# ARCHITECTURAL DECISIONS DOCUMENT

# Pet Adoption-Speed Predictor

Abhilash Neog

## Architectural Components Overview

1. Dataset (use-case)
2. ETL (Extract Transform Load)
3. Feature Creation
    - Data Quality Assessment
    - Feature Engineering
    - Feature Engineering - Image Data
4. Model Definition
5. Model Training
6. Model Evaluation
7. Model Deployment

## 1. DATA SET AND USE CASE

### Data set choice and use case

The dataset used is a database of pet dogs and cats advertised for adoption available on Kaggle. The objective/use-case of the project is to predict the speed of adoption, i.e. given an advertisement of a pet, within how many days does it get adopted (5 targets - 0,1,2,3,4 days)

### Justification

This dataset was chosen because it was an interesting and challenging dataset, given, it contains both **structured** and **unstructured** data, i.e. tabular/relational data + image data.

## 2. ETL (EXTRACT TRANSFORM LOAD)

### Technology Choice

To work with the data, **numpy arrays** and **pandas dataframes** were used.

### Justification

The relational data is present in a csv file and so using a pandas dataframe to load the relational data was more suitable. The image data is basically a list of jpeg images which are loaded into numpy arrays in batches during training. The ETL process was not explicitly implemented as the input data was already in a suitable format (so no such transformation required), plus the data was easily callable (or extractable) and so didn't require storing in object storage (or any warehouse).

# 3. FEATURE CREATION

## 3.1 DATA QUALITY ASSESSMENT

### Technology Choice/ Specific methods used

The specific methods implemented for data quality assessment were,

1. Check for **missing** values and **duplicates**
2. **DataType** check
3. Set and Foreign Key **membership**

### Justification

The above data quality assessment steps are crucial in determining the quality of data, and helps to identify the required procedures for data cleansing/feature engineering.

1. The missing value check was done, as missing value would cause problems in model training. Plus one single missing value may change the datatype of a dataframe column. As such it would hamper certain feature engineering tasks. The missing values are shown as Nan, which were replaced as a separate category of values (a new category was created for such values).
   Duplicate check was done mainly on the primary key (PetID), as other columns can have duplicate values. It was mainly done so as to maintain data integrity. Rows having the same unique id value would only indicate the ML/DL model that, the same image of a pet can have different attribute values and different target labels- which is definitely wrong.
2. Data type check ensures that the content matches the data type of the column. Say, an age column having a string type (possible if age numbers were added as string during dataset creation, or if missing value present), it might get left out (say, for eg.) during some feature engineering tasks involving only int type columns. Or might get called for one-hot encoding of attributes along with the categorical columns.
3. To check if only allowed values are present in the categorical columns (in the same relation, or in a referenced table). This helps to remove the noise present in the dataset.

## 3.2 FEATURE ENGINEERING

### Technology Choice/ Specific methods used

The specific feature engineering tasks implemented were,

1. Range of values/**Distribution** of data in columns (for feature selection)
2. Label **encoding** categorical values, One hot encoding target labels
3. Feature **selection** (using correlation)

### Justification

The above-mentioned feature engineering tasks were important for transforming the data into a suitable model input format, and also to enhance the model training. The justification for the mentioned tasks are,

1. A check on the Dataframe columns - calculating range, min, max, mean. The statistical measures help to understand the distribution of data in a particular column. A column with a max value of 3000, min value of 0, having an average of just 21, speaks of the skewness of the data. In many cases, such columns are filled with 0s, with a few large values here and there that increases the average. Such columns won't contribute much to the training, and hence were safely ignored.
2. Most of the categorical variables contain string values which cannot be fed into the training algorithm directly, thus, label encoding was done. The target labels are converted into one-hot encoded labels, so as to facilitate the training of neural network (makes it easy to use a softmax classifier and learn the labels)
3. Correlation matrices provide a clear picture of the relationship among the columns. 2 highly correlated columns having similar correlation with the target can be combined into 1, thus reducing the training data dimension. Plus, there were certain attributes having ~0 correlation to the target label, and hence was dropped, further reducing the dimension of data.

