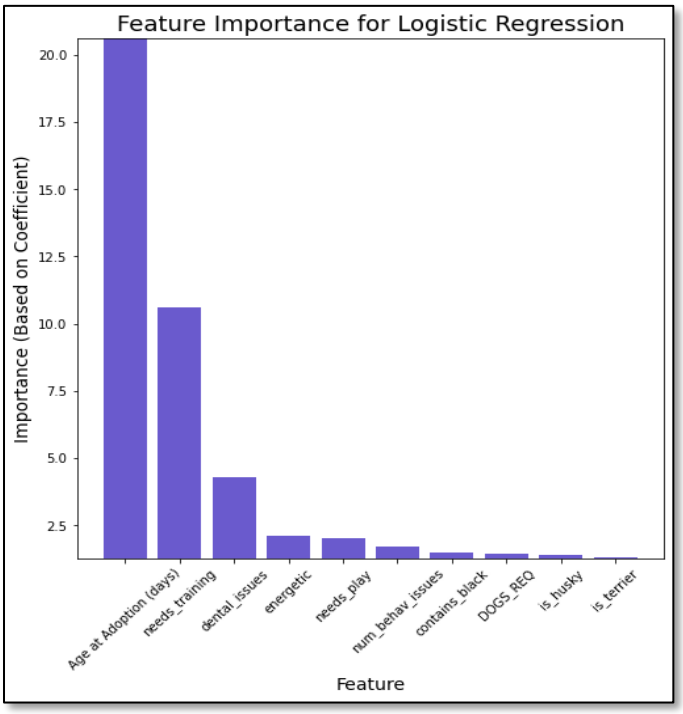
One other benefit to Logistic Regression worth mentioning is that it's a very explainable model. That is, we can specifically call out the values for the coefficients that make up our final equation, like so:

y = 2.0427 – 0.4804 * num_colors + 0.4013 * contains_black + 0.1397 * weight_issues + … + 0.0274 * HW_FIXED

where the coefficients/intercept above are rounded to the 4th decimal point and y is of the form

$$y = \frac{1}{(1 + e^{-z})}$$

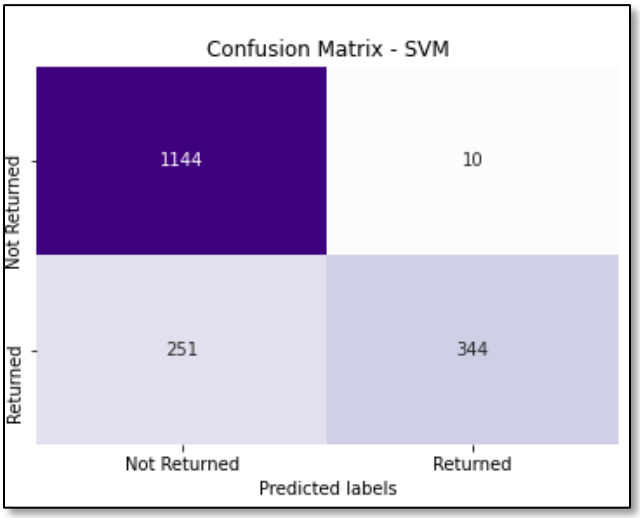
In a binary classification problem, we can simplify our equation by exponentiating both sides, giving us a better sense of the weights of our actual coefficients in calculating P(y = 1). Doing so, gives us the top 10 coefficients in the chart to the right.



Linear SVM

The team attempted to use a Support Vector Machine (SVM) classifier, but set its expectations low. SVM often works off a well-defined mapping space; however, we learned from our attempts at dimensionality reduction through PCA that the data does not lend itself well to such interpretations. This is again because most of the features in our dataset are binary/categorical.

