PetFinder.my - Pawpularity

Kaggle Contest

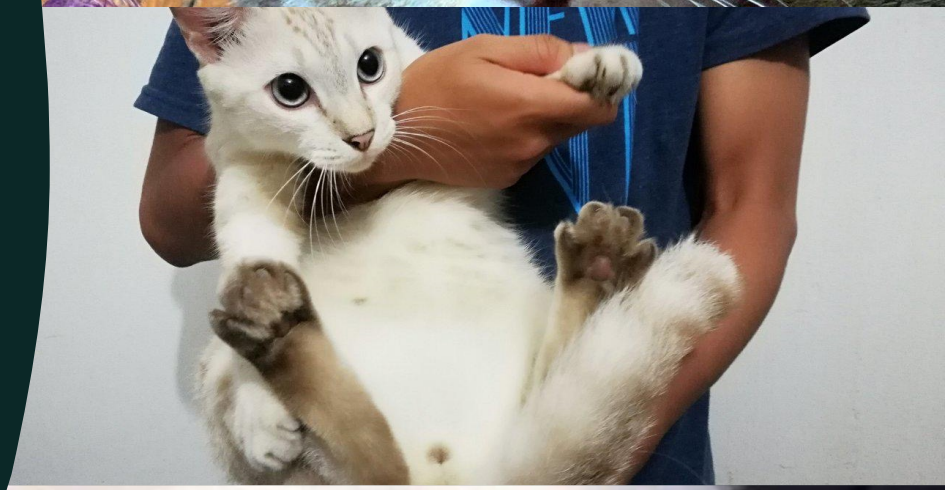
Predict the popularity of shelter pet photos

CS5489

Course project

CONTEXT

0. Background
1. Feature Analysis
2. Traditional Regression
3. Playing with Deep Learning
4. Prediction



Background

PetFinder.my is Malaysia’s leading animal welfare platform, featuring over 180,000 animals with 54,000 happily adopted.

Currently, this website uses a basic Cuteness Meter to rank pet photos. It analyzes picture composition and other factors compared to the performance of thousands of pet profiles. While this basic tool is helpful, it's still in an experimental stage and the algorithm could be improved.

The goal is to analyze raw images and metadata to predict the “Pawpularity” of pet photos. You'll train and test your model on PetFinder.my's thousands of pet profiles.

