Capstone 1: In-Depth Analysis

Additional Project Information: A Home for All: Github Repository

LEARNING OBJECTIVES

1. Practice identifying which supervised and unsupervised learning techniques are best suited for your Capstone Project data.
2. Utilize supervised and unsupervised learning techniques to build predictive models.

DATASET : https://www.kaggle.com/c/petfinder-adoption-prediction

APPROACH

The intent of this project is to utilize machine learning to try and predict the adoption speed of different animals given the standard data provided via the PetFinderAPI. Specifically, the goal is to develop a model that estimates adoption speed at an accuracy at or near the top results in the associated Kaggle Competition, which are around approximately 0.46. Additional information and exploration into the data can be found on the associated Github repository.

PROCESS AND UNDERSTANDING:

It was anticipated prior to beginning analysis that the model would have difficulty predicting same day adoptions as this is generally difficult to predict in a real life setting. The majority of the data chosen was categorical or non-continuous data, so that limited the model choices somewhat.

The first step involved converting some of the features to ordinal classes, to allow for application of specific models and eliminate difficulty of analyzing datasets with mixed data. The two variables specifically targeted were Name and Rescuer ID. Name was converted into a binary presence/absence, and Resucer ID was converted into randomly unique integers by ID. Then all data that was not going to be used was dropped, which included that data that was either converted or would be non-useful in analysis. This included the features Name (converted), Rescuer ID (converted), the index column, randomly assigned Pet ID, and the description, which was not used in this analysis.

Then the data was split using training, testing, and validation sets. The first split was done to segregate 10% of the data for later model validation. The remaining data was split into 75% training, 25% test sets, and stratified using the Adoption Speed feature. Stratification was chosen to handle the amount of imbalance between categories.

Logistic Regression

While the model showed some linear behavior it wasn't definitively linear. Since the statistical analysis did show some linearity between certain variables, it was decided that trying a linear model on the data set would be worth trying.

Analysis during development showed that a model using only those pets who were not adopted on the same day as listing and access to all features performed best. However, when testing the training and validation data, it was found that the best model was to use all the features with a liblinear solver, and to keep the "same day adoptions". There is likely a large component of overfitting here, based on the behavior seen. However, the increased accuracy could also be due to simply having access to more data. Final accuracy score for all data was 0.39483.

Naive Bayes

The next model utilized was Naive Bayes. Multinomial Naive Bayes waas used because the bulk of the data is categorical, and it tends to not be normally distributed. Gaussian was considered but because of the mix of variables, it was decided that the best choice would be multinomial. The benefit of using Naive Bayes is that it assumes the independence of each feature in the data set, and will hopefully find any instances where a particular feature has more weight than the others. The accuracy of the Naive Bayes model was considerably less than that of the logistic regression model, with a score of 0.23333. However, when removing the “same day” adopted pets, the score increased dramatically to 0.35120. Pursuing hyperparameters again decreased the score, so it was decided that Naive Bayes was probably not going to be the best model for this dataset. A test on the entire data set, including validation data, yielded a score of 0.31242. Since we had good success in predicting across all classes for adoption speed, it would be good to further pursue classification algorithms.

Support Vector Machines (SVM)

Since Naive Bayes was successful, it would probably be beneficial to investigate further something that utilizes a categorization approach. Enter support vector machines (SVM) Two types of support vector machines were investigated, both Linear and Nonlinear SVC . SVC was used, rather than SVM, because this is a multiclass problem.

Starting with Linear SVC, the default argument for multiclass was used. This model did not yield good results. The classification was restricted entirely to 1-7 day adoptions, and the accuracy was the lowest so far, at 0.17705. When dropping pets adopted the same day, accuracy increased dramatically, but the confusion matrix still showed that the model was not classifying correctly. The best score for Linear SVC was 0.27527. It is hoped that the different forms of classification in nonlinear SVC methods will perform better.

SVC uses a one-vs-one approach, which would perhaps perform better than the one-vs-rest approach of LinearSVC. Since the default kernel here is rbf, it will also likely work better with the multiclass categorization of our dataset. Utilizing a gamma of 'auto' was found to be more efficient, which was expected because it both allows the model to run faster, and because it will work to correctly scale the amount of influence each single training example has. These settings yielded an accuracy score of 0.39344, so hyper parameter tuning was pursued to see if the model could be further improved. Again this model did not do a good job of predicting “same day” adoptions, so it was repeated without those data points and the model score of 0.42099, which is the best model performance so far.

When testing on all the data, including the validation set the scores were 0.40590 when including all data and 0.44668 when removing the same day adoption values. This is interesting, and is probably due to the way that gamma is being utilized to fit the shape of the Gaussian, which may be filtering out noise.

K-Nearest Neighbors (KNN)

kNN was chosen for exploration because it is a method that allows for investigation into both classification and regression problems, so perhaps it will be a good blend of all the previous models. The only concern prior to application is that the data set may be too large to analyze using this method, even though the decisions made through the analysis process allowed for some of the dataset to be eliminated. Analysis found that the brute force method and the kd-tree method with a leaf size of 1 performed the same. Due to the nature of the kd-tree, it was chosen to stick with brute force. Kd-tree is an axis aligned algorithm, which gives it less flexibility when taking shape and is highly feature dependent. This dataset is small enough that brute force can be run without taking too much time. Final score for this model was highest if eliminating the same-day adoption dogs, and was found to be 0.41656.

Random Forest
It is now time to investigate ensemble methods to see how these perform. The first to explore is Random Forest. A classifier was chosen because we're trying to predict a categorical response variable. Random forest performed best so far of the models, and hyperparameter tuning was pursued. When tuning the number of n-estimators, the score for all animals was 0.47109, and for animals without same day adoption speed 0.48557, which is better than the leader on the Kaggle competition!

LightGBM
It is now time to investigate one final model, LightGBM. This is also an ensemble method and tends to perform well on Kaggle. First, the features were plotted to see if they were consistent with those features that showed the highest weight in our statistical analysis. While breed and age were expected to be heavily weighted, the amount of photos was not after statistical analysis. However when using LightGBM to see the importance of each feature, the photo amount did rank higher than breed. Even with tuning hyperparameters, the model performed best using the default arguments. The best score was again found when dropping same day adoptions, and was 0.47679.

FINAL MODEL RESULTS:

A comprehensive model summary can be found in Appendix A.

The best model choice ended up being Random Forest, which was not unexpected. This method along with LightGBM and XGBoost tend to be most successful in Kaggle competitions. Some of these findings were very unexpected, having looked at many of the other Kaggle notebooks submitted during this competition. The difference in findings can easily be attributed to dropping cats out of the dataset. However, I found it interesting that even without analyzing the descriptions, the final score was comparable to that found by the leader in the kaggle competition. Overall, the findings were consistent with what is understood in real-world dog adoptions. It's very difficult to predict which animals will be adopted on the same day, and is more or less random. The likelihood a dog will be adopted probably has less to do with the animal and more to do with pet trends, seasons, and what people are looking for in a dog. Or, perhaps it's like every other relationship we have as humans, and it's all based on that instant connection an adopter has when meeting a pet.

Appendix A

MODEL PERFORMANCE SUMMARY: