In our work we first downloaded the dataset, got rid of NaN variables, and made sure that all the features are numeric. Then, we visualized some of the features (i.e., gender, type, breed etc.) to show their min, max, mean, std and other properties. Afterwards, we counted how many empty values we have in each column (feature). We assumed that if there is a feature that is mostly empty, it can disturb the prediction. So, we got rid of ‘Name’, ‘Description’, ‘RescuerID’, ‘PetID’.

For better visualization, we used seaborn to generate countplots for each feature in the dataset against the 'AdoptionSpeed', showing the count of each category in the features with respect to different adoption speeds.

In our opinion, the best way to understand the dataset is through correlation matrix. Unfortunately, the matrix was huge, and it was hard to analyze it, so we decided to use the functions that can choose the best features by themselves. To do this, we normalized the data (since higher values can give extra weight).

For comparison, we firstly ran selected model (Random Forest, KNN and Gradient Boosting) on a raw data. So, in the future we will know whether the techniques that we chose improved the prediction.

For feature selection we used RFECV with RepeatedFold with 5 splits, repeated 3 times. We received a new data frame that we used in selected models. Alas, it hasn't yielded significant improvement. With numerous columns, it's worth exploring more sophisticated algorithms. Feature selection is attempted only on KNN since random forest and XGB lack scoring attributes, making feature selection without additional information impossible. Advanced correlation-based feature selection might be a better option.

We decided to analyze the correlation matrix thoroughly to get rid of the features with low correlation with ‘AdoptionSpeed’. We created a new data frame with features that have correlation above the threshold of 0.02. Then we used this dataframe in our models and concluded that XGB still performed the best but still the score was less than when we used raw data.

Afterwards, we check which pair of features have the highest correlation. We created a list of tuples representing these high correlation pairs and then new columns in data frame by multiplying and combining existing columns based on specific relationships. Of course, we dropped the original columns used to create the new ones to avoid redundancy in the data. We ran our models again but still got approximately the same results as using raw data.

Next, we decided to combine some features (‘Dewormed’, ‘Vaccinated’, ‘Sterilized’) because they have high correlation to each-other, meaning that they can be considered one of the same feature. Random forest gave the best results, but it still was not higher than raw data.

Later, we performed additional feature engineering by binning the counts of occurrences of 'RescuerID' into quantile bins and adding a new feature 'RescuerID\_Bin' to data frame. Then, we used it in a machine learning pipeline with XGBoost for classification. We received a much better score, this seems to be due to the fact that the rescuer itself did have correlation to if a pet would be adopted, as some pets arrived from big organizations, which might’ve used additional strategies to try and get these pets adopted.

We also created a new feature by multiplying ‘Dewormed’, ‘Vaccinated’, and ‘Sterilized’ and ran models again. Random Forest and XGB performed didn’t improve the results, although in comparison with raw data we received a great improvement.

Looking for other ways to improve the prediction, we decided to use grid search to find the best set of hyper-parameters. Hyper parameter search for Random Forest didn’t give very promising results, probably because grid search function uses a different scoring system than we did, and so it gave different parameters more value.

Then we used PCA. We experimented with three different PCA configurations: with 5, 10, and 15 components. Surprisingly, all these configurations resulted in worse predictions, suggesting that extracting principal components did not capture meaningful information from the dataset. This aligns with our previous findings during feature selection attempts. Because we were pressed for time, we opted to apply PCA with 15 components solely to Random Forest. Our assumption was that its performance wouldn't exceed the results obtained with XGBoost using PCA 15, which delivered the most favorable scores among the PCA setups we tested.

To sum it all, the best result in raw data was 0.4 and the best result we received after modifications was 0.46 using Random Forest. The difference might seem insignificant, but if you look closely at the competition, you will find that the highest score teams received was 0.45 which is worse than our result. There are many reasons why overall scores are so low for everyone. In our humble opinion, the correlation between the features is extremely low. This is why feature reduction did not help to improve the prediction.

The research that inspired us utilized regression, but we were unable to locate a suitable dataset for regression analysis. In the original study, the error averaged 1.15 weeks, meaning the predictions were typically off by this amount. This aligns well with our dataset, where an error of this magnitude could significantly impact categorization, particularly for adoption periods of 1-7 days and 8-30 days. Despite this, both we and the competition organizers believe that categorizing pets yields more practical benefits than regression. Our results, as per the competition, have been deemed quite satisfactory..