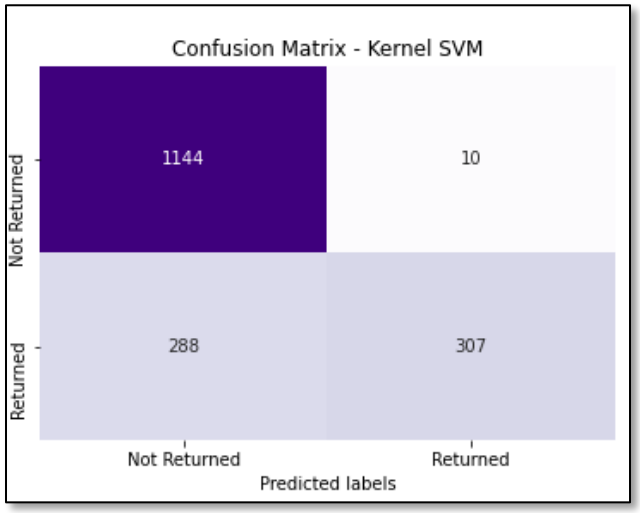
However, with proper scaling of the data, SVM can still work in theory. That is, it can still create linear vectors separating data points between 0 and 1. Thus, building a linear decision boundary would be possible, but not likely as meaningful as if our features were continuous variables. In practice, the accuracy of Linear SVM was 85.4%. Linear SVM produced the confusion matrix to the right.



Kernel SVM

The team had hoped that Kernel SVM would provide a bit more flexibility in capturing the nuances of binary/categorical data than did Linear SVM. That is, because Kernel SVM allows us to use a kernel function to project our data into a higher dimensional space, it may be able to better capture the non-linear shape of our data.

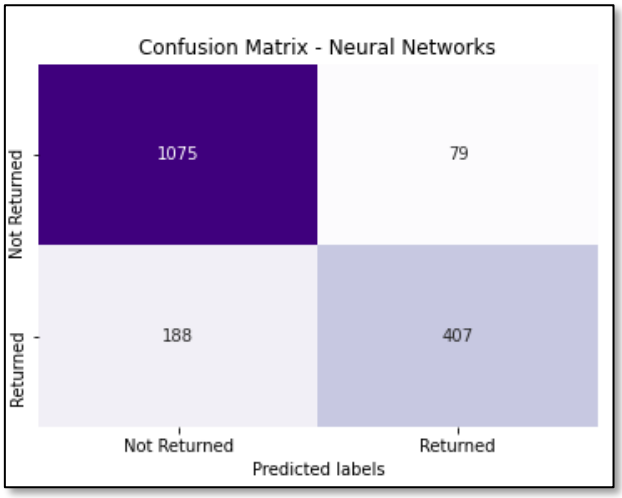
With that, we used a radial basis function (RBF) kernel and achieved an accuracy of 83.85% and the confusion matrix to the right. In fact, Kernel SVM performed worse than did Linear SVM.



With more time, it would be worthwhile to try to define our own kernel function that operates better in a feature space heavily defined by categorical variables, potentially allowing us to increase the performance of the model. This paper by Marco Antonio Villegas Garcia lays out an interesting framework to build a kernel function of that sort.[2]

Neural Networks

The team ran a neural network using a Multi-layer Perceptron (MLP) classifier, incorporating two hidden layers – the first with 20 neurons and the second with 10 neurons. With this, we achieved a general accuracy of 84.25%.



