Regression with an Abalone Dataset

1. Overview

Problem Definition

The goal of this competition is to predict the age of abalone from various physical measurements.

Competition webpage

https://www.kaggle.com/competitions/playground-series-s4e4/overview

Metric

The evaluation metric for this competition is RMSLE (Root Mean Squared Logarithmic Error).

Data

The dataset for this competition (both train and test) was generated from a deep learning model trained on the Abalone dataset.

Strategy for solving the problem

1. Analyze the data;
2. Develop several preprocessing pipelines for the data;
3. Train various tree-based and tabular neural network-based models;
4. Ensemble best-performing models.

2. Training Data

train.csv

Source code

`01_EDA.ipynb` – jupyter notebook with charts and statistics on the dataset,
`DataExplorer.py` – source code for EDA.

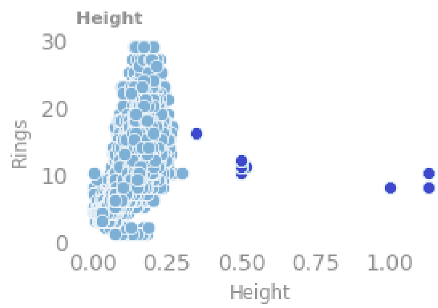
Features

7 numerical features: `Length, Diameter, Height, Whole weight, Whole weight.1, Whole weight.2, Shell weight.`

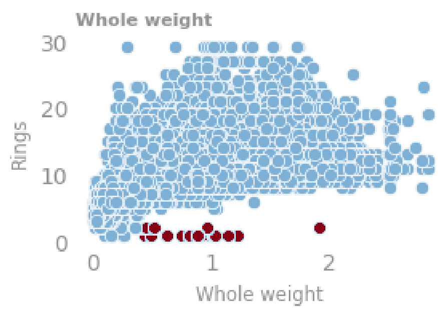
1 categorical feature: `Sex.`

Insights

- All numerical features are highly correlated, pairwise correlation is >0.9;
- All numerical features are correlated to target, correlation values are in the range 0.5-0.7;
- `'Height'` has several datapoints with values much higher than the majority:



- There is a small subset of data with target values of 1 and 2 that is very different from the rest of the data:



- Distribution of data is very similar for the train and test datasets;
- `'Sex'` has 3 unique values, similarly distributed:

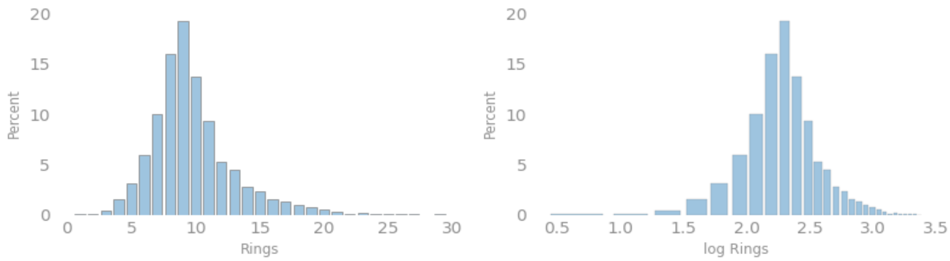
Sex	Train count	Test count
I	33093	22241
M	31027	20783
F	26495	17387

- Quadratic relationship between linear and weight features;
- Distributions for datapoints with feature `'Sex'='I'` are different from `'Sex'≠'I'`;
- Distributions for datapoints with feature `'Sex'='M'` and `'Sex'='F'` are very similar;
- Subset of `'Sex'='I'` is better correlated to target (values are in the range 0.66-0.76, while for the subset `'Sex'≠'I'` values are in the range 0.2-0.5).

Target

Target feature `'Rings'`:

- Integer, values from 1 to 29, value 28 is missing;
- Distribution is skewed;
- Log-distribution is close to normal.



