Capstone Project Proposal

Kaggle Challenge: PetFinder.my Adoption Prediction

> Mahesh Babu Gorantla Feb 18, 2019

Domain Background

Millions of stray animals suffer on the streets or are euthanized in animal shelters every day around the world. If at all we can find those stray animals a forever home, many precious live can be given a new life that they truly deserve and in turn we will see more happier families.

PetFinder.my has been Malaysia's leading animal welfare platform since 2008, with a database of more than 150,000 animals. PetFinder collaborates closely with animal lovers, media, corporations, and global organizations to improve animal welfare.

Problem Statement

The Pet Adoption Prediction is a multi-class classification problem. In this problem, I need to develop algorithms to predict the adoptability of pets - specifically, how quickly is a pet adopted based on it features (which have been explained in the section below).

It is a classificiation problem because a pet's adoption speed is classified into 5 categories.

A relevant potential solution is to use a few classification algorithms such as

- k-Means Clustering
- DBSCAN
- Random Forests
- XGBoost (Baseline model)

And, choose the one with best `cross_validation` score on the training dataset.

Datasets and Inputs

In this problem, I will predict the speed at which a pet is adopted, based on the pet's listing on PetFinder. Sometimes a profile represents a group of pets. In this case, the speed of adoption is determined by the speed at which all of the pets are adopted. The data included text, tabular, and image data.

File Descriptions

- `train.csv` - Tabular/text data for the training set
- `test.csv` - Tabular/text data for the test set
- `sample_submission.csv` - A sample submission file in the correct format
- `breed_labels.csv` - Contains Type, and BreedName for each BreedID. Type 1 is `dog`, 2 is `cat`.
- `color_labels.csv` - Contains ColorName for each `ColorID`
- `state_labels.csv` - Contains StateName for each `StateID`

Data Fields

- `PetID`- Unique hash ID of pet profile
- `AdoptionSpeed` - Categorical speed of adoption. Lower is faster. `This is the value to predict`. See below section for more info.
- `Type` - Type of animal (1 = Dog, 2 = Cat)
- `Name` - Name of pet (Empty if not named)
- `Age` - Age of pet when listed, in months
- `Breed1` - Primary breed of pet (Refer to BreedLabels dictionary)
- `Breed2` - Secondary breed of pet, if pet is of mixed breed (Refer to BreedLabels dictionary)
- `Gender` - Gender of pet (1 = `Male`, 2 = `Female`, 3 = `Mixed`, if profile represents group of pets)
- `Color1` - Color 1 of pet (Refer to ColorLabels dictionary)
- `Color2` - Color 2 of pet (Refer to ColorLabels dictionary)
- `Color3` - Color 3 of pet (Refer to ColorLabels dictionary)
- `MaturitySize` - Size at maturity (1 = `Small`, 2 = `Medium`, 3 = `Large`, 4 = `Extra Large`, 0 = `Not Specified`)
- `FurLength` - Fur length (1 = `Short`, 2 = `Medium`, 3 = `Long`, 0 = `Not Specified`)
- `Vaccinated` - Pet has been vaccinated (1 = `Yes`, 2 = `No`, 3 = `Not Sure`)
- `Dewormed` - Pet has been dewormed (1 = `Yes`, 2 = `No`, 3 = `Not Sure`)
- `Sterilized` - Pet has been `spayed` / `neutered` (1 = `Yes`, 2 = `No`, 3 = `Not Sure`)
- `Health` - Health Condition (1 = `Healthy`, 2 = `Minor Injury`, 3 = `Serious Injury`, 0 = `Not Specified`)

- `Quantity` - Number of pets represented in profile Fee - Adoption fee (0 = `Free`)
- `State` - State location in Malaysia (Refer to StateLabels dictionary)
- `RescuerID` - Unique hash ID of rescuer
- `VideoAmt` - Total uploaded videos for this pet
- `PhotoAmt` - Total uploaded photos for this pet
- `Description` - Profile write-up for this pet. The primary language used is English, with some in Malay or Chinese.

AdoptionSpeed

Contestants are required to predict this value. The value is determined by how quickly, if at all, a pet is adopted. The values are determined in the following way:

- 0 - Pet was adopted on the same day as it was listed.
- 1 - Pet was adopted between 1 and 7 days (1st week) after being listed.
- 2 - Pet was adopted between 8 and 30 days (1st month) after being listed.
- 3 - Pet was adopted between 31 and 90 days (2nd & 3rd month) after being listed.
- 4 - No adoption after 100 days of being listed. (There are no pets in this dataset that waited between 90 and 100 days)

Solution Statement

First, we will import the training dataset `/data/train/train.csv` and then drop irrelevant columns (that do not help our classification algorithm). Next, find the missing rows in each column and also generate descriptive statistics to understand the distribution of data. Usually on close analysis of descriptive statistics we will often find outliers/abnormal values in the `min`/`max` rows for each column.