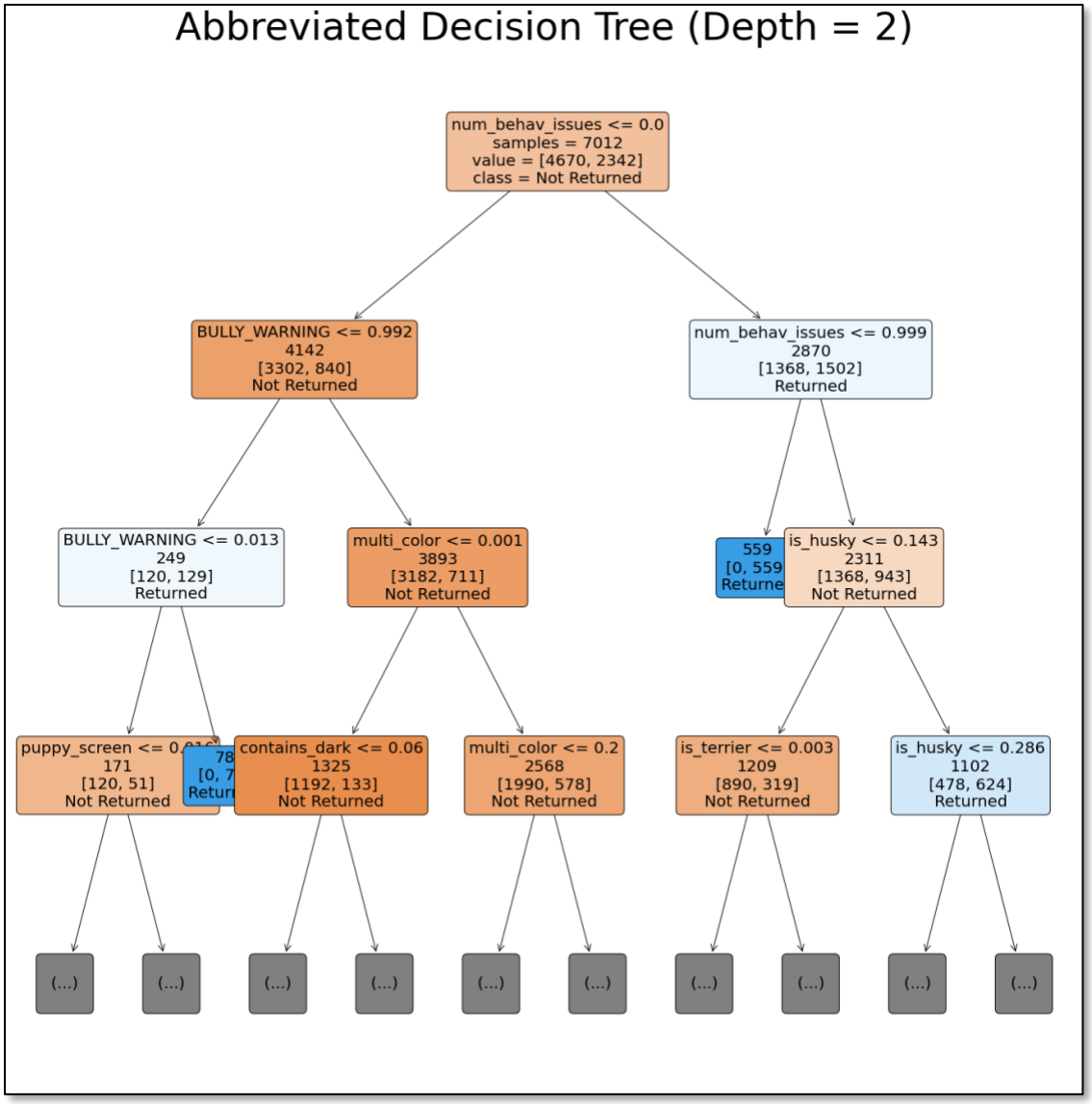
One thing to note is that, unlike other classifiers whose errors were mostly Type II, the Neural Network did a better job balancing Type I and Type II errors.

With more layers, more neurons, a different activation function, and/or weight initialization, we could seek to increase the performance of the neural network, at the cost of more expensive computing.