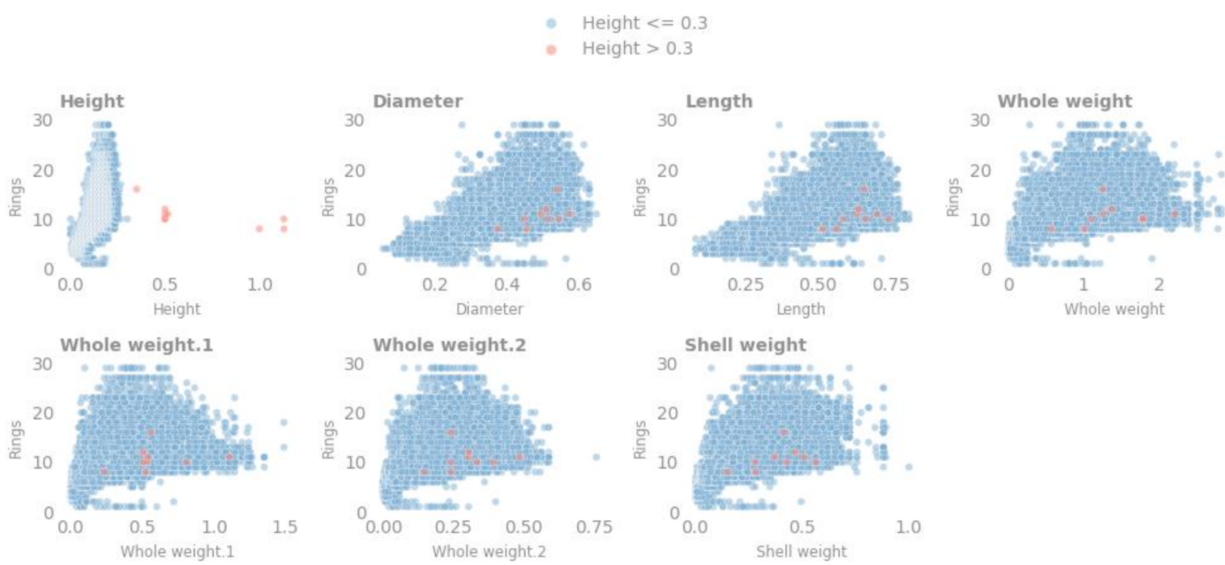
Outlier detection

Source code

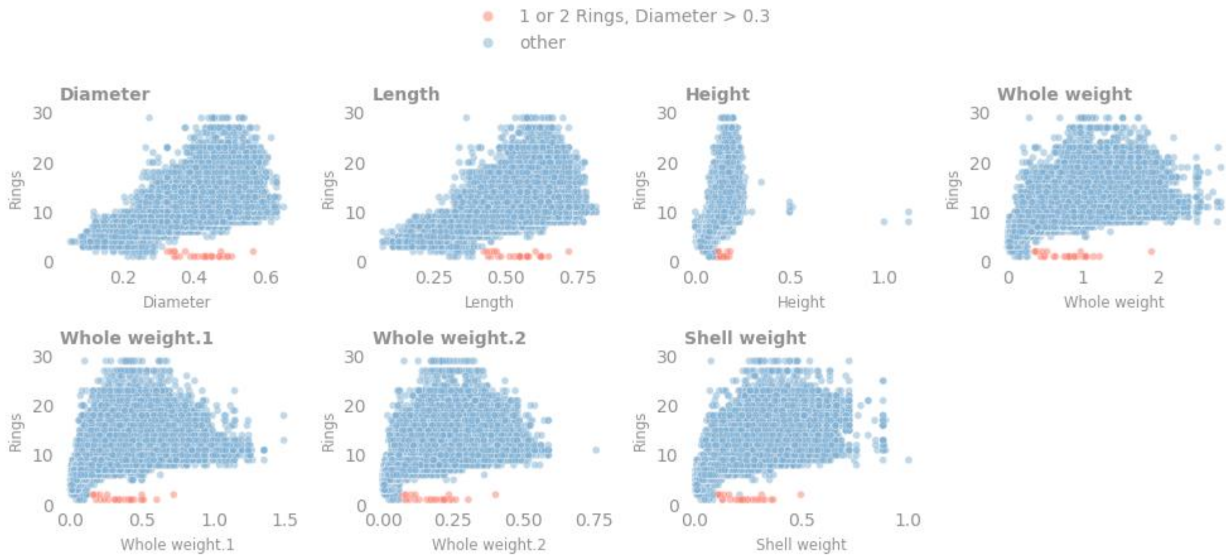
`02_outliers.ipynb` – jupyter notebook for analyzing outliers.

Manual outlier detection

Several datapoints have very high values of the `'Height'` feature. However, those datapoints are within distributions of other features.



There is a small subset of datapoints that are very different from the majority of the data.



Automatic outlier detection

Methods:

- Isolation forest;
- Local outlier factor;
- cleanlab's OutOfDistribution.

Automatic methods haven't identified any definitive outliers: there are no datapoints that all three methods identified as outliers.

3. Explored models

Ensemble of untuned models

Source code

`07_ensemble_untuned.ipynb` – notebook to test several tree-based models.

Results

The following models were fitted on the training dataset. Cross validation score for each model is in the table below:

	Mean RMSLE	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4
lgb_300_depth5	**0.1489**	0.1486	0.1492	0.1497	0.1494	0.1474
catboost	**0.1490**	0.1488	0.1494	0.1497	0.1496	0.1474
HistGradientBoosting	**0.1495**	0.1493	0.1499	0.1499	0.1502	0.1482
random_forest	**0.1498**	0.1495	0.1503	0.1505	0.1504	0.1483
xgb_100_depth5	**0.1500**	0.1498	0.1504	0.1507	0.1504	0.1489
extra_trees	**0.1506**	0.1502	0.1510	0.1512	0.1512	0.1493
knn_50	**0.1547**	0.1545	0.1555	0.1547	0.1554	0.1534

After fitting a linear regression model on out-of-fold predictions, the following weights were obtained for the ensemble:

	Weights
xgb_100_depth5	0.0283
lgb_300_depth5	**0.3808**
catboost	**0.3246**
HistGradientBoosting	0.0352
random_forest	**0.2354**
extra_trees	0.0000
knn_50	0.0000

Ensemble OOF score:
0.1484

Autogluon on 5 folds

Source code

`06_autogluon.ipynb` – jupyter notebook with fitting Autogluon,
`autogluon_wrapper.py` – wrapper class for Autogluon predictions on folds.

Results

The following scores were obtained for the Autogluon predictions:

	Mean RMSLE	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4
Autogluon	**0.1478**	0.1475	0.1482	0.1487	0.1483	0.1465

Base models with the best scores are LightGBM, CatBoost and Random forest, same as with manually assembled ensemble of untuned models.

Neural networks for tabular data

Source code

`abalone_dataset.py` – PyTorch dataset class for current tabular data,
`tabular_nn_tuner.py` – wrapper class for training PyTorch tabular data models,
`*_model.py` – PyTorch models for tabular data (6 different architectures),
`08_NN_models.ipynb` – jupyter notebook for experiments with various neural network architectures,
`09_compare_all_models.ipynb` – jupyter notebook for detailed results.

Results

For the experiments with neural network architectures, the training data is scaled using sklearn's StandardScaler, the categorical feature is one-hot-encoded, and the target is log1p-transformed. MSELoss and Adam optimizer are used.

The performance of all NN models is worse than the performance of the tree-based models, with the best scores of NN models on fold 0 being ~0.151, while untuned LightGBM or CatBoost scores are ~0.149.

Examples of results on fold 0, with some statistics of predictions:

	Min ***($min_{target}=1$)***	***Mean***	***Max*** ***($max_{target}=29$)***	***RMSLE***
saint_8_4	4.1256	9.6375	18.1002	**0.1514**
ft_8_8	3.9785	9.6136	18.5454	**0.1518**
tab_40_2	3.8907	9.4779	17.9753	**0.1519**
autoint_20_2	4.2402	9.6305	18.9098	**0.1524**
tabpfn_16_32	4.6765	9.6352	19.2457	**0.1584**

As the distributions of predictions are different for all architectures, the predictions of NN models can be used as part of ensemble.

Insights from experiments:

1. NODE architecture performs much worse than other architectures and is not included in further analysis,
2. Increasing the complexity of each architecture does not improve the score significantly,
3. Ensembling all trained NN models with weights obtained from linear regression model shows that all non-zero weights belong to models with different architectures.
4. Ensembling all trained NN models with LightGBM and CatBoost (weights obtained from linear regression model) shows that only one NN model has non-zero weight.

4. Solution

Ensemble of tuned LightGBM, Catboost and Neural Network model

Source code

`08_NN_models.ipynb` – jupyter notebook for training NN model on 5 folds,
`10_tune_lgb_catboost.ipynb` – jupyter notebook for tuning hyperparameters of LightGBM and CatBoost models using Weights & Biases sweeps,
`11_ensemble_on_folds.ipynb` – jupyter notebook for ensembling the best models on each fold,
`ft_transformer_model.py` – source code for FTTransformerModel class (PyTorch implementation).

Inference

The solution consists of an ensemble of 15 models in total: for each of the 5 folds three models are tuned:

1. LightGBMRegressor,
2. CatboostRegressor,
3. FTTransformer.

The five folds are obtained using sklearn's StratifiedKFold as the target is discrete.

Training data for each fold is concatenated with the original dataset (see section 1. Overview).

LightGBMRegressor and CatboostRegressor are chosen as the best-scoring single models for the current dataset. Ensembling these two models provides a very good score both for out-of-fold predictions and predictions on the test set.

FTTransformer with 8 hidden dimensions and 4 layers is chosen from the set of NN architectures as a lightweight (~4000 tunable parameters) NN-model. The experiment shows that including an NN-model in the final ensemble improves the CV score. FTTransformer has shown similar results to SAINT and TabTransformer architertures and was chosen arbitrarily.

Weights on each fold are obtained by fitting a linear regression model on out-of-fold preditions.

The scores are:

	RMSLE			*Ensemble*	*Ensemble*
	LightGBM	*CatBoost*	*FTTransformer*	*LightGBM+CatBoost*	*3 models*
Fold 0	0.1477	0.1482	0.1505	0.1475	0.1474
Fold 1	0.1483	0.1489	0.1514	0.1480	0.1480
Fold 2	0.1487	0.1493	0.1520	0.1485	0.1485
Fold 3	0.1488	0.1491	0.1523	0.1485	0.1485
Fold 4	0.1467	0.1471	0.1496	0.1465	0.1464
Mean	0.1480	0.1485	0.1512	*0.14780*	***0.14776***

	Weights obtained with linear regression (2-model ensemble)		*Weights obtained with linear regression (3-model ensemble)*		
	LightGBM	*CatBoost*	*LightGBM*	*CatBoost*	*FTTransformer*
Fold 0	0.63	0.37	0.60	0.25	0.15
Fold 1	0.65	0.35	0.65	0.23	0.12
Fold 2	0.66	0.34	0.64	0.25	0.11
Fold 3	0.59	0.41	0.58	0.39	0.03
Fold 4	0.60	0.40	0.57	0.32	0.11

Predictions on the test dataset are obtained as follows: predictions from models trained on each fold are averaged with weights obtained with linear regression model (resulting in 5 predictions), and then averaged with uniform weights.

Results

	RMSLE test dataset	*RMSLE out-of-fold*
2-model ensemble	0.14579	0.14780
3-model ensemble	0.14581	0.14776
Autogluon ensemble	0.14589	0.14784

The impact of the FTTransformer on the out-of-fold score is very small. On the test set the ensemble without FTTransformer performs slightly better, so for simplicity FTTransformer can be omitted from the ensemble.

