Question 1:

Describe the scheme you will use to validate your model. You may include figures such as the ones shown the lecture on generalization and evaluation. Remember that you are not provided the test labels, as this is an active challenge. To get test scores for a challenge submission, you would need to submit your predictions via Kaggle - this is necessary to earn extra credit, but it's not a requirement for the completion of this project. Instead, to earn 'standard' credit, you'll need to select an appropriate way of testing your model, locally on your machine, using only the data you're provided in the 'training' files. You can either select a subset of it for testing or cross-validate. You'll need to report what we'll call the local test score - a surrogate for the test score that Kaggle would assign. The local test score you report should be an unbiased estimate of the Kaggle test score.

Solution:
I divided my dataset into 90 percent train data and 10% test data. I performed K-cross validation on the train dataset where I put k=5. In k-cross validation , the train dataset is randomly divided into 5 blocks where one block is taken as validation set every iteration and the remaining blocks as training data. The best model out of all the models is selected based the RMSE values. Lower the RMSE , the better will be the model.
This is done by using the KFold from sklearn.model_selection.

Question 2:

[2 points] Describe how you plan to handle the training data set by answering the following questions.
(a) Do you include metadata in your model? If yes, do you pre-process it? How do you pre-process it?

Solution:

Metadata has been used for Linear and ensemble models. There is no preprocessing of data as the metadata is already in the range of (-1,1) , meaning it is already normalized. Data cleaning and Normalization are one of the important data preprocessing steps for any machine learning model . There are no missing values in the dataset and the data seems well normalized .

I performed some exploratory data analysis on the metadata. Data analysis is a way to know more about the dataset we are about to use . By finding the correlation matrix , we can find how different features are correlated to each other.

	Subject Focus	Eyes	Face	Near	Action	Accessory	Group	Collage	Human	Occlusion	Info	Blur	Pawpularity
Subject Focus	1.000000	0.076794	0.038252	0.058672	0.014006	0.020619	-0.052150	-0.038534	-0.075295	-0.076851	-0.040498	-0.046407	-0.009853
Eyes	0.076794	1.000000	0.584484	0.133127	-0.020544	0.052480	-0.084975	0.066361	0.036046	0.022143	0.038179	-0.507323	-0.006686
Face	0.038252	0.584484	1.000000	0.139031	-0.012022	0.034788	-0.108022	0.050847	0.024170	0.013389	0.024704	-0.068198	0.008018
Near	0.058672	0.133127	0.139031	1.000000	-0.027312	0.030234	-0.319107	-0.263498	0.065215	-0.009287	-0.146188	-0.017566	0.001001
Action	0.014006	-0.020544	-0.012022	-0.027312	1.000000	0.025377	-0.002432	-0.004270	-0.009429	-0.010832	-0.017191	0.012009	-0.001373
Accessory	0.020619	0.052480	0.034788	0.030234	0.025377	1.000000	-0.057301	0.065860	-0.041745	-0.038912	0.075063	-0.035012	0.013287
Group	-0.052150	-0.084975	-0.108022	-0.319107	-0.002432	-0.057301	1.000000	0.132007	-0.104287	0.003568	0.063311	0.007899	0.016469
Collage	-0.038534	0.066361	0.050847	-0.263498	-0.004270	0.065860	0.132007	1.000000	0.011476	0.054611	0.482141	-0.026591	0.001732
Human	-0.075295	0.036046	0.024170	0.065215	-0.009429	-0.041745	-0.104287	0.011476	1.000000	0.634381	0.018171	-0.015941	0.003983
Occlusion	-0.076851	0.022143	0.013389	-0.009287	-0.010832	-0.038912	0.003568	0.054611	0.634381	1.000000	0.117725	-0.006338	0.001979
Info	-0.040498	0.038179	0.024704	-0.146188	-0.017191	0.075063	0.063311	0.482141	0.018171	0.117725	1.000000	-0.022604	-0.004735
Blur	-0.046407	-0.507323	-0.068198	-0.017566	0.012009	-0.035012	0.007899	-0.026591	-0.015941	-0.006338	-0.022604	1.000000	-0.023540
Pawpularity	-0.009853	-0.006686	0.008018	0.001001	-0.001373	0.013287	0.016469	0.001732	0.003983	0.001979	-0.004735	-0.023540	1.000000

For the above matrix , we can see that some of the features seem to be substantially correlated to each other. For example, "Human" and "Occlusion".There will be features which have a low correlation to the target variable. Linear models trained on these features can perform poorly.

```
Subject Focus   -0.009853
Eyes            -0.006686
Face             0.008018
Near             0.001001
Action          -0.001373
Accessory        0.013287
Group            0.016469
Collage          0.001732
Human            0.003983
Occlusion        0.001979
Info            -0.004735
Blur            -0.023540
Name: Pawpularity, dtype: float64
```

Of all the features, "Blur" seems to be the most predictive for the target.