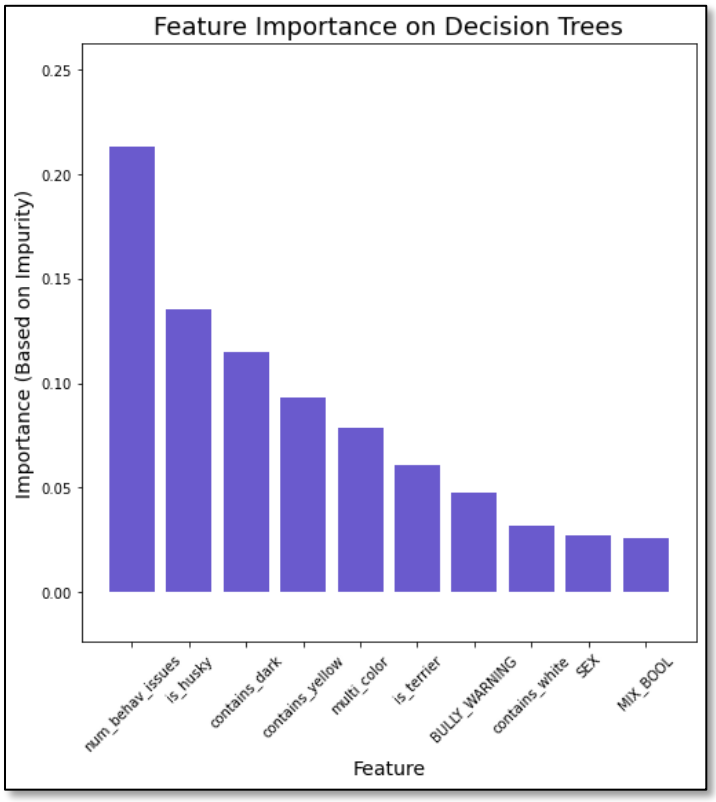
[2] An investigation into new kernels for categorical variables by Marco Antonio Villegas Garcia, January 2013

Decision Tree

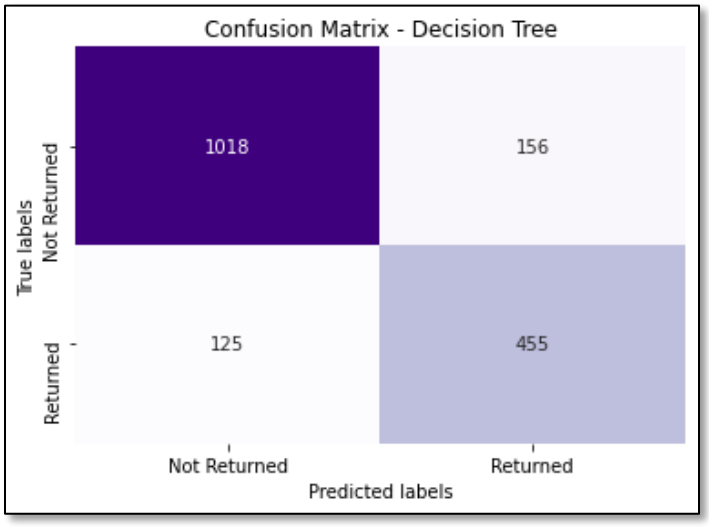
The team used a Decision Tree classifier and achieved an accuracy of 84.94%, and the second highest recall score of 78.89%. When looking at the first few layers of the decision tree, we found promising results. Variables that we would have expected to be key decision makers appear at the top. For example, *num_behav_issues*, which tracks the number of behavioral issues related to a given dog, is the first node. If there are 0 behavioral issues reported, we look to see if the dog has a warning attached to them that they are a bully (in the *BULLY_WARNING* variable).



Which variables ended up being the most important? **Not necessarily the variables at the top nodes**. Why is that? Well, feature importance in decision trees is based on an algorithm that also looks at the number of times a feature appears at a given node. With that, we get the following top variables:

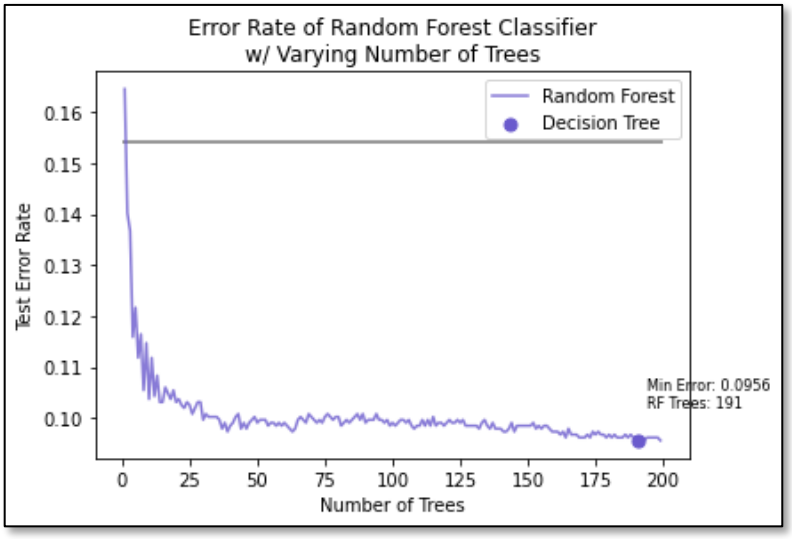


This also lines up generally with what we expected. For example, dogs that have many behavioral issues tend to get returned more frequently than dogs that do not. With these variables of interest, the Decision Tree classifier achieved an accuracy of 84.94% and the confusion matrix to the right.

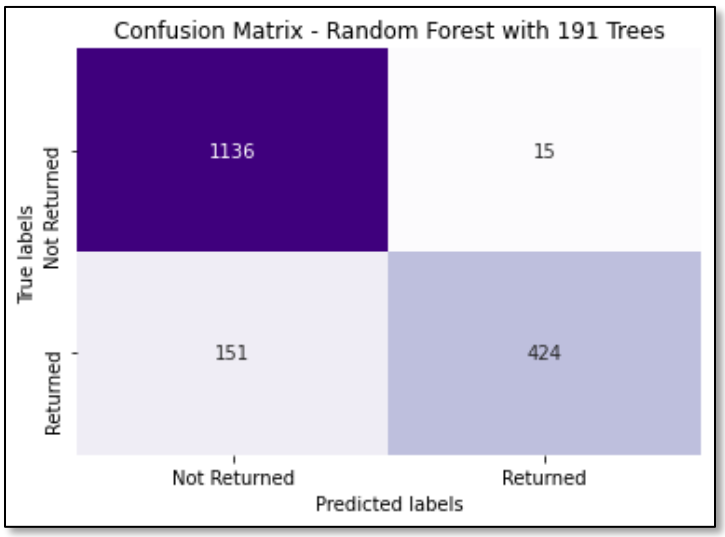


Random Forest

Our most successful classifier in terms of accuracy is the Random Forest, achieving an accuracy rate of 90.92%. As seen below, we tested a varying number of trees, ultimately finding that the optimal number of estimators is at 191. Across most values for the number of trees, Random Forest performed better, in terms of accuracy, than did the Decision Tree classifier.

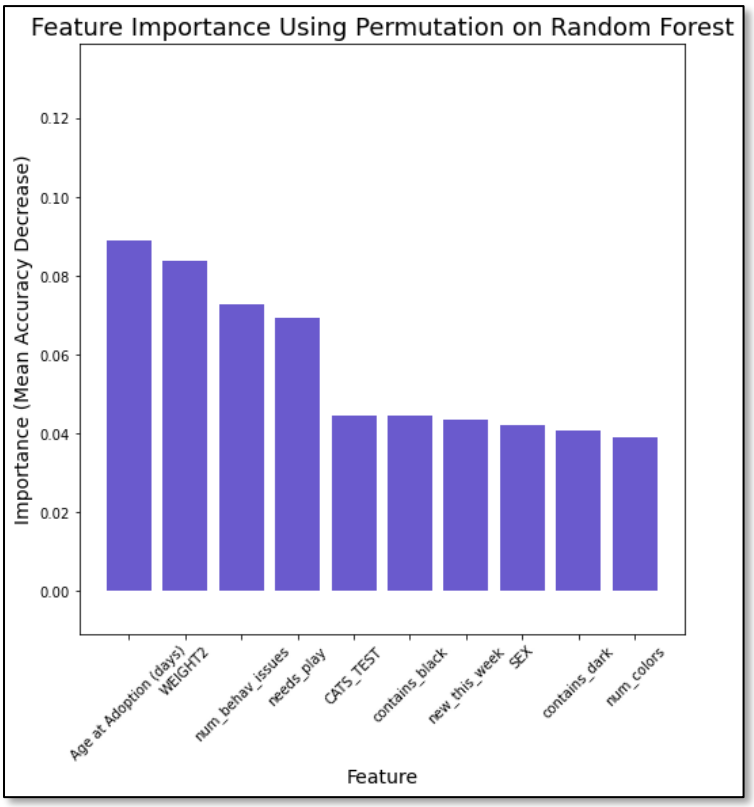


After identifying the optimal number of trees to be 191, we re-ran the Random Forest classifier to get the following confusion matrix:



