31

100

0 - Data overview

train.csv: 9912 x 14 with 12 binary features

test.csv: unknown x 13

9912 RGB images in train folder

Features:

Id	Pawpularity	Subject Focus	Eyes	Face	Near	Action	Accessory	Group	Collage	Human	Occlusion	Info	Blur
Hex string	Integer in [1,100]	{0,1}	{0, 1}	{0,1}	{0, 1}	{0, 1}	{0, 1}	{0, 1}	{0, 1}	{0, 1}	{0, 1}	{0, 1}	{0, 1}

0 - Constraints

Test data not given, need to submit Kaggle notebook

GPU runtime ≤ 9 hours

- Don’t overcomplicate!

Metric:

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

0 - Project methodology

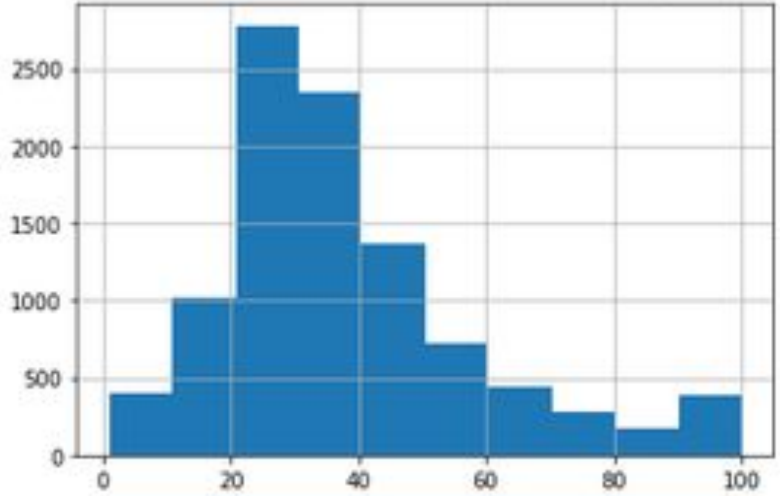
1. Exploratory data analysis
2. Simple models on tabular data
3. Models for score prediction from raw images

0 - Project methodology

1. Exploratory data analysis
2. Simple models on tabular data
    a. Why? Ease of interpretability
3. Models for score prediction from raw images