Feature Analysis

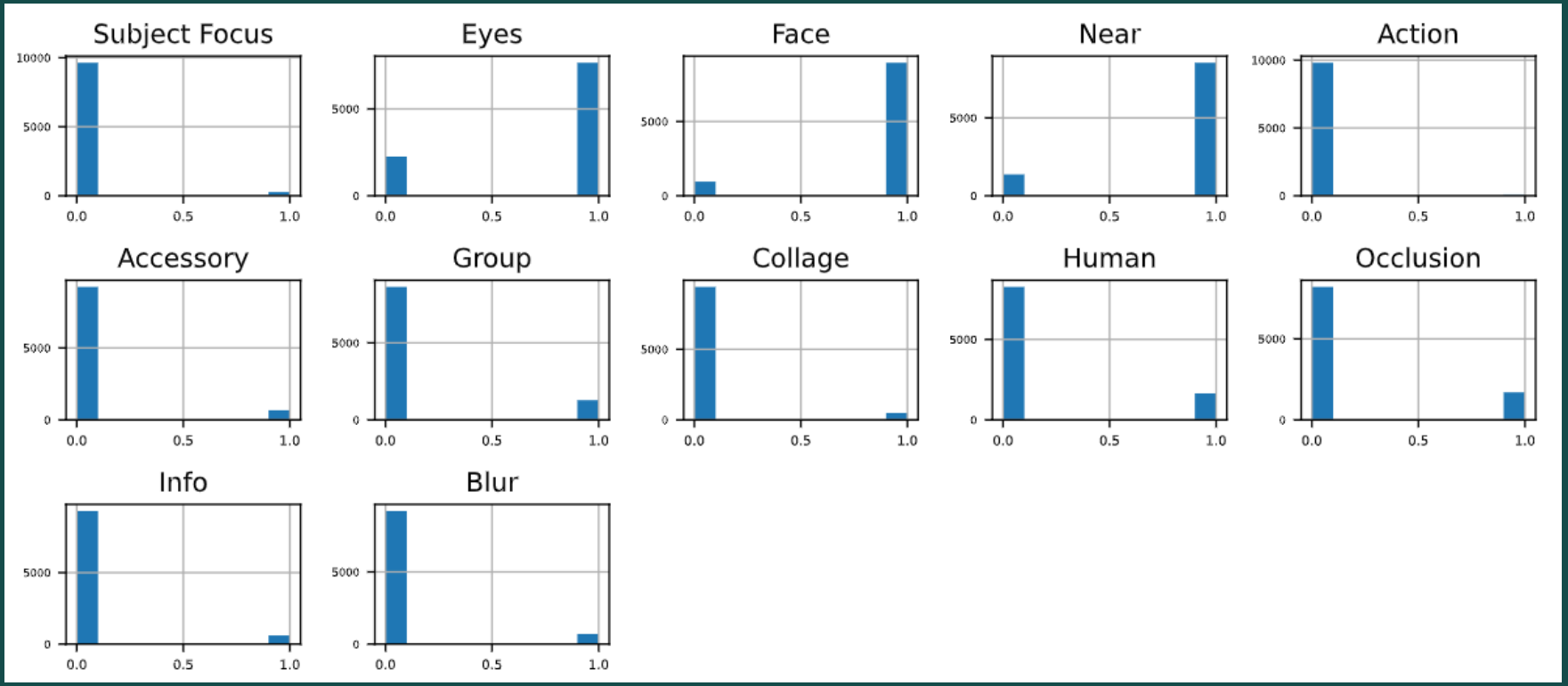
Feature Analysis

First we take a look at train metadata. It has 12 columns of features. The number of Sample is 9912.

	Subject Focus	Eyes	Face	Near	Action	Accessory	Group	Collage	Human	Occlusion	Info	
count	9912.000000	9912.000000	9912.000000	9912.000000	9912.000000	9912.000000	9912.000000	9912.000000	9912.000000	9912.000000	9912.000000	9912.0
mean	0.027643	0.772599	0.903955	0.861582	0.009988	0.067797	0.129338	0.049637	0.166263	0.172014	0.061239	0.0
std	0.163957	0.419175	0.294668	0.345356	0.099444	0.251409	0.335591	0.217204	0.372335	0.377411	0.239780	0.2
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	0.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
50%	0.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
75%	0.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0