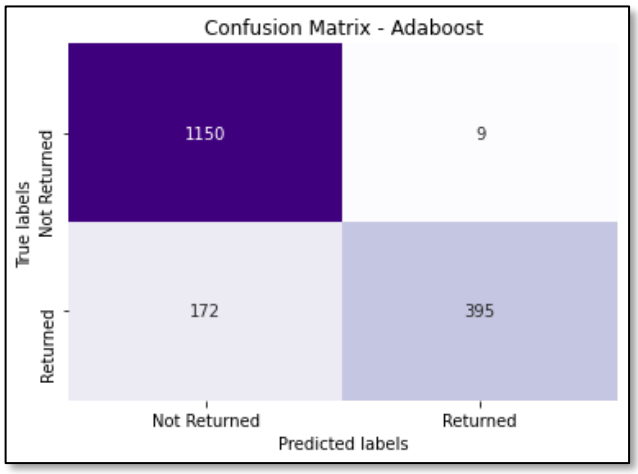
In terms of accuracy and F1 score, this is our best result out of all the classifiers tested. When we look at the features that prove to be the most meaningful (using a permutation importance), we find promising results, as shown below. That is, features which we would

expect to be indicative of a return (a dog being too old, over/underweight, having many behavioral issues, etc.) are all reflected as the most important features.



Adaboost

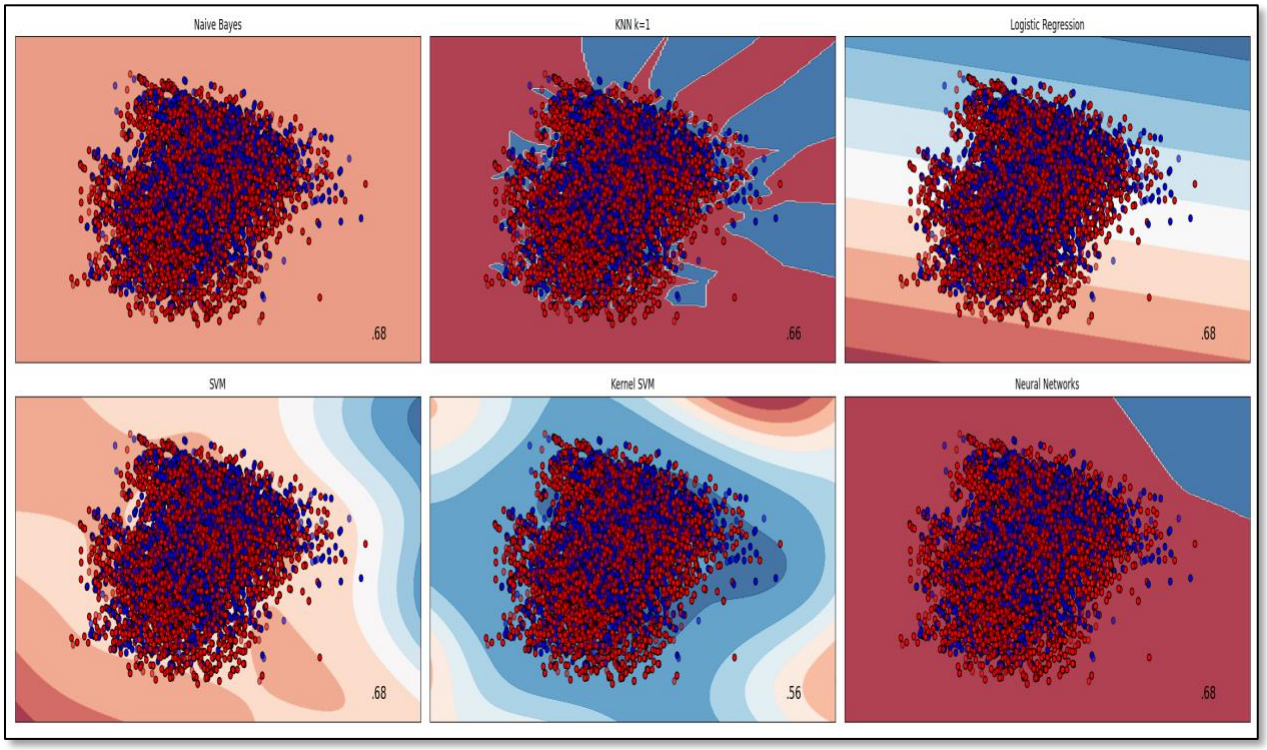
The last classifier we ran, [Adaboost classifier], achieved a final accuracy of 89.43%. We tested various parameters, including the maximum number of estimators at which boosting is terminated and the learning rate, which shrinks the contribution of each successive classifier (since Adaboost is an ensemble of weak learners). We found that *n_estimators = 500* and *learning_rate = .1* performed best, yielding the confusion matrix to the right.



