NADA & NOURA

MACHINE LEARNING PROJECT

Table of Contents

List of Figures

1 Introduction:

In this world, many animals are not lucky enough to be adopted by caring families because of their characteristics like breed, color, age etc. In this project, we developed a combination of multi-layer perceptron and convolutional neural network to predict how long it will take animal to be adopted. This project takes the characteristics as well as images of pets for classification task, where characteristics were used by multi-layer perceptron and images were by convolutional neural network and their results were combined to predict the adoption rate. The result of our method are shown in Section 4.

Dataset:

- **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.

Class:

- **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).

Images:
For each pet in .csv file there are one or few images having the same name as PetID in csv.

2 Data preprocessing and visualization:

Before giving an input to machine learning model, it is important to perform certain actions on the data to improve the performance of the model and make the data compatible for the model. The process on the data before giving it to machine learning or deep learning model is known as Data Pre-processing. There are certain steps which were performed in this dataset which includes, one-hot encoding of all the categorical features (Type, Breed1, Breed2, Gender, Color1, Color2, Color3, MaturitySize, FurLength, Vaccinated, Dewormed, Sterilized, Health and State), scaling down the continuous variables (Fee and Age) between [0,1] and converting AdoptionSpeed into one-hot representation to make it compatible with keras model. Images were also resized into 32x32 shape to make all the images of equal shape.

There are few records in .csv file but don't have any image for it so we have ignored these record and the dataset was prepared in such a way that images and features are stored in parallel arrays as in every index of training and testing features (Age, Breed1, Breed2, Color1, etc.) corresponds to the same training and testing pictures.

The major problem that can occur is imbalance class distribution which makes the machine learning model biased to a particular class(es). A dataset is said to be imbalance when there is a huge difference in the number of examples or samples of each class, that lead the machine learning model biased to the class which have higher samples. The question arise why is it so? Let's assume having 1,000 samples for 'class A' which says a patient have a heart disease and 100,000 samples for 'class B' which says a patient do not have a heart disease. The machine learning model gets train on this dataset but the model will understand the features of 'class B' better than the 'class A' because there are more number of samples for it. The model will have a better prediction for 'class B' but what about 'class A'? Assume a scenario when unseen sample comes from the heart patient but the model predicts that the patient does not have a heart disease, the model is going to be lethal in this case. So, to avoid this problem it is always recommended to make these number of samples equal for all classes if it isn't already.

The below figure 2.1 showing the distribution of almost all classes are equal other than class 0 which can affect the end result.

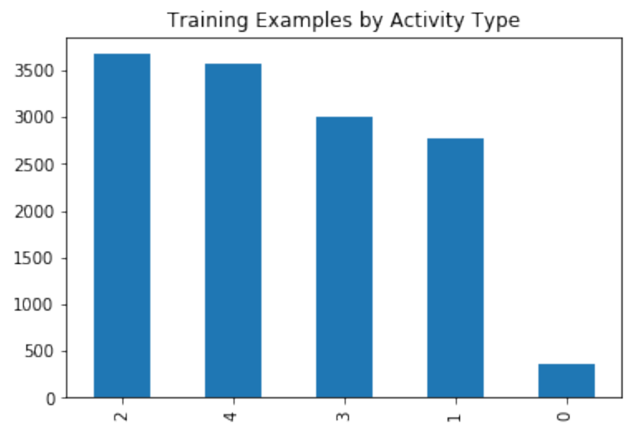
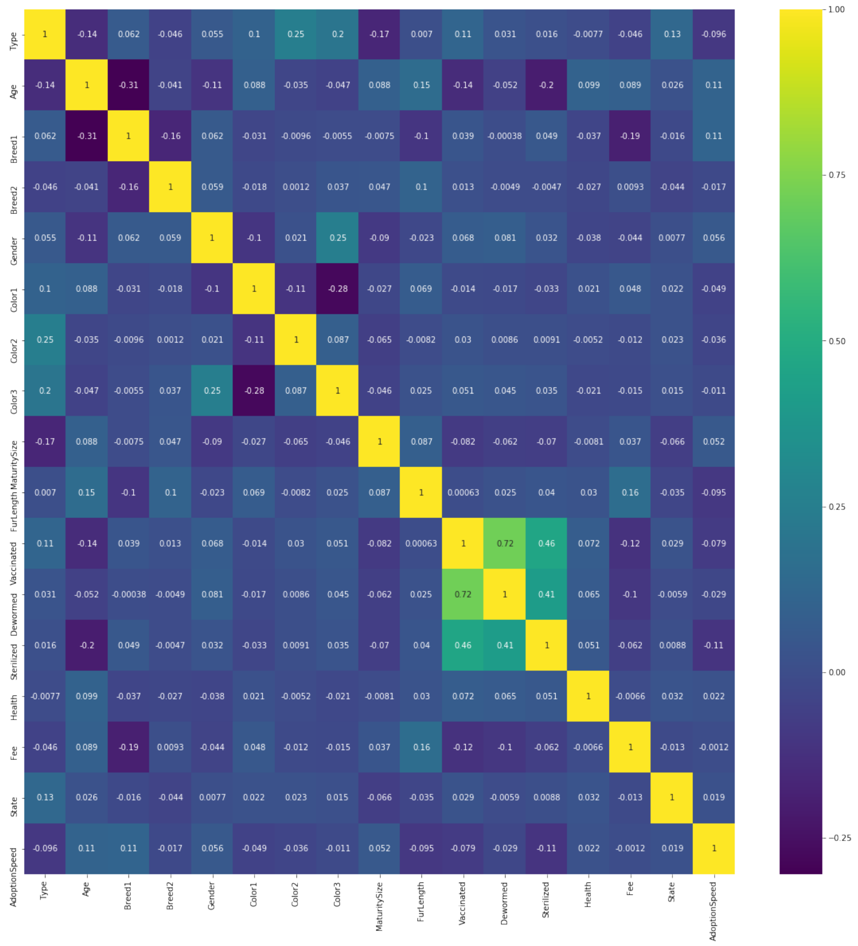