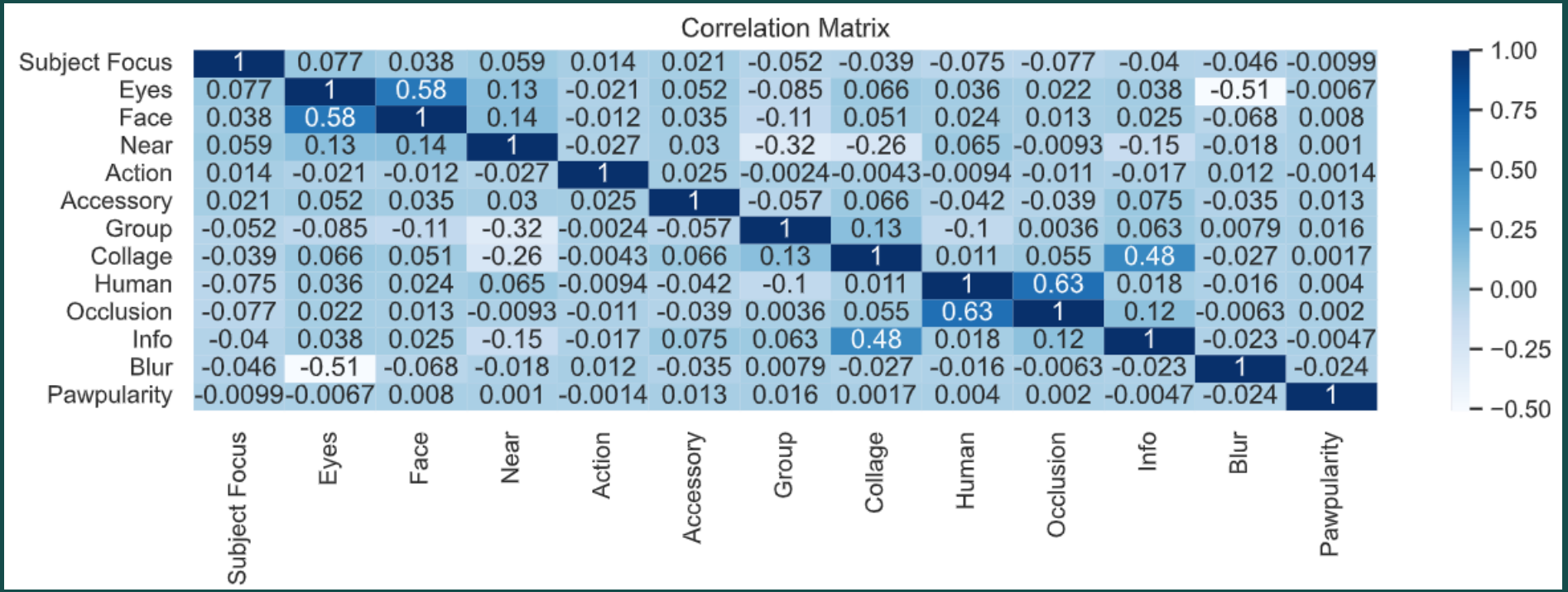
Feature Analysis

Visualize the distribution of all features. All features are label value. Now we wonder are this features get enough differences to properly partition the data?



