CS589 Machine Learning - Fall 2021
MiniProject

Submitted By- Shriya Sehgal
Spire ID -33060871

---

1 Describe the scheme you will use to validate your model. You may include figures such as the ones shown in the lecture on generalization and evaluation.

To validate the models, the dataset has been shuffled and divided into 3 parts in the ratio of 60:20:20 for training ,validation and testing.

- The training data is used to teach the machine learning models the patterns in the images and the metadata to produce the desired results.
- The validation data is used to pick the best models, tune the hyperparameters and get an estimate of the performance of the model.
- The best models are then applied to the test data to check how the model performs on the data it has never seen before.

We first calculate the rmse loss for the validation data for all the iterations of all the models, and the one with the least rmse loss is chosen and finally applied to the test data to give the final results. The formula of the RMSE loss is given below.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Of all the models implemented, the stacking regressor had the least RMSE loss of **19.8418.**

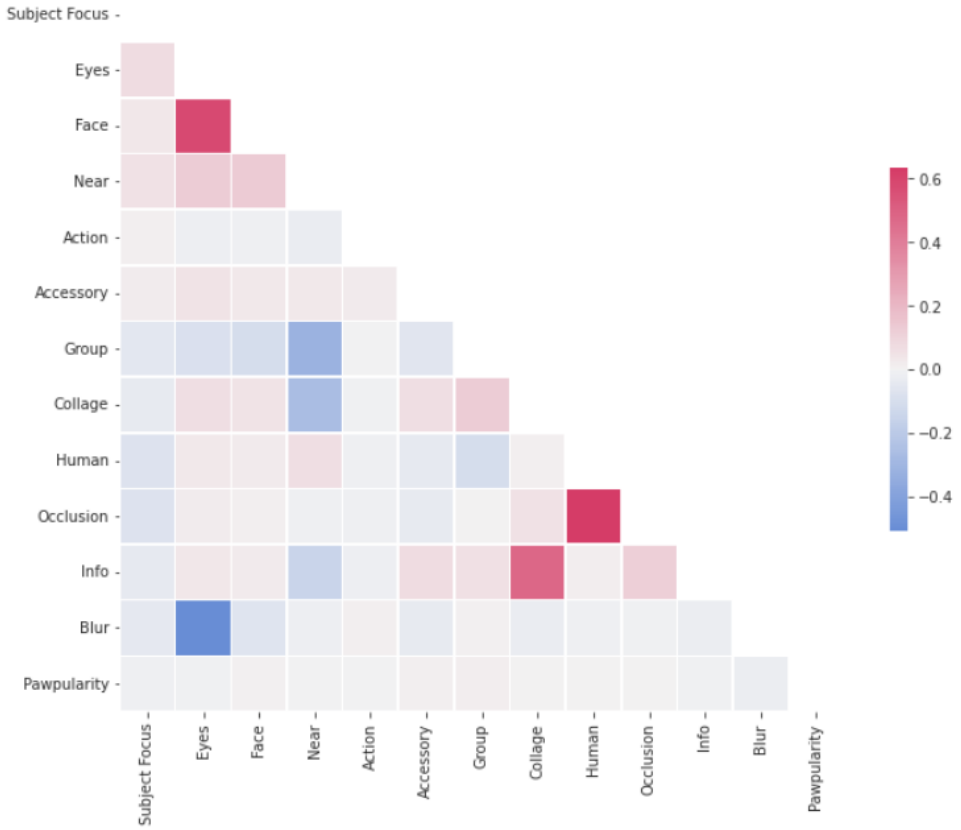
2. Describe how you plan to handle the training data set by answering the following questions.
(a) Do you include metadata in your model? If yes, do you pre-process it? How do you pre-process it?

We have used the metadata in the linear and ensemble models. The data is extracted in the form of a dataframe containing the key visual quality and composition parameters.Each value is a binary number stating whether the feature is present in the image of the animal or not. The following properties are included.

```
Index(['Id', 'Subject Focus', 'Eyes', 'Face', 'Near', 'Action', 'Accessory',
       'Group', 'Collage', 'Human', 'Occlusion', 'Info', 'Blur',
       'Pawpularity'],
```

With the help of the metadata I created X_train,y_train,X_val, Y_val, X_test, Y_test. No preprocessing was necessary in the case of the linear and ensemble model as the features given in the metadata are directly taken from the train.csv file.

