AutoML and Explainable AI

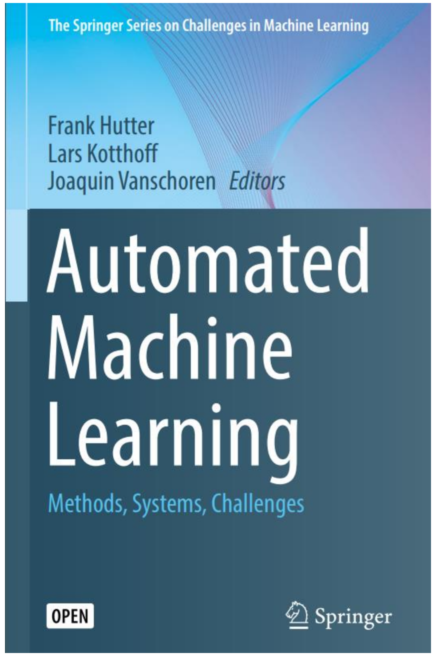
Assoc. Prof. Karl Ezra Pilario, Ph.D.
Process Systems Engineering Laboratory
Department of Chemical Engineering
University of the Philippines Diliman

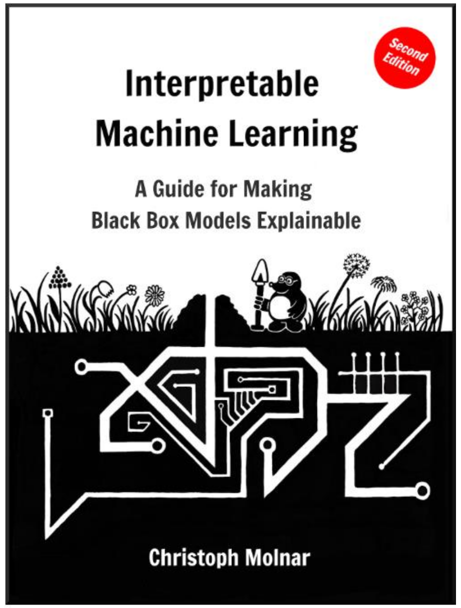
Outline

- AutoML Packages
    - Lazy Predict
    - Auto-sklearn
    - Optuna
    - TPOT
    - PyCaret

- Explainable AI (XAI)
    - Definitions and Concepts
    - Permutation Feature Importance
    - Drop-column Feature Importance
    - Mean-Decrease-in-Impurity
    - Shapley Additive Explanations

Hutter, Kotthoff, Vanschoren (2019)

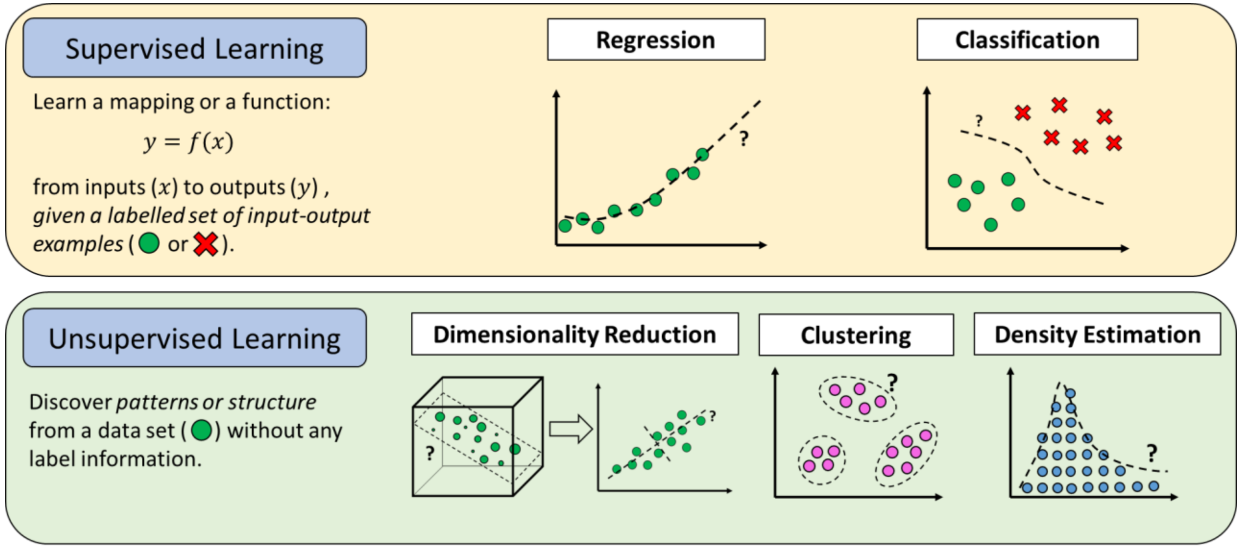


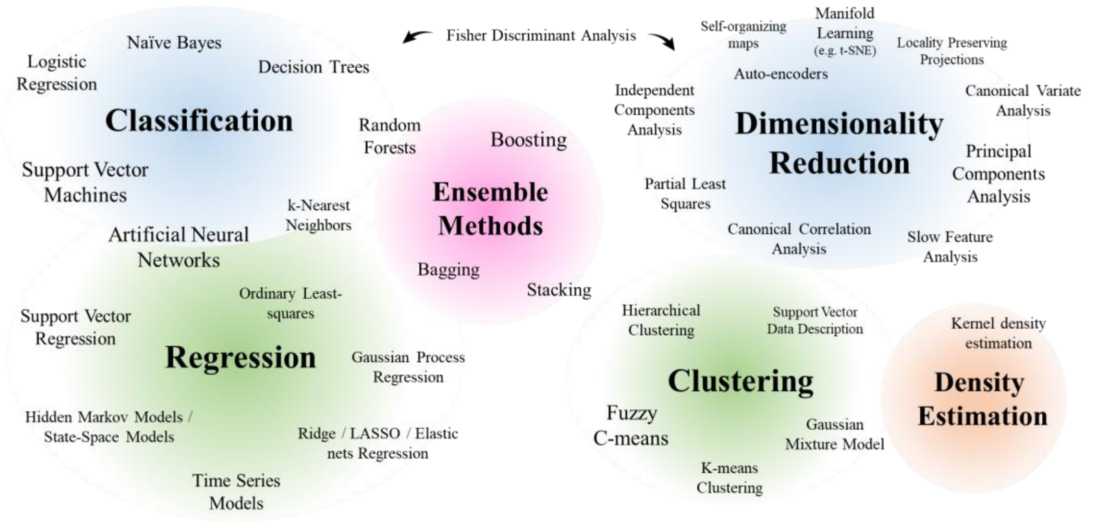


Molnar (2022)

AutoML

- Automatically discover best-performing models with little user involvement.
- For model comparison, AutoML offers a single hyper-parameter optimization toolkit for all models.
- **Meta-learning:** *Learning to Learn*
  - The science of systematically observing how different ML approaches perform on a wide range of tasks, then learning from this experience to improve ML itself.
- **CASH:** *Combined Algorithm Selection and Hyper-parameter Optimization (Kotthoff et al., 2019)*
  - Automatically and simultaneously choosing a learning algorithm and setting its hyperparameters to optimize empirical performance.

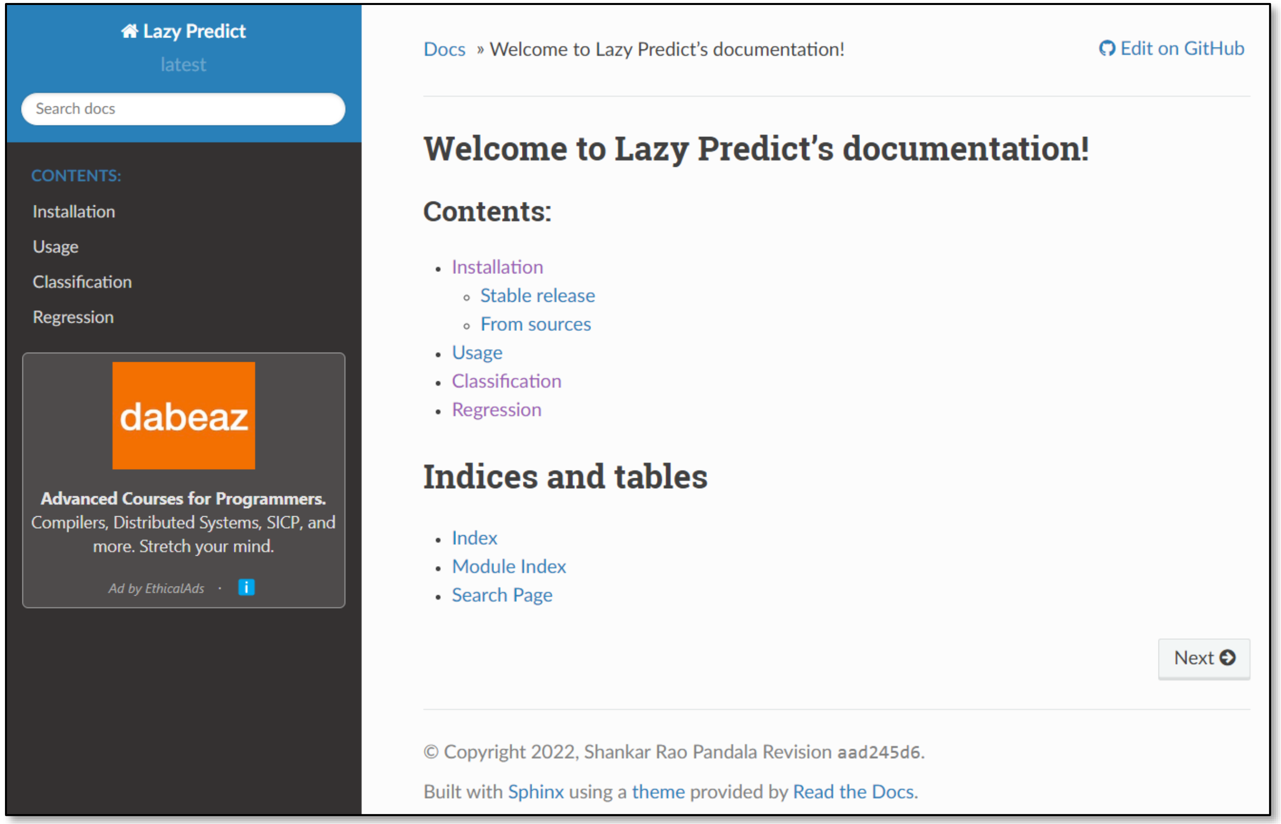




Lazy Predict

- Shankar Rao Pandala (Last Update: 2022)
- https://github.com/shankarpandala/lazypredict/tree/master
- https://lazypredict.readthedocs.io/en/latest/

- Fits a number of **scikit-learn** models on the data with default settings for all.
- Results: Accuracy, R2, F1-score, etc.
- No automatic model selection nor hyper-parameter tuning.
- For classification or regression only.



Lazy Predict

Example:

Use LazyClassifier on the **Breast Cancer Data Set** (this is the example from the website)

Ranked by Accuracy

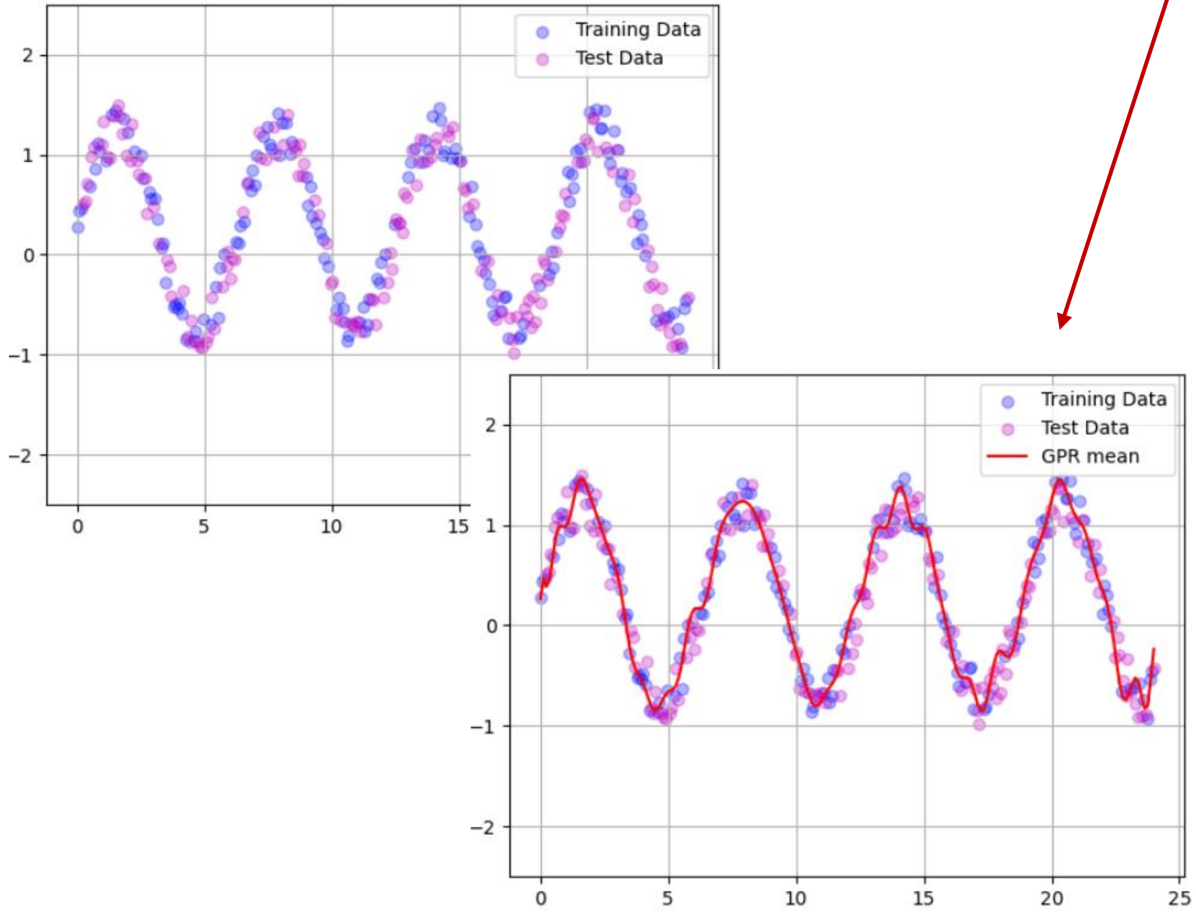
```
100%|██████████| 29/29 [00:01<00:00, 16.79it/s]
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LinearSVC	0.99	0.99	0.99	0.99	0.02
Perceptron	0.99	0.98	0.98	0.99	0.02
LogisticRegression	0.99	0.98	0.98	0.99	0.03
SVC	0.98	0.98	0.98	0.98	0.02
XGBClassifier	0.98	0.98	0.98	0.98	0.13
LabelPropagation	0.98	0.97	0.97	0.98	0.03
LabelSpreading	0.98	0.97	0.97	0.98	0.03
BaggingClassifier	0.97	0.97	0.97	0.97	0.07
PassiveAggressiveClassifier	0.98	0.97	0.97	0.98	0.02
SGDClassifier	0.98	0.97	0.97	0.98	0.02
RandomForestClassifier	0.97	0.97	0.97	0.97	0.29
CalibratedClassifierCV	0.98	0.97	0.97	0.98	0.07
LGBMClassifier	0.97	0.97	0.97	0.97	0.17
QuadraticDiscriminantAnalysis	0.96	0.97	0.97	0.97	0.03
ExtraTreesClassifier	0.97	0.96	0.96	0.97	0.21
RidgeClassifierCV	0.97	0.96	0.96	0.97	0.02
RidgeClassifier	0.97	0.96	0.96	0.97	0.02
AdaBoostClassifier	0.96	0.96	0.96	0.96	0.29
KNeighborsClassifier	0.96	0.96	0.96	0.96	0.04
BernoulliNB	0.95	0.95	0.95	0.95	0.02
LinearDiscriminantAnalysis	0.96	0.95	0.95	0.96	0.03
GaussianNB	0.95	0.95	0.95	0.95	0.02
NuSVC	0.95	0.94	0.94	0.95	0.03
ExtraTreeClassifier	0.94	0.93	0.93	0.94	0.02
NearestCentroid	0.95	0.93	0.93	0.95	0.02
DecisionTreeClassifier	0.93	0.93	0.93	0.93	0.02
DummyClassifier	0.64	0.50	0.50	0.50	0.02

Lazy Predict

Example:

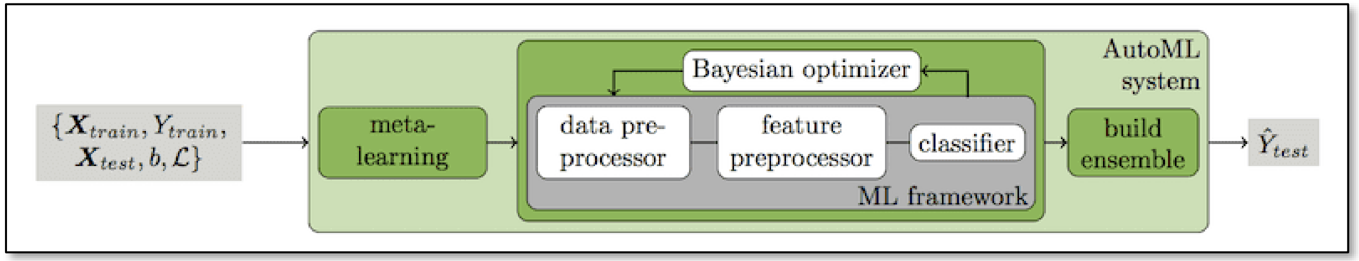
Use LazyRegressor on the **Sine Data Set.**

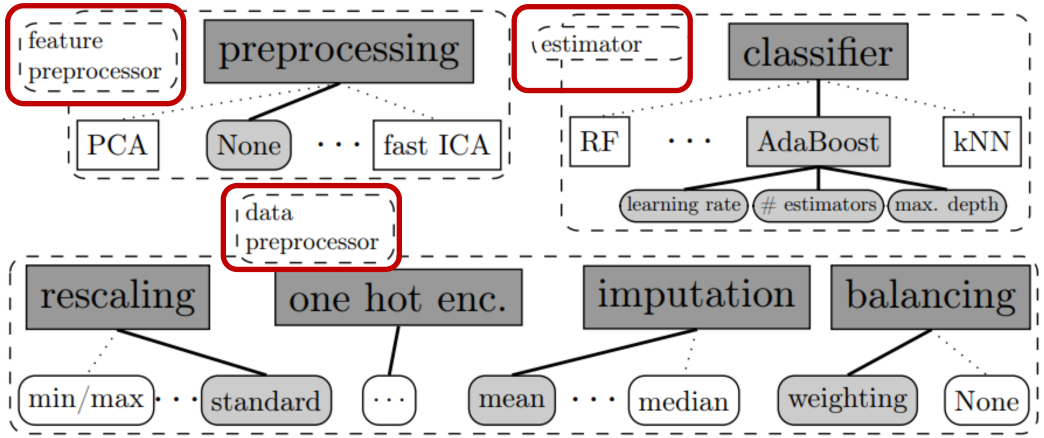



