

3/26/25 - ALL

- Hosted a meeting about our organization, how we were going to document as we go, and how we want to get started

3/29

- Transported the CSV into a google sheets to see it better
 - <https://docs.google.com/spreadsheets/d/1excOeFCz1pAn5sld59E3xcBU6U7hhiQY6DqsxePFbLw/edit?usp=sharing>
- Created a data cleaning guide to consider the features we wanted to keep (check Data Cleaning Guide)
 - Made visualizations and observations about the data we wanted (check Data Cleaning Guide)

Colab link:

https://colab.research.google.com/drive/1n3lR0vf3ojc9V_-tGVvYsFPR_n9bbILQ?usp=sharing

- Created a data cleaning guide

4/5

- Going through our data engineering and feature engineering guide
 - Added in dog_breeds.csv
 - https://www.kaggle.com/datasets/yonkotoshiro/dogs-breeds?select=dogs_cleaned.csv
- Finished up our exploration and moved on to our official Data Cleaning for our first attempt

4/6

- More feature engineering (see data cleaning guide)

4/8

- Decided to make different models for cats and for dogs
- Made a data cleaning and feature engineering code to run on test.csv and train.csv
- Changed the model from GridSearchCV -> decision trees to randomized search CV -> XGBoost because training decision trees took too long. Kernel crashed while running, but we achieved a cv score of 0.59 i the best iteration (dubya)
 - For unseen data, pipeline + transformer will set to 0 (might want to change later to handle unseen variables as medical conditions like Panleuk)
- Got a CSV file with predictions using XGBoost and a downsampled dataset
- Later trained XGBoost with whole dataset, got a score of ~37%
- Now trying various things – not splitting the cats and dogs, adding more parameters/iterations etc.
- Realized we should have been using **balanced_accuracy** and switch to that for internal checking

4/9

- Nneoma creating Naive Bayes **Multinomial** model
 - Multinom. bc it can handle categorical data that is not in a normal dist.
 - Note that breed and color are NOT independent - going to just assume they are right now, will adjust later by either removing breed or removing color
 - Downsampling the majority class and NOT splitting on cat/dog got me better results but in either case, the most i could get was a balanced accuracy of .34 with an alpha of .4 while testing it in the CV loop → not sure why we are overpredicting the majority class - **used a confusion matrix to get better insights**

```

ut[col] = ut[col].map(freq_series)
Confusion Matrix:
[[4339    0    1  227  364]
 [  54    0    0    5   43]
 [ 152    0    7   42   96]
 [ 742    0    0  672   49]
 [1760    0    1  159 1287]]
Classification Report:

```

	precision	recall	f1-score	support
Adoption	0.62	0.88	0.72	4931
Died	0.00	0.00	0.00	102
Euthanasia	0.78	0.02	0.05	297
Return to Owner	0.61	0.46	0.52	1463
Transfer	0.70	0.40	0.51	3207
accuracy			0.63	10000
macro avg	0.54	0.35	0.36	10000
weighted avg	0.64	0.63	0.60	10000

- Tried to implement **SMOTE sampling** in order to see if i can get a better training when i do that
 - Used SMOTENC that handles numerical and categorical with onehotencoder in pipeline set to **sparse_output=True** (stops MemoryError)
 - GOT 50% ACCURACY WHEN I USED IT WITH DOWNSAMPLE!!!!
 - Did not split dog / cat
 - Downsample to 50K
 - SMOTENC X_train, y_train
 - OneHotEncoder in pipeline should have sparse_output=True
- Rebecca's neural net
 - Couldn't run it on local machine (ran out of memory)
 - Couldn't finish on CS machine puffins (killed after 1 hour)
 - Ran on Condor using these steps:

1. Made executable using this [link](#)
2. Compiled .sub file:

```
#####
# Neural Net Training for CS 363M, Rebecca Wang
#####

+Group          = "UNDER"
+Project         = "OTHER"
+ProjectDescription = "training a neural net on austin animal center
data"

Executable      = ./dist/neural_net
RequestMemory    = 8192
RequestCpus      = 4
Requirements     = InMastodon
Rank             = Memory

Error           = err.log
Input          = in.txt
Output         = out.txt
Log            = train.log

notify_user     = rwang1129@gmail.com

queue
```