To understand the data, the heatmap visualising the correlation among all the features and pawpularity is given below.



(b) Do you pre-process the image? If yes, describe how you plan to do it.

Yes. For the CNN model, images are used to calculate the weights and the loss. Paths of all the images corresponding to their Ids are added to a dataframe. The images are read and loaded into a numpy array X. The images are resized into (128,128,3) which is the input of the architecture built and then normalised.

```
def load_img(image_path):
    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, (128, 128))
    return image
```

These images are then passed through the model in the batches of 32 for the network to learn the weights and calculate the Root mean square loss for each epoch.

3. 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.

The linear model used was the Ridge Regression where the loss function is the linear least squares function and uses L2 regularization technique. L2 norm deals with the multicollinearity problems between independent variables and penalize the insignificant features.

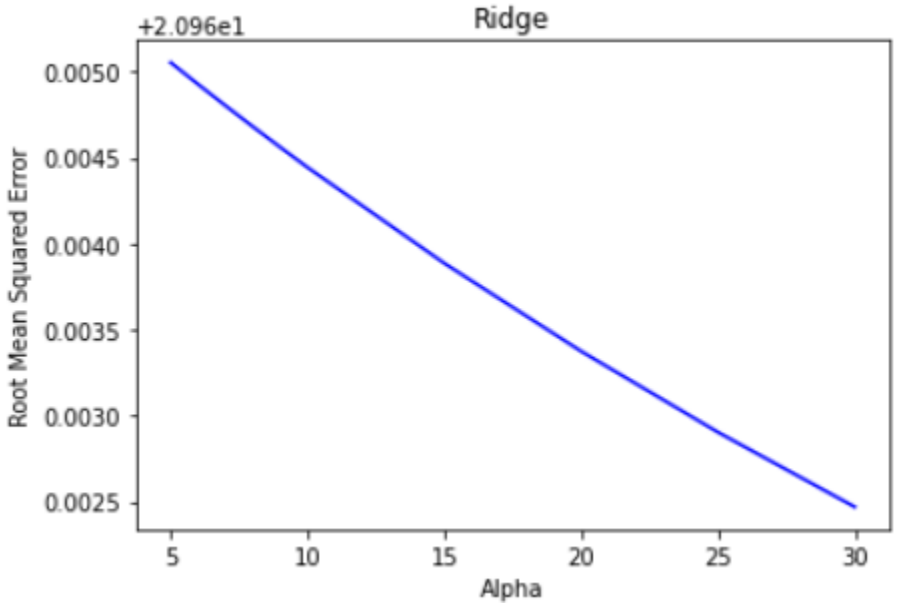
$$\|\mathbf{w}\|_2 = \left(|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2\right)^{\frac{1}{2}}$$

2-norm (also known as L2 norm or Euclidean norm)

I tuned the hyperparameters by setting multiple values of alphas, [5,6,7,8,9,10,15,20,25,30] after which the best value of RMSE **19.8630** with the alpha value of 30 was achieved.

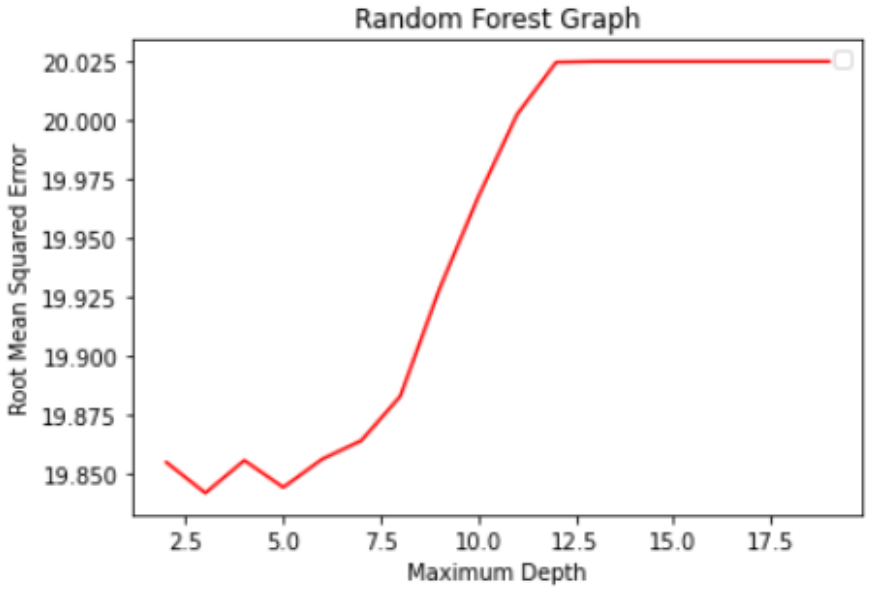
The iterations with the values of Root mean Squared Error corresponding to each alpha value and its graph is given below.

```
Alpha: 5
RMSE: 19.865421132935865
Alpha: 6
RMSE: 19.86530782785057
Alpha: 7
RMSE: 19.865196381890797
Alpha: 8
RMSE: 19.865086756759766
Alpha: 9
RMSE: 19.864978914996254
Alpha: 10
RMSE: 19.864872819961068
Alpha: 15
RMSE: 19.864367319100538
Alpha: 20
RMSE: 19.863900438124215
Alpha: 25
RMSE: 19.863468434624632
Alpha: 30
RMSE: 19.863067970119353
Best Values: RSME= 19.863067970119353, Model= Ridge(alpha=30), Alpha= 30
```