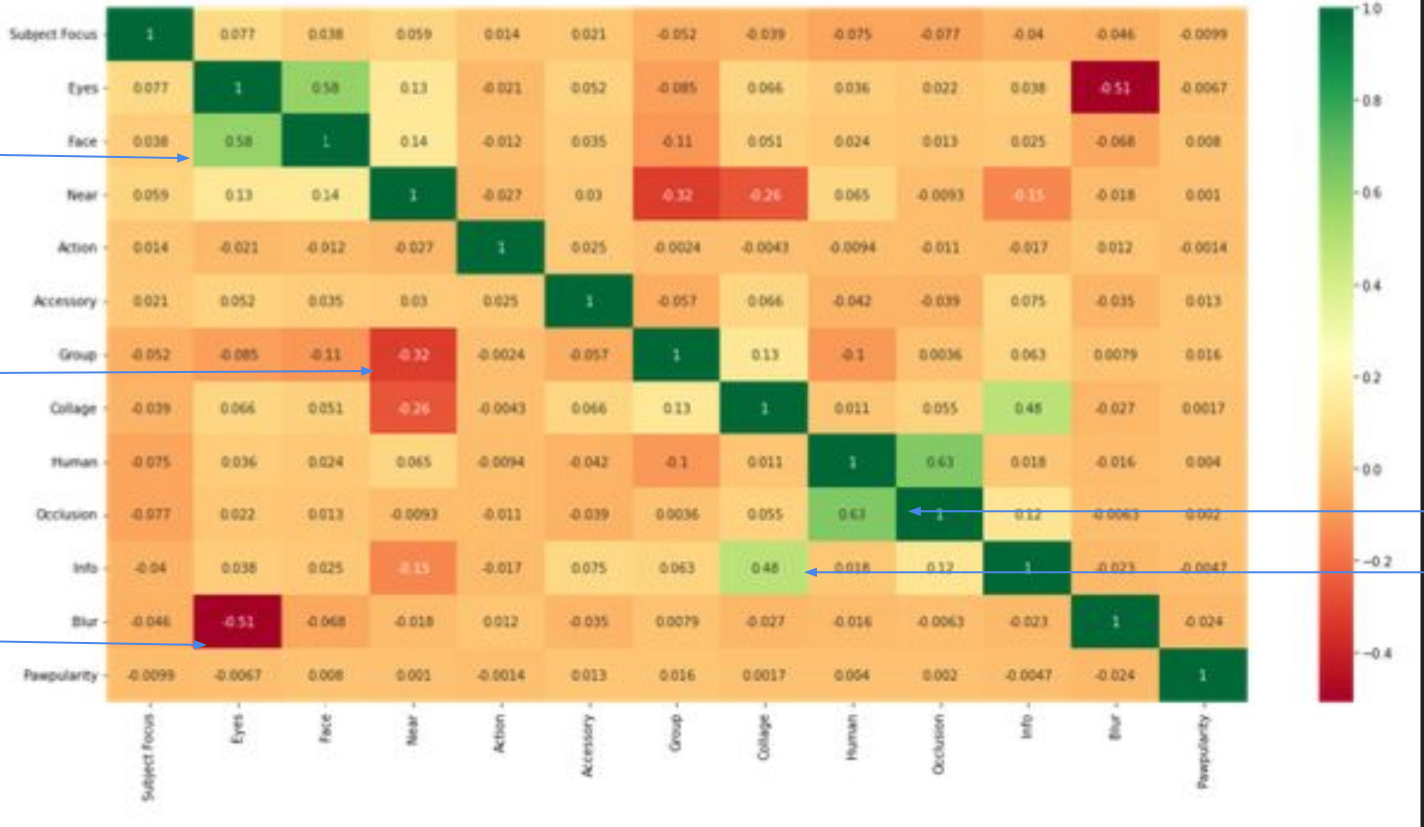
1 - Exploratory data analysis

Figure 1: Histogram of pawpularity scores (x=Scores, y=Number)



1 - Exploratory data analysis

Figure 2: Pairwise correlation heatmap of features



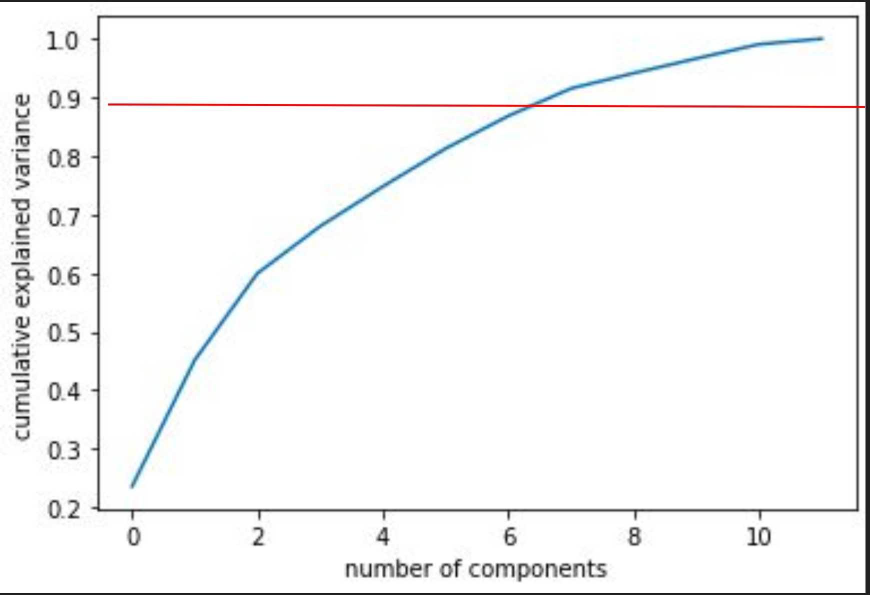
1 - Exploratory data analysis

Check VIF for each predictor < 10

	feature	VIF
0	Subject Focus	1.048292
1	Eyes	10.118170
2	Face	13.715668
3	Near	5.762924
4	Action	1.010174
5	Accessory	1.090942
6	Group	1.163850
7	Collage	1.452023
8	Human	2.064939
9	Occlusion	2.073562
10	Info	1.412621
11	Blur	1.595109