Figure 2.1 Class Distribution

Figure 2.2 is **Correlation Matrix** with Heatmap represent how all the features of a dataset are related to each other and with the target label. The relationship between the features can be positive or negative. Correlation Matrix with Heatmap makes it easy to visualize the best features towards the target label.

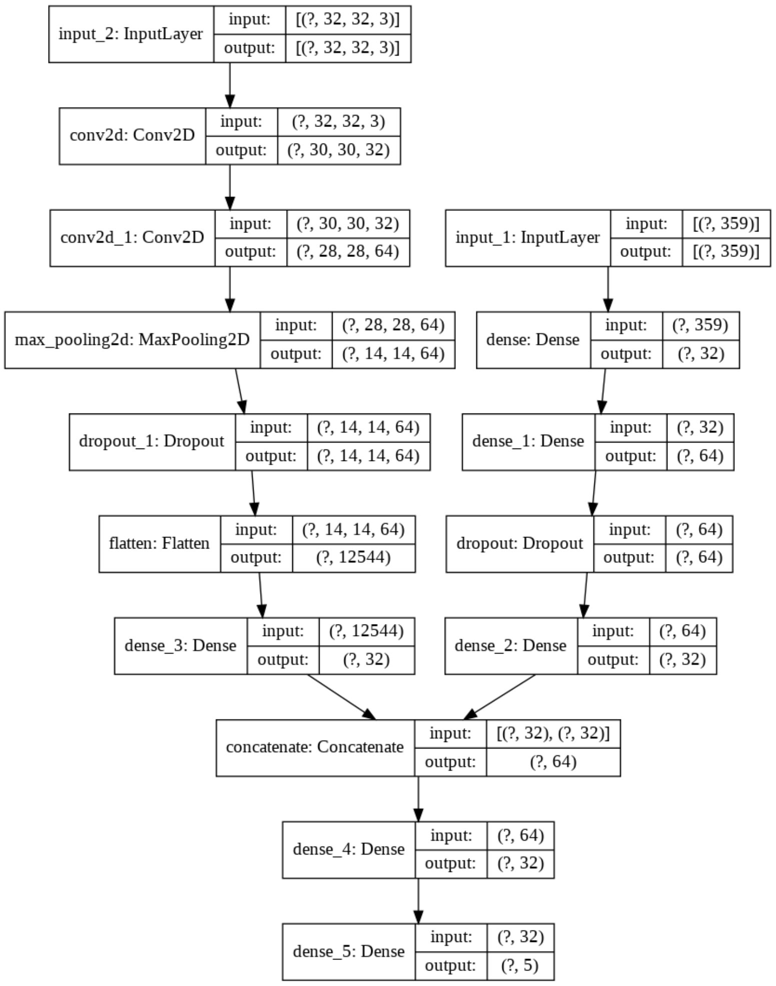


	Type	Age	Breed1	Breed2	Gender	Color1	Color2	Color3	MaturitySize	FurLength	Vaccinated	Dewormed	Sterilized	Health	Fee	State	AdoptionSpeed
Type	1	-0.14	0.062	-0.046	0.055	0.1	0.25	0.2	-0.17	0.007	0.11	0.031	0.016	-0.0077	-0.046	0.13	-0.096
Age	-0.14	1	-0.31	-0.041	-0.11	0.088	-0.035	-0.047	0.088	0.15	-0.14	-0.052	-0.2	0.099	0.089	0.026	0.11
Breed1	0.062	-0.31	1	-0.16	0.062	-0.031	-0.0096	-0.0055	-0.0075	-0.1	0.039	-0.00038	0.049	-0.037	-0.19	-0.016	0.11
Breed2	-0.046	-0.041	-0.16	1	0.059	-0.018	0.0012	0.037	0.047	0.1	0.013	-0.0049	-0.0047	-0.027	0.0093	-0.044	-0.017
Gender	0.055	-0.11	0.062	0.059	1	-0.1	0.021	0.25	-0.09	-0.023	0.068	0.081	0.032	-0.038	-0.044	0.0077	0.056
Color1	0.1	0.088	-0.031	-0.018	-0.1	1	-0.11	-0.28	-0.027	0.069	-0.014	-0.017	-0.033	0.021	0.048	0.022	-0.049
Color2	0.25	-0.035	-0.0096	0.0012	0.021	-0.11	1	0.087	-0.065	-0.0082	0.03	0.0086	0.0091	-0.0052	-0.012	0.023	-0.036
Color3	0.2	-0.047	-0.0055	0.037	0.25	-0.28	0.087	1	-0.046	0.025	0.051	0.045	0.035	-0.021	-0.015	0.015	-0.011
MaturitySize	-0.17	0.088	-0.0075	0.047	-0.09	-0.027	-0.065	-0.046	1	0.087	-0.082	-0.062	-0.07	-0.0081	0.037	-0.066	0.052
FurLength	0.007	0.15	-0.1	0.1	-0.023	0.069	-0.0082	0.025	0.087	1	0.00063	0.025	0.04	0.03	0.16	-0.035	-0.095
Vaccinated	0.11	-0.14	0.039	0.013	0.068	-0.014	0.03	0.051	-0.082	0.00063	1	0.72	0.46	0.072	-0.12	0.029	-0.079
Dewormed	0.031	-0.052	-0.00038	-0.0049	0.081	-0.017	0.0086	0.045	-0.062	0.025	0.72	1	0.41	0.065	-0.1	-0.0059	-0.029
Sterilized	0.016	-0.2	0.049	-0.0047	0.032	-0.033	0.0091	0.035	-0.07	0.04	0.46	0.41	1	0.051	-0.062	0.0088	-0.11
Health	-0.0077	0.099	-0.037	-0.027	-0.038	0.021	-0.0052	-0.021	-0.0081	0.03	0.072	0.065	0.051	1	-0.0066	0.032	0.022
Fee	-0.046	0.089	-0.19	0.0093	-0.044	0.048	-0.012	-0.015	0.037	0.16	-0.12	-0.1	-0.062	-0.0066	1	-0.013	-0.0012
State	0.13	0.026	-0.016	-0.044	0.0077	0.022	0.023	0.015	-0.066	-0.035	0.029	-0.0059	0.0088	0.032	-0.013	1	0.019
AdoptionSpeed	-0.096	0.11	0.11	-0.017	0.056	-0.049	-0.036	-0.011	0.052	-0.095	-0.079	-0.029	-0.11	0.022	-0.0012	0.019	1

Figure 2.2 Correlation

3 Model:

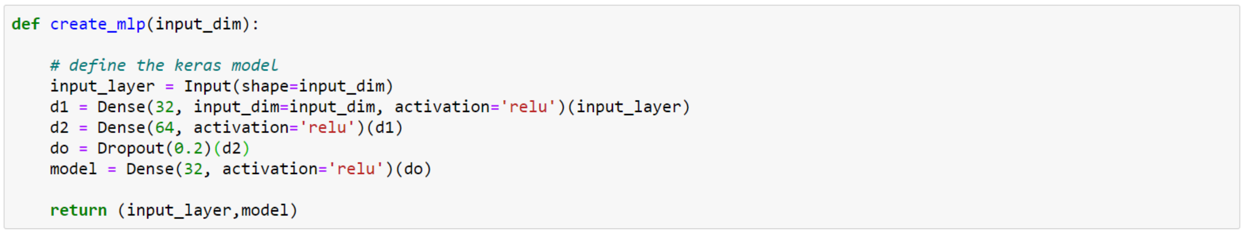
Our model is a combination of multi-layer perceptron and convolutional neural network which predicts the final class label. Below figure 3.1 represents the architecture of our model where the left represents CNN and right side represents MLP, then these two models were combined together using concatenation layer. After concatenation there is fully connected layer which accepts the outputs from CNN and MLP then there is output layer which predicts the final label.



input_2: InputLayer | input: [(?, 32, 32, 3)] | output: [(?, 32, 32, 3)]
conv2d: Conv2D | input: (?, 32, 32, 3) | output: (?, 30, 30, 32)
conv2d_1: Conv2D | input: (?, 30, 30, 32) | output: (?, 28, 28, 64)
input_1: InputLayer | input: [(?, 359)] | output: [(?, 359)]
max_pooling2d: MaxPooling2D | input: (?, 28, 28, 64) | output: (?, 14, 14, 64)
dense: Dense | input: (?, 359) | output: (?, 32)
dropout_1: Dropout | input: (?, 14, 14, 64) | output: (?, 14, 14, 64)
dense_1: Dense | input: (?, 32) | output: (?, 64)
flatten: Flatten | input: (?, 14, 14, 64) | output: (?, 12544)
dropout: Dropout | input: (?, 64) | output: (?, 64)
dense_3: Dense | input: (?, 12544) | output: (?, 32)
dense_2: Dense | input: (?, 64) | output: (?, 32)
concatenate: Concatenate | input: [(?, 32), (?, 32)] | output: (?, 64)
dense_4: Dense | input: (?, 64) | output: (?, 32)
dense_5: Dense | input: (?, 32) | output: (?, 5)

Figure 3.1 Model Architecture

Figure 3.2 shows the **MLP** model where we have two Dense layers followed by a Dropout out of 20% to reduce overfitting then there is another Dense layer with **'relu'** as its activation function.



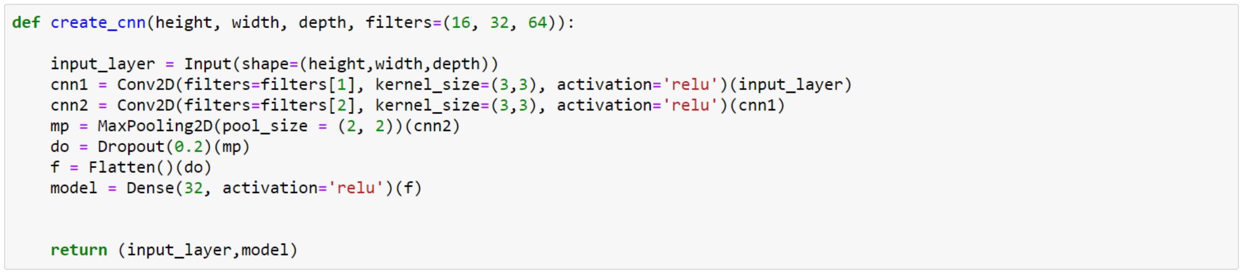
```
def create_mlp(input_dim):

    # define the keras model
    input_layer = Input(shape=input_dim)
    d1 = Dense(32, input_dim=input_dim, activation='relu')(input_layer)
    d2 = Dense(64, activation='relu')(d1)
    do = Dropout(0.2)(d2)
    model = Dense(32, activation='relu')(do)

    return (input_layer,model)
```

Figure 3.2 MLP Model

Below is the figure 3.3 of CNN (Convolutional Neural Network) model which shows that the input is passed to the **two Convolutional Layers** which extract the features and the **Max Pooling** layer was applied to reduce the image size by preserving important features only. Then there is Dropout of 20% to avoid overfitting. After that, all the images are being flattened into a single vector so that model train and then there is **Fully Connected** Layer before concatenating it with MLP.



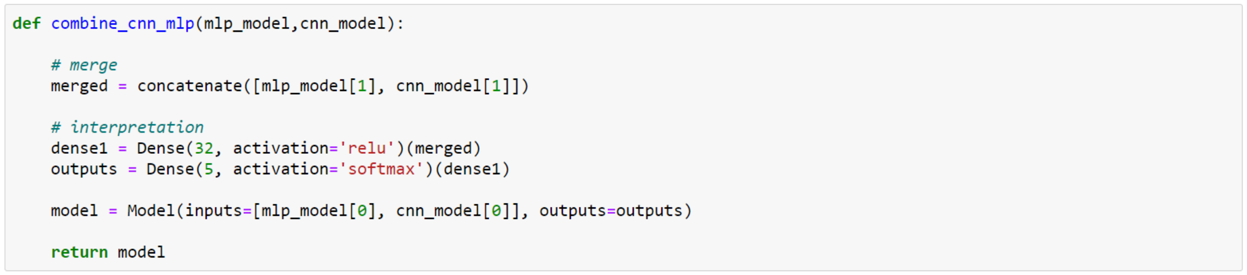
```
def create_cnn(height, width, depth, filters=(16, 32, 64)):

    input_layer = Input(shape=(height,width,depth))
    cnn1 = Conv2D(filters=filters[1], kernel_size=(3,3), activation='relu')(input_layer)
    cnn2 = Conv2D(filters=filters[2], kernel_size=(3,3), activation='relu')(cnn1)
    mp = MaxPooling2D(pool_size = (2, 2))(cnn2)
    do = Dropout(0.2)(mp)
    f = Flatten()(do)
    model = Dense(32, activation='relu')(f)


    return (input_layer,model)
```

Figure 3.3 CNN Model

Below is the figure 3.4 of combining the MLP and CNN using concatenation layer there is fully connected layer followed by an output layer which predicts the final label.



```
def combine_cnn_mlp(mlp_model,cnn_model):

    # merge
    merged = concatenate([mlp_model[1], cnn_model[1]])

    # interpretation
    dense1 = Dense(32, activation='relu')(merged)
    outputs = Dense(5, activation='softmax')(dense1)

    model = Model(inputs=[mlp_model[0], cnn_model[0]], outputs=outputs)

    return model
```

Figure 3.4 Combining MLP and CNN

Rectifier Linear Unit was selected as activation function in CNN because to increase the non-linearity in our images. For example, the image on the left side is original image and the image on the right side is after applying **Rectifier Linear Unit** which contains only non-negative pixels.

Softmax was used in the output layer because this is multi-class problem. **Softmax** assigns the probability to each class.

4 Result and discussion:

By comparing the actual and predicted labels and getting few results to check the performance of the model which are Accuracy, Precision, Recall, F1-score and Confusion Matrix. Let us discuss these results one by one.

While training the model 20% of training data was selected for validating the model performance and the graph of training and validation accuracy is shown in figure 4.1. The figure shows that the model started over-fitting after from 2nd epoch.

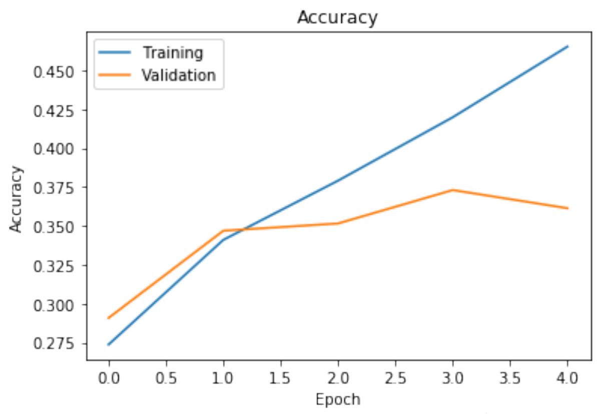


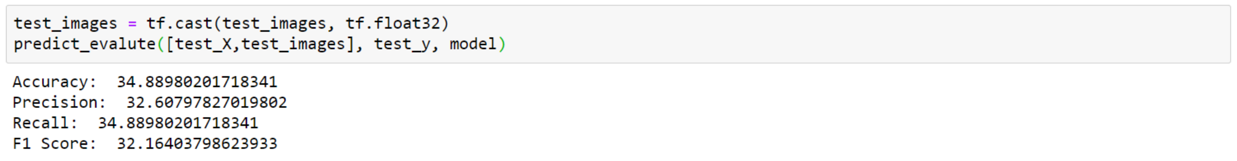
Figure 4.1 Accuracy Graph

Accuracy refers to how many samples were predicted correctly out of all? Let's assume having 50 samples to predict and the model predicted 35 out of 50 correctly then the accuracy will (35/50) * 100 = 70%. The accuracy reported by our model is shown in figure 4.2.