3. ssh'd into darmok machine
4. Condor_submit train_neural_net.sub

- Rebecca's catboost
 - Using SMOTENC to offset majority class = Adopted while encoding categorical features
 - Using Casey's confusion matrix + balanced accuracy scorer by importing through ml_project
 - SMOTE with threshold at < 5% chooses Euthanasia and Died as rare classes
 - Catboosting is nice because it doesn't require encoding for any features!
 - Added and pushed sklearn class weighting but didn't run it yet

	precision	recall	f1-score	support
0	0.66	0.79	0.72	11009
1	0.04	0.16	0.06	208
2	0.23	0.48	0.31	690
3	0.64	0.50	0.56	3319
4	0.67	0.41	0.51	7005
accuracy			0.61	22231
macro avg	0.45	0.47	0.43	22231
weighted avg	0.64	0.61	0.61	22231

4/10

- Nneoma tinkering with Naive Bayes and SVM
 - Ramping up Naive Bayes with feature engineering (?) / categoricalNB
 - Observed the probabilities of different features using the **nb.feature_log_prop** and discovered we had 270K features in our model 🐼

	Adoption	Died	Euthanasia	Return to Owner	Transfer
intake_time_1380613860	0.000003	0.000003	0.000003	0.000004	0.000003
intake_time_1380616380	0.000003	0.000003	0.000003	0.000003	0.000004
intake_time_1380623908	0.000003	0.000003	0.000003	0.000004	0.000003
intake_time_1380625140	0.000003	0.000003	0.000003	0.000003	0.000004
intake_time_1380626880	0.000004	0.000003	0.000003	0.000003	0.000003
...
primary_color_15171	0.001019	0.001004	0.000980	0.000884	0.001092
primary_color_15241	0.000940	0.000664	0.000958	0.001657	0.000782
primary_color_24265	0.001367	0.002656	0.001804	0.000403	0.001724
primary_color_24993	0.001469	0.001036	0.001894	0.002267	0.001526
primary_color_54492	0.003141	0.004713	0.003631	0.003296	0.003062

271111 rows × 5 columns

- Sincerely promoted doing more feature engineering
- **IDEAS** (not yet implemented)
 - Hierarchical modeling with multiclass SVM (Adopted vs Not Adopted, and then separate models again for the different subgroups)
 - Either stacking where those Not Adopted can never be voted adopted again (Adopted gets looked at better) or just a straight up different model after separation
 -
 - Ensemble mix of Multiclass SVM and NB - adopted needs to get 80-90% of notes to be considered, else we will go to the next one (adjusting threshold)
 - Ensemble of just SVM for now?

1. Train the First Stage Classifier (Adopted vs. Not Adopted):

- Use a binary classifier (like SVM or any classifier you prefer) to separate **Adopted** and **Not Adopted** classes. Make sure it's highly accurate at this stage.

2. Ensemble of SVMs for Stage 2A (Adopted):

- For the **Adopted** class, train an **ensemble of SVM models** (each with slightly different hyperparameters or kernel functions) to divide the **Adopted** class into its subcategories.
- Use a **voting system** to ensure that all models in the ensemble agree before deciding the class. If they disagree, decide based on the majority or a weighted decision.

3. Train a Multiclass Classifier for Stage 2B (Not Adopted):

- For the **Not Adopted** class, train a **multiclass classifier** (e.g., SVM, Random Forest) to classify a sample into one of the other 5 categories.
- Here, ensure no confusion or overlap with **Adopted** classes.

4. Final Prediction:

- For each sample, first apply the **Stage 1 (Adopted vs. Not Adopted)** classifier.
- If it's classified as **Adopted**, pass it to **Stage 2A** (Adopted ensemble classifier).
- If it's **Not Adopted**, pass it to **Stage 2B** (multiclass classifier).