1 - Exploratory data analysis

Check VIF for each predictor < 10
PCA

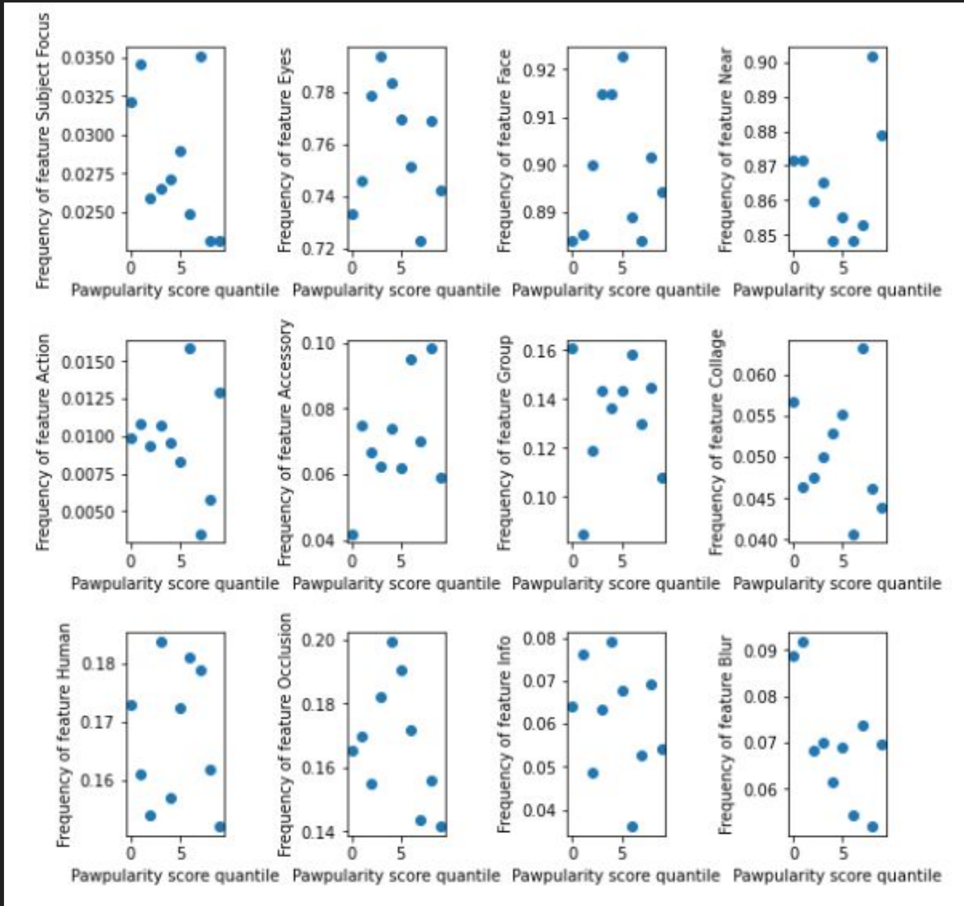


	feature	VIF
0	Subject Focus	1.048292
1	Eyes	10.118170
2	Face	13.715668
3	Near	5.762924
4	Action	1.010174
5	Accessory	1.090942
6	Group	1.163850
7	Collage	1.452023
8	Human	2.064939
9	Occlusion	2.073562
10	Info	1.412621
11	Blur	1.595109

1 - Exploratory data analysis

Separate into 10 score quantiles (1-10, 11-20, …)

- Since predictors are <u>binary</u>, plot <u>frequency of each predictor in each quantile</u>



2 - Simple models with tabular metadata

Approaches attempted

1. Logistic regression (as 100-class classification problem)
2. Linear regression
3. ElasticNet regression (L1 and L2 penalties)
4. Decision tree regressor
5. Random forest regressor (number of estimators = 50)

Assessment:

- Mean 5-fold cross validation RMSE

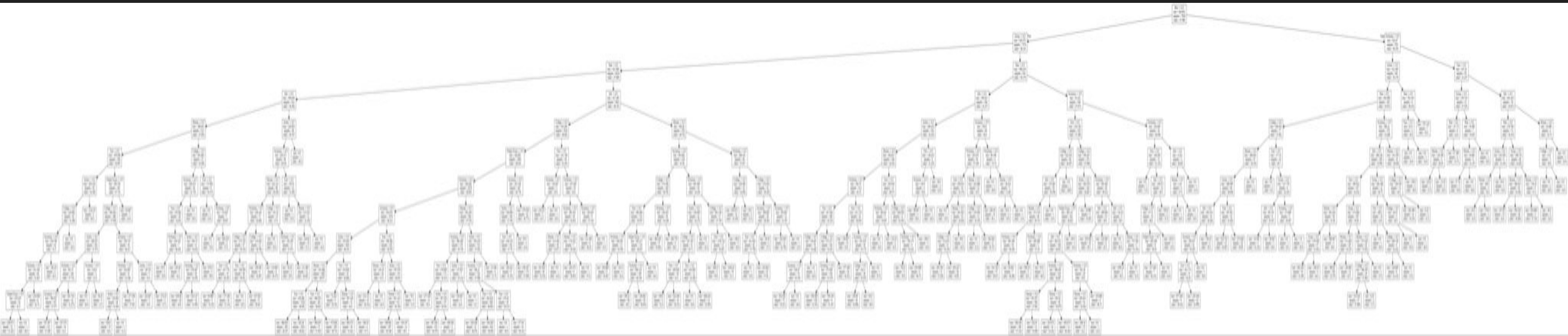
2 - Simple models with tabular metadata

Method	Logistic regression	Linear regression	ElasticNet	Decision tree regressor	Random forest regressor (DTR as base)
RMSE	23.386	20.600	20.589	20.854	20.763

2 - Simple models with tabular metadata

Base decision tree

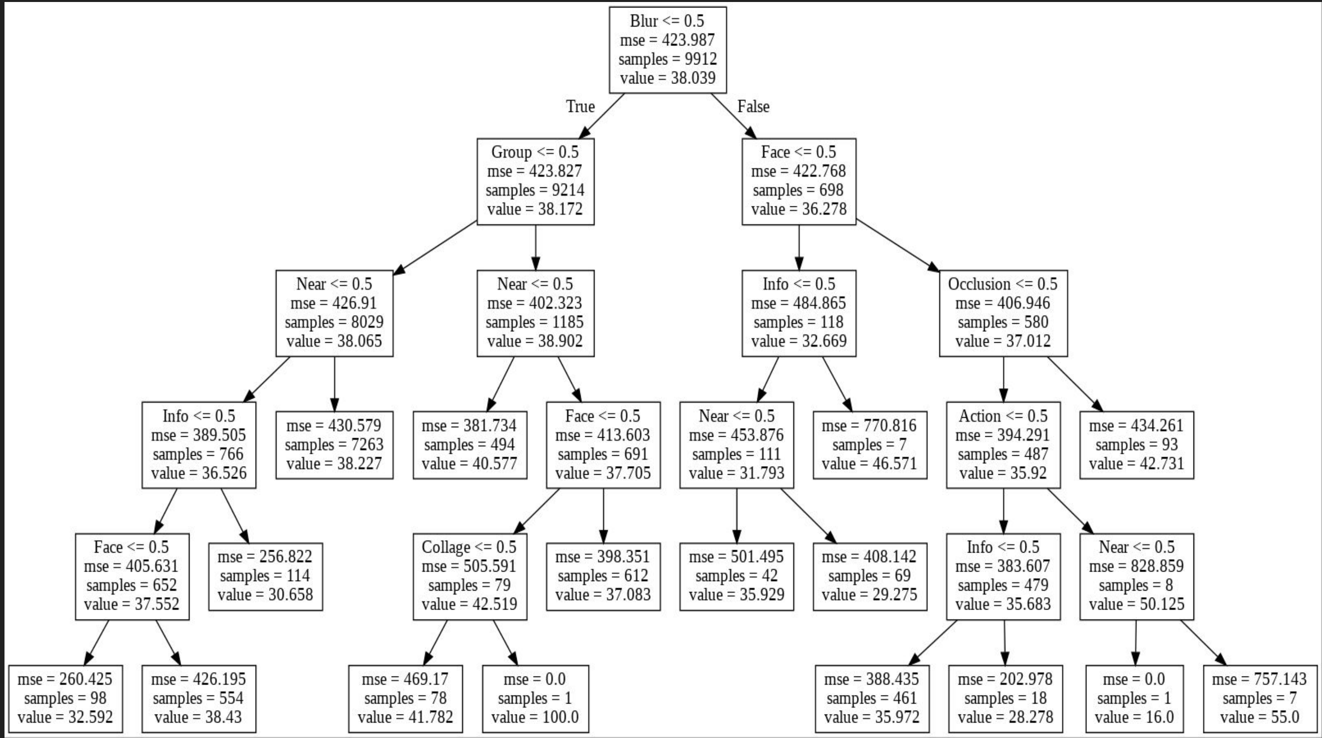
- Too many leaf nodes!