Precision refers to the ratio of true positive predictions out of total positive predictions which can be calculated as (TP/TP+FP). The precision score is also shown in figure 4.2.

Recall refers to the ratio of true positive predictions out of total positive examples which can be calculated as (TP/TP+FN). The recall score is also shown in figure 4.2.

F1-score is harmonic mean of precision and recall which can be calculated as 2*[(precision*recall)/(precision+recall)]. The f1-score is also shown in the figure 4.2.


```
test_images = tf.cast(test_images, tf.float32)
predict_evalute([test_X,test_images], test_y, model)
```

```
Accuracy:  34.88980201718341
Precision:  32.60797827019802
Recall:  34.88980201718341
F1 Score:  32.16403798623933
```

Figure 4.2 Results

Below figure 4.3 is of **Confusion Matrix** which shows the number of samples predicted correctly and wrongly for each class.

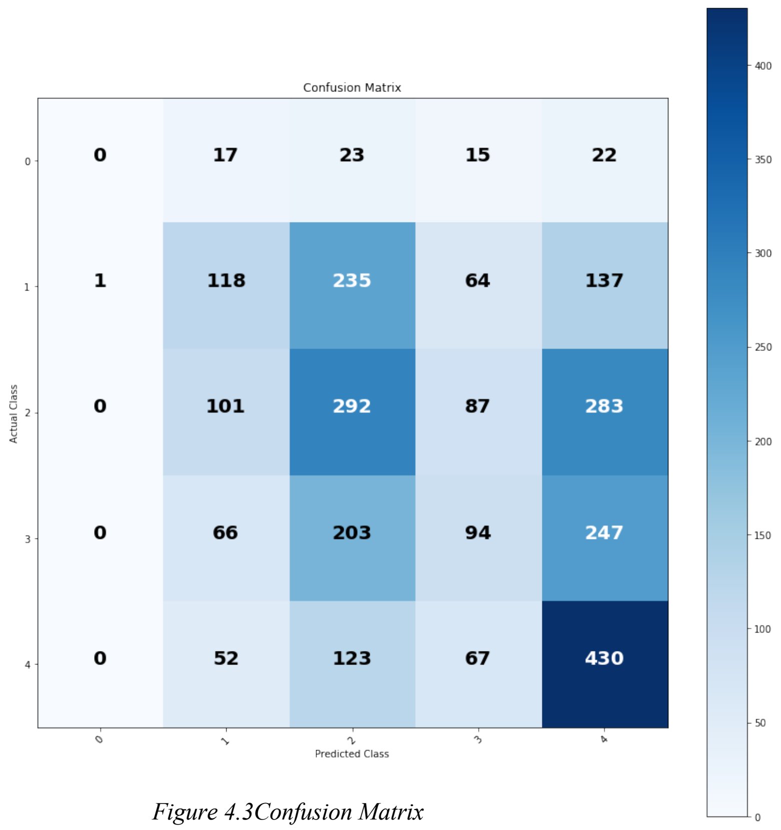
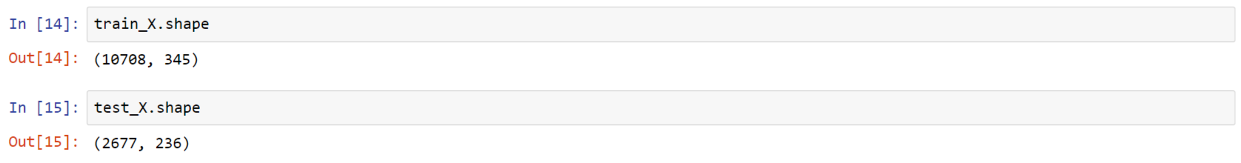