## 3.3 FEATURE ENGINEERING - IMAGE DATA

### Technology choice/specific methods used

The following tasks were done on image data before feeding into the model,

1. Image Resizing and **Normalization**

### Justification

The images were fed into a pretrained network (DenseNet169), to obtain a 1-D feature representation of the image, which can then be fed into a different model. In order to feed into the pretrained network, images were resized.

The pretrained network was trained on the imagenet dataset. The imagenet images are first normalized using mean and std dev. per channel, before getting fed into the network. Hence, Pet images were also normalized (z-score normalization) using the standard imagenet mean and std dev. before feeding into the network (so that the pretrained network can generate accurate features/ or recognize the images easily).

## 4. MODEL DEFINITION

### Technology choice/Specific Algorithm used

The architecture consists of 2 Neural Network Models + 1 pretrained network. The first neural network is trained on tabular data/**categorical** data (without the images). To train the 2nd neural network (the primary network that takes both image (or **continuous** data) and relational/categorical data), tabular data is passed through the trained 1st neural network, and a 1-D feature representation is obtained from a certain layer. Similarly, the image data (for training) is passed through the pretrained network to obtain a 1-D feature vector. Both the feature vectors are then concatenated to form a new 1-D **feature vector**, which is then passed onto the 2nd NN as input. The output generated by this NN is the predicted output of this model.

Another **technique** was implemented by using Machine Learning models in conjunction with the NN. A ML model (like a DT) was trained on the tabular data, and the output of this classifier was combined with the proposed model output.

### Justification

The dataset consisted of both tabular data and image data. In order to train a model that would take both types of data as input, this particular algorithm was designed. A neural network was used to train on the tabular data, and not a ML model like DT or SVM, since a feature representation of the tabular data was required to combine with the image data. For obtaining feature vector representation of images, it is ideal to use a pretrained network (on imagenet) having high accuracy. Combining the 2 different feature vectors and feeding into a network would make the network learn both types of information at the same time.

A ML model could also be trained on the concatenated feature vector, but was not done so, since the number of images (> 150k) were large and as such had to be loaded in batches (not suitable for training a SVM, kNN, etc.)

## 5. MODEL TRAINING

### Technology choice

The project has been written in python3. For model training, **Keras**, a high-level deep-learning library over tensorflow was used. The kaggle notebook platform was used to train the model.

### Justification

The primary reason for choosing Keras, and not any other f/w is my proficiency/confidence in developing and training models in Keras compared to another f/w. Keras, being a high-level library provides easy development of deep learning models. And with the incorporation of this library into the tensorflow f/w, makes it easy to use different tf functionalities with the keras code without any compatibility issues.

The model was trained on the kaggle platform, because the dataset used was a kaggle dataset, and as such was easier to import into the working notebook. More importantly, the platform provides free GPU hours (30 a week) which was helpful in the model training.

## 6. MODEL EVALUATION

### Technology choice/Performance Indicators used

The model performance indicators used are **precision**, **recall**, **F1-score** and the **confusion matrix**. Scikit-learn library was used to evaluate these metric values.

### Justification

The above-mentioned metrics were used to measure the performance of the model, as it was a **classification** problem. F1-score is the harmonic mean of precision and recall, and so is a better indicator compared to only precision or recall. Confusion matrices evaluate the number of TP, FP, TN and FN which provide a raw idea of the classification performance.

## 7. MODEL DEPLOYMENT

### Technology choice

The model is deployed as a **jupyter notebook** containing all the data analysis, model training, evaluation code and outputs

### Justification

The model is deployed as a jupyter notebook, and I believe it to be a better option (specifically in this case/project) compared to deploying through REST APIs or containerizing as a data product, because there a lot of data analysis being done (with visualizations and documentation) along with model training and evaluation. The notebook captures all these in a document-style and provides a better understanding from scratch.