2 - Simple models with tabular metadata

Pruned decision tree

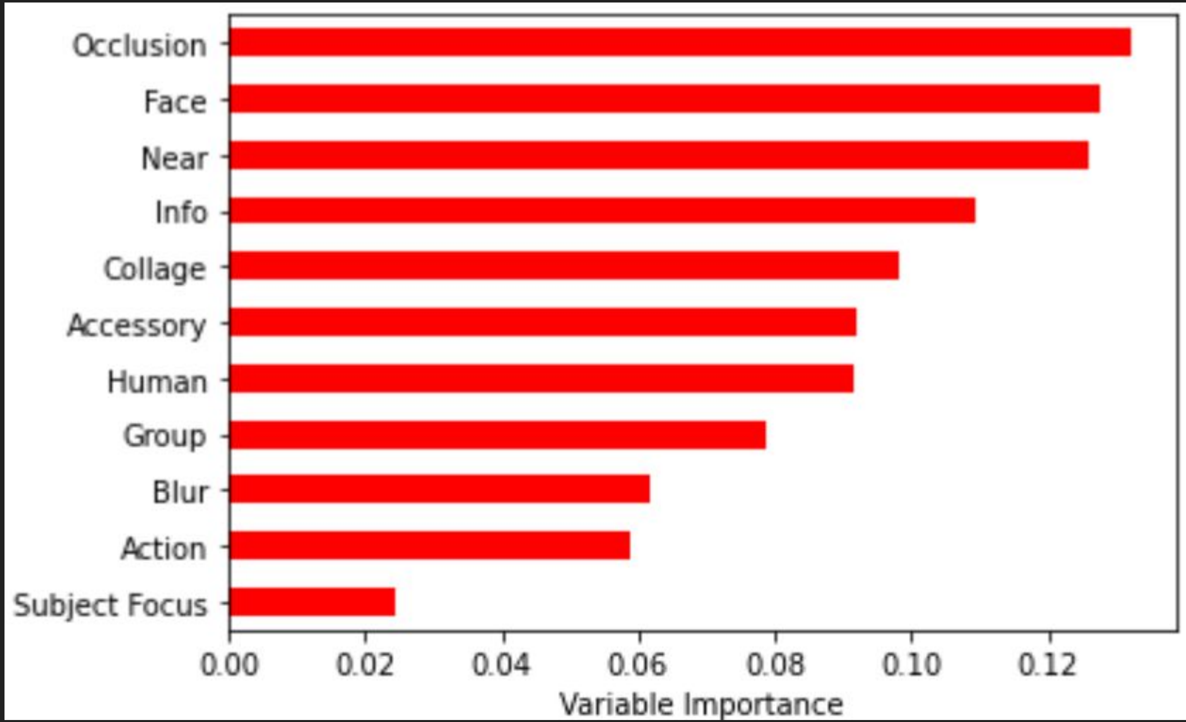
- alpha in [10^{-6}, 10^{-1}]
- Optimal alpha = 0.1



