Machine learning with Galaxy

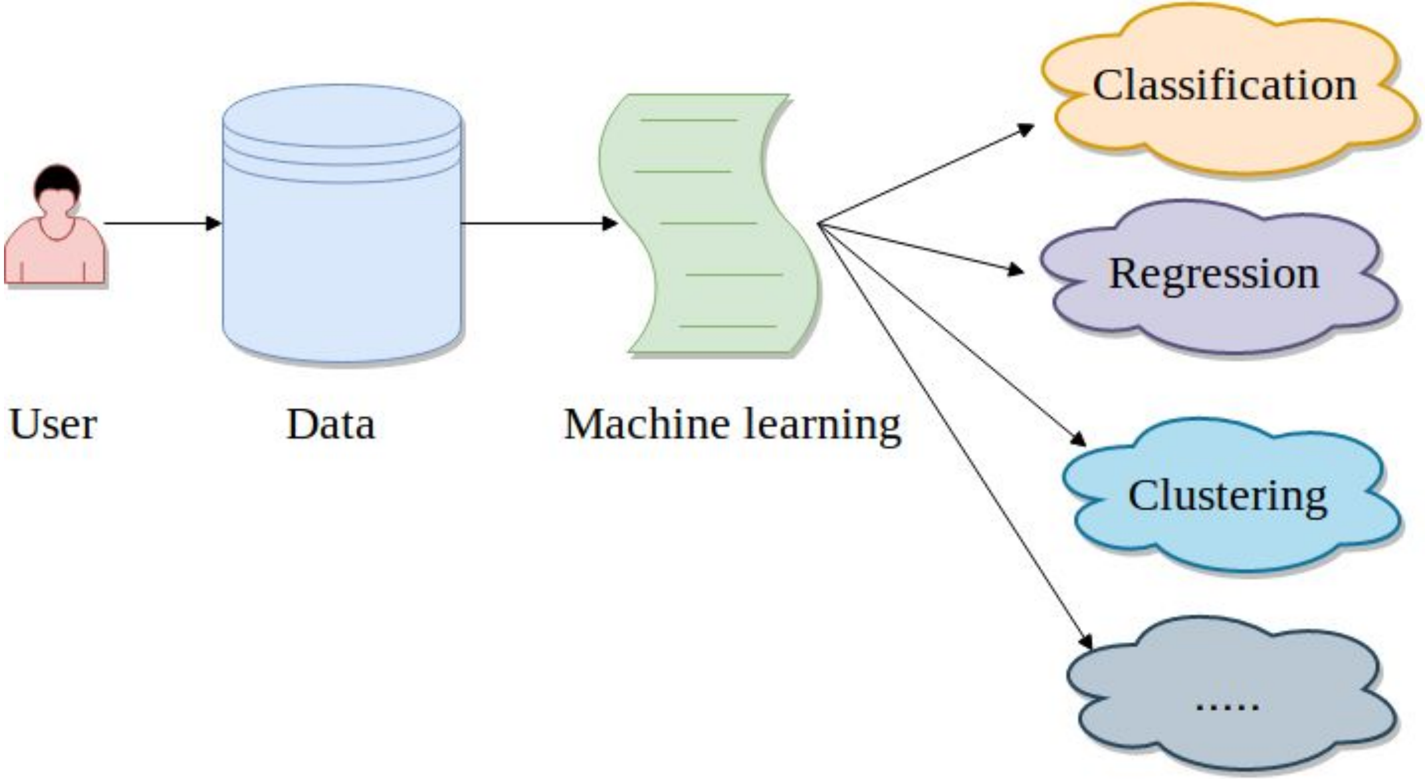
Qiang Gu and Anup Kumar

European Galaxy Days, 2018
Faculty of Engineering,
Freiburg

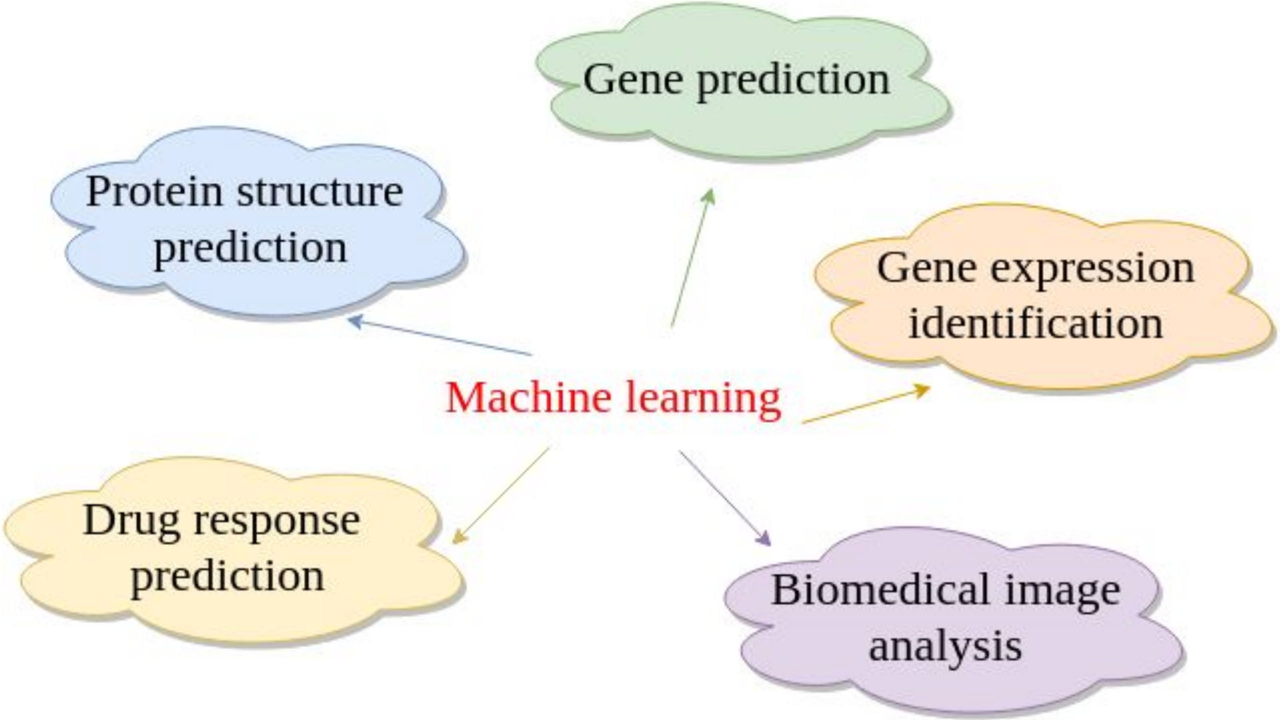
Outline

- Machine learning
- Significance in Bioinformatics
- Galaxy’s machine learning tools
- Use-case: Regression
    - Penn Machine Learning Benchmark datasets

Machine learning



Application in Bioinformatics



Gene prediction

Protein structure prediction

Gene expression identification

Machine learning

Drug response prediction

Biomedical image analysis

Scikit-learn in Galaxy

- Scikit-learn modules as Galaxy tools
- Scikit-learn:
    - Open source, python
    - 89 releases
    - 1,200+ contributors
    - 15,000+ forks
    - 30,000+ stargazers

Various machine learning tools in Galaxy

- 20+ classifiers (classification)
- 20+ regressors (regression)
- 30+ data preprocessing techniques
- 2 hyperparameter search approaches
  - Grid search
  - Random search
- 2+ plotting tools
  - ROC, AUC, confusion matrix
  - True vs predicted, residual plot

