For attempts that we submitted through Kaggle, please write down the commit ID / link AND

1. The time the code was submitted
2. The accuracy score
3. The rank we got at the leaderboard
4. Notes about how this model is different from other models submitted

Commit ID / link	Date, Time submitted	Accuracy score	Ranking	Notes
75cc470e6b5c6b7c21e1e67cd6685ac84ec6f86f	4/8 7:53p	0.26686	16/16	From when we were doing just decision trees
	4/8 8p	0.31225	17/17	Casey using XGBoost on his local machine (downsampled)
d2fa0d2bf58781ef1f9d4ec7f14fc06558397a4a	4/8 11:48p	0.37876	16/19	XGBoost, no downsampling, entire dataset
3978eaa80d6f11c4c87ee94d6ac4fc6bf181e546	4/9 7:08p	0.47192	10/24	Naive Bayes multinominal, SMOTE, entire dataset, 20 seconds
dd4408669c0a42044b8bd1f95e951b4766ae4c06	4/9 9:04a	0.4502	12/25	CatBoost with SMOTE
fe93aebb6f05080ae118fcc3c178c030bd2f7f07	4/11 12p	0.51301	1/30	Catboost with NO SMOTE, stratified k-folding, and class weighting
c1d0fb0d7a86694ff3bd91a8eb07a38796b43512	4/12 1:25a	0.47973	14/33	Extra random trees, class weighting and stratified k-folding (so fast, took like 1 min to run)
ecfbc77db07e84f2a1f1db20433a11ebd673ffde	4/12 5:48p	0.51973	2/38	XGBoost with no SMOTE, fixed one hot encoding (previously it was one hot encoding numerical values as well), added class weighting to fit (weight rarer classes higher)
6044d7ee873af5b8bb613c10886ec560fe6fcf03	4/13 1:41p	0.50608	9/42	Random forest with seasons replacing months

cd712123adc89cdd94cb72cc37110e65be7c69a8	4/14 12:11a	0.51286	8/51	Xg_boost with dogs as clusters and cats as “common”, “rare” - may have one hot encoded color though
	4/14 9:12a	0.50721		Same as above, but with frequency encoded color (250 fits)
	4/14 9:21a	0.52388	2/54	XGBoost with new clustered breed, and used uniform/randint distribution to select variables
2e736e9105a4dc827a2f67eb9a7e0b882438ff3f	4/15 3p	0.40496		Multiclass OVR with Random Forests, fixed outputs - outputted accuracies were too optimistic

1. Attempt 1: We were going to start with Decision Trees → Switched over to XGBoost because 1) it was much faster and ran within 15 minutes whereas DTs never ran after hours 2) it was more optimized and finds more interesting splits by building up
    a. We didn’t really know which features were the most critical overall, so we felt that DTs were good on helping us learn more about the strengths of different features