Mapping the Decision Space

In order to get a better picture of how some of the classifiers split the data, we plotted the first two principal components as separated by the classifiers below. As outlined in the dimensionality reduction section earlier on in the report, the first few principal components do not represent a large percent of the variance in the data, so these plots cannot fully represent our classifiers' performance. We do see a trend of blue (or returns) being concentrated in the top-right corner for these models, but it is not completely helpful in trying to show the classification of the dogs, as the data is much more reliant on multiple components/features, which is difficult to plot as a two-dimensional image.

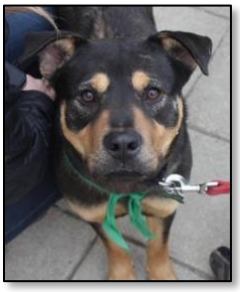
The nature of this data is thus why the linear classifiers did not perform as well. Our data is difficult to separate solely by a boundary, so the classification methods of other models, such as Neural Networks, KNN, Decision Tree, Random Forest, and AdaBoost suit the data better. Those models work by separating the data into smaller groups based on their features (DT & RF), nearby known datapoints (KNN), combining weak classifiers (AB), or grouping the points by weights and thresholds (NN). However, as you can see below, as PCA is incapable of fully representing our data, the accuracy scores of the methods below (run on the first two principal components) decrease drastically compared to the full data set.



An Example

What does all this mean? Well, let's use our most accurate classifier, Random Forest, to predict whether the following two dogs will be returned:

- Zach (*ID #PICK-MD-14-0037*), who was adopted and subsequently returned
    - Gender: Male
    - Weight: 50lbs
    - Age (days): 1460
    - Breed: American Staffordshire Terrier Mix



- Cajun (*ID #MTHY-MD-15-0081*), who was adopted and stayed adopted (thus, never being returned)
    - Gender: Male
    - Weight: 40lbs
    - Age (days): 730
    - Breed: Shepherd

If we run these two dogs through the Random Forest model, we find that Zach is predicted as being 'returned' and Cajun is predicted as being 'not returned'. This aligns with the true results of the two dogs. Even so, it is instructive to see why this might be the case. Earlier we saw that the variable `BULLY_SCREEN` and `BULLY_WARNING` played a strong role in the classification models. Zach was, in fact, returned due to a 'Facial Bite'. We can see this reflected in his behavioral notes (`num_behav_issues = 6`) and him being marked for a bully screen (`BULLY_SCREEN = 1`). Cajun, on the other hand, had minimal behavioral issues reported and was not screened or warned as a bully. Thus, the model did a good job in correctly predicting the results of these two.