```
100%|██████████| 37/37 [00:01<00:00, 28.27it/s]
```

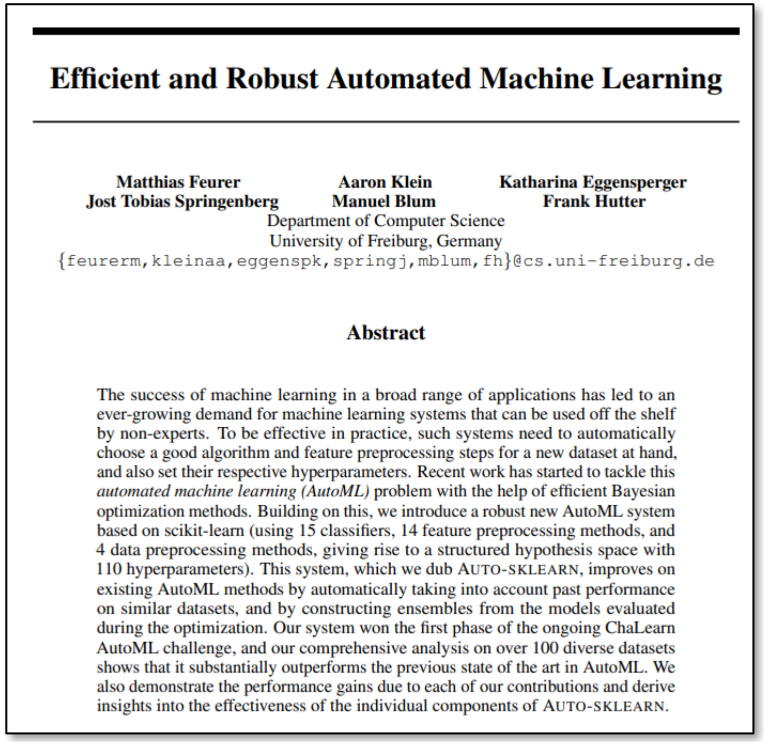
Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
GaussianProcessRegressor	0.95	0.95	0.16	0.03
KNeighborsRegressor	0.94	0.94	0.17	0.02
ExtraTreesRegressor	0.94	0.94	0.19	0.18
BaggingRegressor	0.93	0.93	0.19	0.05
GradientBoostingRegressor	0.93	0.93	0.20	0.08
ExtraTreeRegressor	0.91	0.91	0.22	0.01
DecisionTreeRegressor	0.91	0.91	0.22	0.01
XGBRegressor	0.90	0.90	0.23	0.07
HistGradientBoostingRegressor	0.78	0.78	0.34	0.10
LGBMRegressor	0.76	0.76	0.36	0.07
AdaBoostRegressor	0.55	0.56	0.49	0.08
NuSVR	0.12	0.12	0.69	0.02
SVR	0.11	0.11	0.70	0.02
MLPRegressor	0.04	0.05	0.72	0.10
LinearSVR	0.02	0.03	0.73	0.02
HuberRegressor	0.01	0.02	0.73	0.02
SGDRegressor	0.01	0.02	0.73	0.01
Lars	0.01	0.01	0.73	0.01
TransformedTargetRegressor	0.01	0.01	0.73	0.02
OrthogonalMatchingPursuit	0.01	0.01	0.73	0.01
LinearRegression	0.01	0.01	0.73	0.02
RidgeCV	0.01	0.01	0.73	0.02
TweedieRegressor	-0.00	0.00	0.74	0.01
BayesianRidge	-0.02	-0.01	0.74	0.02
ElasticNetCV	-0.02	-0.01	0.74	0.09
LassoLarsIC	-0.02	-0.01	0.74	0.01
LassoLarsCV	-0.02	-0.01	0.74	0.02
LassoLars	-0.02	-0.01	0.74	0.01
LassoCV	-0.02	-0.01	0.74	0.08
DummyRegressor	-0.02	-0.01	0.74	0.01
LarsCV	-0.02	-0.01	0.74	0.02
ElasticNet	-0.02	-0.01	0.74	0.01
Lasso	-0.02	-0.01	0.74	0.01
KernelRidge	-0.08	-0.07	0.76	0.01
PassiveAggressiveRegressor	-0.90	-0.89	1.02	0.02

Auto-Sklearn





- Feurer et al. (2015) and Feurer et al. (2022)
- https://automl.github.io/auto-sklearn/master/
- For regression and classification with pre-processing.
- A total of 110 tunable hyper-parameters across all models (2015).
- Can discover ensembles.
- Uses Bayesian Optimization and meta-learning.



Efficient and Robust Automated Machine Learning

Matthias Feurer **Aaron Klein** **Katharina Eggensperger**
Jost Tobias Springenberg **Manuel Blum** **Frank Hutter**
Department of Computer Science
University of Freiburg, Germany
{feurerm,kleinaa,eggenspk,springj,mblum,fh}@cs.uni-freiburg.de

Abstract

The success of machine learning in a broad range of applications has led to an ever-growing demand for machine learning systems that can be used off the shelf by non-experts. To be effective in practice, such systems need to automatically choose a good algorithm and feature preprocessing steps for a new dataset at hand, and also set their respective hyperparameters. Recent work has started to tackle this *automated machine learning (AutoML)* problem with the help of efficient Bayesian optimization methods. Building on this, we introduce a robust new AutoML system based on scikit-learn (using 15 classifiers, 14 feature preprocessing methods, and 4 data preprocessing methods, giving rise to a structured hypothesis space with 110 hyperparameters). This system, which we dub AUTO-SKLEARN, improves on existing AutoML methods by automatically taking into account past performance on similar datasets, and by constructing ensembles from the models evaluated during the optimization. Our system won the first phase of the ongoing ChaLearn AutoML challenge, and our comprehensive analysis on over 100 diverse datasets shows that it substantially outperforms the previous state of the art in AutoML. We also demonstrate the performance gains due to each of our contributions and derive insights into the effectiveness of the individual components of AUTO-SKLEARN.

Optuna

- Akiba et al. (2019)
- Suitable for CASH (algorithm selection + hyper-parameter tuning)
- Models and hyper-parameters are user-defined.
- Uses Bayesian Optimization.

OPTUNA

Optuna: A hyperparameter optimization framework

Akiba et al., (2019) Optuna: A Next-generation Hyperparameter Optimization Framework. https://arxiv.org/pdf/1907.10902.pdf

Optuna has modern functionalities as follows:

Lightweight, versatile, and platform agnostic architecture
- Handle a wide variety of tasks with a simple installation that has few requirements.

Pythonic search spaces
- Define search spaces using familiar Python syntax including conditionals and loops.

Efficient optimization algorithms
- Adopt state-of-the-art algorithms for sampling hyperparameters and efficiently pruning unpromising trials.

Easy parallelization
- Scale studies to tens or hundreds of workers with little or no changes to the code.

Quick visualization
- Inspect optimization histories from a variety of plotting functions.

- Grid Search implemented in `GridSampler`
- Random Search implemented in `RandomSampler`
- Tree-structured Parzen Estimator algorithm implemented in `TPESampler`
- CMA-ES based algorithm implemented in `CmaEsSampler`
- Algorithm to enable partial fixed parameters implemented in `PartialFixedSampler`
- Nondominated Sorting Genetic Algorithm II implemented in `NSGAIISampler`
- A Quasi Monte Carlo sampling algorithm implemented in `QMCSampler`

The default sampler is `TPESampler`.

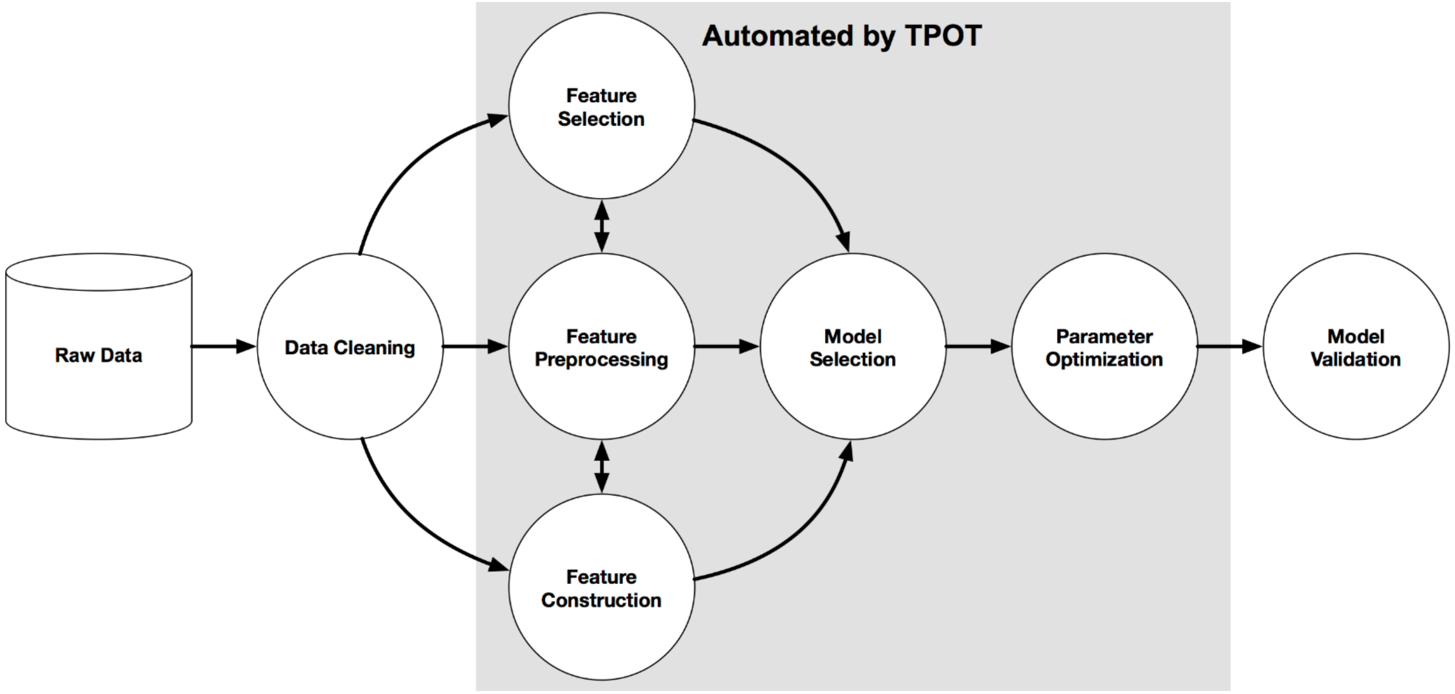
Main algorithm:

TPE (Tree-structured Parzen Estimator)

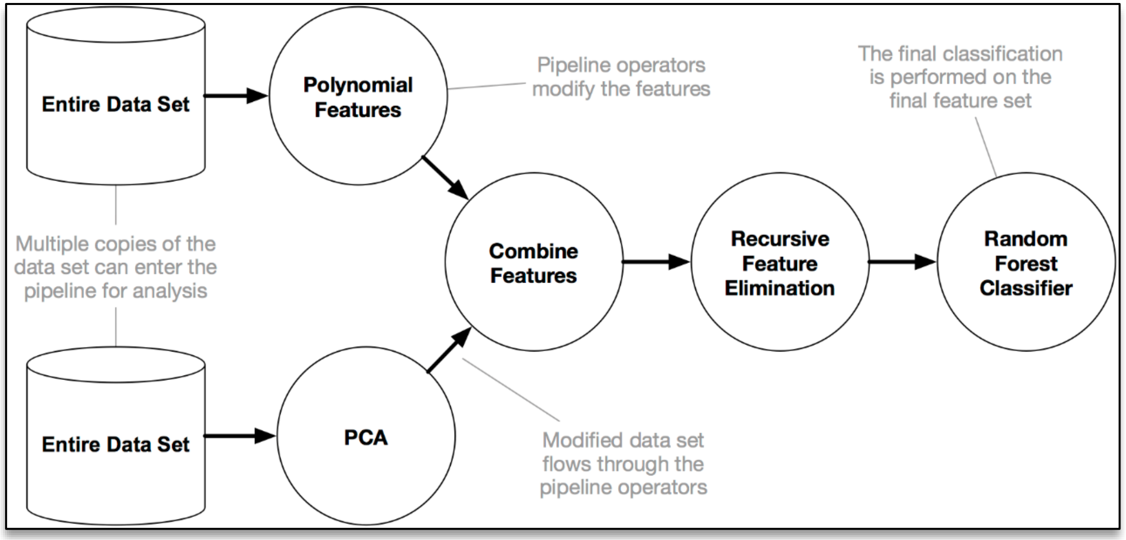
- A variant of Bayesian Optimization

TPOT

- Olson and Moore (2016)
- http://epistasislab.github.io/tpot/
- TPOT = **T**ree-based **P**ipeline **O**ptimization **T**ool
- TPOT optimizes machine learning pipelines using genetic programming.

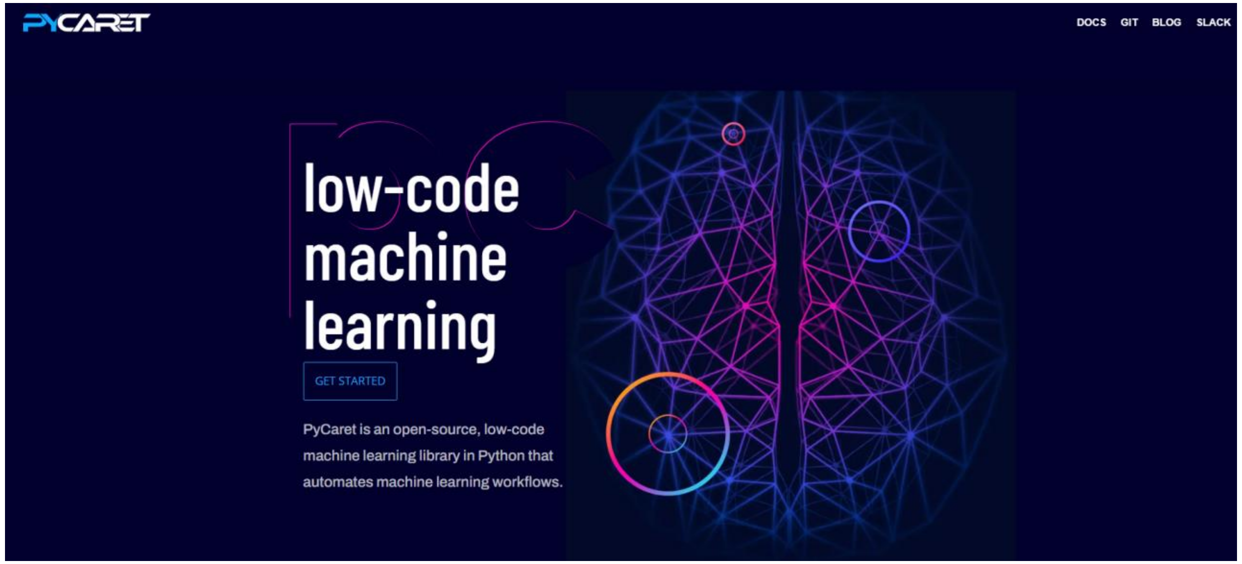


Sample Result:



PyCaret

- Moez Ali (2020)
- https://pycaret.gitbook.io/docs/
- low-code library: replace hundreds of lines of code with a few lines only.



Example: **Regression**

```python
# Regression Functional API Example

# loading sample dataset
from pycaret.datasets import get_data
data = get_data('insurance')

# init setup
from pycaret.regression import *
s = setup(data, target = 'charges', session_id = 123)

# model training and selection
best = compare_models()

# evaluate trained model
evaluate_model(best)

# predict on hold-out/test set
pred_holdout = predict_model(best)

# predict on new data
new_data = data.copy().drop('charges', axis = 1)
predictions = predict_model(best, data = new_data)

# save model
save_model(best, 'best_pipeline')
```

Example: **Classification**

```python
# Classification Functional API Example

# loading sample dataset
from pycaret.datasets import get_data
data = get_data('juice')

# init setup
from pycaret.classification import *
s = setup(data, target = 'Purchase', session_id = 123)

# model training and selection
best = compare_models()

# evaluate trained model
evaluate_model(best)

# predict on hold-out/test set
pred_holdout = predict_model(best)

# predict on new data
new_data = data.copy().drop('Purchase', axis = 1)
predictions = predict_model(best, data = new_data)

# save model
save_model(best, 'best_pipeline')
```

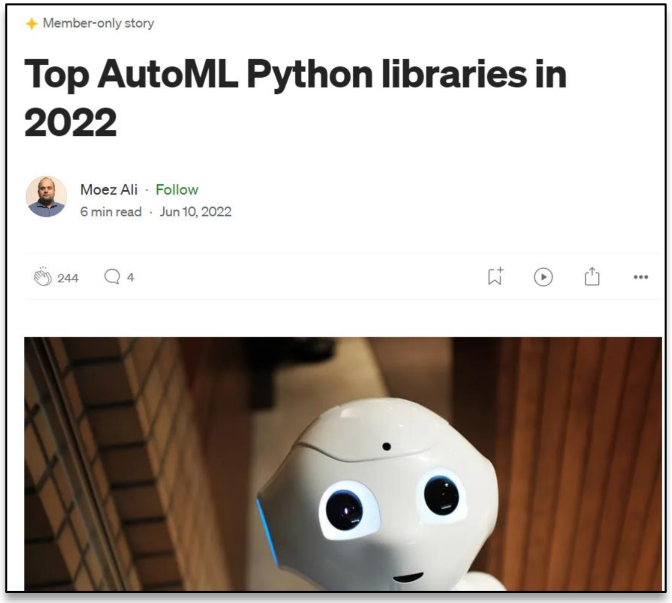
Example: **Anomaly Detection**