- Currently building SVM One v. All model for first stage - Adopted vs Not Adopted
- Stochastic Gradient Descent with SVM

4/11

- Ideas for alternate Feature Engineering
 - Bin age
 - Don't use the full intake time - use just month and/or date

there are so many breeds with "Breed X / Breed Y"
but bc the clean_breed sees the string "Breed X / Breed Y" as one breed
when really, we need to include with Breed X and Breed Y

- Rebecca fighting for her life catboost again
 - Added stratified k-folds so that we are guaranteed rare classes in every CV fold - balanced accuracies in CV seem to be doing better!
 - Added weighting classes -> CV normal accuracies seem to go down a lot
 - Confusion matrix looks much more balanced - better performance on minority classes!

	precision	recall	f1-score	support
0	0.73	0.54	0.62	11009
1	0.04	0.46	0.08	208
2	0.20	0.56	0.30	690
3	0.42	0.76	0.54	3319
4	0.65	0.36	0.47	7005

accuracy			0.52	22231
macro avg	0.41	0.54	0.40	22231
weighted avg	0.63	0.52	0.54	22231

```
df['intake_time_cos'] = np.c
```

If `intake_time` was a timestamp:



And you just converted it to an integer (`int64` Unix timestamp), then:

- It's not very informative — raw timestamps don't carry meaningful patterns for ML models unless transformed.

Summary:

- Yes, converting to `int64` works syntactically, but you'll likely get better performance by extracting features like:
 - Hour of day
 - Day of week
 - Sine/cosine transformation for cyclicity

- Nneoma trying to figure out why her OneHotEncoder made 271,115 features 😭
 - Naive Bayes should not be using one hot encoder
 - She should be using stratified k fold to try and help with class imbalances
 - Naive Bayes is not doing frequency encodings and its putting way to many features for breed / color, so i have given up on this
 - Nneoma went to do data exploration for intake time (updated in data exploration tab) + decided to change her intake time into month, hr, weekend, etc in her branch
 - Nneoma tinkered a bit with a Linear SVC model
 - Used enhanced data cleaning with transforming intake time and treating them as **numerical values** for LinearSVC
 - For naive bays, they would have to be categorical
 - Had many hyperparamters values (param_distributions) and just to look at it, toyed with both ovr (one v all) and where they check all combos of classes against each other
 - Squared hinge is better bc it's more punishing for misclassifications
 - Linear SVC kind of flopped, so Nneoma tried looking at regular SVC before doing her hierarchical + ensembling idea
 - Realized i wasn't scaling... I need to scale:
 - Year (standardization)
 - Transforming hour / month into sin/cos already standardizes it to be in a similar range
- Rebecca finding more datasets

-  dog breed info (what we had before but looks nicer in an Excel sheet)
-  dog popularity + group type holds interesting group information about dog groups (herding, toy, terrier...) + popularity statistics (probably just use 2023)