(b) Do you pre-process the image? If yes, describe how you plan to do it.Please remember that any augmentation needs to be performed on training data only.

Image data has been preprocessed for the deep-learning model.The data is split into training 80%,valid 10% and test 10%.

It is preprocessed in the following way :

1. The jpeg image is taken from the image path and decoded into a tensor.
2. The tensor is resized into 128 x 128 x 3
3. Code:

   image_height = 128

   image_width = 128

   raw = tf.io.read_file(image_path)

   image = tf.image.decode_jpeg(raw, channels=3)

   image = tf.cast(image, tf.float32) / 255.0

   image = tf.image.resize(image, (image_height, image_width))

4. Example :
5.

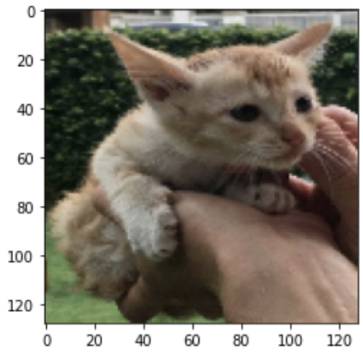
   Original image:

```
(470, 393, 3)
```

First Training Image

Preprocessed Image:

```
type:  <class 'tensorflow.python.framework.ops.EagerTensor'>
 shape:  (128, 128, 3)
```



6. Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data.Here we do data augmentation by using Imagedatagenerator from tensorflow.keras.preprocessing.image import ImageDataGenerator

   data_augmentation = ImageDataGenerator(

   rotation_range = 15,

   zoom_range = 0.15,

   width_shift_range = 0.2,

   height_shift_range = 0.2,

   shear_range = 0.1,

horizontal_flip = True,

fill_mode = "nearest")

Question 3:

[2 points] Try out a linear model. Describe any regularization you're using, how you are setting your hyperparameters and other details a reader would need to replicate your work. Report the local test score for this model.

Solution:

Model 1: Linear Regression

- Divided the dataset into 90% train and 10% test.
- Performed K-cross validation on the training data with K=5.
- Min RMSE = 20.12582752830501for validation set.
- For the unseen test data set (10 percent dataset)) , the best Linear Regression model got a RMSE = 19.825563748263995
- Pawpularity scores for the images in the test.csv :

Id	Pawpularity
4128bae22183829d2b5fea10effdb0c3	35.960492
43a2262d7738e3d420d453815151079e	35.219454
4e429cead1848a298432a0acad014c9d	35.147034
80bc3ccafcc51b66303c2c263aa38486	37.435658
8f49844c382931444e68dffbe20228f4	34.592593
b03f7041962238a7c9d6537e22f9b017	41.014331
c978013571258ed6d4637f6e8cc9d6a3	30.292042
e0de453c1bffc20c22b072b34b54e50f	33.827219

In this Model , there was no regularization used

Model 2: Ridge Regression (L2 Regularization)

- Same train, test split and performed K-cross validation with K=4
- Hyperparameter tuning of alpha and Max-iterations with values

param_grid = dict(alpha=[0.1, 0.2,0.3 ,0.4, 0.5, 0.6, 0.7, 0.8 , 0.9 , 1 , 1.2 , 1.4] ,max_iter = [2000,4000,8000,12000, 16000,20000, 50000,100000, 500000,1000000])

- Best Model after hyperparameter tuning :Ridge(alpha=0.1, max_iter=2000)
- Min RMSE = 20.108858302208745 for validation set.
- For the unseen test data set (10 percent dataset)) , the best Linear Regression model got a RMSE = 19.82160578205504
- Pawpularity scores for the images in the test.csv :

	Id	Pawpularity
0	4128bae22183829d2b5fea10effdb0c3	36.993339
1	43a2262d7738e3d420d453815151079e	36.903271
2	4e429cead1848a298432a0acad014c9d	35.881024
3	80bc3ccafcc51b66303c2c263aa38486	37.254772
4	8f49844c382931444e68dffbe20228f4	33.966199
5	b03f7041962238a7c9d6537e22f9b017	39.569343
6	c978013571258ed6d4637f6e8cc9d6a3	31.168533
7	e0de453c1bffc20c22b072b34b54e50f	36.046670