```python
# Anomaly Detection Functional API Example

# loading sample dataset
from pycaret.datasets import get_data
data = get_data('anomaly')

# init setup
from pycaret.anomaly import *
s = setup(data, session_id = 123)

# model training
iforest = create_model('iforest')

# assign labels from trained model
results = assign_model(iforest)

# evaluate trained model
evaluate_model(iforest)

# predict on new_data
new_data = data.copy()
predictions = predict_model(iforest, data = new_data)

# save model
save_model(iforest, 'iforest_pipeline')
```

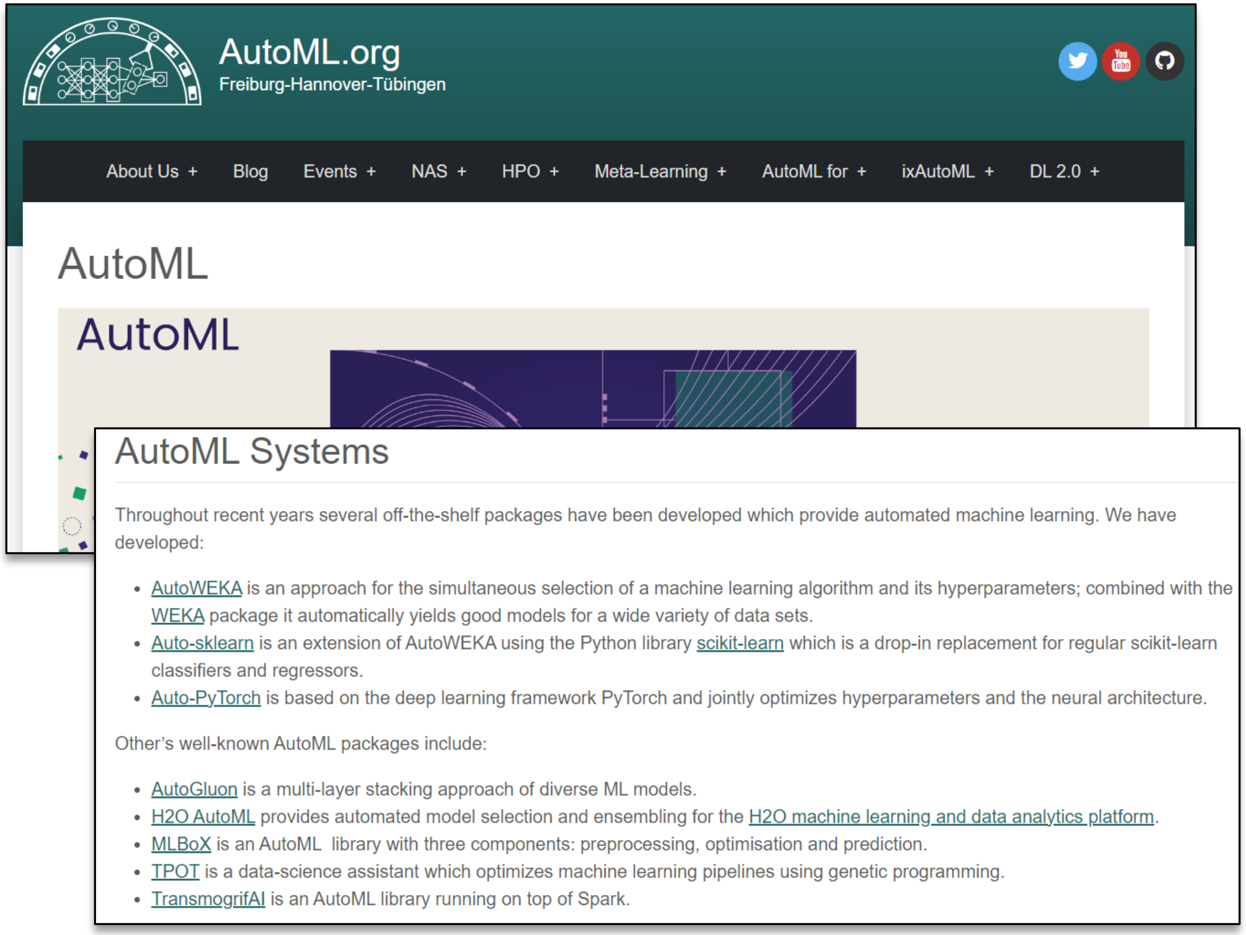
Other AutoML Libraries

According to Moez Ali (from PyCaret), here are the **top AutoML libraries** in 2022.

1. PyCaret
2. H_2O AutoML
3. TPOT
4. Auto-sklearn
5. FLAML
6. EvalML
7. AutoKeras
8. Auto-ViML
9. AutoGluon
10. MLBox



https://www.automl.org/automl/



AutoML Systems

Throughout recent years several off-the-shelf packages have been developed which provide automated machine learning. We have developed:

- AutoWEKA is an approach for the simultaneous selection of a machine learning algorithm and its hyperparameters; combined with the WEKA package it automatically yields good models for a wide variety of data sets.
- Auto-sklearn is an extension of AutoWEKA using the Python library scikit-learn which is a drop-in replacement for regular scikit-learn classifiers and regressors.
- Auto-PyTorch is based on the deep learning framework PyTorch and jointly optimizes hyperparameters and the neural architecture.

Other's well-known AutoML packages include:

- AutoGluon is a multi-layer stacking approach of diverse ML models.
- H2O AutoML provides automated model selection and ensembling for the H2O machine learning and data analytics platform.
- MLBoX is an AutoML library with three components: preprocessing, optimisation and prediction.
- TPOT is a data-science assistant which optimizes machine learning pipelines using genetic programming.
- TransmogrifAI is an AutoML library running on top of Spark.

Comparison of AutoML Libraries

A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost

Luís Ferreira
EPMQ - IT
CCG ZGDV Institute
ALGORITMI Center
University of Minho
Guimarães, Portugal
luis.ferreira@ccg.pt

André Pilastri
EPMQ - IT
CCG ZGDV Institute
Guimarães, Portugal
andre.pilastri@ccg.pt

Carlos Manuel Martins
WeDo Technologies
Braga, Portugal
carlos.mmartins
@mobileum.com

Pedro Miguel Pires
WeDo Technologies
Braga, Portugal
pedro.mpires
@mobileum.com

Paulo Cortez
ALGORITMI Center
Dep. Information Systems
University of Minho
Guimarães, Portugal
pcortez@dsi.uminho.pt

***Abstract*—This paper presents a benchmark of supervised Automated Machine Learning (AutoML) tools. Firstly, we analyze the characteristics of eight recent open-source AutoML tools (Auto-Keras, Auto-PyTorch, Auto-Sklearn, AutoGluon, H2O AutoML, rminer, TPOT and TransmogrifAI) and describe twelve popular OpenML datasets that were used in the benchmark (divided into regression, binary and multi-class classification tasks). Then, we perform a comparison study with hundreds of computational experiments based on three scenarios: General Machine Learning (GML), Deep Learning (DL) and XGBoost (XGB). To select the best tool, we used a lexicographic approach, considering first the average prediction score for each task and**

algorithm selection; Deep Learning (DL) selection and XGBoost (XGB) hyperparameter tuning. Each tool is measured in terms of its predictive performance (using an external 10-fold cross-validation) and computational cost (measured in terms of time elapsed). Moreover, the best AutoML tools are further compared with the best public OpenML predictive results (which are assumed as the "gold standard").

The paper is organized as follows. Section 2 presents the related work. Next, Section 3 describes the AutoML tools and datasets. Section 4 details the benchmark design. Then,