Create custom data imputers to fill in the missing values for both quantitative (`mean`/`median` value for the column) and categorical columns (`mode` value for the column). Create a pipeline to fit and transform the data using the custom data imputers. After imputing the missing values, plot histograms per pet category (1 - Dogs and 2 - Cats) to further examine the distributions per column, determine any anomalies found and correct those anomalies with appropriate data correction techniques. And, plot the column correlation heatmap to find any closely related columns (if any).

Once all the processes are implemented as mentioned above, the data at this point can now safely be assumed with no missing or anomalous values. Now, we need to dummify any categorical columns and scale each column either using `StandardScaler` or `MinMaxScaler` (I chose to use `MinMaxScaler` for now).

Now, lets split the cleaned and scaled data into training features and prediction variable (i.e. `AdoptionSpeed`).

The above mentioned problem is a supervised multi-class classification problem. To predict `AdoptionSpeed` based on the training features of a pet, I would like to use classification algorithms like

1. Classification and Regression Trees (CART)

2. kNN Classifier
3. Support Vector Machines (SVM)
4. Random Forests
5. XGBoost

to predict the adoptibility of a pet.

Find the model with best parameters using `GridSearchCV` and save the model. Load the model weights, preprocess the test data and obtain the performance of the model on the testing dataset at `data/test/test.csv`.

Benchmark Model

As we know that the above mentioned prediction task is a supervised classification problem, we should use tree based classification models which typically outperform other classification models. Hence, I would like to use `kNN Classfier` and `XGBoost` algorithms as a benchmark and beat the benchmark perfomance both in terms of time to fit the model as well as increase in prediction accuracy.

Currently, the baseline prediction accuracy (without feature transformation) is as follows

Classification Model	Model Fit Time (in mins)	Cross Validation Accuracy
KNeighborsClassifier	1.01	31.17%
XGBoost	0.99	38.01%

From the above results, we can infer that `XGBoost` should outperform all the above mentioned tree-based classification algoritms because we are able to get higher prediction accuracy given the higher dimensional space in lesser time than `kNN`.

Check Benchmark Model Notebook for information on feature engineering techniques used to impute the missing data in the given dataset.

To improve the performance of the model over the benchmark model, we will need to find more anomalies in the data (if there are any and correct them). Reduce the dimensionality of the dataset using **`LDA`** - **`Linear Discriminant Analysis`** techique and test for the performance on the `train` and `test` datasets.

Evaluation Metrics

I would like to propose an evaluation metric called `Cross Validation Score`. The simplest way to use cross-validation is to call the `cross_val_score` helper function on the estimator and the dataset.

Given, the problem is to develop a supervised multi-class classification model, the `cross_val_score` method uses `StratifiedKFold` cross-validation technique. `StratifiedKFold` cross-validation method is a variation of `KFold` that returns startified folds. The folds are made by preserving the percentage of samples for each class.

The model whose cross validation score is greater than the benchmark model will be my solution model.

Project Design

Theoretical Workflow for designing the Solution Model

- **Import the Training Dataset**
    - Load the necessary libraries such as `numpy`, `pandas`
    - Load the `data/train/train.csv` dataset
    - Drop irrelevant columns
- **Data Exploration**
    - Find missing rows in the each column using `df.isnull().sum()`
    - Obtain and Understand Descriptive Statistics
- **Feature Engineering**
    - Decipher more missing values from descriptive statistics (if any)
    - Impute missing values using Custom Quantitative/Categorical Imputers
    - Visualize per column distribution using histograms
    - Find any anomalous data distributions from the plots
    - Dummify Categorical Columns
    - Scale all the columns either by using `z-scaling` (`StandardScaler`) or `min-max scaling` (`MinMaxScaler`) technique
- **Evaluate Classification Models**
    - Choose a few classification models
    - Split the dataset into train/split test using `StratifiedKFold` cross-validation technique
    - Score each classification model using `cross_val_score`
    - Choose the classification model with best cross validation score
- **Feature Transformation**
    - Reduce the dimensionality of features
        - Using LDA - Linear Discriminant Analysis (Best for multi-class classification)
- **Tune the Classification Model**
    - Use `GridSearchCV` to improve the accuracy
- **Save the best model**
    - with best prediction time and accuracy
- **Final Inference and Conclusion**

> Please check the Notebook to understand what I meant by Feature Engineering