Conclusion

After running all nine of our classification models and looking at their resulting performance, we came to the following conclusions. The models with the best performing accuracy and F1 scores are Random Forest and Adaboost; however, KNN has the highest recall and does the best job at minimizing the Type II Errors (prediction: Not Returned | actual: Returned) which is a factor that we care about the most. However, this is at the expense of a slightly higher misclassification and lower specificity rate. As KNN places 168 dogs as False positives vs. Random Forest's 15, depending on the shelter's resources, a lot of time and effort could be expensed on dogs that are not at risk of return, or the nonprofit might not even have enough resources to provide extra care for all the dogs classified as return. Thus, in choosing our best model, we sought to strike a balance

between accuracy and the recall score in order to taken into consideration the shelter's resources.

So, we chose the Random Forest model as our "best model." It has the highest accuracy and F1 score, and the third highest recall score. Overall, this model gives a good balance of dogs marked as returned compared to overall accuracy, which allows the shelter to reach out and provide extra training, support, and adopter guidance to the dogs marked as return.

But as we mentioned previously, recall is an important metric for the nonprofit. Thus, if the nonprofit gains additional resources or can provide support for more dogs at risk of return, we could later reassess and look at switching to the Decision Tree or KNN models, which had the highest two recall scores. But to reiterate, this would result in more dogs being falsely classified as returns, since these two models had lower accuracy scores.

Additionally, it's important to note that we were able to achieve a high accuracy on the test set despite using KNN to impute missing values and ADASYN to resample the data, both of which add a degree of error to our final model. Thus, given the scores we achieved overall, we believe the Random Forest classifier will be a good choice for the nonprofit to use to help the dogs at risk of return.

Opportunities for Further Analysis

During this project, the team, along with the *Program Manager for Volunteers and Data Integrity* at the nonprofit, discussed a variety of different analytical approaches. Some approaches we opted not to pursue in the interest of time; however, we would be remiss if we did not mention some of these opportunities for further analysis.

One of the most pertinent opportunities lies in the viewpoint of *why* a dog is returned. Our project mainly focused on attributes related to the dog; thus, making a core assumption that the reason why a dog is returned is *solely* attributable to the dog. However, attributes of the dog only make up half of the picture when an adoption (and subsequent return) takes place. The other half of the picture lies on the shoulders of the adopter: the individual/family adopting the dog.

The animal rescue tracks extensive details and features about the adopters (largely collected from a thorough 45 minute interview and accompanying questionnaire) to better understand the individuals adopting the dogs. As is sometimes the case, the reason a dog is returned may be because of the adopter: a lack of experience with adoptions, an inability to train a dog, an allergy that they did not know existed, etc. To complete a thorough analysis, we would try to **incorporate these features about the adopter to better make predictions**. That being said, the nonprofit is limited in how it can change the behavior of adopters; it can provide extra training, treatment, care, or socialization for dogs at risk of return, but it is difficult to do the same for humans. Thus, we believe our models are helpful in identifying the factors that the nonprofit is in control of, and is a good first stepping stone in reducing the amount of returns.

In terms of preparing our data, we likely could have employed **feature selection methods like LASSO regression** to determine which features are meaningful and which ones are redundant (i.e. highly correlated). The way the features were built leads us to believe that inclusion of some of the features increases how overfit the models are; removing some of them (like the medical issues or behavioral issues) would likely not seriously degrade the performance of the models used.

Another option to reassess the results of our models would be to look at changing the classification threshold to help focus on returns/increasing the recall score. However, as we want to avoid overclassifying dogs as returns, we would have to fine tune the threshold to create a balance between the overall accuracy and recall scores.

Additionally, the results of this project would be most beneficial to the animal rescue if there were a way to **act on the results of the model**. That is, if the model identifies a dog will be returned, can we alert the animal rescue so they can follow-up with the family and provide better resources to ensure the dog happily remains adopted? There are programmatic ways of doing so, but finding a method that can work in near-real-time (and, most importantly, before a dog is actually returned) would require a bit more software engineering and likely a bit of webscraping than we initially implemented for this project.