2 - Simple models with tabular metadata

Random forest regressor

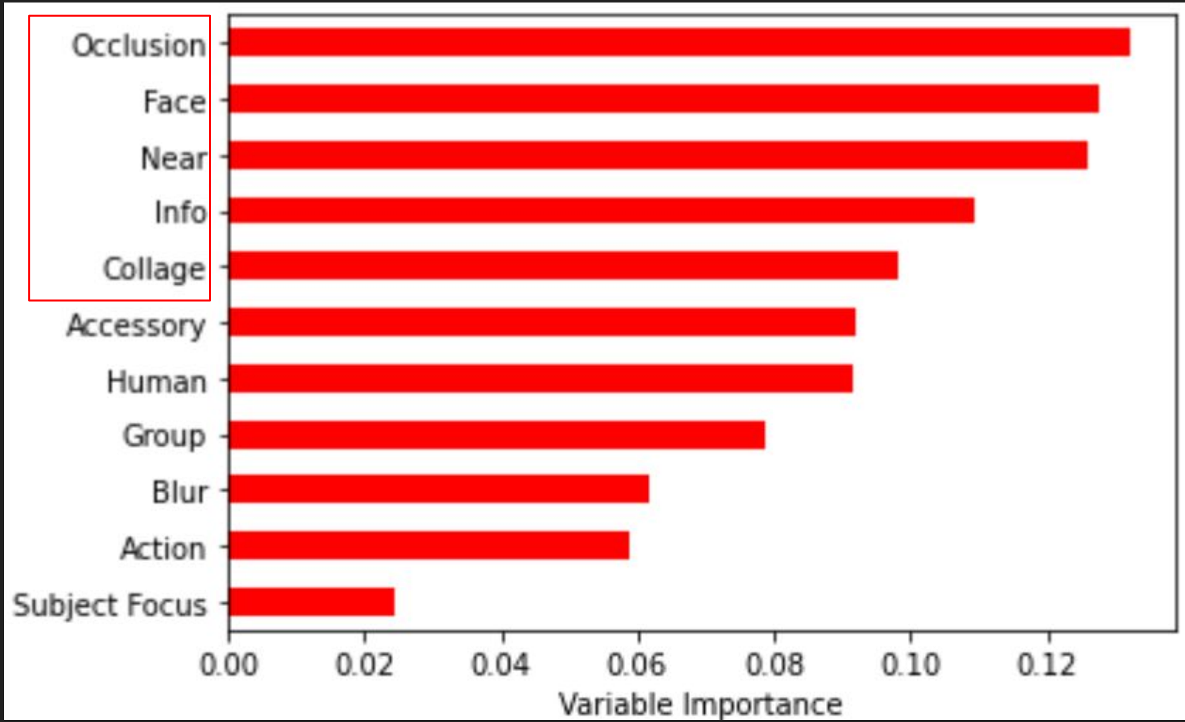
- Find most significant features
- Simplest model within 1SD of lowest mean 10-fold CV RMSE has features:



2 - Simple models with tabular metadata

Random forest regressor

- Find most significant features
- Simplest model within 1SD of lowest mean 10-fold CV RMSE has features:



2 - Simple models with tabular metadata

Method	Logistic regression	Linear regression	ElasticNet	Decision tree regressor	Random forest regressor (DTR as base)
RMSE	23.386	20.600	20.589	20.854	20.763
				20.666	20.630

Other simple model adjustments:

1. Random forest with number of estimators = 300 (RMSE: 20.623)
2. XGBoost (RMSE: 20.812)

- Adding more estimators in sequence would only lead to overfitting

2 - Simple models with tabular metadata

The end of the line for simple models?

Formulation of our problem:

- As regression: the output range of values should be constrained, but it isn’t
- As classification: but some classes are more similar than others

2 - Simple models with tabular metadata

Ordinal regression

learning a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ from data $(X_1, Y_1), ..., (X_n, Y_n)$

where $X_i \in \mathcal{X}$, $Y_i \in \mathcal{Y} = \{1, 2, ..., k\} \forall i \in \{1, ..., n\}$ such that the average loss L over all (X, Y) pairs (i.e. $\mathbb{E}_{X \times Y}[L(Y, h(X))]$) is minimised. [2]

Loss differs from cross-entropy

- “5” misclassified as “6” less wrong than “5” misclassified as “12”
- But in cross entropy, same penalty

2 - Simple models with tabular metadata

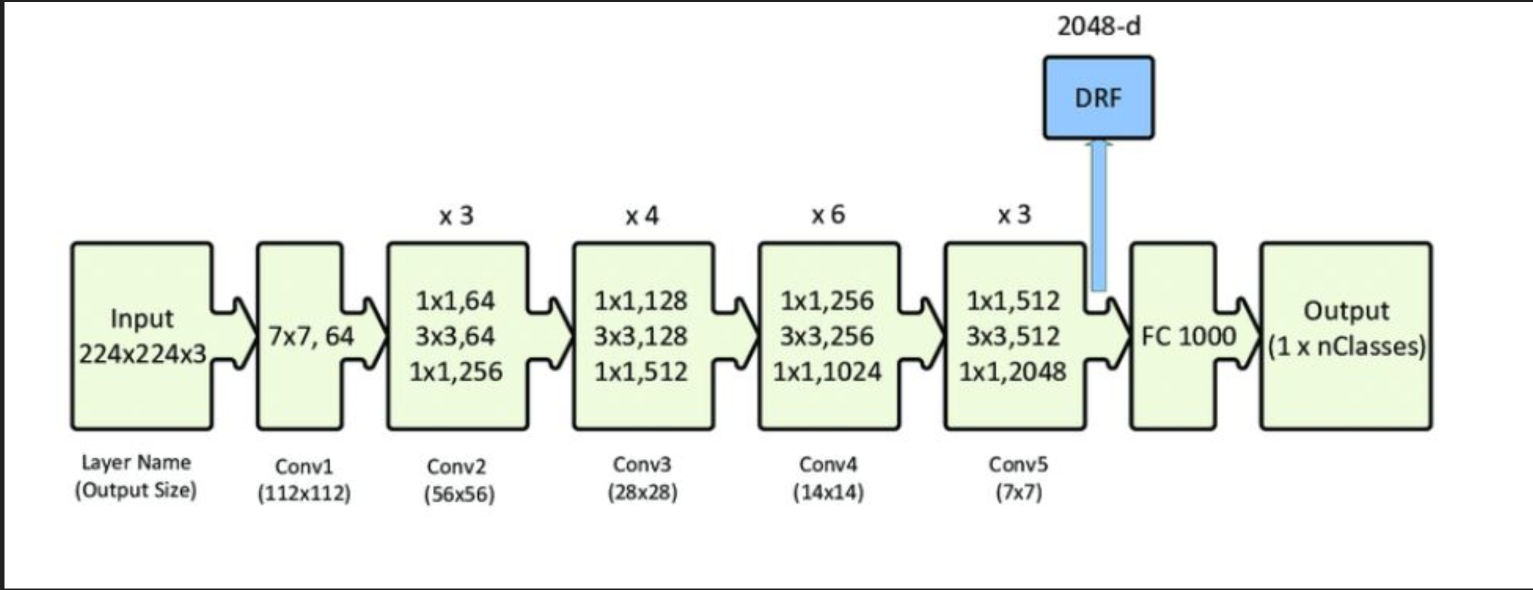
Method	Ordinal logistic regression	Linear ordinal ridge regression	Logistic regression	Linear regression	ElasticNet	Decision tree regressor	Random forest regressor (DTR as base)	XGBoost
RMSE	21.182	20.601	23.386	20.600	20.589	20.623	20.630	20.812

Other simple model adjustments:

1. Random forest with number of estimators = 300 (RMSE: 20.623)
2. XGBoost (RMSE: 20.812)

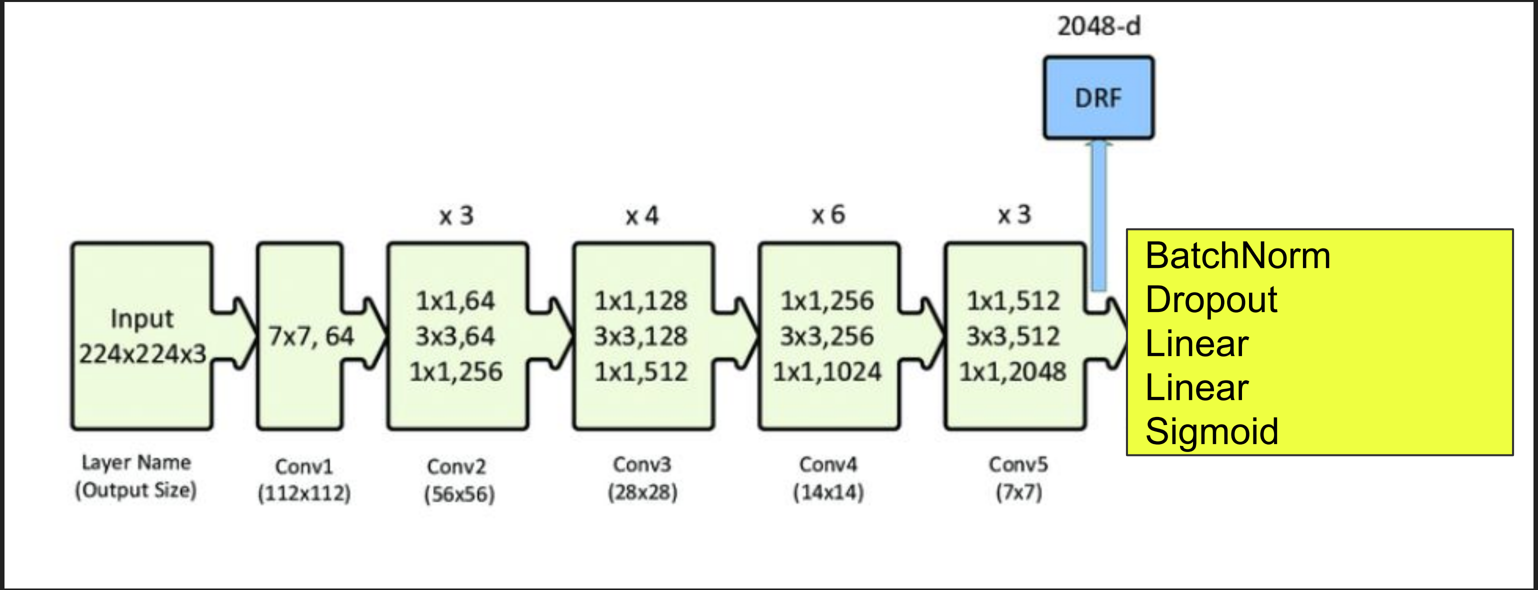
- Adding more estimators in sequence would only lead to overfitting

3 - CNN with images



3 - CNN with images

CNN architecture: ResNet
BatchNorm: increased network training speed and stability
Dropout: increases generalization
Sigmoid: maps all outputs into [0,1] which can be rescaled to [1,100]



3 - CNN with images

Data preprocessing and augmentation

Hypothesise that significant cropping, random erasure, blur or colour jitter has unpredictable and adverse effect on Pawpularity

- Limit augmentation to horizontal flips on-the-fly to save RAM
- Resize to 224x224x3
- Normalize input channels of each images