Reference: Ferreira et al. (2021). A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost. *Proceedings of the International Joint Conference on Neural Networks.*

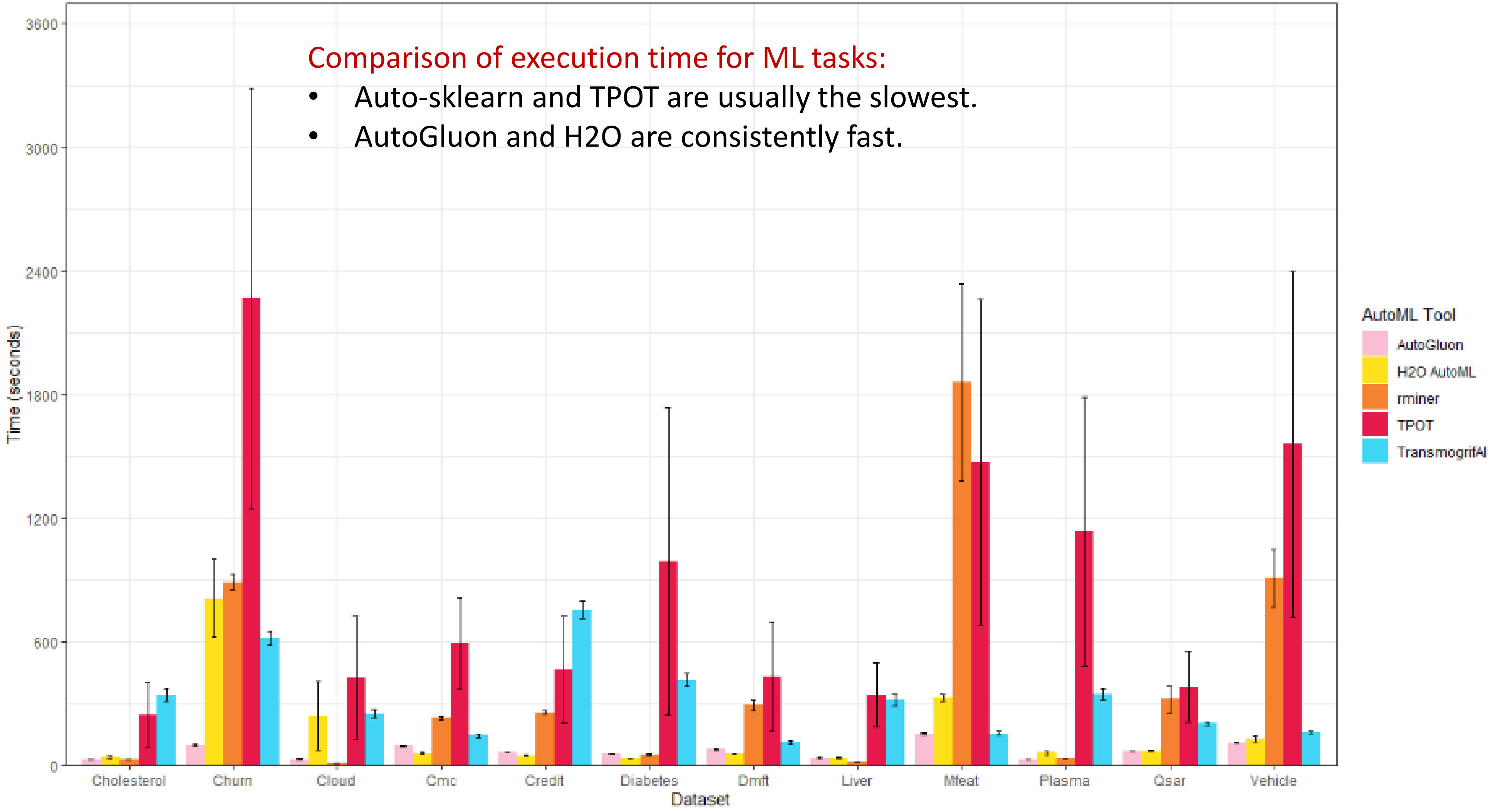
AutoML Tool	Framework	API Lang.	Operating Systems	DL	Scenario GML	DL	XGB
Auto-Keras	Keras	Python	MacOs Linux Windows	Yes (only)		✓	
Auto-PyTorch	PyTorch	Python	MacOs Linux Windows	Yes (only)		✓	
Auto-Sklearn	Scikit-Learn	Python	Linux	No	✓		
AutoGluon	PyTorch	Python	MacOS (P.) Linux	Yes	✓	✓	
H2O AutoML	H2O	Java Python R	MacOs Linux Windows (P.)	Yes	✓	✓	✓
rminer AutoML	rminer	R	MacOs Linux Windows	No	✓		✓
TPOT	Scikit-Learn	Python	MacOs Linux Windows	No	✓		
TransmogrifAI	Spark (MLlib)	Scala	MacOs Linux Windows	No	✓		

- In this work, the authors compared 8 different AutoML tools (see Table).
- Twelve different OpenML data sets were used to benchmark the AutoML tools.
  - Binary classification
  - Multi-class classification
  - Regression

Comparison of AutoML Libraries

Comparison of execution time for ML tasks:

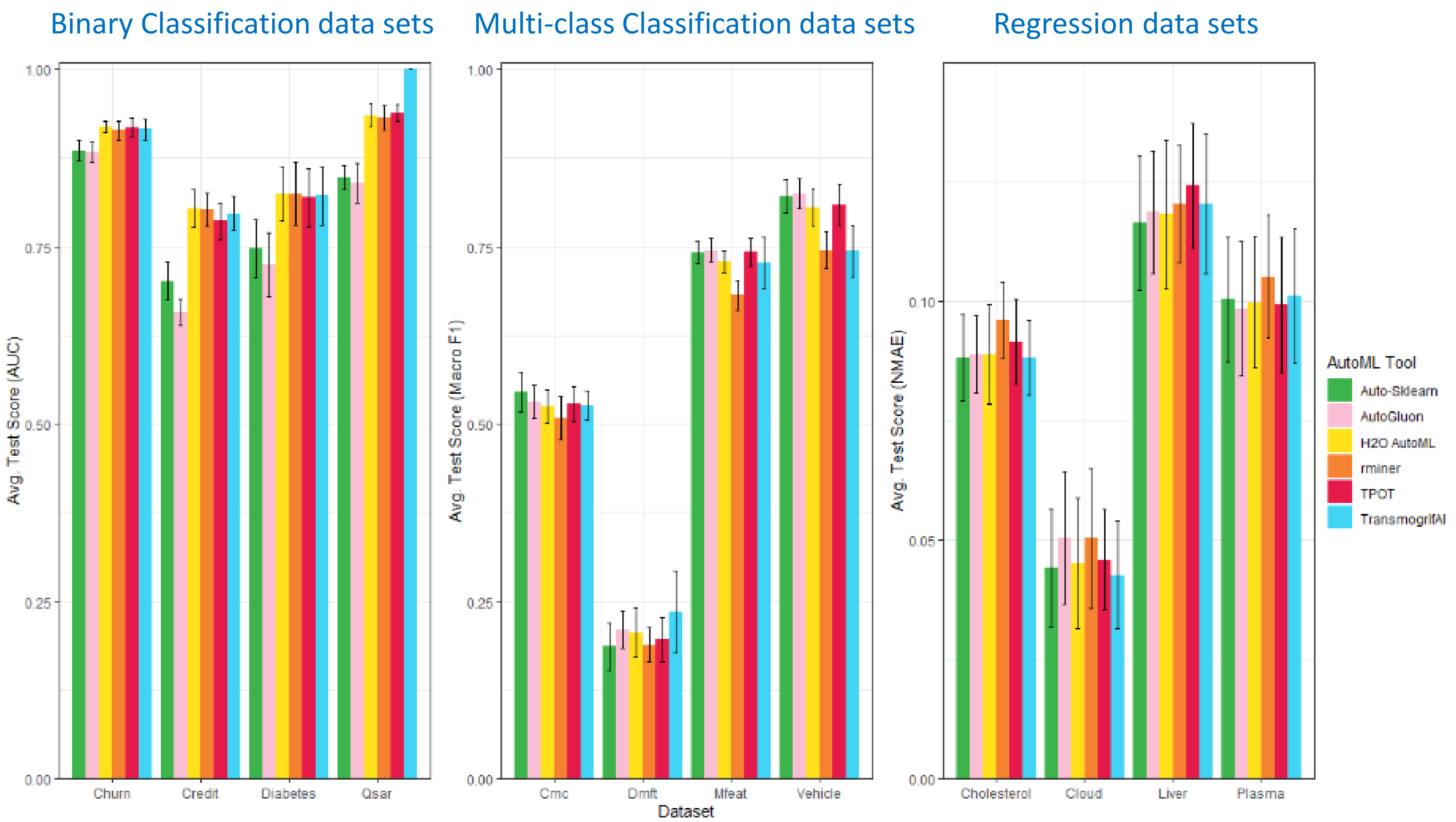
- Auto-sklearn and TPOT are usually the slowest.
- AutoGluon and H2O are consistently fast.



Comparison of AutoML Libraries

Comparison of performance for ML tasks:

- For binary classification, **TransmogrifAI** is best for 3 out of 4 data sets. **AutoGluon** and **Auto-sklearn** produced the worst overall results.
- For multi-class classification, **AutoGluon** and **Auto-sklearn** are best.
- For regression, differences between tools are not that significant. But the best overall is **rminer**.



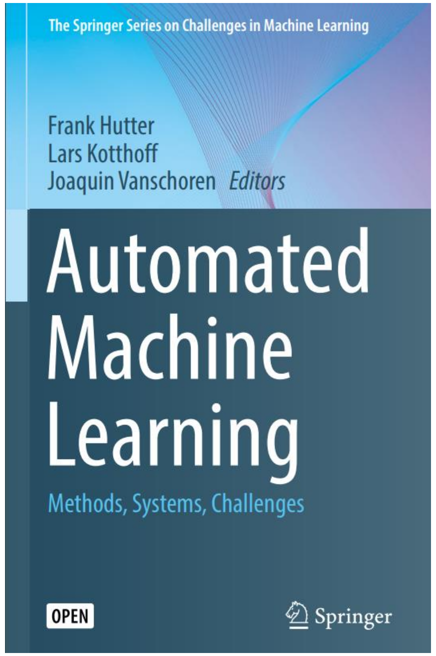
Reference: Ferreira et al. (2021). A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost. *Proceedings of the International Joint Conference on Neural Networks.*

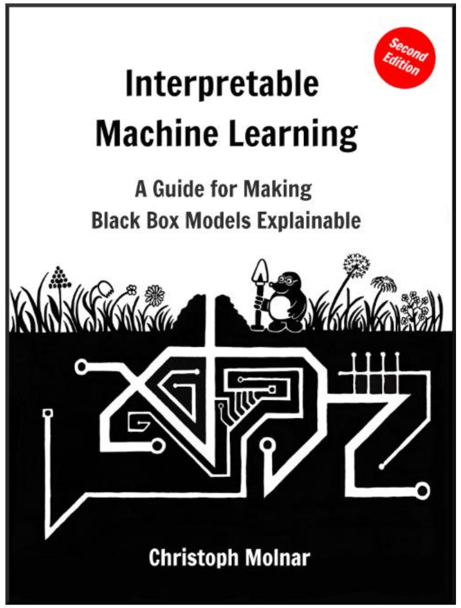
Outline

- AutoML Packages
  - Lazy Predict
  - Auto-sklearn
  - Optuna
  - TPOT
  - PyCaret

- Explainable AI (XAI)
  - Definitions and Concepts
  - Permutation Feature Importance
  - Drop-column Feature Importance
  - Mean-Decrease-in-Impurity
  - Shapley Additive Explanations

Hutter, Kotthoff, Vanschoren (2019)





Molnar (2022)

Explainable AI (XAI)

IBM

What is explainable AI (XAI)?

Explainable artificial intelligence (XAI) is a set of processes and methods that allows human users to **comprehend and trust** the results and output created by machine learning algorithms. Explainable AI is used to describe an AI model, its expected **impact** and **potential biases**. It helps characterize model **accuracy**, **fairness**, **transparency** and outcomes in AI-powered decision making.

Explainable AI is crucial for an organization in building trust and confidence when putting AI models into production. AI explainability also helps an organization adopt a **responsible** approach to AI development.

- IBM (https://www.ibm.com/watson/explainable-ai)

Explainable AI (XAI)

Understandability

Ability of a model to make a human understand its *internal structure* and how it works *algorithmically*.

Comprehensibility

Ability of a learning algorithm to represent its learned knowledge in a human understandable fashion.

Interpretability

Refers to how accurate a machine learning model can associate a cause to an effect.

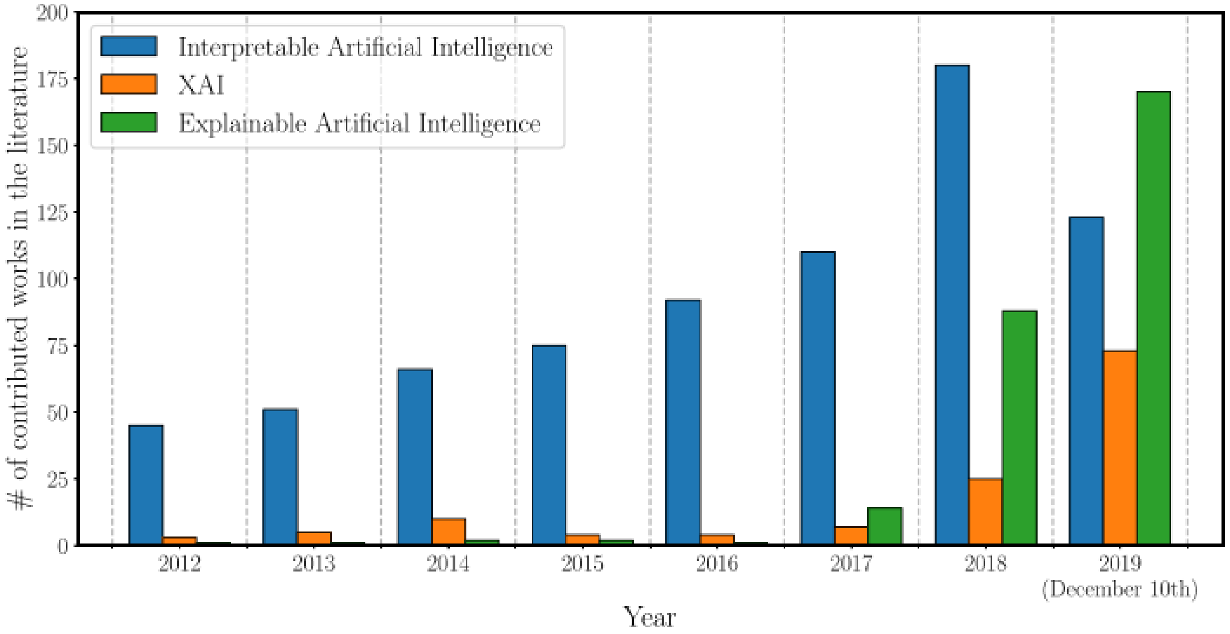
Transparency

A model is transparent if, by itself, it is already understandable.

Explainability

Ability of a model to explain its results to humans:

- How did it arrive at its decisions?
- Which inputs in the data prompted the decision to change?
- Which features have a significant effect on the prediction?

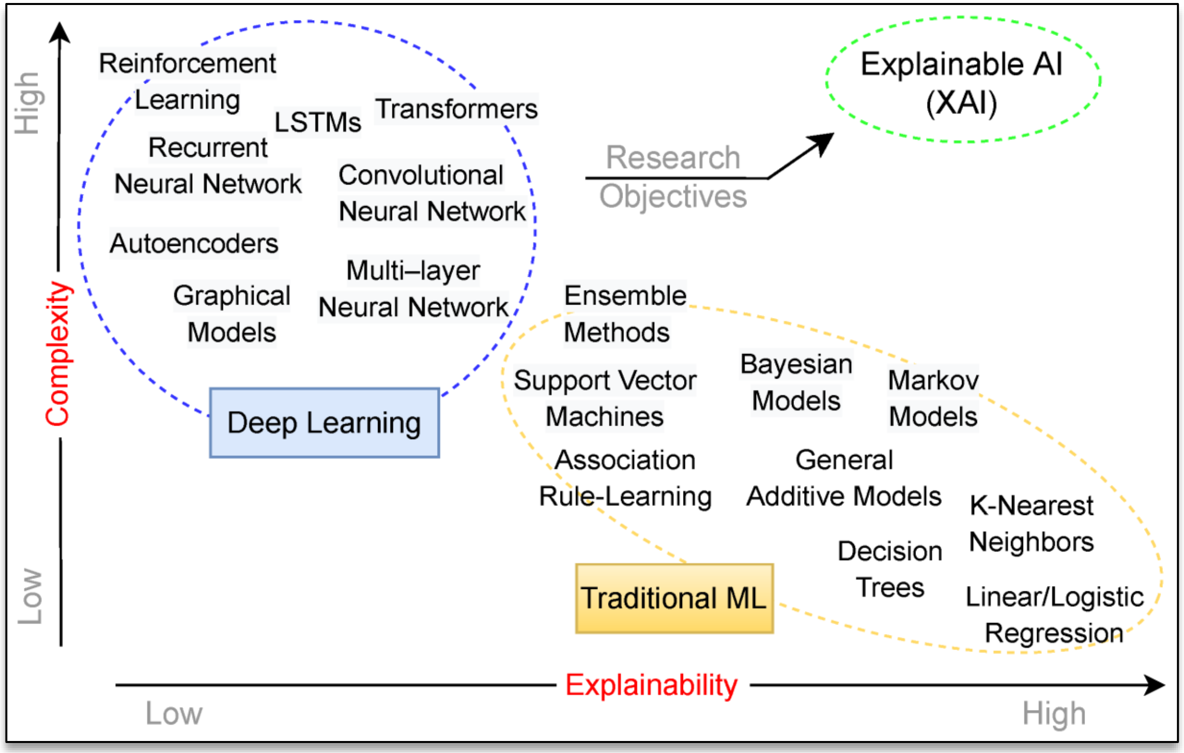


Number of Papers in Literature that mentioned XAI

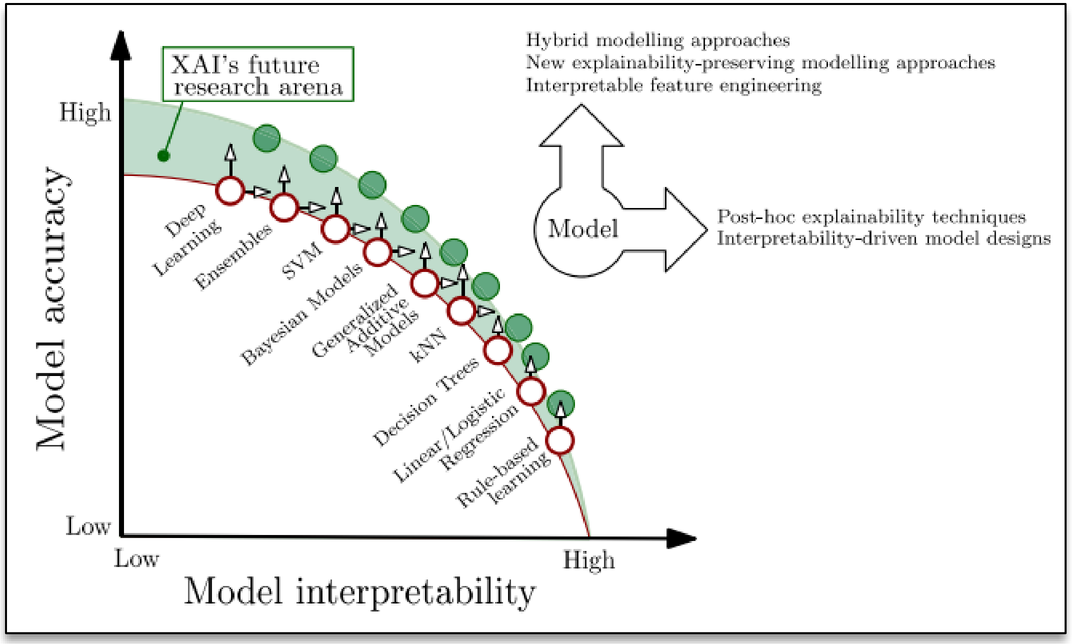
Reference: Arrieta et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion, Vol. 58, June 2020, 82-115. https://doi.org/10.1016/j.inffus.2019.12.012