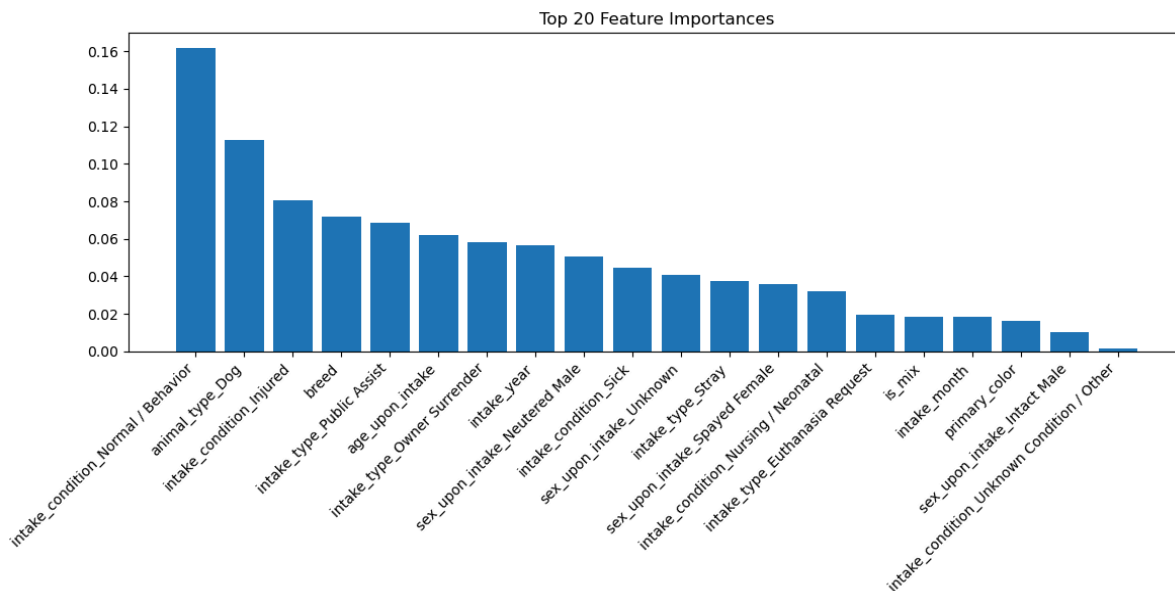
4/12

- Rebecca catboosting again
 - Adding SMOTE with 3000 new minority class records makes accuracy go down to ~0.45 instead of consistent .5+ w/o SMOTE
 - Trying no SMOTE + fixed intake times (from time -> month, year): accuracies seem to be similar and it's taking a bit longer (~10 minutes at its longest when before it was ~8 min)
 - Trying to boost threshold for predicting adopted with custom_cross_val_with_threshold + added prints for performance on cats vs. dogs
 - This gives me an error with tricolor and I can't figure out why T_T
- Rebecca extra random trees
 - Learned extra random trees are faster than random forests and work better on large datasets, I feel like this is more ideal
 - First run: fixed intake time from Nneoma, class weighting
 - Frequency encoded breed + color, one hot encoded everything else
 - Chose to keep stratified k-folding and class weighting
 - It ran in 1 minute and got a .47 accuracy
 - Added One vs Rest classifier wrapper but seems to be doing worse - best CV balanced accuracy = 0.41
- Nneoma working on feature engineering / data cleaning tasks
 - Made the improvements to intake time as found in Data Cleaning
 - Exploring correlation with dog breed facts to see if there are alternate ways to work with dog breeds from the dog breeds csv file
 - Edited dogbreeds.csv to make 'Chihuahua' more specific for shorthair v longhair
 - What exactly to tweak came from chat gpt
 - Put anything with a / (Ex: Labrador / Pitbull) into the Mixed Dog Group
- Rebecca clustering dogs_cleaned.csv breeds (*not* clustering original dataset)
 - Figuring out best k-value using elbow method + sil score averages
 - K-means recommended because I want to cluster all breeds, and it works well with the high amount of features we have unlike DBSCAN which would have been hard to tune and taken longer to run
 - Ran k-means with k=9 and generated file output that can be used to map breed names to a cluster

4/13

- Rebecca tried the feature engineering (seasons replacing months) on Catboost (no thresholding because that's causing bugs), but seems to be doing meh with CV scores ~0.50

- Rebecca testing feature importance using extra random trees (orange = random forest, blue = extra random trees)

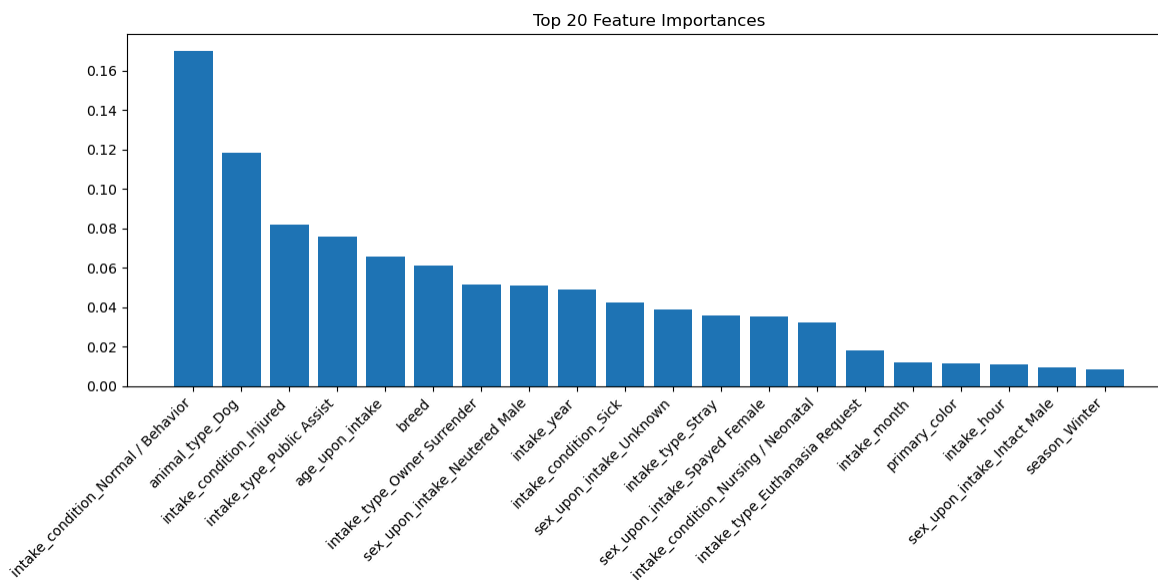


Dropping intake_hour

Observations:

- Breed is already a good determinant, might be improved with clustering

Generalization accuracy: 0.479

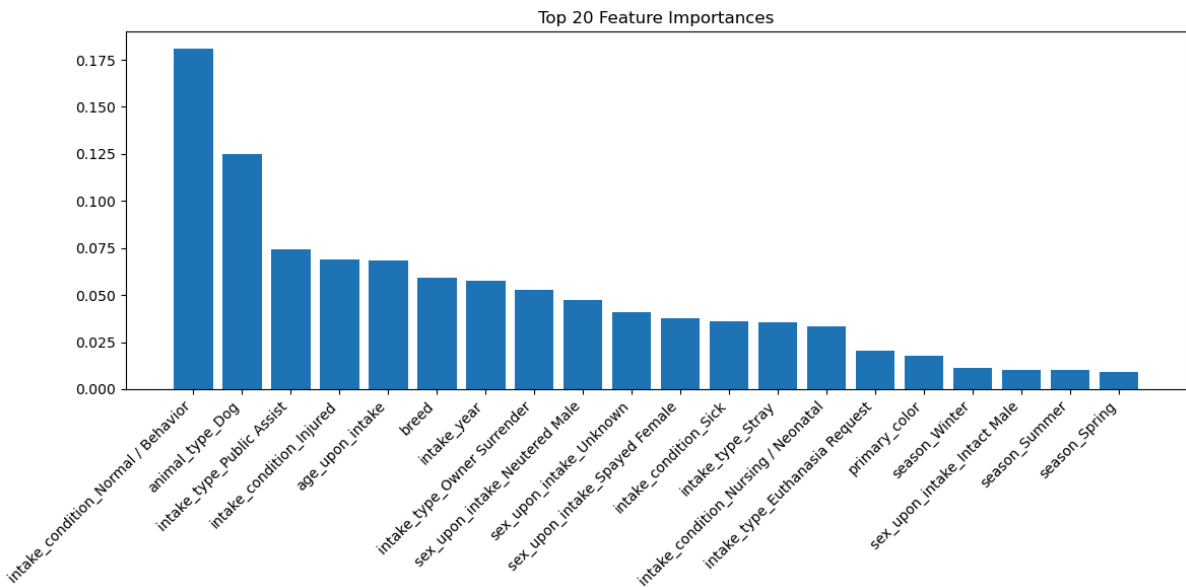


Bucketing seasons, keeping intake_hour, dropped is_mix

Observations

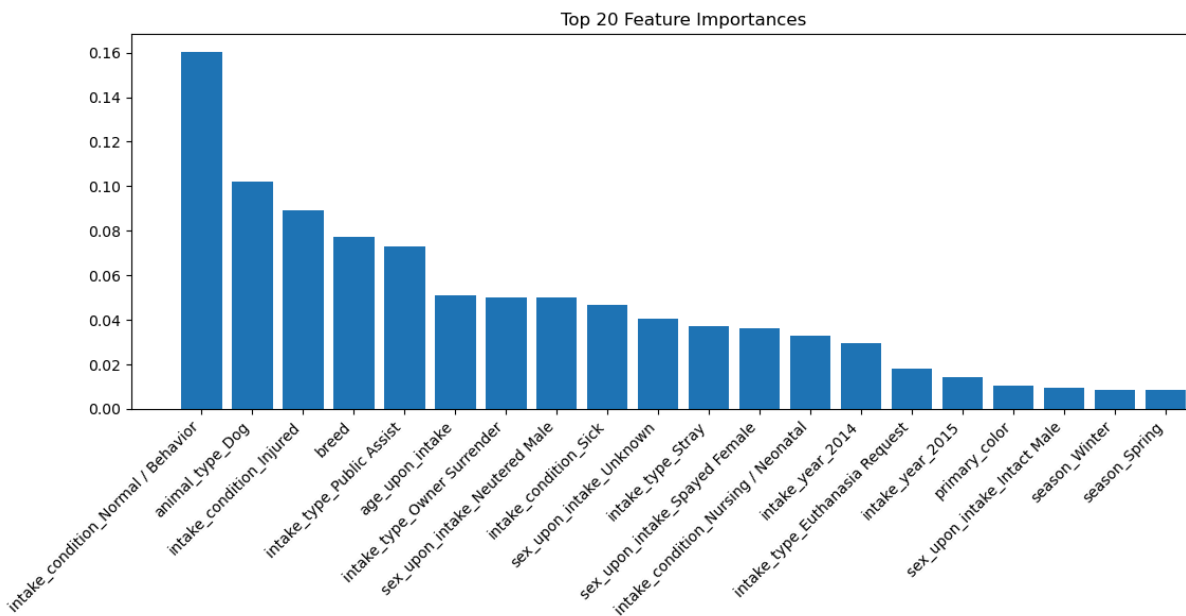
- Season isn't a good determinant
- Dropping is_mix might have improved generalization accuracy?