4. 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.

For the ensemble model, inbuilt RandomForestRegressor is used. The Decision Tree regressor works as the base classifier. I tuned the hyperparameters by setting multiple values of max_depth in the range (2,20), n_estimator = 100 and max_features = log2. The graph with the value of RMSE corresponding to all the models with varying depths is shown below.
The best RMSE score received was **19.8421** with the max_depth of 3.



Random Forest Graph with the best RSME =19.842183360400313 max_depth =3

Along with that I also trained a StackingRegressor which uses the strength of each individual estimator by using their output as input of the final estimator. This is built by stacking the output of the following models and received an rmse loss of **19.8418.**

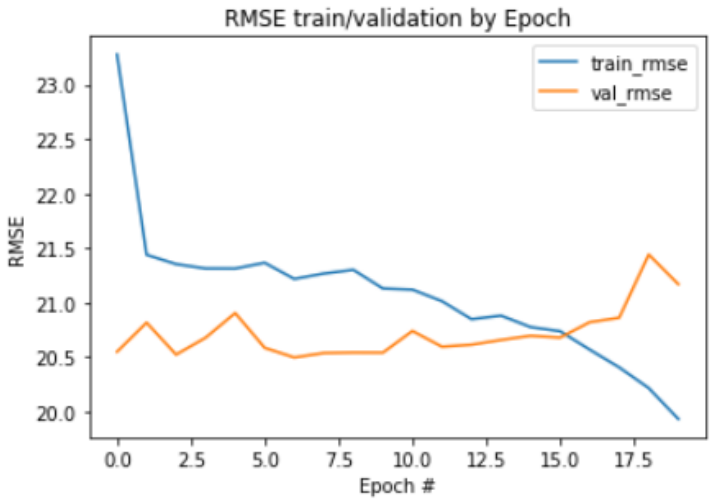
```
StackingRegressor(estimators=[('lr', RidgeCV(alphas=array([ 0.1,  1. , 10. ]))),
                              ('svr', LinearSVR(random_state=21))],
                  final_estimator=RandomForestRegressor(max_depth=4,
                                                        max_features='sqrt',
                                                        random_state=21))
```

5. 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.

To find the popularity score of all animals, I have built a CNN architecture from scratch. The model's summary is given below.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 128, 128, 3)]     0

 conv2d (Conv2D)             (None, 64, 64, 128)       3584

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        73792

 conv2d_2 (Conv2D)           (None, 16, 16, 32)        18464

 conv2d_3 (Conv2D)           (None, 8, 8, 16)          4624

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 128)               131200

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 231,793
Trainable params: 231,793
Non-trainable params: 0
_________________________________________________________________
```

I used Adam optimizer with the learning rate of 0.001 and trained the model for 20 epochs. RMSE, MSE and MAPE loss was calculated at each iteration The model consists of .4 Convolution layers followed by Relu activation, 2 dense layers and a dropout layer with p = 0.3. The graph of the training and Validation RMSE loss corresponding to each epoch is shown below,



The RMSE loss for the given model on the test dataset is **20.2523.**

We can also combine the deep learning model with the ensemble method by taking the average of the scores produced by the network and all the models in the ensemble. This will give us a better approximation of the RMSE loss and hence a better pawpularity scores of the test images.