Explainable AI (XAI)

- It is said that traditional ML models are explainable, but are low-performing.
- On the other hand, deep learning models are not explainable but high-performing.
- Explainable AI aims to provide models that are *explainable yet high-performing*.



Reference: Clement, T.; Kemmerzell, N.; Abdelaal, M.; Amberg, M. XAIR: A Systematic Metareview of Explainable AI (XAI) Aligned to the Software Development Process. Mach. Learn. Knowl. Extr. 2023, 5, 78-108. https://doi.org/10.3390/make5010006



Reference: Arrieta et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion, Vol. 58, June 2020, 82-115. https://doi.org/10.1016/j.inffus.2019.12.012

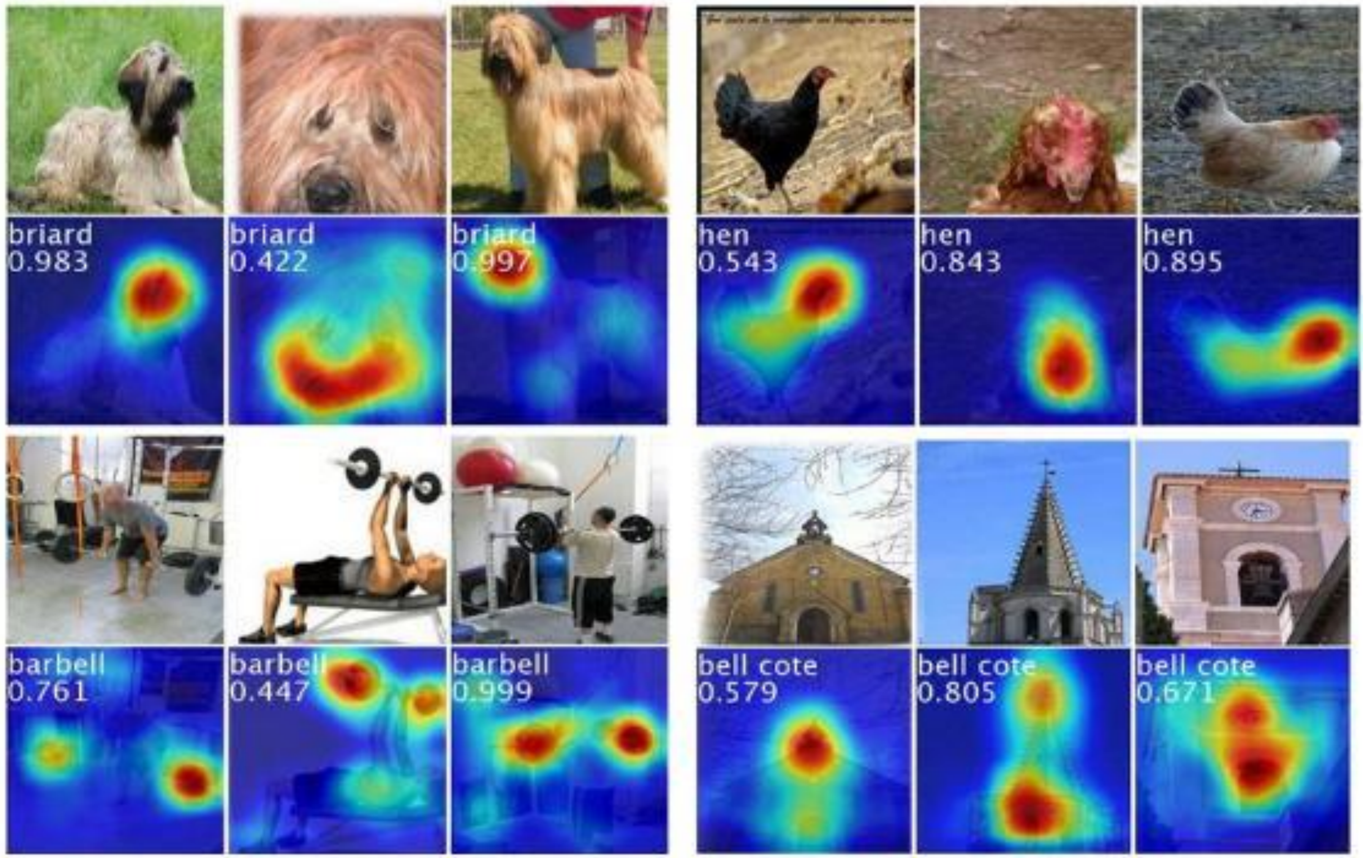
How to Explain ML models?

Intrinsically Explainable
Some models are explainable / interpretable on their own.

Post-hoc Explainability
If an ML model is not transparent, additional analysis must be done *after training the model* in order to provide an explanation.

Some examples of **post-hoc explainability** methods:

- Visual explanation
- Saliency maps (images)
- Model simplification
- Uncertainty Quantification
- **Look at the Features!**
  - Feature Importance
  - Feature Relevance
  - Feature Attribution
  - Feature Significance



https://debuggercafe.com/saliency-maps-in-convolutional-neural-networks/

How to Explain ML models?

ML explainers can be categorized into:

Model-specific Explainers vs. **Model-agnostic Explainers**

The explainability method is only applicable to a certain ML model only.

The explainability method is applicable to any ML model.

Local Explainers vs. **Global Explainers**

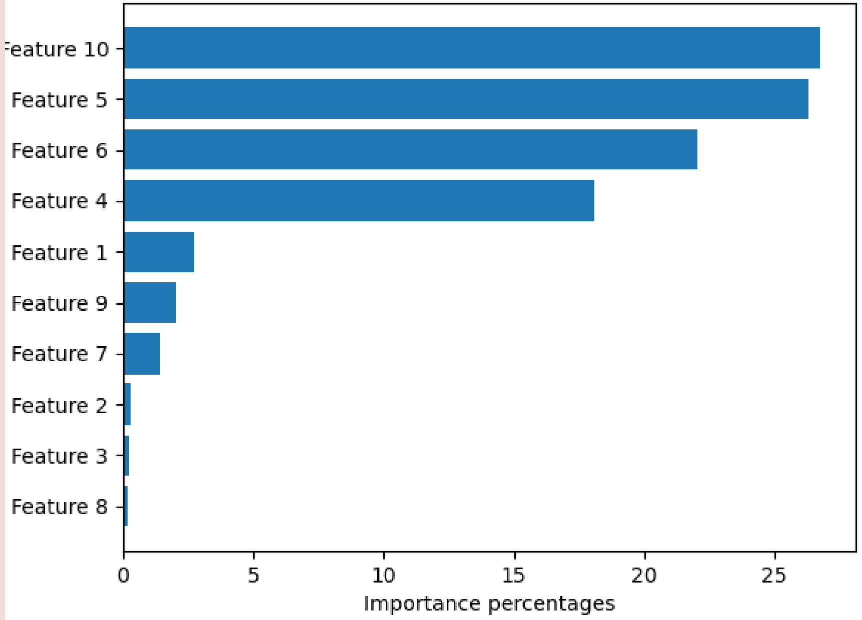
An explanation is provided for a specific data sample only.

An explanation is provided for the model behavior across the entire data space.

Feature Importance

- A mechanism to identify features that have the most *relevant impact* to the model predictions.
- Typically model-agnostic; can be local or global
- Packages: LIME, SHAP, DeepLIFT, etc.

Sample Result



How to Explain ML models?

Permutation Feature Importance (PFI)

- PFI is defined as the decrease in the model score when a single feature value is randomly shuffled.
- If 2 or more features are correlated, PFI is biased to give them lower importance. Cluster correlated features first, then pick only one from each cluster to shuffle.

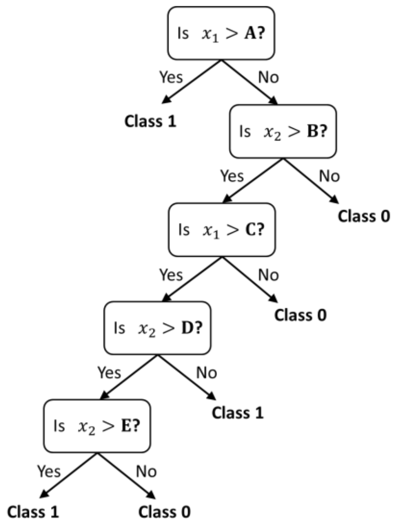
> - Inputs: fitted predictive model m, tabular dataset (training or validation) D.
> - Compute the reference score s of the model m on data D (for instance the accuracy for a classifier or the R^2 for a regressor).
> - For each feature j (column of D):
>   - For each repetition k in $1, \ldots, K$:
>   - Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\tilde{D}_{k,j}$.
>   - Compute the score $s_{k,j}$ of model m on corrupted data $\tilde{D}_{k,j}$.
>   - Compute importance i_j for feature f_j defined as:
>
> $$i_j = s - \frac{1}{K} \sum_{k=1}^{K} s_{k,j}$$

Drop-Column Feature Importance

- Defined as the decrease in the model score when a single feature is removed from the data set.
- Requires model to be re-trained. Hence, it is not purely a post-hoc explainer.

Mean-Decrease-in-Impurity Feature Importance

- Applicable to tree-based models only (e.g. Random Forest)
- Idea: Features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples.
- “Decrease in impurity” quantifies the fraction of the samples a feature contributes to.



How to Explain ML models?

Shapley Additive Explanations (SHAP)

- Uses "Shapley values" from coalitional game theory.

- Feature importance, I_j, is computed as:

$$I_j = \sum_{i=1}^{n} \left|\phi_j^{(i)}\right|$$

where ϕ is the Shapley value (contribution of the jth feature at the ith sample).

- Calculation of the Shapley value involves iterating over all possible subsets of the feature set, then checking the changes in the model score.