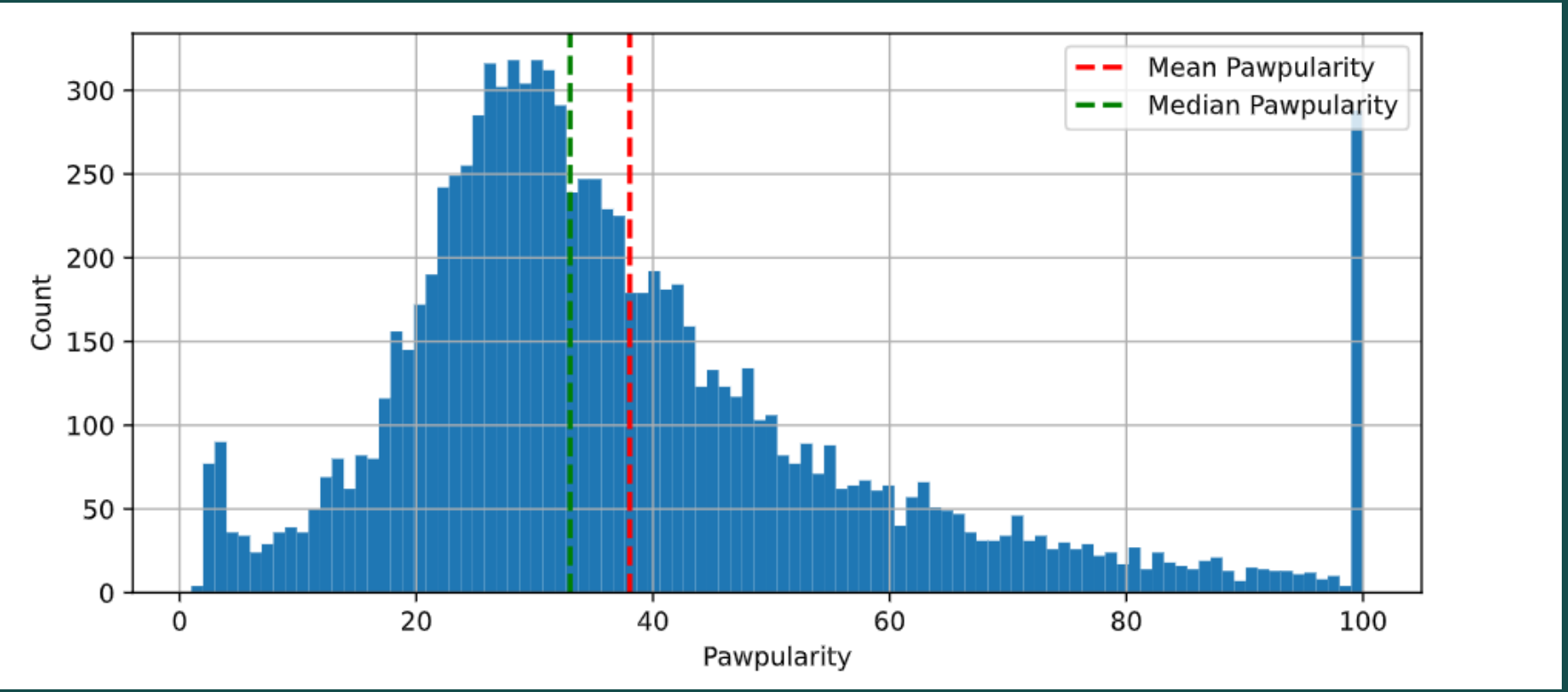
Feature Analysis

From the correlation matrix we can see that actually features do not have strong correlation with pawpularity.



Feature Analysis

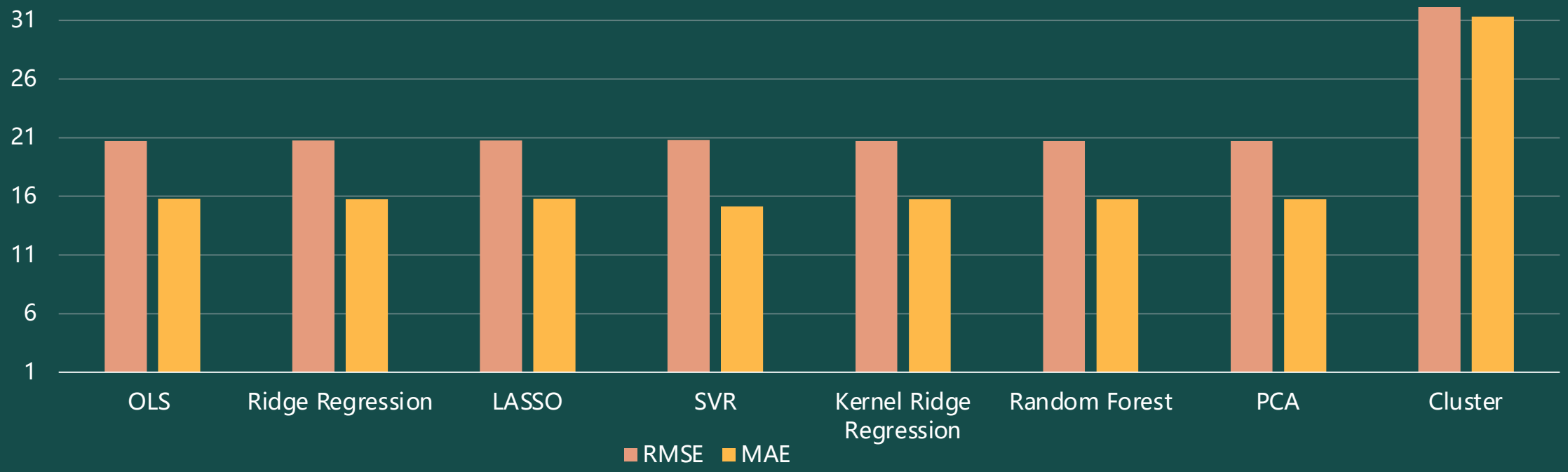
Pawpularity is our target, we can see that most pawpularity scores locate in 25 to 30, and numbers of extreme situation is small.