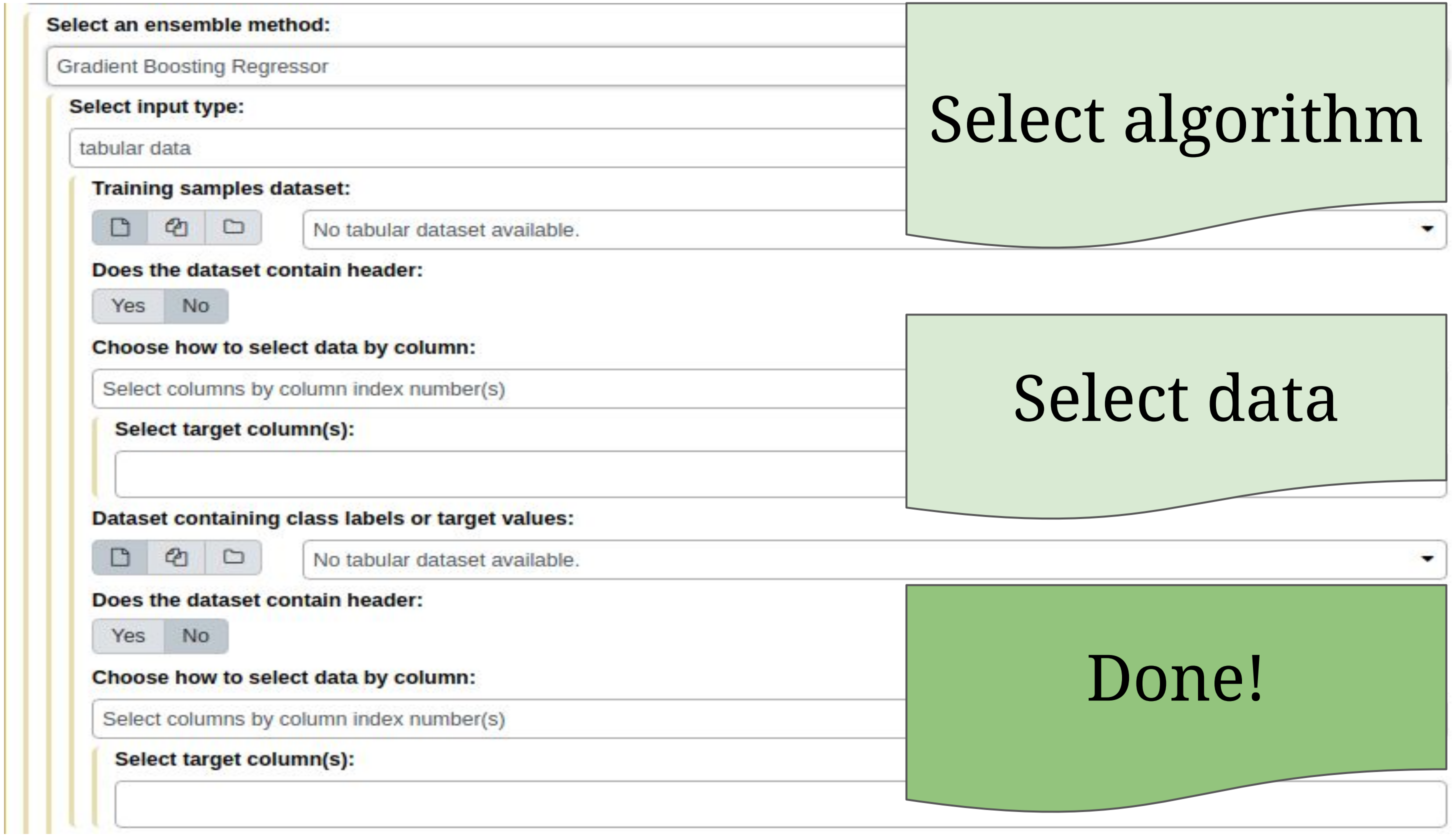
- Linear models
- Support vector machines
- Nearest neighbours
- Tree and ensemble methods
- Naive bayes
- ...

Usability and accessibility

- No programming knowledge required
- Easy to use UI
- Create workflows for end-to-end analysis
- Parallel processing (a few modules)
- Submit jobs (dataset collection)
- Save and reuse trained models
- 1 training material online, 1 in progress

- Select algorithm
- Select data
- Run the analysis

Tool



Select an ensemble method:

Gradient Boosting Regressor

Select input type:

tabular data

Training samples dataset:

No tabular dataset available.

Does the dataset contain header:

Yes [✓ No]

Choose how to select data by column:

Select columns by column index number(s)

Select target column(s):

Dataset containing class labels or target values:

No tabular dataset available.

Does the dataset contain header:

Yes [✓ No]

Choose how to select data by column:

Select columns by column index number(s)

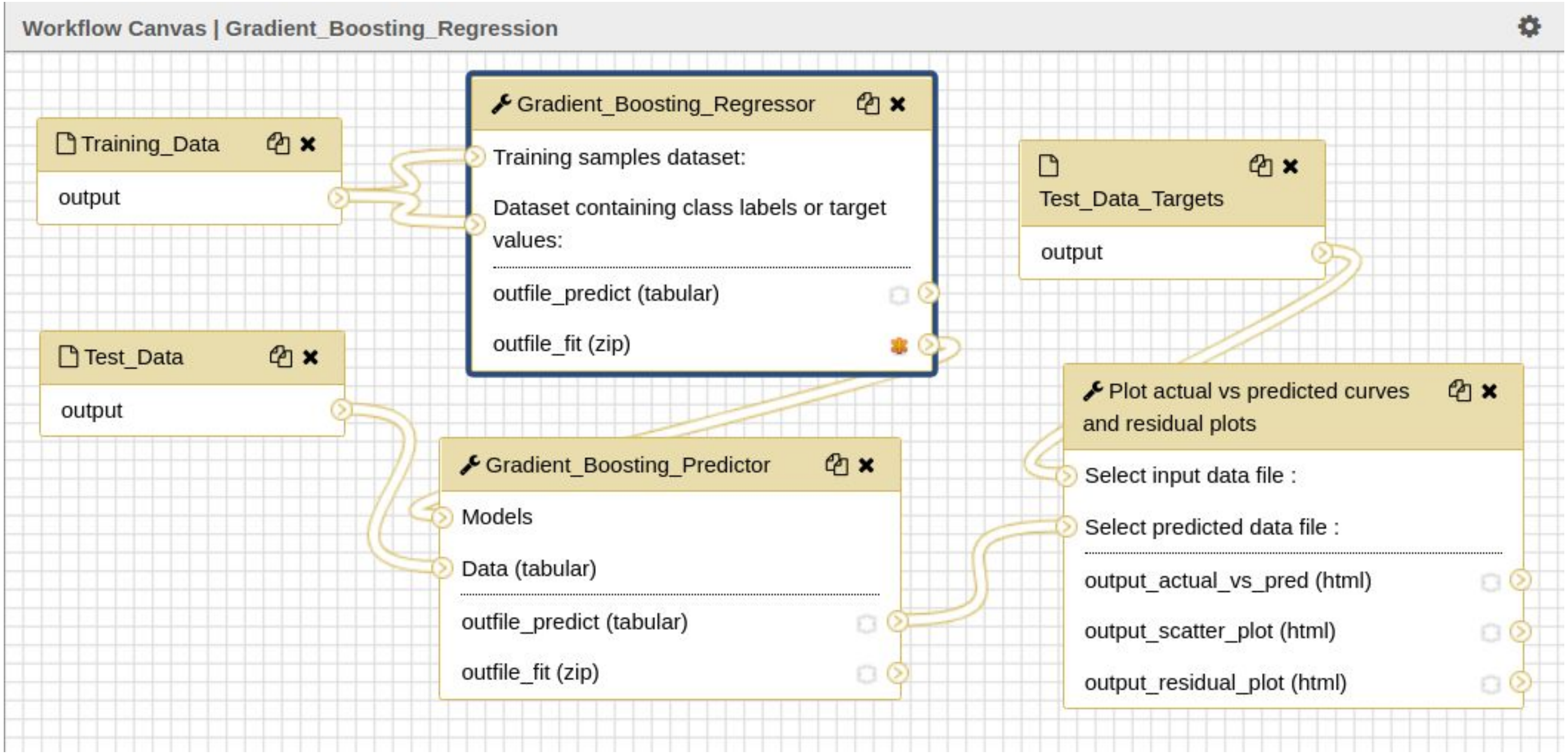
Select target column(s):

Select algorithm

Select data

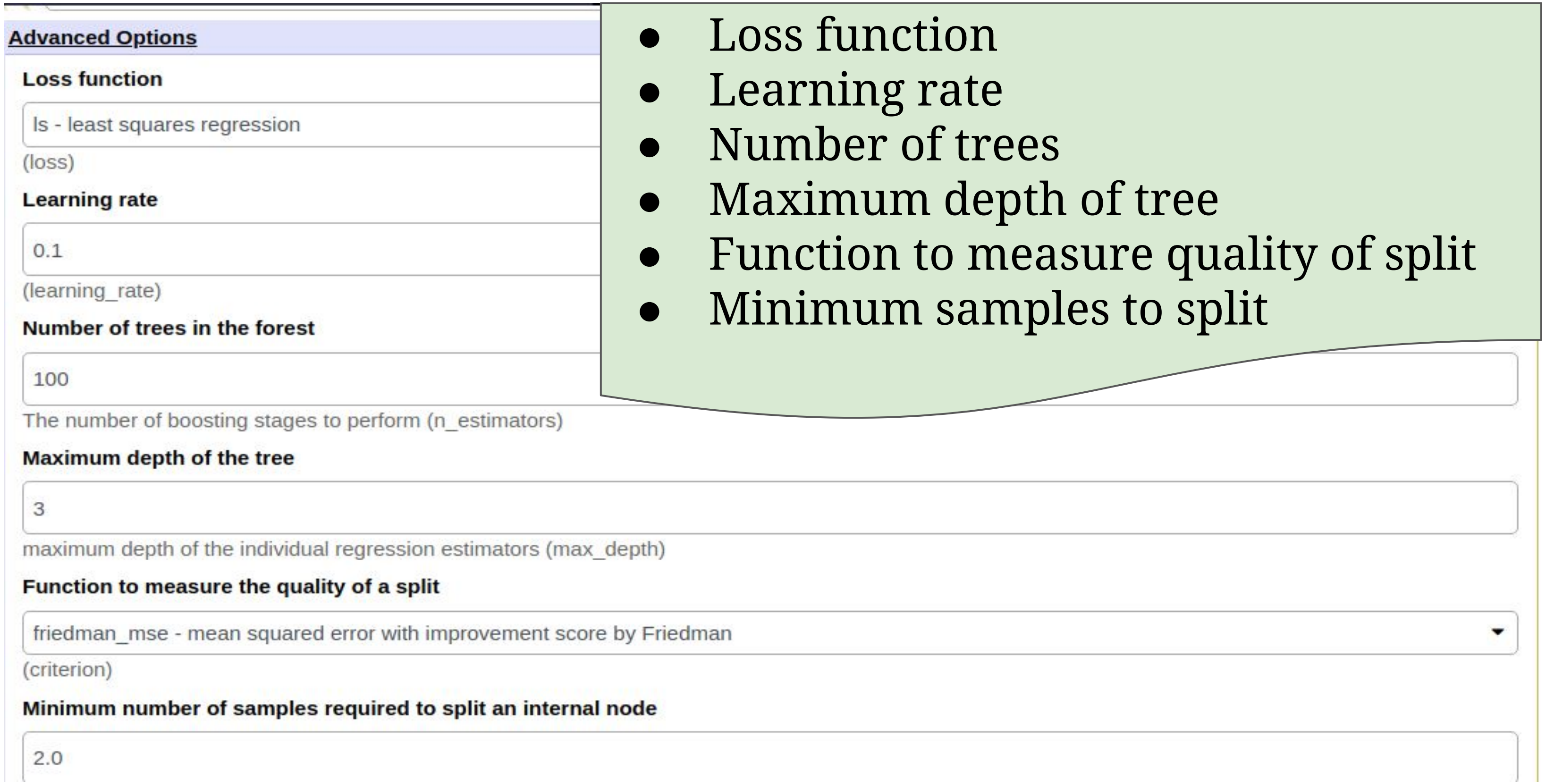
Done!

Workflow



Workflow Canvas | Gradient_Boosting_Regression
Training_Data
output
Gradient_Boosting_Regressor
Training samples dataset:
Dataset containing class labels or target values:
outfile_predict (tabular)
outfile_fit (zip)
Test_Data_Targets
output
Test_Data
output
Gradient_Boosting_Predictor
Models
Data (tabular)
outfile_predict (tabular)
outfile_fit (zip)
Plot actual vs predicted curves and residual plots
Select input data file :
Select predicted data file :
output_actual_vs_pred (html)
output_scatter_plot (html)
output_residual_plot (html)

Hyperparameters



Advanced Options

Loss function

ls - least squares regression

(loss)

Learning rate

0.1

(learning_rate)

Number of trees in the forest

100

The number of boosting stages to perform (n_estimators)

Maximum depth of the tree

3

maximum depth of the individual regression estimators (max_depth)

Function to measure the quality of a split

friedman_mse - mean squared error with improvement score by Friedman

(criterion)

Minimum number of samples required to split an internal node

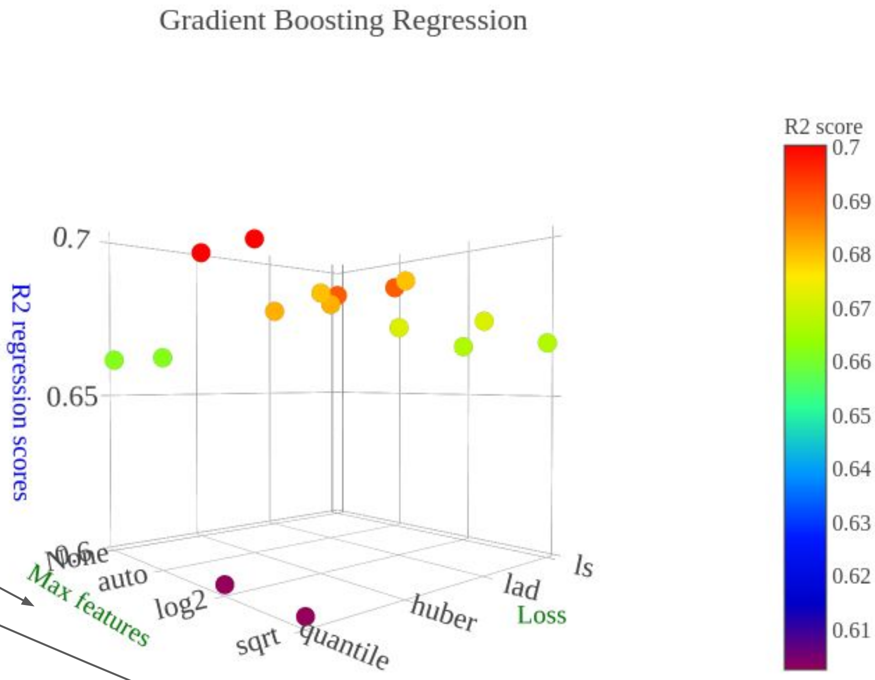
2.0

- Loss function
- Learning rate
- Number of trees
- Maximum depth of tree
- Function to measure quality of split
- Minimum samples to split