Generalization accuracy: 0.4847547907094233



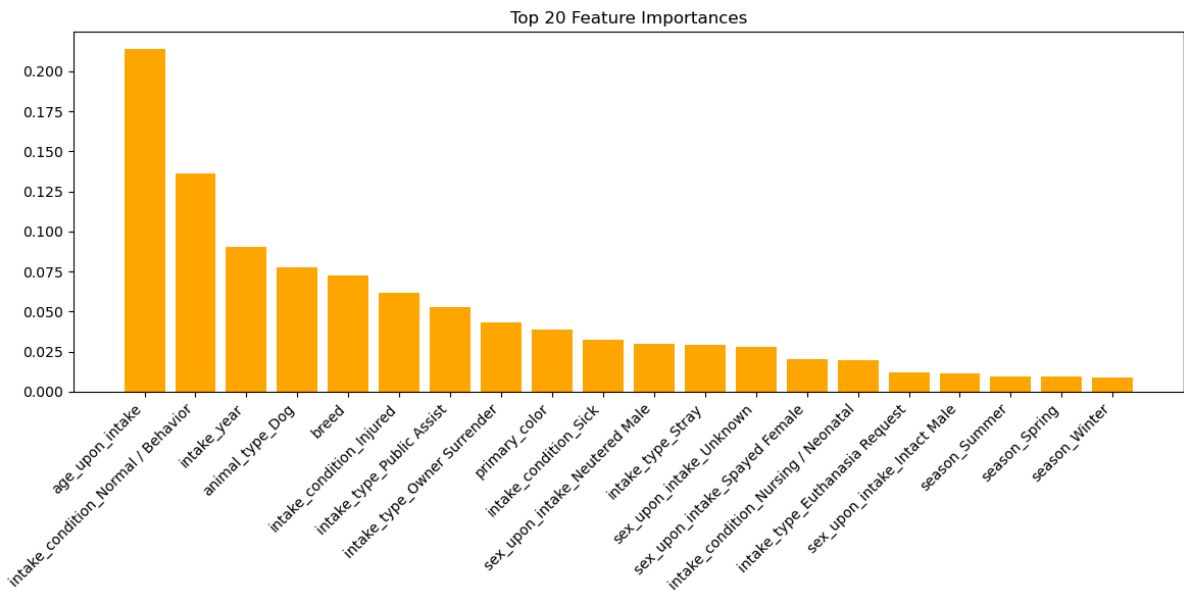
Replace months with seasons instead

0.4851183256631935



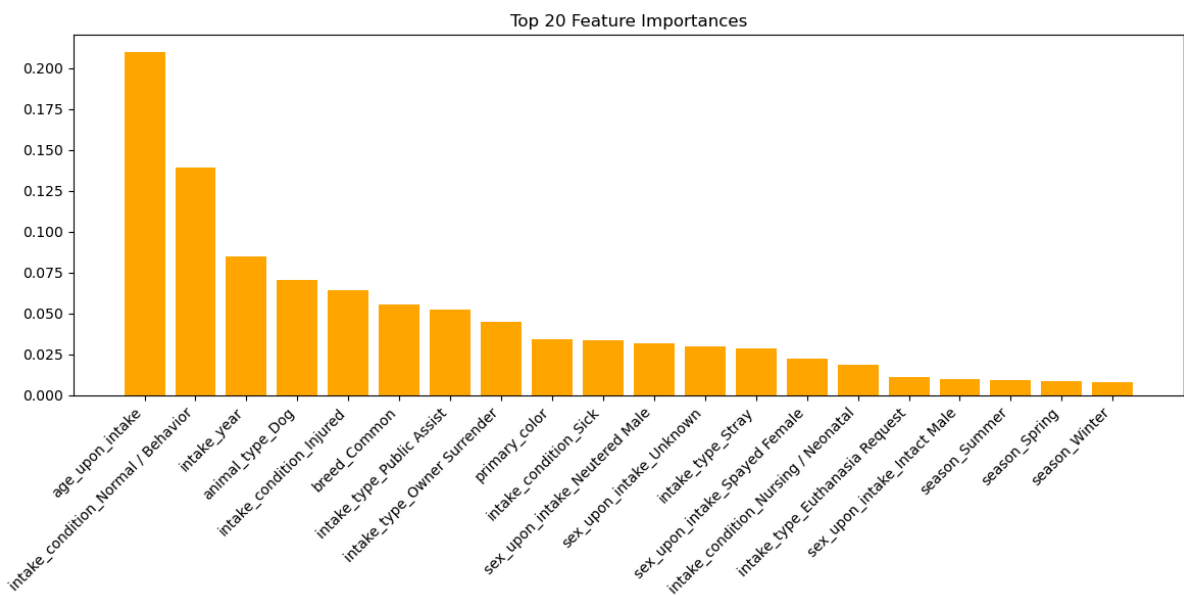
One hot encoded year

0.4792550046078617



Same features as above

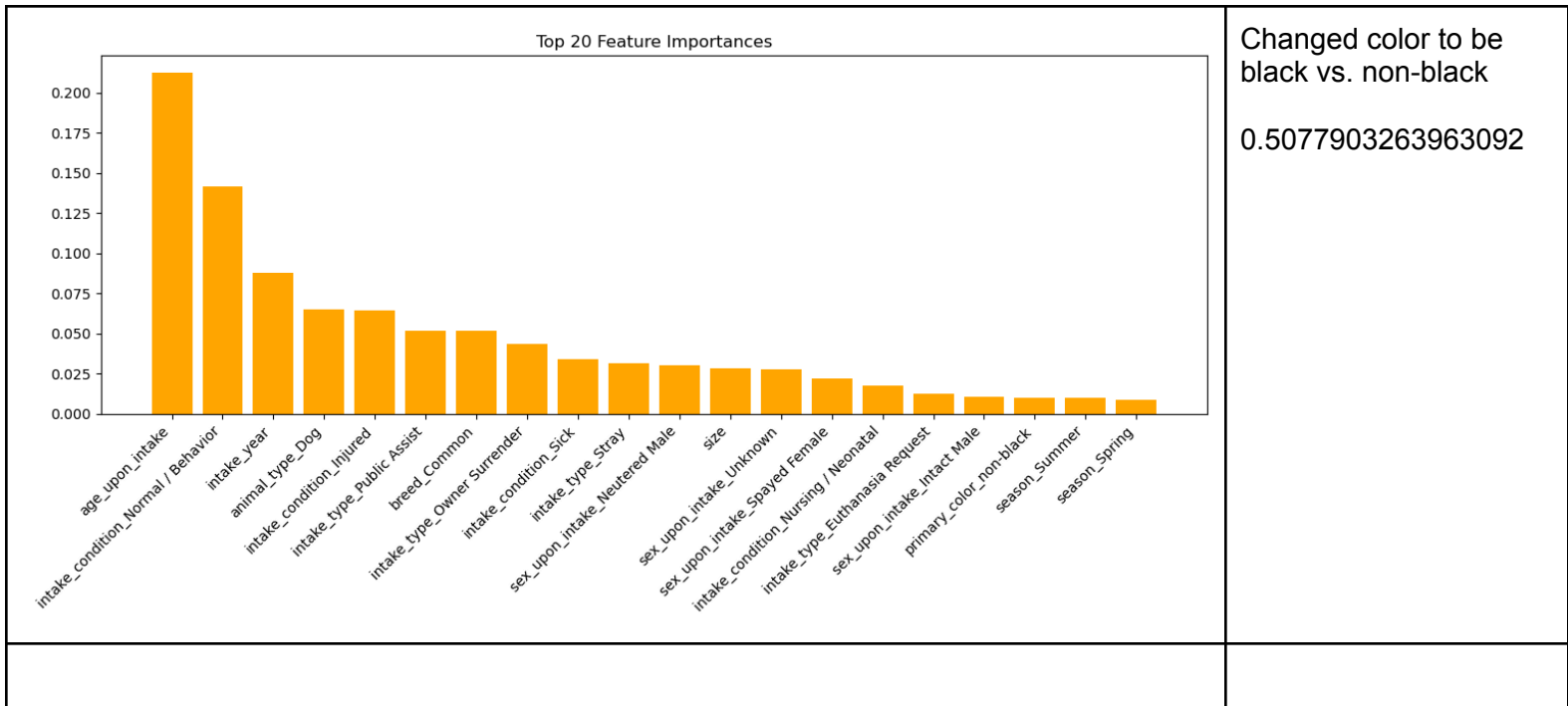
0.5072532290300937



After splitting dog breeds into clusters and cats into “rare” and “common” breeds

- Cat common breed seemed to do the best!
- Dog breeds actually went down

0.5055542690696367



- Nneoma suggested to use PCA for things using trees
 - Nneoma cleaning the dog breeds to match in our data set with the dogbreeds.csv data set
 - In dogbreeds.csv
 - Note that Australian Shepherd and Australian Shepherd Mix are different but do to our data cleaning, the Mixes are not represented tho they should be - 🙄 sorry
 - Because of majority of the data is under 1 yr, we decided to use age in months instead for a feature engineering
 - Rebecca dog clustering
 - Chose to send first match (i.e A in A/B) as a priority when mapping to dogs_info because it's likely that's what the dog looks like more
 - Mapped cats to rare and common to try and reduce dimensionality
 - Unknown dogs were named "Rare"
 - Rebecca ran xg_boost with new breed mappings (but lowkey colors may have been one hot encoded and gotten really high dimensionality) - it did well (51%)