Traditional Regression

Traditional Regression

First we split data into train set(90%) and validation set(10%), then fit it. We try different methods for both linear and non-linear regression. The result shows that Random Forest has the best performance. In the whole, There's not much difference in performance of all traditional models .



Some tips

Cross-validation

```
:
# parameters for cross-validation
paramgrid = {'C':       logspace(-2,2,5),
             'gamma':   logspace(-2,2,5),
             'epsilon': logspace(-2,2,5)}

# do cross-validation
svrcv = model_selection.GridSearchCV(
    svm.SVR(kernel='rbf'),   # estimator
    paramgrid,                        # parameters to try
    scoring='neg_mean_squared_error',   # score function
    cv=5,
    n_jobs=-1, verbose=1)                 # show progress
svrcv.fit(trainX, trainY)

print(svrcv.best_score_)
print(svrcv.best_params_)
```

```
Fitting 5 folds for each of 125 candidates, totalling 625 fits
-430.85835459826467
{'C': 10.0, 'epsilon': 10.0, 'gamma': 0.1}
```

Principle component analysis

```
# run PCA
pca = decomposition.PCA(n_components=10)
trainW   = pca.fit_transform(trainX)  # returns the coefficients
valW   = pca.transform(valX)
testW   = pca.transform(test)
v = pca.components_  # the principal component vector
m = pca.mean_        # the data mean
(trainW.shape, valW.shape)
```

```
((8920, 10), (992, 10))
```

Playing with Deep Learning

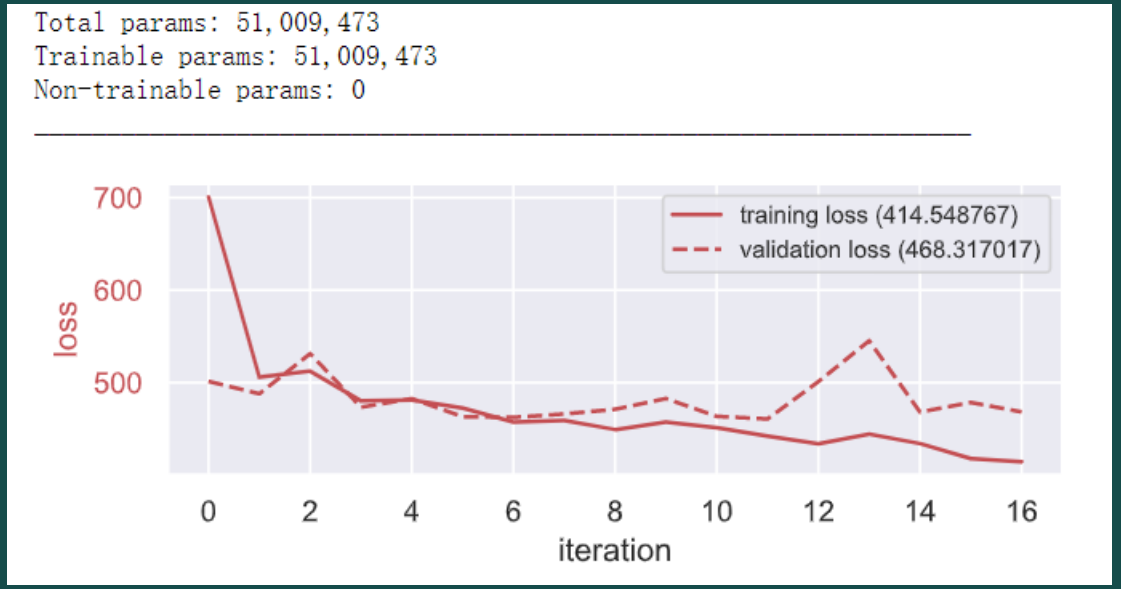
Why we use Neural Network

The metadata we used are not enough to partition the sample.

We need more features from the image to build a prediction. So we consider Neural Networks as it can extract information from the raw image.