3 - CNN with images

Training configuration

- Stratified 80% train, 20% validation
- 20 epochs with early stopping
- MSE loss
- SGD, learning rate={various}, L2 regularization 0.001, momentum=0.9
- dropout=0.5

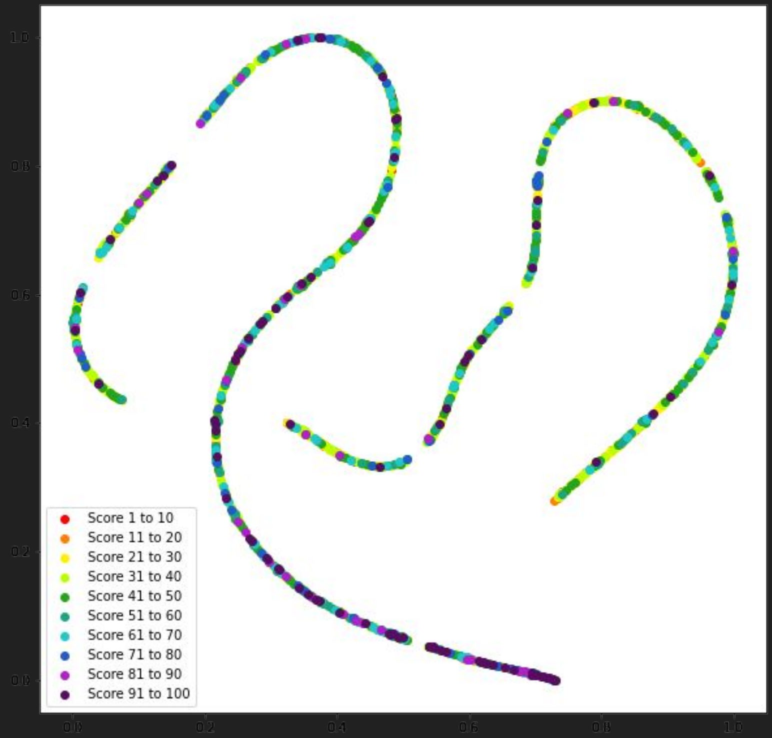
Why?
Empirically, research has shown dropout + L2 regularization + high momentum = best performance

3 - CNN with images

Model/Parameters	Learning rate = 0.01	Learning rate = 0.001	Learning rate = 0.0001
ResNet-18	21.1673	18.5846	18.6221
ResNet-50	19.8544	*17.9965*	18.0132

3 - CNN with images

Model output visualisation (t-SNE)



3 - CNN with images

Training configuration

- Due to imbalance in score quantiles, improve generalizability with distribution-aware (weighted) RMSE loss function proposed by Yang et al.
- Computes average proportion of each score quantile in the training set, then use its inverse as weight
  - Weighted RMSE vs RMSE similar to LDAM vs cross-entropy

Let p_i $\forall i \in \{0, ..., 9\}$ denote the proportion of pets with scores in range $[10i + 1, 10(i + 1)]$. Let $m = max\{p_0, ..., p_9\}$. Our weighted RMSE loss is $RMSE_{weighted} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \frac{m}{p_i}(y_i - \hat{y}_i)^2}$.

3 - CNN with images

Training configuration for new model

- Stratified 80% train, 20% validation
- 20 epochs with early stopping
- Distribution-aware RMSE loss
- SGD, learning rate={various}, L2 regularization 0.001, momentum=0.9
- dropout=0.5

3 - CNN with images

Training configuration for new model

- Model A (MSE loss, learning rate = 0.001): 17.9965

Model/Parameters	Learning rate = 0.01	Learning rate = 0.001 (Model B)	Learning rate = 0.0001
ResNet-50	17.1055	*13.3793*	13.6822

3 - CNN with images

Kaggle test set results

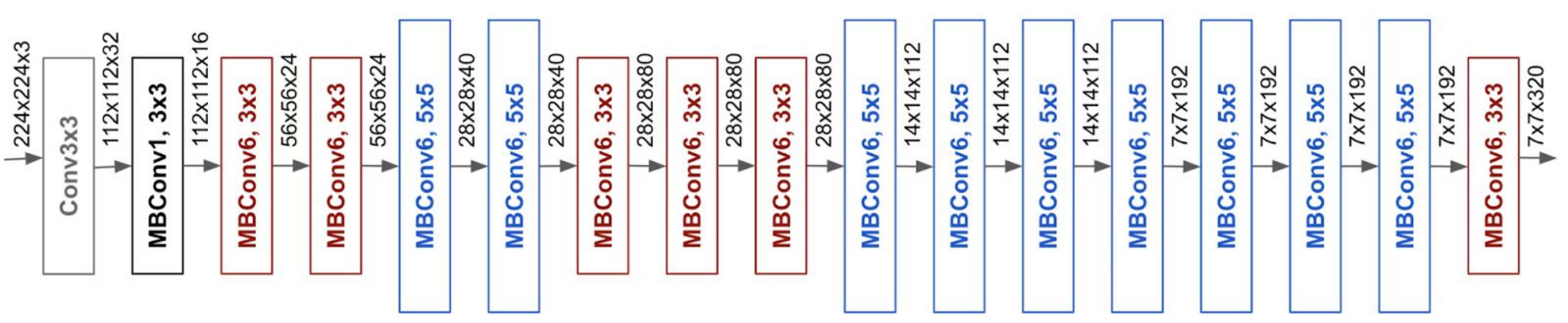
- Not great…
- Perhaps alternative CNN structures?
    - EfficientNet

Model/Parameters	Model A	Model B
Test set RMSE	20.4521	*19.11171*

3 - CNN with images

Why EfficientNet?