Model3: Lasso Regression (L1 Regularization)

- Same train, test split and performed K-cross validation with K=4
- Hyperparameter tuning of alpha and Max-iterations with values

param_grid = dict(alpha=[0.1, 0.2,0.3 ,0.4, 0.5, 0.6, 0.7, 0.8 , 0.9 , 1 , 1.2 , 1.4] ,max_iter = [2000,4000,8000,12000, 16000,20000, 50000,100000, 500000,1000000])

- Best Model after hyperparameter tuning :Lasso(alpha=0.1, max_iter=2000)
- Min RMSE = 20.1362512453177 for validation set.
- For the unseen test data set (10 percent dataset)) , the best Linear Regression model got a RMSE = 19.837202535746968
- Pawpularity scores for the images in the test.csv :

	Id	Pawpularity
0	4128bae22183829d2b5fea10effdb0c3	38.325453
1	43a2262d7738e3d420d453815151079e	38.325453
2	4e429cead1848a298432a0acad014c9d	38.325453
3	80bc3ccafcc51b66303c2c263aa38486	38.088129
4	8f49844c382931444e68dffbe20228f4	38.088129
5	b03f7041962238a7c9d6537e22f9b017	38.325453
6	c978013571258ed6d4637f6e8cc9d6a3	38.088129
7	e0de453c1bffc20c22b072b34b54e50f	38.088129

My conclusion with Linear Models would be : The features have a low correlation to the target variable from the correlation matrix. Linear models trained on these features are therefore likely to perform poorly.

Question 4:

[2 points] Try out an ensemble model. What are you using as the base classifiers? How are they combined in the ensemble? What are the pertinent values for the hyperparameters – ensemble size, for instance. Report the local test score.

Solution:

Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

Model 1: Using Random Forest

- Used RandomForestRegressor() from sklearn.ensemble package.
- Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression.A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.
- Hypertuned the parameters of n_estimators and max_depth with values of n_estimator=range(1,151) and max_depth=range(2,7).Best Model=RandomForestRegressor(max_depth=6, n_estimators=66, random_state=0)
- Min RMSE = 20.053794474029093 for validation set.
- For the unseen test data set (10 percent dataset)) , the best Linear Regression model got a RMSE = 19.74950837376964
- Pawpularity scores for the images in the test.csv :

	Id	Pawpularity
0	4128bae22183829d2b5fea10effdb0c3	31.906285
1	43a2262d7738e3d420d453815151079e	39.077922
2	4e429cead1848a298432a0acad014c9d	59.498232
3	80bc3ccafcc51b66303c2c263aa38486	33.179369
4	8f49844c382931444e68dffbe20228f4	25.954558
5	b03f7041962238a7c9d6537e22f9b017	50.269451
6	c978013571258ed6d4637f6e8cc9d6a3	38.703619
7	e0de453c1bffc20c22b072b34b54e50f	31.311410

Model2: Adaboost:

- Best Model after hyperparameter tuning is AdaBoostClassifier(base_estimator=DecisionTreeClassifier (max_depth=5),learning_rate=0.1, n_estimators=125)
- Min RSME loss for validation set:24.55918876971783
- For unseen test dataset (10 percent dataset), RMSE:22.668699999722513
- Pawpularity scores for the images in the test.csv :

	Id	Pawpularity
0	4128bae22183829d2b5fea10effdb0c3	37
1	43a2262d7738e3d420d453815151079e	28
2	4e429cead1848a298432a0acad014c9d	100
3	80bc3ccafcc51b66303c2c263aa38486	54
4	8f49844c382931444e68dffbe20228f4	32
5	b03f7041962238a7c9d6537e22f9b017	55
6	c978013571258ed6d4637f6e8cc9d6a3	30
7	e0de453c1bffc20c22b072b34b54e50f	27

Model3: By Stacking

- Used RidgeCV,GradientBoostingRegressor,KNeighborsRegressor,LinearSVR for Level 0 and RandomForestRegressor for Level 1.
- For unseen test dataset (10 percent dataset), RMSE:20.576178378682165
- Pawpularity scores for the images in the test.csv :

	Id	Pawpularity
0	4128bae22183829d2b5fea10effdb0c3	36.937681
1	43a2262d7738e3d420d453815151079e	41.034545
2	4e429cead1848a298432a0acad014c9d	46.177269
3	80bc3ccafcc51b66303c2c263aa38486	47.400000
4	8f49844c382931444e68dffbe20228f4	52.100000
5	b03f7041962238a7c9d6537e22f9b017	40.566667
6	c978013571258ed6d4637f6e8cc9d6a3	38.310556
7	e0de453c1bffc20c22b072b34b54e50f	39.188333

Question 5:

[2 points] Try out deep learning for this task. What network architectures made the most sense here? Can you combine deep learning with some of the other models you've tried? Report the local test score for your best DL-enhanced model.