The MLP model we use consist of 6 layers with relu activate function.

```
Total params: 51,009,473
Trainable params: 51,009,473
Non-trainable params: 0
```



We use the simplest MLP model as the baseline.

Preprocess

Normalization:

Neural Networks prefer value of 0~1, the pixel value is 0~255 in the image file, we need to normalize it .

Preprocess:

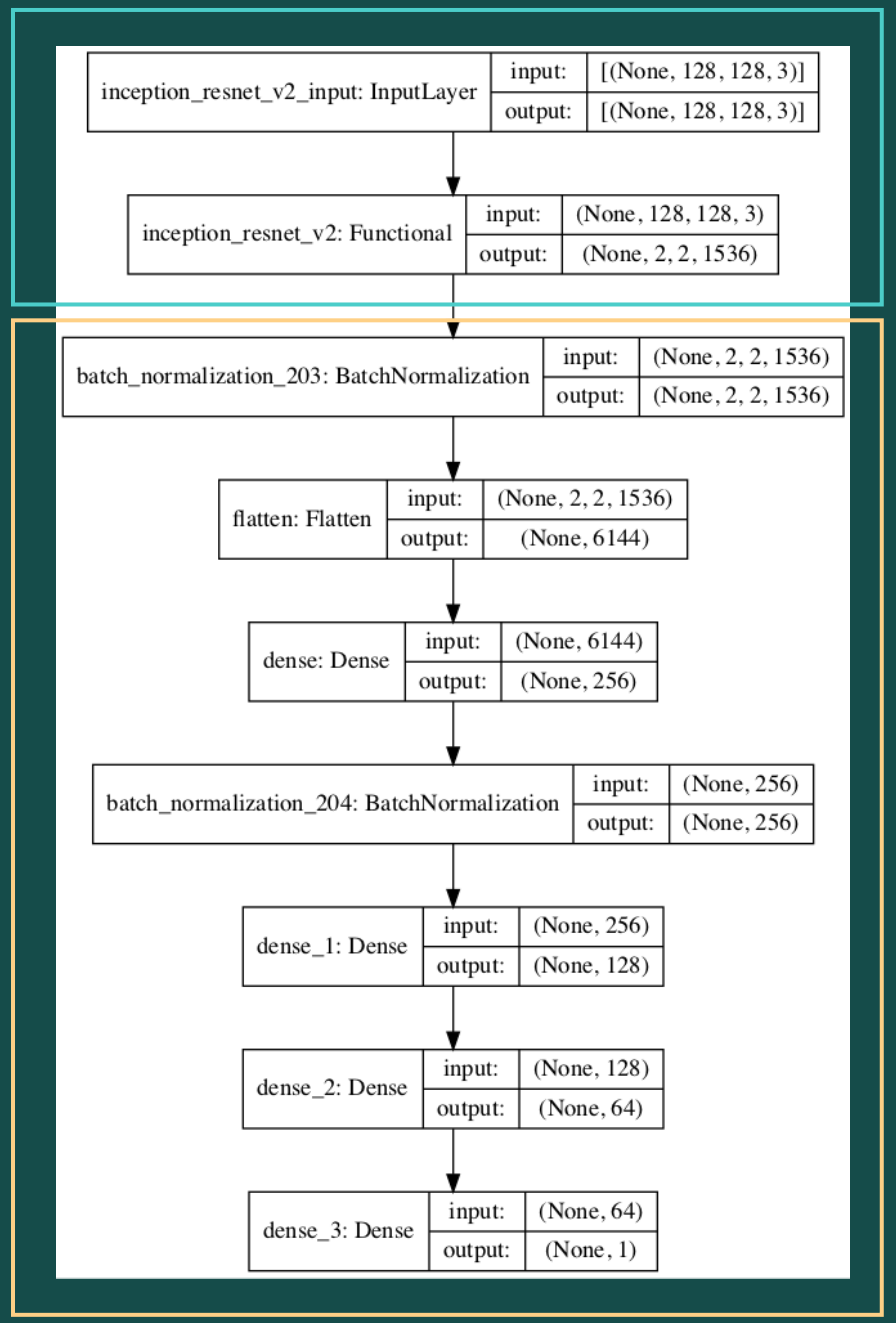
Next we are going to use popular pretrained models in keras, each Keras Application expects a specific kind of input preprocessing, so we should apply preprocess_input function to input.