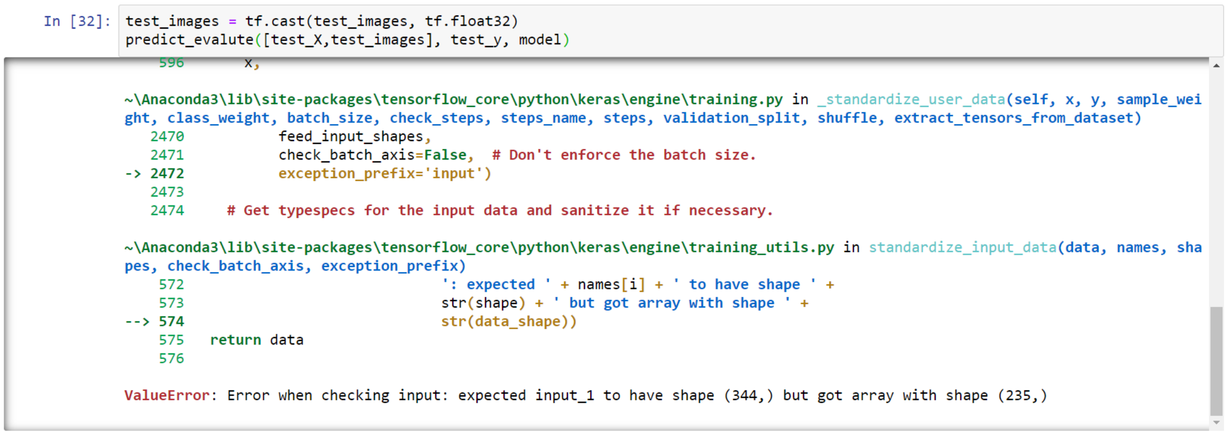
Figure 4.3Confusion Matrix

One problem was encountered while doing this project which was about the different input shapes of training and testing features because of one hot encoding. The problem basically was when we split the data into training and testing data using split function after that train_X, test_X, train_y and test_y was passed to preprocess_data function which performed one hot encoding to categorical features. While performing the one hot encoding the number of columns for both train_X and test_X became different which later on were not compatible for model. This incompatible shapes problem occurred because we were asked to split the data before preprocessing it and both train_X and test_X contains different range of values as in MaturitySize can be (0,1,2,3,4) but it is possible that all MaturitySize values for train_X can be (0,1,2,3,4) and all MaturitySize values for test_X can be (0,1,2). So, in this case after one hot encoding the number of columns for train_X will be greater than the number of columns of test_X. This issue is shown in figure 4.4 and when we passed the test_X to evaluate the model it says the test_X shape is different than the shape of data for which the model was trained and it is shown in figure 4.5.


```
In [14]: train_X.shape
Out[14]: (10708, 345)

In [15]: test_X.shape
Out[15]: (2677, 236)
```

Figure 4.4 Different no. of feature columns



```
In [32]: test_images = tf.cast(test_images, tf.float32)
         predict_evalute([test_X,test_images], test_y, model)
```

```
    596         x,

~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training.py in _standardize_user_data(self, x, y, sample_wei
ght, class_weight, batch_size, check_steps, steps_name, steps, validation_split, shuffle, extract_tensors_from_dataset)
   2470           feed_input_shapes,
   2471           check_batch_axis=False,  # Don't enforce the batch size.
-> 2472           exception_prefix='input')
   2473 
   2474     # Get typespecs for the input data and sanitize it if necessary.

~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training_utils.py in standardize_input_data(data, names, sha
pes, check_batch_axis, exception_prefix)
    572                            ': expected ' + names[i] + ' to have shape ' +
    573                            str(shape) + ' but got array with shape ' +
--> 574                            str(data_shape))
    575   return data
    576 

ValueError: Error when checking input: expected input_1 to have shape (344,) but got array with shape (235,)
```

Figure 4.5 Incompatible shapes

To resolve this issue, we have called the preprocess_data inside the split function and preprocessed the data before splitting into training and testing data.

5 Conclusion:

This project is about prediction of pet adoption speed based on the given pet's dataset such as (Type, Breed1, Breed2, Gender, Color1, Color2, Color3, MaturitySize, FurLength, Vaccinated, Dewormed, Sterilized, Health and State) and images for each pet. We have used an approach which combines the multi-layer perceptron and convolutional neural network to predict the adoption speed. After all the preprocessing and training steps we concluded that we are getting 34.88% accuracy with our approach. The other results such as precision, recall, score and confusion matrix are also shown in above figure 4.2 and figure 4.3.