Project Kojak Summary

This project originally focused on classifying animals entering shelters in to one of five outcomes. It morphed into binary classification of animals in shelters as adopted or non-adopted. I built a model to classify animals in to five classes, but this summary will focus on the binary classification model.

Data

I obtained my data from the Austin Animal Center via Kaggle. My data had 80,000 animals in it. A significant challenge in this data was the large number of categorical variables and the small number of numerical variables. I'll talk more about this below.

Tools

Python

- **Pandas**
- **NumPy**
- **Matplotlib**
- **scikit-learn**
- **Imb-learn**
- **Catboost**
- **Skopt**
- **Wikipedia module**

Google Docs
Google Slides
Javascript
HTML
Flask
Jupyter Notebook
VSCode
Git
Vim
iMovie
Quicktime

Modeling

The biggest technical challenge of this project was in dealing with a large number of categorical variables that had huge amounts of categories in them. Most notably, the breed and color

features of animals had a large amount of categories. I needed to reduce these two features into smaller categories while still retaining *most* of their informational value for my model. For breeds, I ended up using the Wikipedia module to get the group each dog breed in my dataset belonged to. I downloaded the Wikipedia page “List of Breeds Recognized by the American Kennel Club” to do this. This meant that instead of “American Staffordshire Terrier,” the breed feature would simply be “Terrier.” Lots of dog breeds were entered with a typo, or had their breed listed slightly differently than the American Kennel Association, so this meant lots of manual editing via Vim.

I reduced the depth of the color feature using a similar method. This ended up being a much greater challenge, and I think some information was lost in this case. This is because the specific color of an animal is very distinguishing. It’s a bit of a stretch to say that a “taupe” dog is the same as a “golden” dog, but I ended up having to label both “taupe” and “golden” as “yellow.”

The color field in my dataset had another important piece of information buried in it: the pattern of the animal. This is a very important feature to know about an animal, but I ended up being unable to use it. The format of many entries was “color / pattern,” e.g. “Brown / Sable” which would indicate the animal has a brown body with some black parts, like the nose, mouth, and paws. I would have loved to create a pattern feature, but the data was just too dirty. Many features had weird inputs, like “Yellow / Yellow”, and many of the inputs totally omitted the pattern feature. I ended up having to drop the pattern feature, but I think with more time I could have used the breed of the animal to impute the pattern of the animal.

I did a lot more feature engineering not related to breed / color. This included datetime features such as the day of the week the animal was brought in, the season the animal was brought in, the fixed/intact status of the animal, and perhaps most importantly the name frequency of the animal.

The frequency with which the animal’s name appears in my dataset turned out to be the most important feature for my model (with the animal’s age in days close behind). From what I can tell, the more unique the name the more likely the animal is to get adopted.

It’s important to note I was modeling almost the entire time I was feature engineering, to see what was working and what wasn’t. I initially started with Random Forest and XGBoost because I’ve had good results with those models for classification in the past. XGBoost was giving me the best results for a while, but then I overheard that Catboost is good for handling large amounts of categorical variables. I plugged my data in to Catboost and got the best results I had seen thus far.

After hyperparameter tuning, my cross-validated AUC was .78, but AUC was .71 on test data. I still don’t know why I got such a big drop on test data, but I observed this phenomenon with every model I tried. I’m guessing my particular train test split might have led to this, or perhaps

hyperparameter tuning led me to overfitting on the training and validation sets despite the inclusion of L2 leaf regularization in my model.

I eventually realized precision for the non-adopted class is a more important metric for me. This is because correctly identifying animals that are unlikely to get adopted provides the most value to the shelter, allowing the shelter to allocate resources to those animals which are less likely to be adopted. I ended with a precision of .80 for the non-adopted class on test data.

Conclusions

My model had mixed success in the end. I'm happy that it had high precision on the non-adopted class, but of course it would have been nice to be able to be able to reliably identify those animals that went on to get adopted as well.

I think there are several things that my data didn't totally capture in terms of an animal's potential to be adopted. In fact, I don't know if data *can* capture everything about an animal's potential to be adopted. I think a lot of the reason an animal gets adopted or does not get adopted has to do with hard-to-quantify factors, like an animal's socialization, and how trusting it is of potential adopters who come in to the shelter.

I was very surprised to find that the name (or rather an animal's name's frequency in my dataset) is the most important feature in my model. The name of an animal makes up almost 25% of the information gain according to the mean impurity decrease method of determining feature importance in my model. Coincidentally, the name of an animal is also the one thing the person bringing the pet to the shelter has total control over. With this knowledge (and a suggestion from Roberto) in hand, I built a Flask app that 1) predicts whether an animal will get adopted or not, and 2) suggests a different name to maximize the probability of adoption of the animal.

I'm happy with my Flask app -- it generally gives a 10-20% boost to the probability of adoption, depending on how frequently the name you put in to the app appears in the dataset, and I think this functionality is the coolest part of my project.

Overall, I'm proud of this project. I think its best use case is when it's used in combination with the expertise of shelter workers -- it's not quite precise enough to be used as the sole tool in identifying less adoptable animals.

Next Steps

My goal is to use this project to get some volunteer analytics experience. I spoke to the owner of adopt-a-pet.com and he directed me towards an organization call Shelter Animals Count, who have indicated they might have some interest in free analytics. I'm going to follow up with them

after the project, and hopefully either use this model or any other general analytics skills I've picked up to help them out.

Shelter Animals Count collects data from thousands of shelters in a much more standardized way than the dataset I had. I believe the dataset they have available would add to the predictiveness of my model, especially if it has socialization data.

I wanted to do a regression on the number of days an animal spends in the shelter, but never had time for it while I was doing this project. I'd like to add this to my analysis in the future as well.

What I'd do differently

I spent too much time modeling in retrospect. I wanted to make my model better so badly, but I hit a ceiling that I wasn't able to improve upon about halfway through the project. I could have focused more on improving my Flask app and how I was going to communicate my results and really show why my model provides value to a shelter.