Solution :

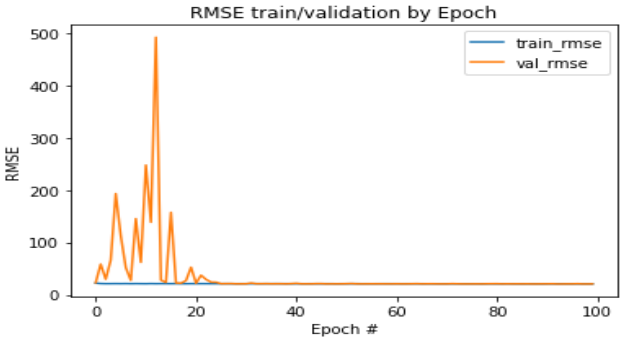
- I split the dataset into 80 percent training , 10 percent validation and 10 percent test dataset.
- I explained the preprocessing steps in the Question 2 above.

- After experimenting with various CNN models and tweaking hyperparameters like the values of dropout. The best model I got is described below.
- The Architecture I used is:

```
model.summary()
```

```
batch_normalization_1 (Batc  (None, 31, 31, 32)       128
hNormalization)

dropout (Dropout)            (None, 31, 31, 32)       0

conv2d_3 (Conv2D)            (None, 31, 31, 64)       18496

batch_normalization_2 (Batc  (None, 31, 31, 64)       256
hNormalization)

conv2d_4 (Conv2D)            (None, 16, 16, 64)       36928

batch_normalization_3 (Batc  (None, 16, 16, 64)       256
hNormalization)

dropout_1 (Dropout)          (None, 16, 16, 64)       0

conv2d_5 (Conv2D)            (None, 16, 16, 128)      73856

batch_normalization_4 (Batc  (None, 16, 16, 128)      512
hNormalization)

max_pooling2d (MaxPooling2D  (None, 8, 8, 128)        0
)

conv2d_6 (Conv2D)            (None, 8, 8, 128)        147584

batch_normalization_5 (Batc  (None, 8, 8, 128)        512
hNormalization)

dropout_2 (Dropout)          (None, 8, 8, 128)        0

flatten (Flatten)            (None, 8192)             0

dense (Dense)                (None, 512)              4194816

dropout_3 (Dropout)          (None, 512)              0

dense_1 (Dense)              (None, 1)                513

=================================================================
Total params: 4,490,241
Trainable params: 4,489,345
Non-trainable params: 896
_________________________________________________________________
```

- Root mean squared error vs epoch while training:



- For Validation Dataset RMSE :20.152287307398147
- For unseen test data set, RMSE=19.882104736020107
- Pawpularity scores for the images in the test.csv :

	Id	Pawpularity
0	4128bae22183829d2b5fea10effdb0c3	35.602200
1	43a2262d7738e3d420d453815151079e	35.325481
2	4e429cead1848a298432a0acad014c9d	34.034252
3	80bc3ccafcc51b66303c2c263aa38486	34.142452
4	8f49844c382931444e68dffbe20228f4	34.622528
5	b03f7041962238a7c9d6537e22f9b017	34.389511
6	c978013571258ed6d4637f6e8cc9d6a3	34.720512
7	e0de453c1bffc20c22b072b34b54e50f	33.953896

Yes , we can combine our deep learning model with other models which we tried by taking an average of the predicted values produced by both the models . For example , we can combine our deep learning with Linear regression or ensemble models. The RMSE Loss is least for Random forest . Combining it with CNN model above , we get the following result for the images in the test set:

	Id	Pawpularity
0	4128bae22183829d2b5fea10effdb0c3	35.460002
1	43a2262d7738e3d420d453815151079e	36.305497
2	4e429cead1848a298432a0acad014c9d	36.057287
3	80bc3ccafcc51b66303c2c263aa38486	44.913538
4	8f49844c382931444e68dffbe20228f4	38.300450
5	b03f7041962238a7c9d6537e22f9b017	34.494457
6	c978013571258ed6d4637f6e8cc9d6a3	36.072785
7	e0de453c1bffc20c22b072b34b54e50f	34.447570