Data Augmentation

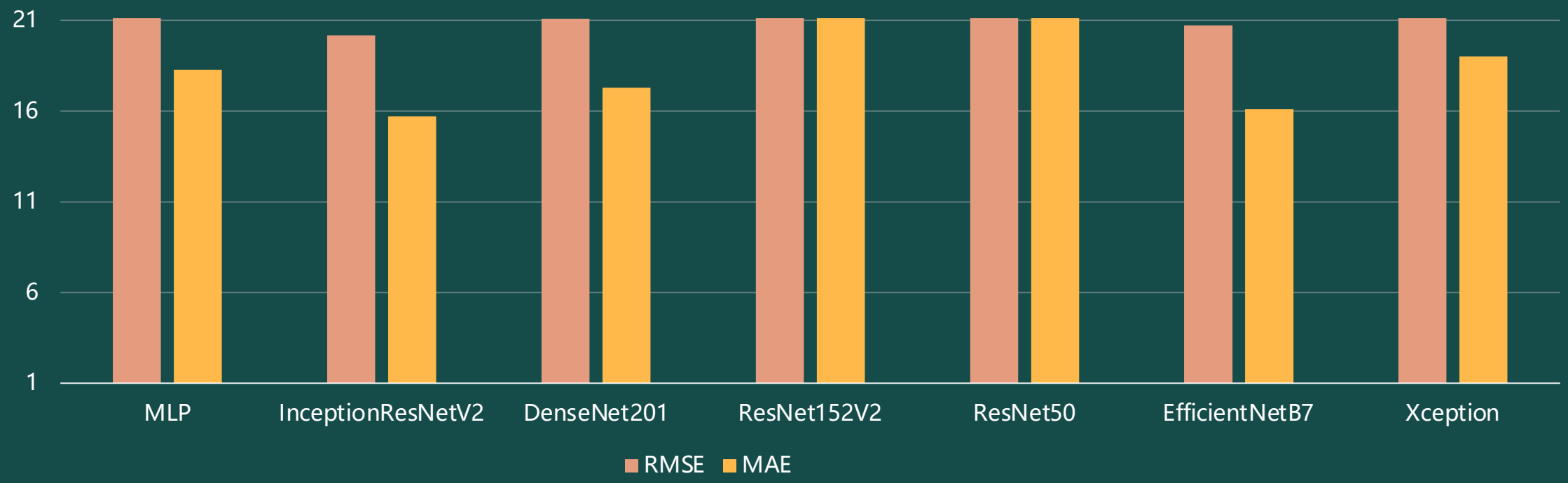
Data augmentation makes the model more robust, we try a different series of augmentation methods including adding noise and scale.

Transfer Learning

Due to limit of resources and time, we use pretrained model to extract features from train_data, then do the prediction.