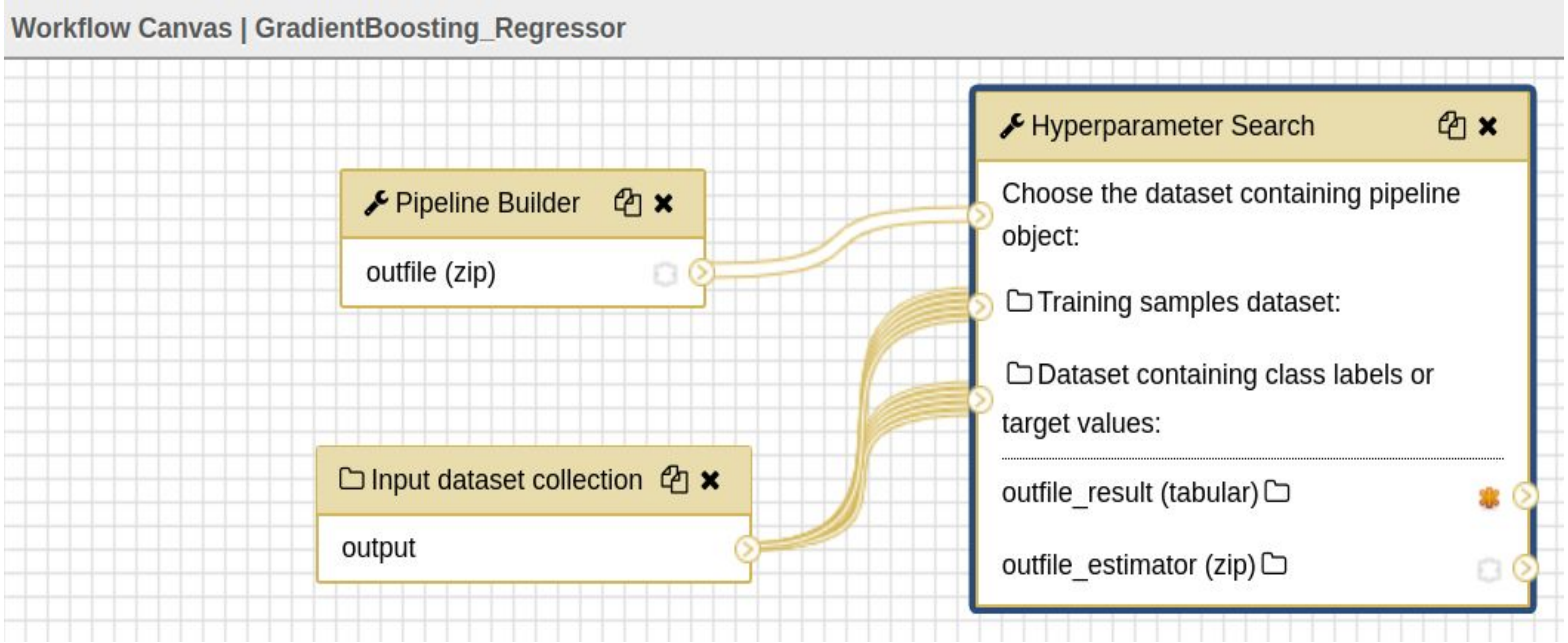
Hyperparameter optimisation

- Grid search

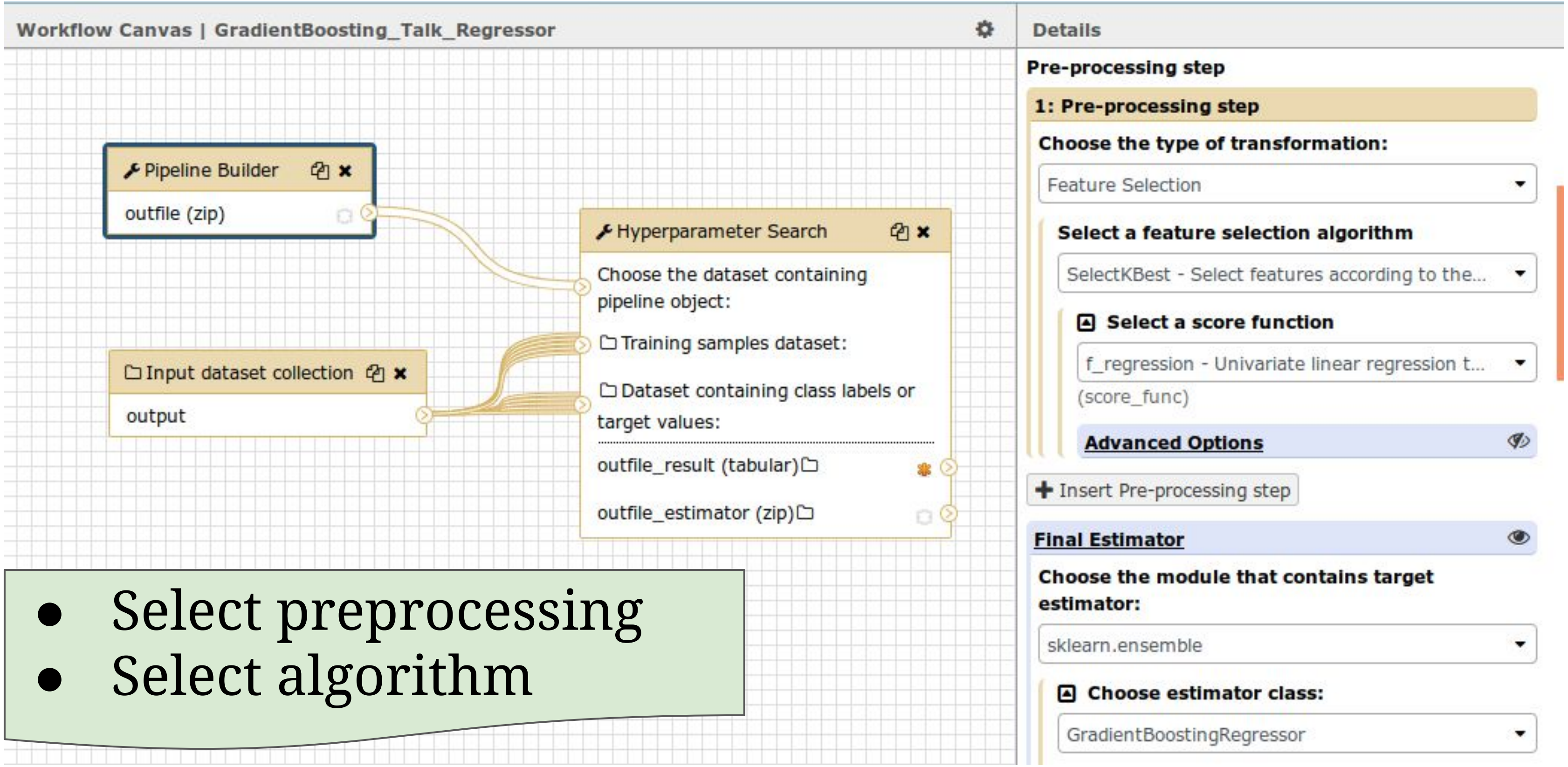
# models: 4 x 4 = 16



Hyperparameter optimisation


Workflow Canvas | GradientBoosting_Regressor
Pipeline Builder
outfile (zip)
Input dataset collection
output
Hyperparameter Search
Choose the dataset containing pipeline object:
Training samples dataset:
Dataset containing class labels or target values:
outfile_result (tabular)
outfile_estimator (zip)

1. Create pipeline



Workflow Canvas | GradientBoosting_Talk_Regressor

Details

Pre-processing step

1: Pre-processing step

Choose the type of transformation: Feature Selection

Select a feature selection algorithm: SelectKBest - Select features according to the...

Select a score function: f_regression - Univariate linear regression t... (score_func)

Advanced Options

+ Insert Pre-processing step

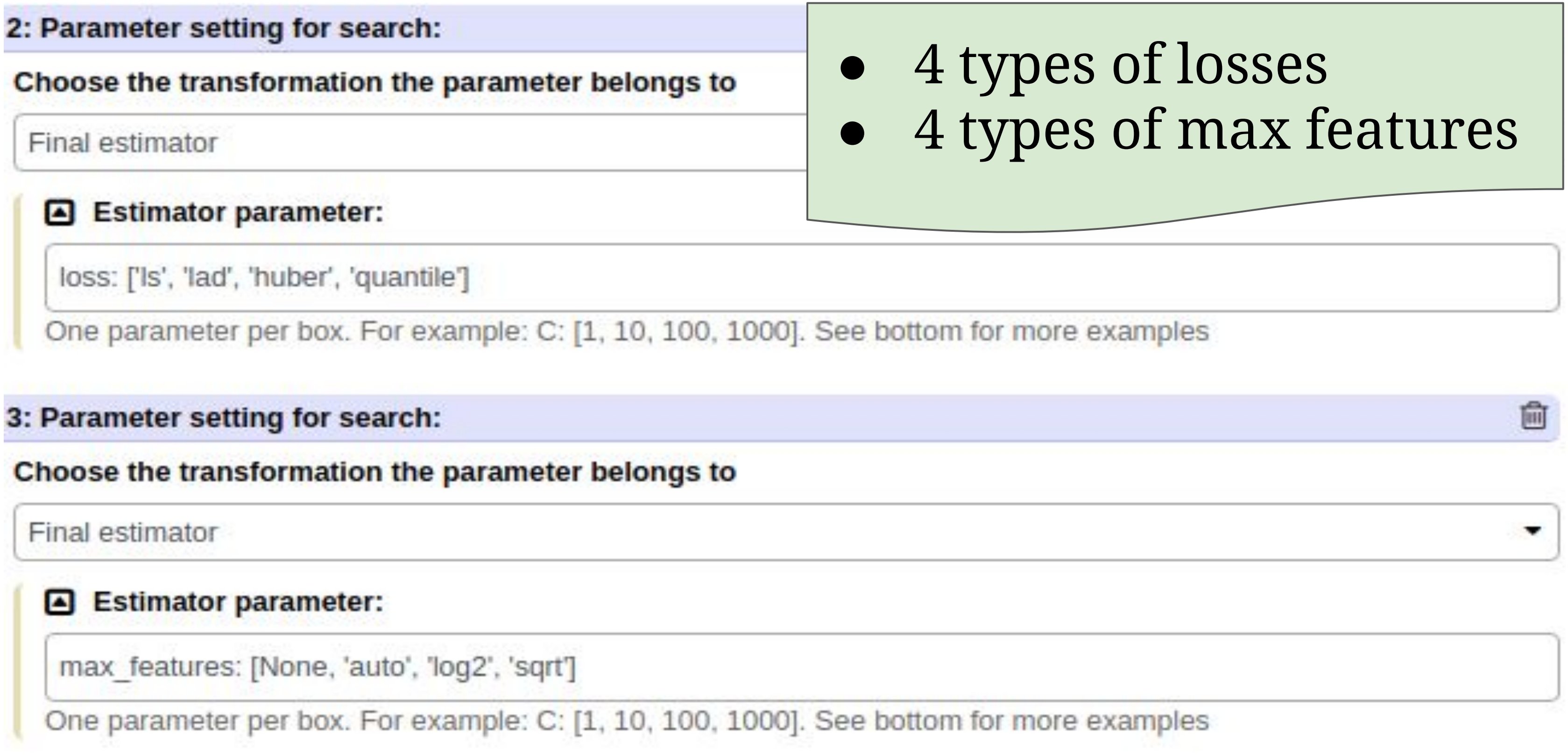
Final Estimator

Choose the module that contains target estimator: sklearn.ensemble

Choose estimator class: GradientBoostingRegressor

- Select preprocessing
- Select algorithm

2. Set hyperparameters



2: Parameter setting for search:

Choose the transformation the parameter belongs to

Final estimator

Estimator parameter:

loss: ['ls', 'lad', 'huber', 'quantile']

One parameter per box. For example: C: [1, 10, 100, 1000]. See bottom for more examples

3: Parameter setting for search:

Choose the transformation the parameter belongs to

Final estimator

Estimator parameter:

max_features: [None, 'auto', 'log2', 'sqrt']

One parameter per box. For example: C: [1, 10, 100, 1000]. See bottom for more examples

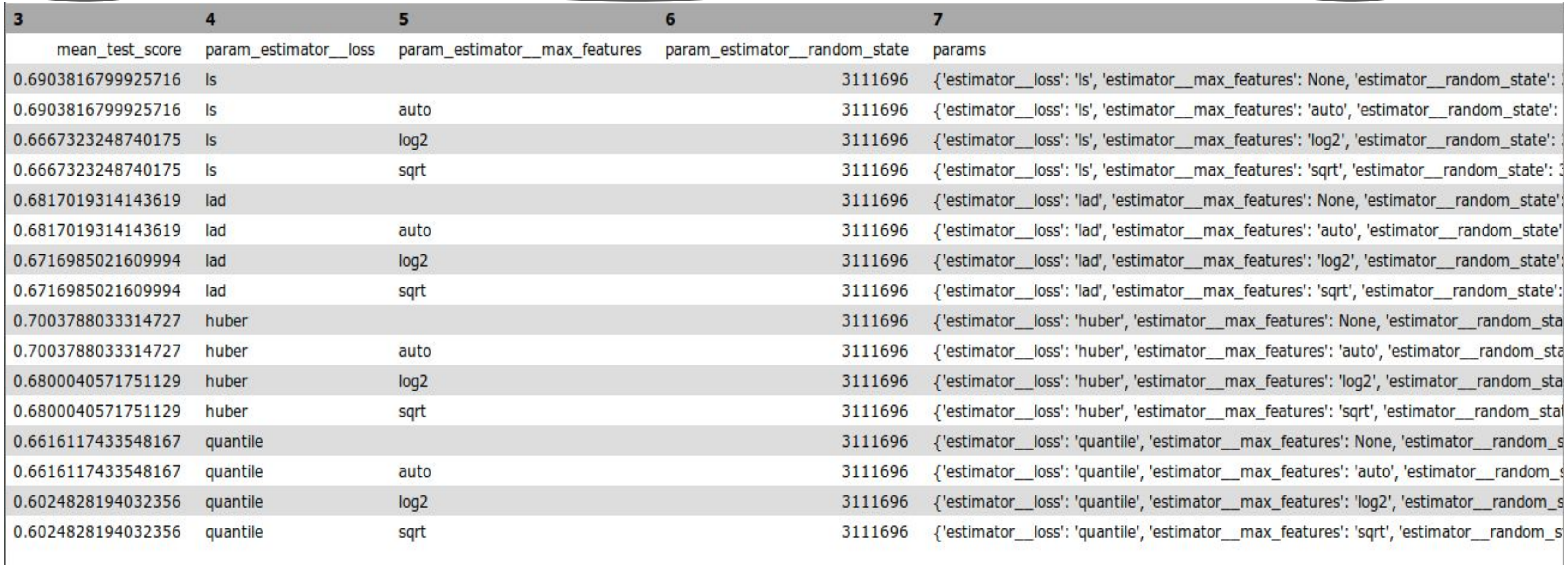
- 4 types of losses
- 4 types of max features

3. Search results

Accuracy

Hyperparameters

Models



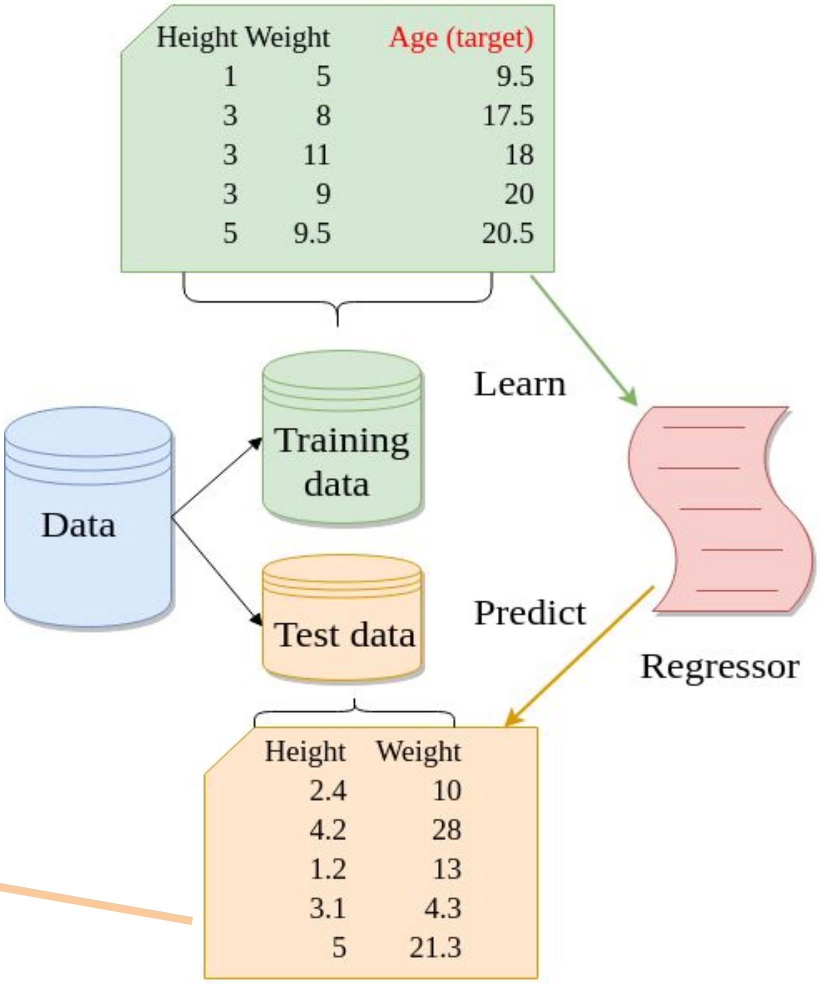
3	4	5	6	7
mean_test_score	param_estimator__loss	param_estimator__max_features	param_estimator__random_state	params
0.6903816799925716	ls		3111696	{'estimator__loss': 'ls', 'estimator__max_features': None, 'estimator__random_state':
0.6903816799925716	ls	auto	3111696	{'estimator__loss': 'ls', 'estimator__max_features': 'auto', 'estimator__random_state':
0.6667323248740175	ls	log2	3111696	{'estimator__loss': 'ls', 'estimator__max_features': 'log2', 'estimator__random_state':
0.6667323248740175	ls	sqrt	3111696	{'estimator__loss': 'ls', 'estimator__max_features': 'sqrt', 'estimator__random_state':
0.681701931414361 9	lad		3111696	{'estimator__loss': 'lad', 'estimator__max_features': None, 'estimator__random_state'
0.681701931414361 9	lad	auto	3111696	{'estimator__loss': 'lad', 'estimator__max_features': 'auto', 'estimator__random_state'
0.6716985021609994	lad	log2	3111696	{'estimator__loss': 'lad', 'estimator__max_features': 'log2', 'estimator__random_state'
0.6716985021609994	lad	sqrt	3111696	{'estimator__loss': 'lad', 'estimator__max_features': 'sqrt', 'estimator__random_state':
0.7003788033314727	huber		3111696	{'estimator__loss': 'huber', 'estimator__max_features': None, 'estimator__random_sta
0.7003788033314727	huber	auto	3111696	{'estimator__loss': 'huber', 'estimator__max_features': 'auto', 'estimator__random_sta
0.6800040571751129	huber	log2	3111696	{'estimator__loss': 'huber', 'estimator__max_features': 'log2', 'estimator__random_sta
0.6800040571751129	huber	sqrt	3111696	{'estimator__loss': 'huber', 'estimator__max_features': 'sqrt', 'estimator__random_sta
0.6616117433548167	quantile		3111696	{'estimator__loss': 'quantile', 'estimator__max_features': None, 'estimator__random_s
0.6616117433548167	quantile	auto	3111696	{'estimator__loss': 'quantile', 'estimator__max_features': 'auto', 'estimator__random_
0.6024828194032356	quantile	log2	3111696	{'estimator__loss': 'quantile', 'estimator__max_features': 'log2', 'estimator__random_s
0.6024828194032356	quantile	sqrt	3111696	{'estimator__loss': 'quantile', 'estimator__max_features': 'sqrt', 'estimator__random_s

Regression
on
Penn Machine Learning Benchmark datasets

Regression

- Supervised learning
- Real valued targets (output)
- R2 scoring metric

Predict “Age”

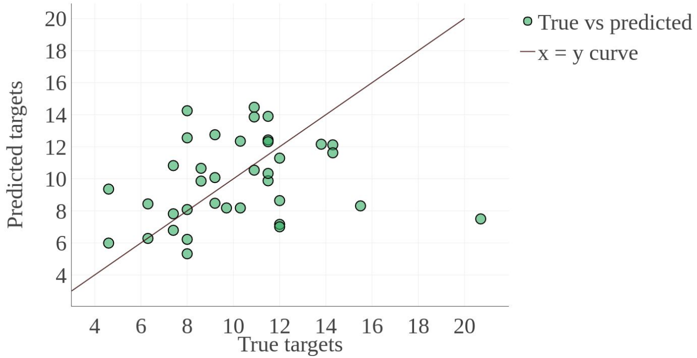


Height	Weight	Age (target)
1	5	9.5
3	8	17.5
3	11	18
3	9	20
5	9.5	20.5

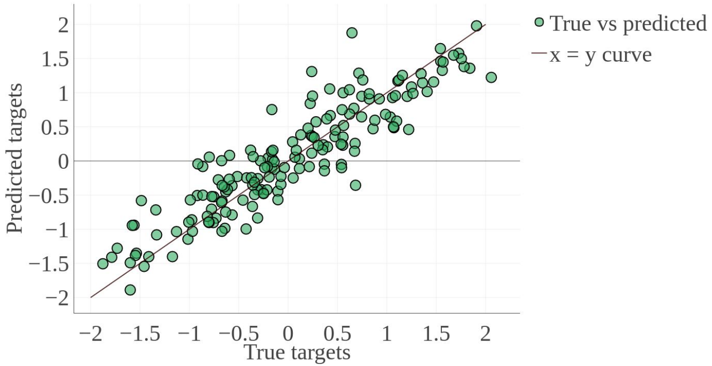
Height	Weight
2.4	10
4.2	28
1.2	13
3.1	4.3
5	21.3

Regression metric (R2)

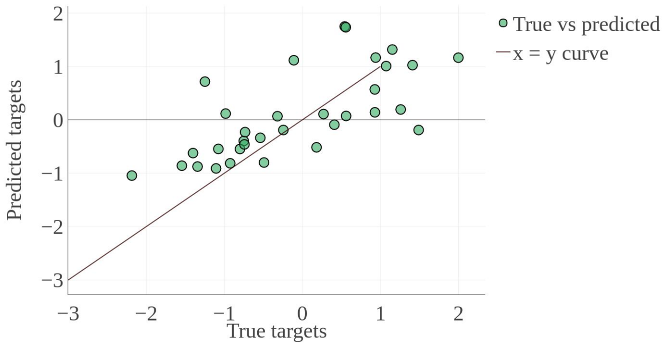
True vs predicted targets (R2: -0.33)



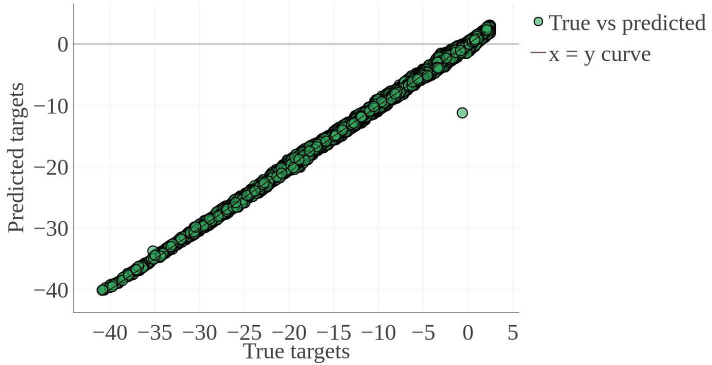
True vs predicted targets (R2: 0.82)

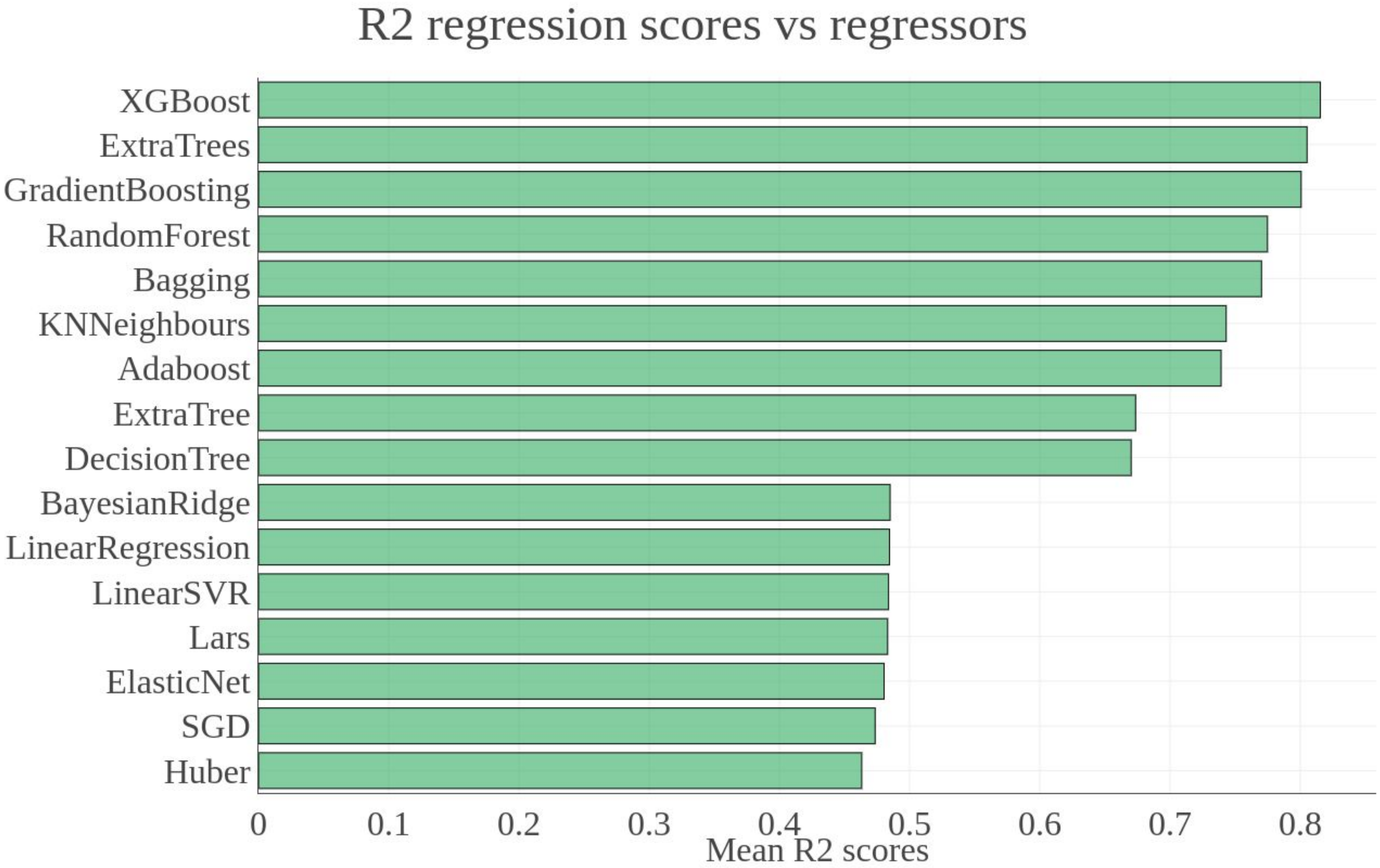


True vs predicted targets (R2: 0.44)