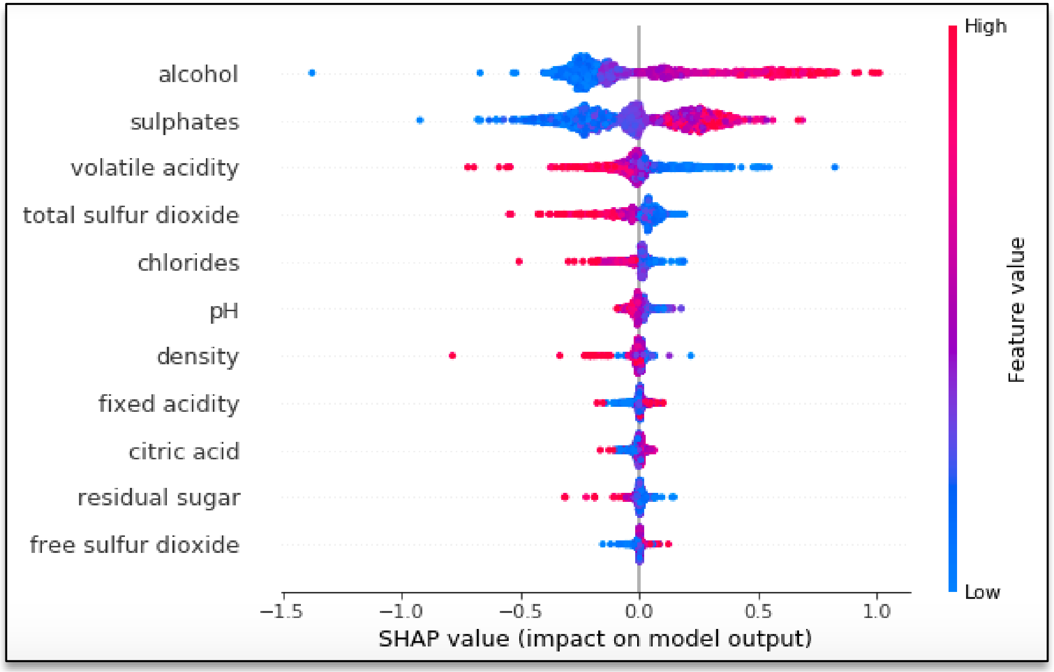
- Computation time grows exponentially with the number of features. Hence, we can approximate the Shapley value using only local samples →Kernel SHAP.

Shapley values are calculated as:

$$\phi_j = \sum_{S \in j} \frac{|S|!\ (|F| - |S| - 1)!}{|F|!} \left[f_{S\cup j}(\boldsymbol{x}_{S\cup j}) - f_S(\boldsymbol{x}_S)\right]$$

F = set of all input features
S = coalition which is a subset of F
$|\cdot|$ = cardinality of the set

$f_{S\cup j}(\boldsymbol{x}_{S\cup j}) - f_S(\boldsymbol{x}_S)$ = marginal contribution of feature j in coalition S.

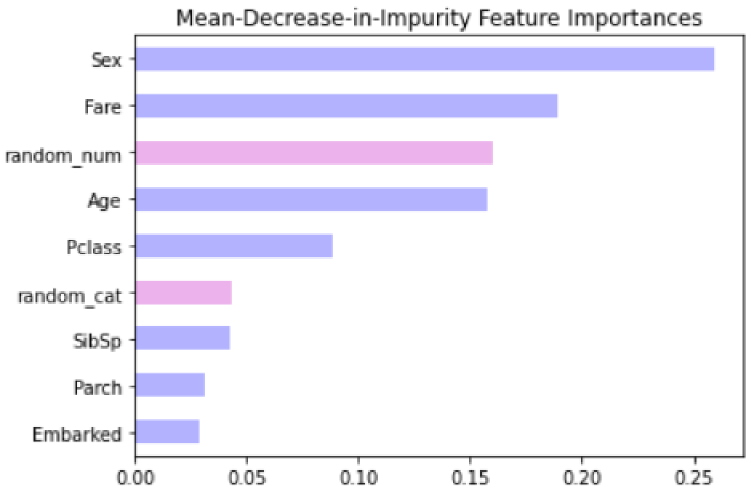


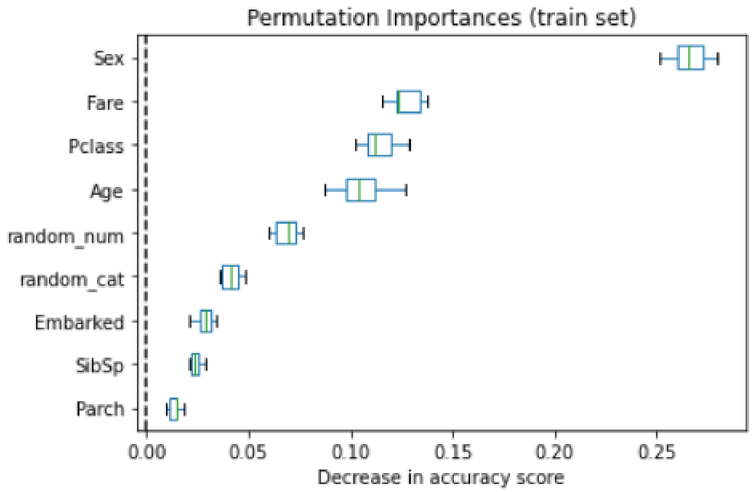
How to Explain ML models?

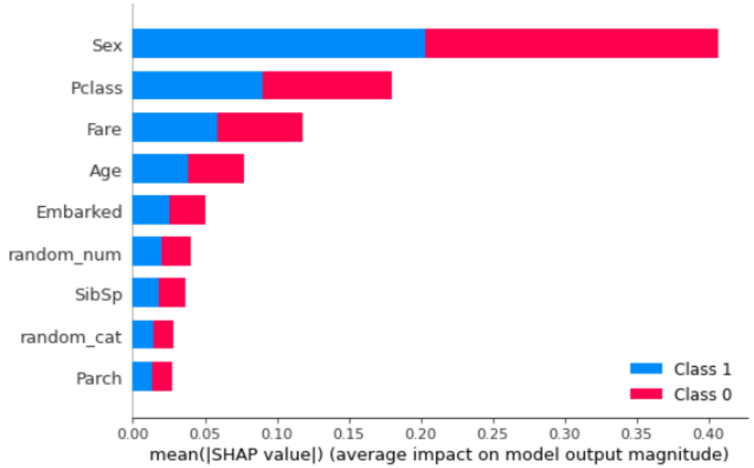
Example:

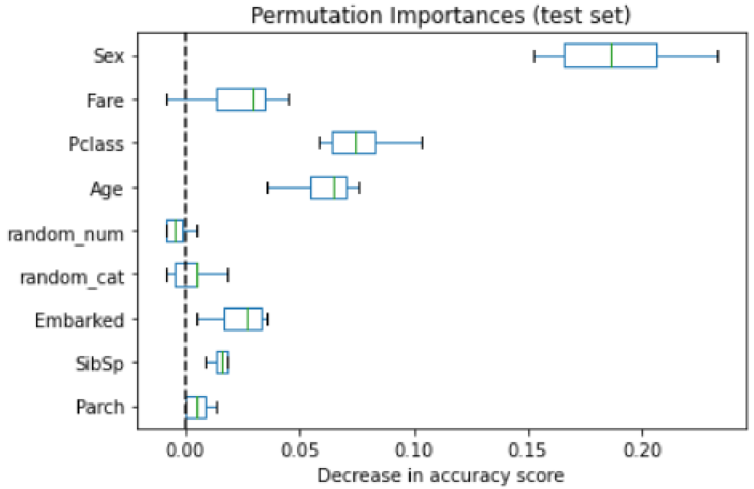
Apply feature importance techniques on a Random Forest classifier trained on the Titanic data set.

Type of feature	Feature	Feature values
	Survived	If survived or no (0 = No, 1 = Yes) (Target variable)
Numeric variables	PassengerId	Unique ID of each passsenger (in integers)
	Age	Age in years
	SibSp	Number of siblings / spouses aboard the Titanic
	Parch	Number of parents / children aboard the Titanic
	Fare	Passenger fare
Strings:	Name	Name of passenger
	Cabin	Cabin number
	Ticket	Ticket number
Categorical variables:	Pclass	Ticket class (1 = 1st, 2 = 2nd, 3 = 3rd)
	Sex	Sex (string : 'male' 'female')
	Embarked	Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)







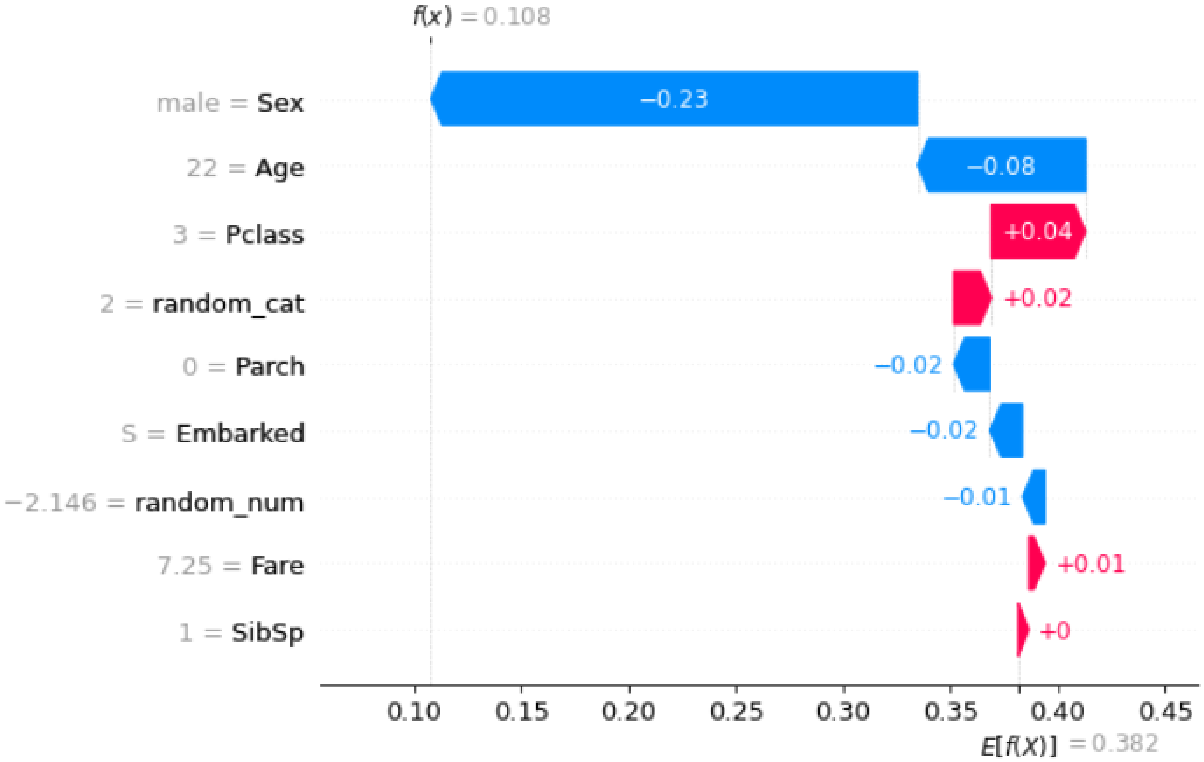


How to Explain ML models?

Example: Titanic Data Set – Local Explanations

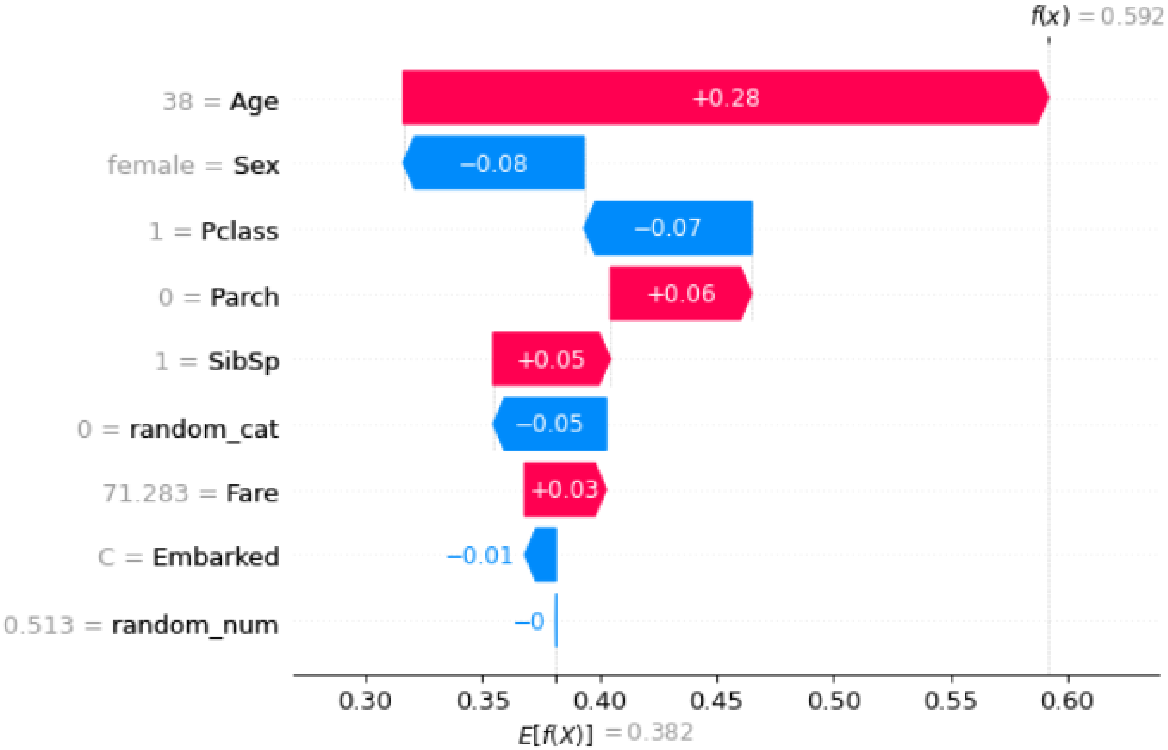
Passenger 288

Random Forest Prediction: Did not survive



Passenger 869

Random Forest Prediction: Survived

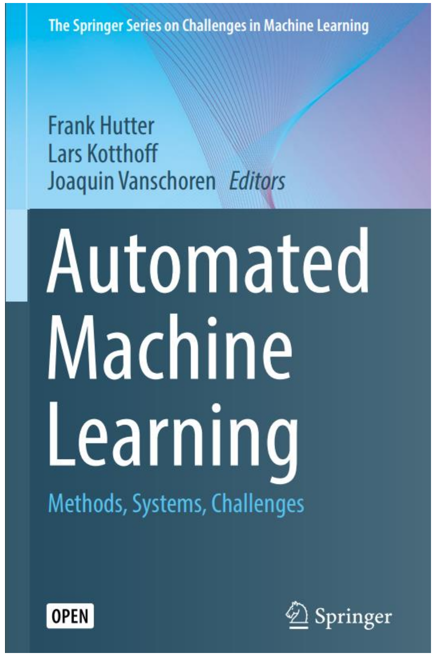


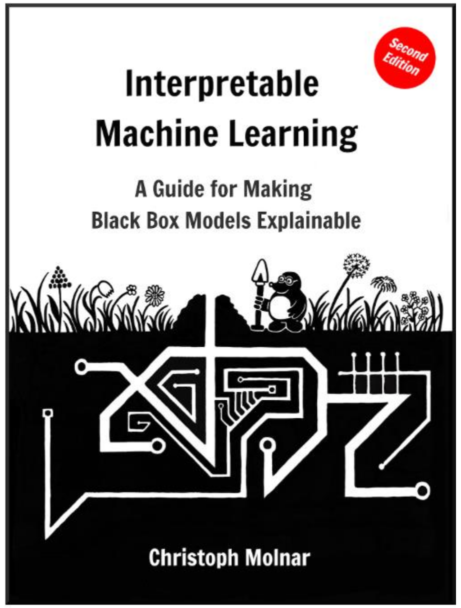
Outline

- AutoML Packages
  - Lazy Predict
  - Auto-sklearn
  - Optuna
  - TPOT
  - PyCaret

- Explainable AI (XAI)
  - Definitions and Concepts
  - Permutation Feature Importance
  - Drop-column Feature Importance
  - Mean-Decrease-in-Impurity
  - Shapley Additive Explanations

Hutter, Kotthoff,
Vanschoren (2019)





Molnar (2022)

Further Reading

- https://www.automl.org/automl/
- https://www.automl.org/wp-content/uploads/2019/05/AutoML_Book.pdf
- https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html
- https://machinelearningmastery.com/auto-sklearn-for-automated-machine-learning-in-python
- Feurer et al. (2015). Efficient and Robust Automated Machine Learning. Advances in Neural Information Processing Systems 28 (NIPS 2015).
- Feurer et al. (2022). Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning. https://arxiv.org/abs/2007.04074
- Olson and Moore (2016). TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning. http://proceedings.mlr.press/v64/olson_tpot_2016.pdf
- Lundberg and Lee (2017). A Unified Approach to Interpreting Model Predictions. https://arxiv.org/abs/1705.07874
- Ribeiro et al. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. https://arxiv.org/abs/1602.04938
- Jang, **Pilario**, Lee, Na. (2023). Explainable Artificial Intelligence for Fault Diagnosis of Industrial Processes. IEEE Trans. On Industrial Informatics. doi: 10.1109/TII.2023.3240601
- Khan, Pao, **Pilario**, Sallih, Rehan (2023). Two-phase flow regime identification using multi-method feature extraction and explainable kernel Fisher discriminant analysis. International Journal of Numerical Methods for Heat & Fluid Flow. Emerald Publishing, Ltd. doi: 10.1108/HFF-09-2023-0526
- Clement, T.; Kemmerzell, N.; Abdelaal, M.; Amberg, M. XAIR: A Systematic Metareview of Explainable AI (XAI) Aligned to the Software Development Process. Mach. Learn. Knowl. Extr. 2023, 5, 78-108. https://doi.org/10.3390/make5010006
- Arrieta et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion, Vol. 58, June 2020, 82-115. https://doi.org/10.1016/j.inffus.2019.12.012