Playing with Deep Learning



Performance summary

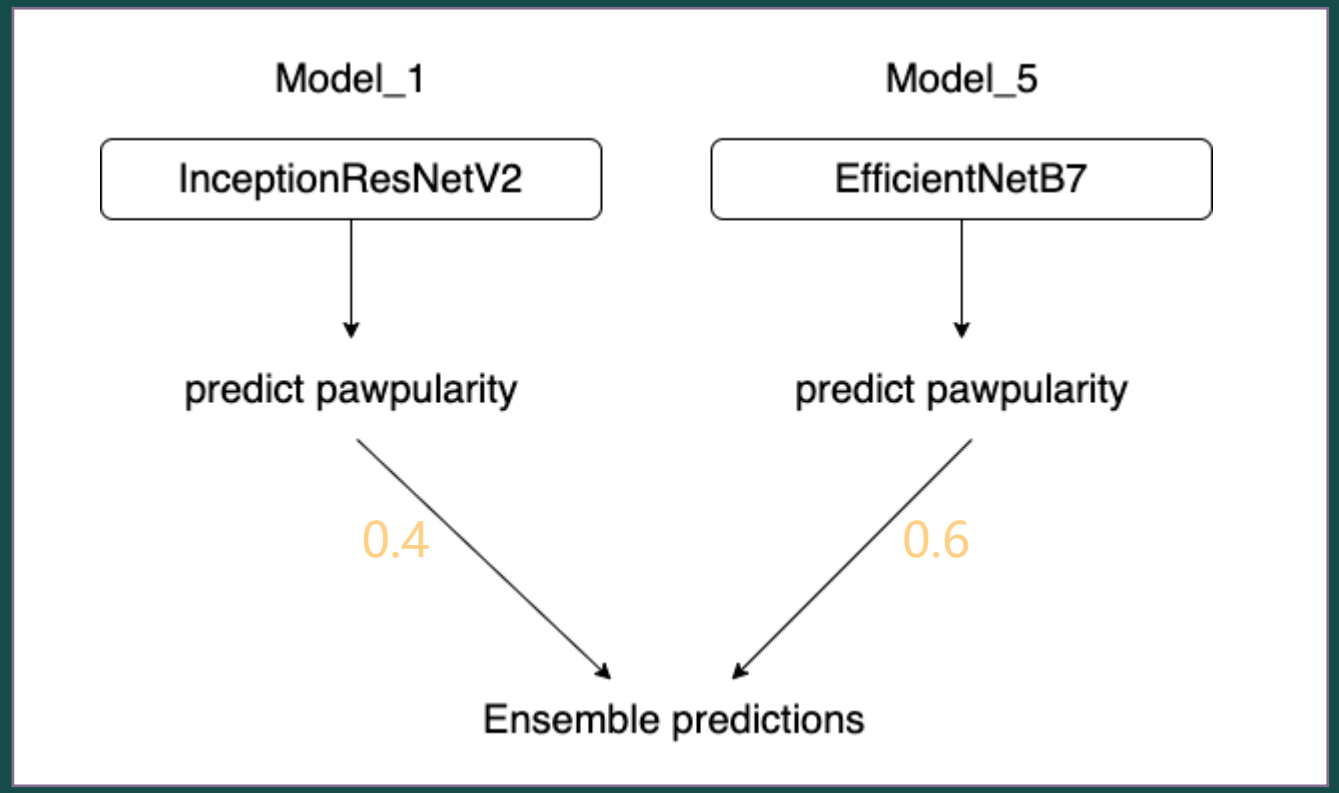
We use 6000 samples to run Network models, and all the sample for traditional models.

MODEL	RMSE	MAE
OLS	20.73824764611121	15.76785972406625
Ridge Regression	20.743712171549397	15.756259950364006
LASSO	20.75573688225275	15.767690402140891
Kernel Ridge Regression	20.717123104065067	15.736461514585494
Support Vector Regression	20.817379745431104	15.129932661840236
Random Forest Regression	20.716464671286573	15.739780845467504
PCA	20.718815193707346	15.747754263136668
Kmeans Cluster	37.62706965693371	31.336693548387096
MLP	21.64063389579775	15.9560591562589
Model_1(InceptionResNetV2)	20.965820018274336	15.996552220980327
Model_2(DenseNet201)	21.230656647422844	17.058412809371948
Model_3(ResNet152V2)	32.38418958619872	29.42764493306478
Model_4(ResNet50)	31.414984160993658	28.457210210164387
Model_5(EfficientNetB7)	20.718224338984903	16.092647444407145
Model_6(Xception)	27.282674789492592	24.244772679011028

Prediction

Prediction

To make our model more robust and accurate, we choose top2 performance network model to assemble the final predict model.



Thanks!

Team member

J.B.Zhang

S.X.Wang