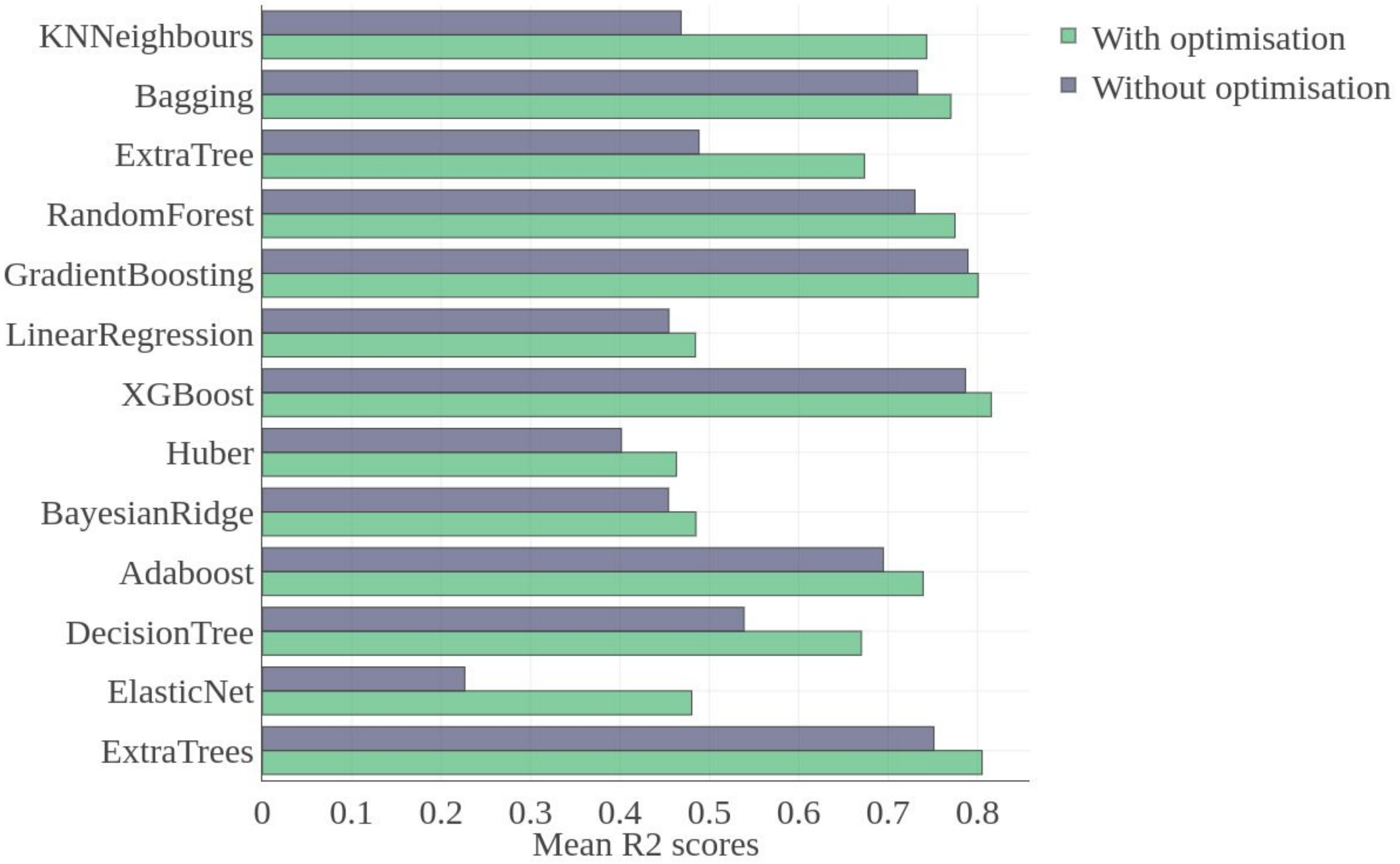


True vs predicted targets (R2: 1.00)

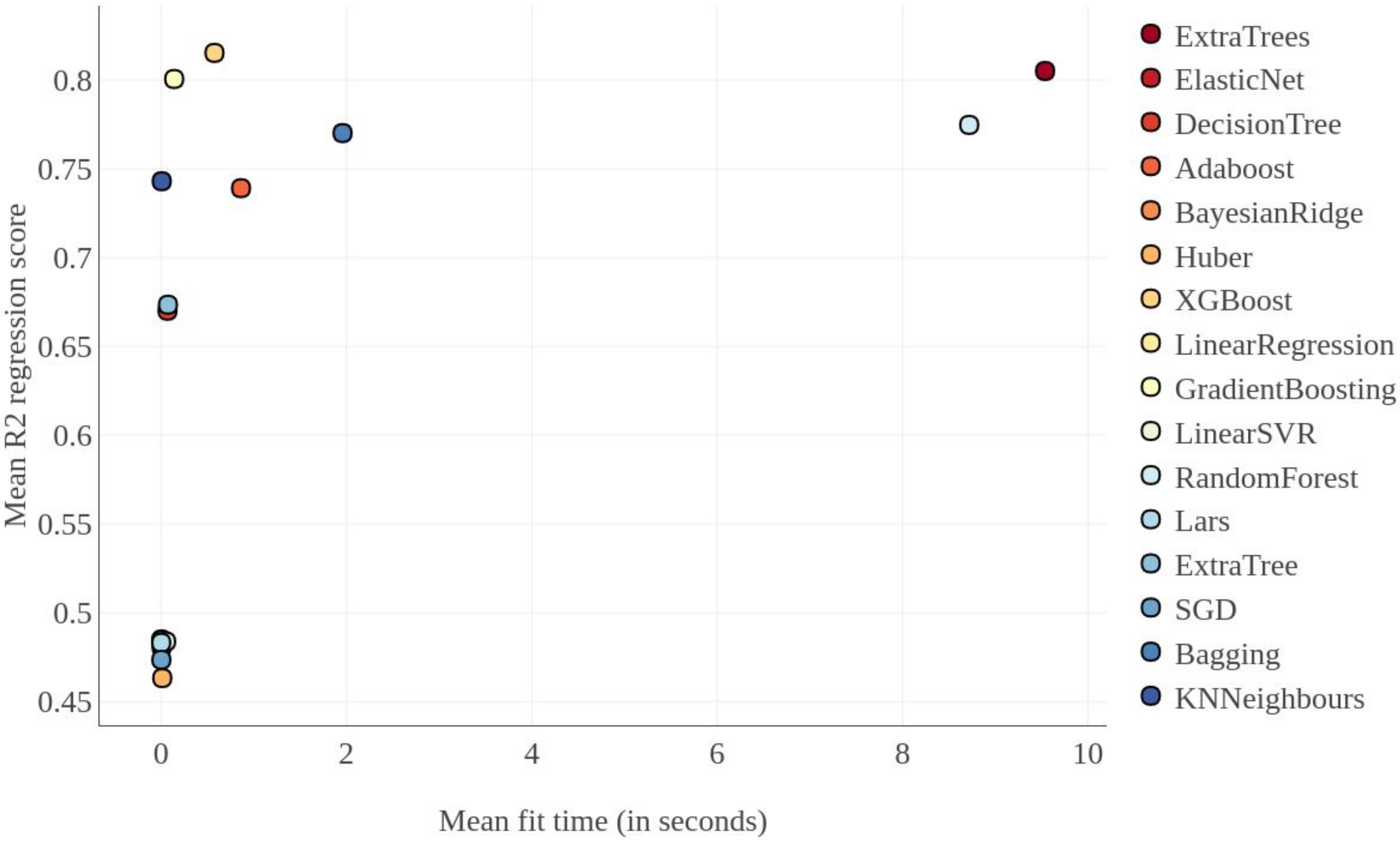



R2 regression scores vs regressors
XGBoost
ExtraTrees
GradientBoosting
RandomForest
Bagging
KNNeighbours
Adaboost
ExtraTree
DecisionTree
BayesianRidge
LinearRegression
LinearSVR
Lars
ElasticNet
SGD
Huber
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Mean R2 scores

R2 scores of regressors with and without hyperparameter optimisation



Fit time vs R2 regression score



Conclusion

- Machine learning with Galaxy
- Variety of preprocessing and learning techniques
- Applications in Bioinformatics
- Usage:
    - Create workflows
    - Optimise hyperparameters
    - Analyse results using plots
    - Regression on a public dataset

Thank you for your attention

Questions?

References

- Original paper [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5890912/]
- Machine learning in Bioinformatics [https://academic.oup.com/bib/article/7/1/86/264025]
- Drug prediction [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4396063/]
- Penn Machine Learning Benchmarks [https://github.com/EpistasisLab/penn-ml-benchmarks]
- Protein structure prediction with machine learning [https://www.ncbi.nlm.nih.gov/pubmed/22274898]
- Gene prediction [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5698827/]
- Machine learning and genome annotation [https://genomebiology.biomedcentral.com/articles/10.1186/gb-2013-14-5-205]
- Machine learning on gene expression microarray data [https://www.ncbi.nlm.nih.gov/pubmed/18366602]
- Medical image analysis with machine learning [https://www.sciencedirect.com/science/article/pii/S1361841516301098]
- First training material [https://galaxyproject.github.io/training-material/topics/statistics/tutorials/machinelearning/tutorial.html]