 - Rebecca retry xg_boost after frequency encoding primary colors
 - Nneoma retrying new features with CATEGORICAL Naive Bayes model
 - Binned ages according to distribution
 - After i made the necessary feature changes, i only got 40% accuracy 😞
 - A bit improvement with SMOTE (sampled from catboost)
- 4/14
- Nneoma is fixing Categorical Naive Bayes again 🙄

- Used ordinal encoder + implemented a rare class for each category to handle unknown values
- Rebecca trying ensembling with CatBoost w/ 10 models -> 51% accuracy
- Rebecca wrote code for stacking xg_boost + catboost -> logistic regression
- Rebecca reran catboost with the new breed encodings/clustering -> 51.9%
- Casey ran XGBoost for a long time, getting better results yippee 52%

4/15

- Rebecca adding OVR in *multiple* random forests
 - There was data leakage I think? Printed accuracies were way too optimistic
- Nneoma created sizes by joining with dogs_info - things that are missing we filled with "Medium" (size = 2)
 - Explored the data from the dogbreeds.csv
 - Found a cat breeds.csv
- Nneoma is trying out logistic Regression just bc
 - 47% not the worst
 - SMOTE flopped
 - Trying to adjust threshold
- Casey trying to run stacking

4/16

- Nneoma trying out bagging with logistic regression while building voting classifier
 - Testing it with fast classifiers: log reg, extra random trees, random forest, XGBoost
 - **IDEA:** group primary color into more color groups (black, yellow, blue, etc)
- Nneoma trying to tune threshold for MANUAL VOTING algo
- Trying to see if k-means clustering can give us any info
- Rebecca tried reclassifying primary color
- Nneoma is trying log_age
 - SLOWING down log reg and not really improving stuff
- Trying out balanced random forest
- ~~- Rebecca HDBSCAN: we thought this could work because maybe died records were very similar to each other, but it took 15 min to run and gave .28 accuracy~~
- Rebecca doing OVR with XGBoost as the base classifier - SO overfitted
- Nneoma tried seeing if other dog_breed traits were good
- Casey running XGBoost and stacking, seeing if new feature engineering would improve
- Casey Short stint with LightGBM, but not enough time to train for long enough to get better results imo
- Casey testing out an "in_austin" feature

PREPROCESSING NEEDS

	Naive Bayes	SVC	XG Boost	Cat Boost	Extra Random Trees	Logistic Regression
Needs Scaling	No	Yes		No	No	
Handles numerical data	No	Yes	Yes	Yes	Yes	
Subject to curse of dimensionality	No		Yes	Yes	Yes	