Manually curated ensembles perform a little better than the Autogluon ensemble. In future Autogluon can be used to quickly obtain baseline score.

5. Approaches that didn't work

PCA

Source code

`03_pca_pipelines.ipynb` – jupyter notebook for PCA with various feature transformations.

Results

During 5-fold cross-validation experiments the regression model was fixed to the following (LightGBM model was chosen as the best-performing single model):

```
LGB(random_state=42, n_estimators=300, verbose=-1)
```

However, none of the data preprocessing transformations that included PCA improved the score:

	Mean RMSLE	***Fold 0***	***Fold 1***	***Fold 2***	***Fold 3***	***Fold 4***
original features	**0.1489**	0.1488	0.1491	0.1497	0.1497	0.1474
concat PCA after sqrt Weight features	**0.1500**	0.1497	0.1507	0.1507	0.1505	0.1486
concat PCA after sqr linear features	**0.1501**	0.1498	0.1505	0.1504	0.1512	0.1484
concat PCA	**0.1501**	0.1499	0.1505	0.1504	0.1509	0.1486
PCA after sqr linear features	**0.1521**	0.1521	0.1524	0.1525	0.1531	0.1504
PCA	**0.1524**	0.1523	0.1527	0.1526	0.1533	0.1510
PCA after after sqrt Weight features	**0.1526**	0.1520	0.1535	0.1531	0.1535	0.1507

Feature engineering

Source code

`04_feature_engineering.ipynb` – jupyter notebook with new feature creation and selection.

Results

After manually creating 18 new features, several methods of feature selection were employed:

1. SelectKBest,
2. RFECV,
3. Boruta.

During 5-fold cross-validation experiments the regression model was fixed to the following (LightGBM model was chosen as the best-performing single model):

```
LGB(random_state=42, n_estimators=300, verbose=-1)
```

However, none of the new features improved the score from original features:

	Mean RMSLE	***Fold 0***	***Fold 1***	***Fold 2***	***Fold 3***	***Fold 4***
original features	**0.1489**	0.1488	0.1491	0.1497	0.1497	0.1474
SelectKBest	**0.1492**	0.1490	0.1496	0.1501	0.1498	0.1475
RFECV	**0.1499**	0.1494	0.1504	0.1506	0.1508	0.1484
Boruta	**0.1505**	0.1500	0.1512	0.1511	0.1514	0.1489

Training separate models on subsets of the training dataset

Source code

`05_baseline.ipynb` – jupyter notebook with untuned models,
`06_autogluon.ipynb` – jupyter notebook with fitting Autogluon.

Results

During 5-fold cross-validation experiments LightGBM and XGBoost regression models were fitted to each subset of the training dataset. However, the best score is achieved on the whole dataset for both models:

	Mean RMSLE	**Fold 0**	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**
LGB original	**0.1492**	0.1490	0.1493	0.1499	0.1497	0.1480
LGB split 2 subsets: 'Sex'='I' | 'Sex'≠'I'	**0.1495**	0.1493	0.1497	0.1501	0.1507	0.1477
XGB original	**0.1503**	0.1502	0.1505	0.1510	0.1505	0.1494
LGB split 3 subsets: 'Sex'='I' | 'Sex'='F' | 'Sex'='M'	**0.1506**	0.1502	0.1508	0.1513	0.1514	0.1493
XGB split 2 subsets: 'Sex'='I' | 'Sex'≠'I'	**0.1508**	0.1507	0.1509	0.1516	0.1519	0.1490
XGB split 3 subsets: 'Sex'='I' | 'Sex'='F' | 'Sex'='M'	**0.1518**	0.1516	0.1518	0.1525	0.1528	0.1503

During the experiments with Autogluon on 5 folds the best score was achieved on the whole dataset (and this is also true for each subset):

	'Sex'='I'	**'Sex'≠'I'**
AutoGluon on all data	0.1336	0.1555
AutoGluon on each subset	0.1341	0.1557