Project 1 Report

COE 379L

Prof. Joe Stubbs

Owen Scott

25 September 2024

Data Exploration, Preparation, and Insights

Upon opening the dataset, I saw that there were about 130,000 rows of data about pets from animal shelters, including Animal ID, Date of Birth (of animal), Name (of animal), DateTime (of admission to the shelter), MonthYear (of admission to the shelter), OutcomeType (either "Transfer" or "Adoption"), Outcome Subtype, Animal Type, Sex upon Outcome, (animal's biological sex and whether it was neutered/spayed/neither), Age upon Outcome, Breed, and Color. I handled each of these as follows

Animal ID: There wasn't a lot of special handling here. I left it in the dataset until the end to prevent over-de-duping. I excluded it from my dataset models because it isn't a factor that I believe would be meaningful in predicting OutcomeType.

Date of Birth: I didn't think date of birth would be a particularly valuable set of data. People might consider how old an animal was when they were considering adopting it, but no one I know stops to think, "Hmm, I wonder what day/month/year they were born." Many pet owners aren't even aware of that for their own pets. I did try and use Date of Birth in combination with DateTime of admission to the shelter and Age upon Outcome to identify when the animals entered the shelters and the durations that they were present in the shelter, but I got a negative duration in the shelter 97% of the time, which lead me to believe that the Age upon Outcome metric was more of an age approximation than a mathematically reliable duration. Because I couldn't obtain reliable data on shelter duration, I dropped those attempts, as well as the Date of Birth column.

DateTime & MonthYear: I validated that there weren't any rows where MonthYear was present but DateTime was not, and then dropped the MonthYear column because it was redundant. I thought the year and time components of when animals entered the shelter would probably be

meaningless when it came to predicting OutcomeType, so I dropped those and converted them to Weekday and Month values, and then converted those values into ‘circular’ values ie, Weekday_sin and Weekday_cos, and the same idea for Month. That was a trick that ChatGPT taught me for handling circular data, such as weekdays and months. Python marks days 0-6 and months 0-11, but days 0 and 6 and months 0 and 11 are right next to each other despite being as far as possible from each other in KNN dimensions. Converting them into sin and cos values captures that relationship and makes it more obvious to the KNN that 0/6, 0/11 are neighbors in a circle and not opposites.

OutcomeType: I left this as-is, but binary encoded it into 0 (Transfer) and 1 (Adoption)

Outcome SubType: I dropped this column. If the point of the assignment is to use ML to predict an outcome, it doesn’t seem fair to make that prediction based on another thing we know about the outcome.

Animal Type: This column was fully present, so I just one-hot encoded it.

Sex upon Outcome: This column was mostly full, and I filled in the unknown values (about 3% of the data) with the mode of the set since I thought that would be the least significantly impactful on the set.

Age upon Outcome: I converted this variable into integer days and later, when it was time for training, applied StandardScaler for statistical normalization of the variable.

Breed: This column was dropped per assignment instructions.

Color: I found that the color column often described multiple patterns *and* multiple colors or types of cats. Conflating patterns with types of cats does blur the lines a bit, but I think trying to separate out these details into more individually meaningful columns is more useful than just blindly one-hot encoding a bunch of different color strings. So, I set up four columns, Colors A

and B and Patterns A and B. If 0 colors were specified, they were both ‘none’; if 1 was specified, they were both that color; and if 2 were specified, one column was allocated to each color. If 3 were specified, I either rounded (in cases like silver/white/grey/cream) or excluded them from the dataset. To do all this classification, I applied some standardization, regex, and string splitting, and then tried basic functions that parsed the data and reported the set of unique values they couldn’t parse with value_counts(). That told me what to prioritize in my parser. Ultimately, I wrote a color/pattern parser that got 99.5% of all the rows into the 4-column description of their coats. I dropped the remaining .5% since I didn’t feel the juice was worth the squeeze. Some of that .5% were odd niche animals I had never heard of as well, so it might have been a good thing to prevent those outliers from clouding my dataset.

Deduping: I did dedupe the data after dropping the breed and outcome subtype columns. That removed a small part of the data, and I didn’t do it again after that because all the data I had removed or standardized made it more likely that I would drop “duplicates” that actually used to be distinct rows.

Model Training

I trained 3 models using basic sklearn functions. I created a reproducible test/train split and then fit a KNN, a GridSearchCV, and a LogisticRegression to it.

KNN Results: The KNN performs pretty well, with a weighted average F1 of .83. It does a great job predicting Adoptions (1), but often thinks that Transfers (0) are going to be adopted instead. This is likely due to bias in the dataset and, if I investigated and confirmed that, I may be able to improve model performance by resampling toward a more even distribution in the dataset.

GridSearch CV Results: The GridSearch CV performs slightly better (0.01 higher in weighted average F1 score) but has the same issues of bias toward incorrectly picking Adoption when

presented with an input that should be marked Transfer. This bias was actually slightly higher than in the KNN.

Logistic Regression: The logistic regression model actually performed the best, although it beat GridSearch CV by only 0.01 in weighted average F1 score. It too has the bias issue, to the most extreme extent of the 3 models.

Overall, I am quite satisfied with the models' performance. Being able to accurately predict if an animal is going to be adopted or transferred, based on only the input features I prepared, more than 80% of the time, is impressive in my view. And, I think with more fine-tuning, a couple more percentage points of performance could be squeezed out of it. We also dropped the Breed column for some reason… perhaps that information would help too.