- Grid search for optimal scaling coefficient for each dimension of network
- MBConv blocks computationally cheap
  - Turns convolution into depthwise THEN pointwise
- Linear bottleneck in final layer of each block



3 - CNN with images

EfficientNet training configuration (Model C)

- Stratified 80% train, 20% validation
- 20 epochs with early stopping
- MSE loss
- Adam, learning rate=0.001 (Adam converges faster)
- dropout=0.2

Note: Validation RMSE continuing to decrease
- Suggesting further improvement possible

Model/Parameters	Model A	Model B	Model C
Best validation RMSE	17.9965	*13.3793*	14.8244

3 - CNN with images

Kaggle test set results

- Model C outperformed all previous models, even without using weighted RMSE!
- Model C with weighted RMSE may perform even better, but didn't submit due to time constraints

Model/Parameters	Model A	Model B	Model C
Test set RMSE	20.4521	19.1117	*18.0441*

4 - Future analysis

How to improve?

- Build CNN for automatically labelling a picture’s metadata
  - May help us associate certain features with Pawpularity!

4 - Future analysis

How to improve?

- Build CNN for automatically labelling a picture’s metadata
  - May help us associate certain features with Pawpularity!
- Separate Pawpularity by animal species and breed (collect more info)

4 - Future analysis

How to improve?

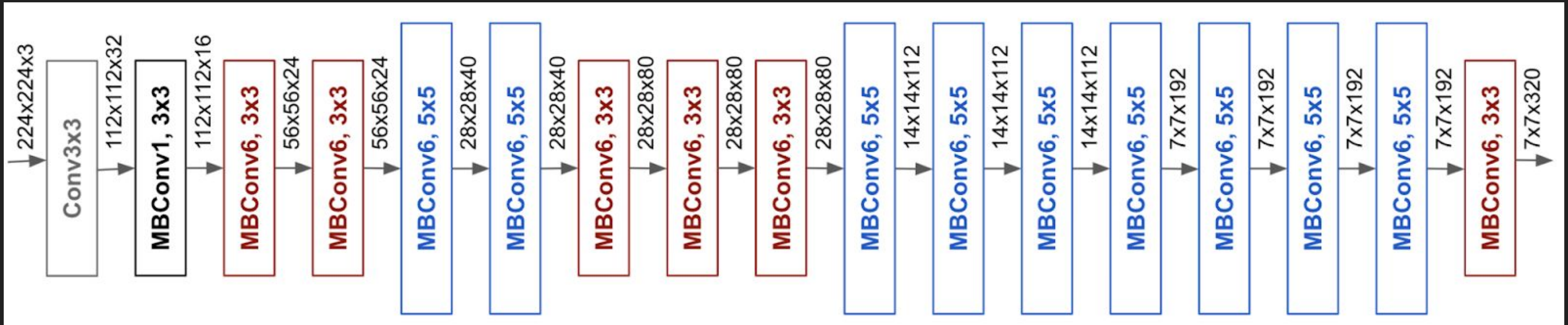
- Build CNN for automatically labelling a picture’s metadata
  - May help us associate certain features with Pawpularity!
- Separate Pawpularity by animal species and breed (collect more info)
- Rigorous search of hyperparameter space (eg cross-validation with NN)
- CNN ensembling

4 - Future analysis

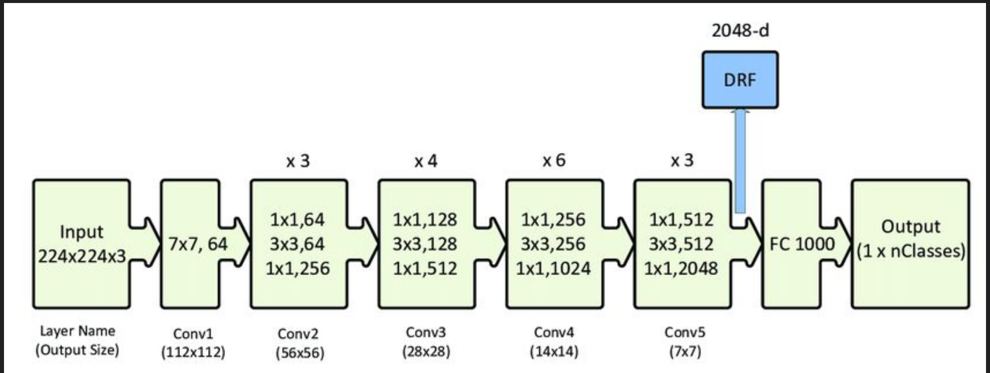
How to improve?

- Build CNN for automatically labelling a picture's metadata
  - May help us associate certain features with Pawpularity!
- Separate Pawpularity by animal species and breed (collect more info)
- Rigorous search of hyperparameter space (eg cross-validation with NN)
- CNN ensembling
- Don't use CNN for feature extraction! Vision transformer?
  - Encode each image as sequence of patches
  - Feed into encoder
  - Self-attention detects relationships between patches

5 - Conclusion



>




Thank you!