Automated Machine Learning (AutoML) --Fall 2021 IRisk Lab Project

Dec. 9, 2021

Participants: Panyi Dong, Ruiyu Guo, Weiyang Wang

Instructors: Xiaochen Jing, Yuxuan Li, Frank Quan

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

1. Introduction to AutoML

Machine Learning (ML) is a type of Artificial Intelligence (AI)

“Machine learning is the science of getting computers to act without being explicitly programmed.” -- Andrew Ng

Applications: self-driving cars, practical speech recognition, effective web search, image generation, individualized recommendation, ….

Popular in various industry fields and academic research for past few decades

1. Introduction to AutoML

Better data preprocessing techniques

Better selection of ML model

Better selection of hyperparameters

Better prediction/performance

However,

Heavily dependent on expertise & Experience

How to select a proper model from all options

All sorts of hyperparameters to tune (especially neural network)

Difficult for non-experts to gain benefit from ML

1. Introduction to AutoML

Automated Machine Learning (AutoML) comes the rescue

Automated ML processing
(preprocessing, model selection, hyperparameter optimization)

Easy to use

```
save_path = 'agModels-predictClass'  # specifies folder to store trained models
predictor = TabularPredictor(label=label, path=save_path).fit(train_data)
```

Fig. AutoGluon [26]

```
import autosklearn.classification
cls = autosklearn.classification.AutoSklearnClassifier()
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

Fig. auto-sklearn [27]

```
>>> import autoPyTorch
>>> cls = autoPyTorch.api.tabular_classification.TabularClassificationTask()
>>> cls.search(X_train, y_train)
>>> predictions = cls.predict(X_test)
```

Fig. Auto-PyTorch [28]

1. Introduction to AutoML

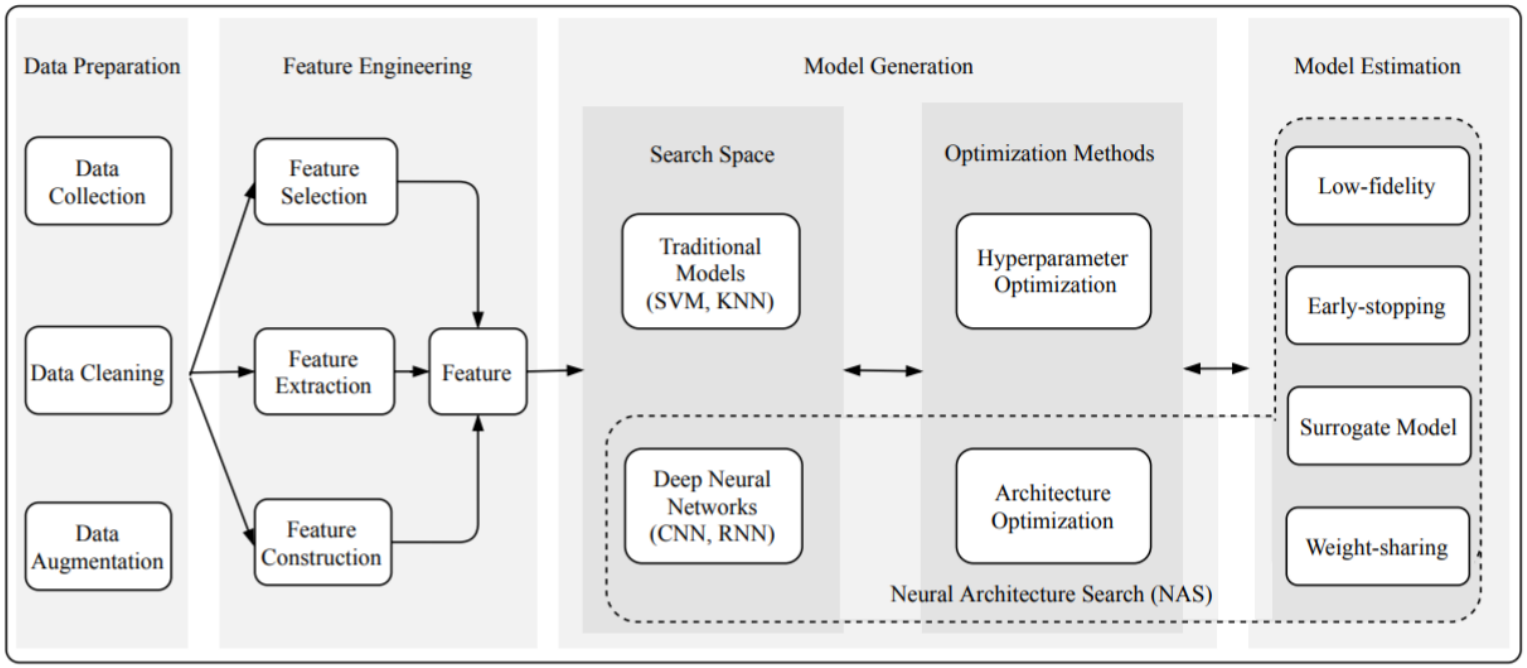


Fig. AutoML underhood [24]

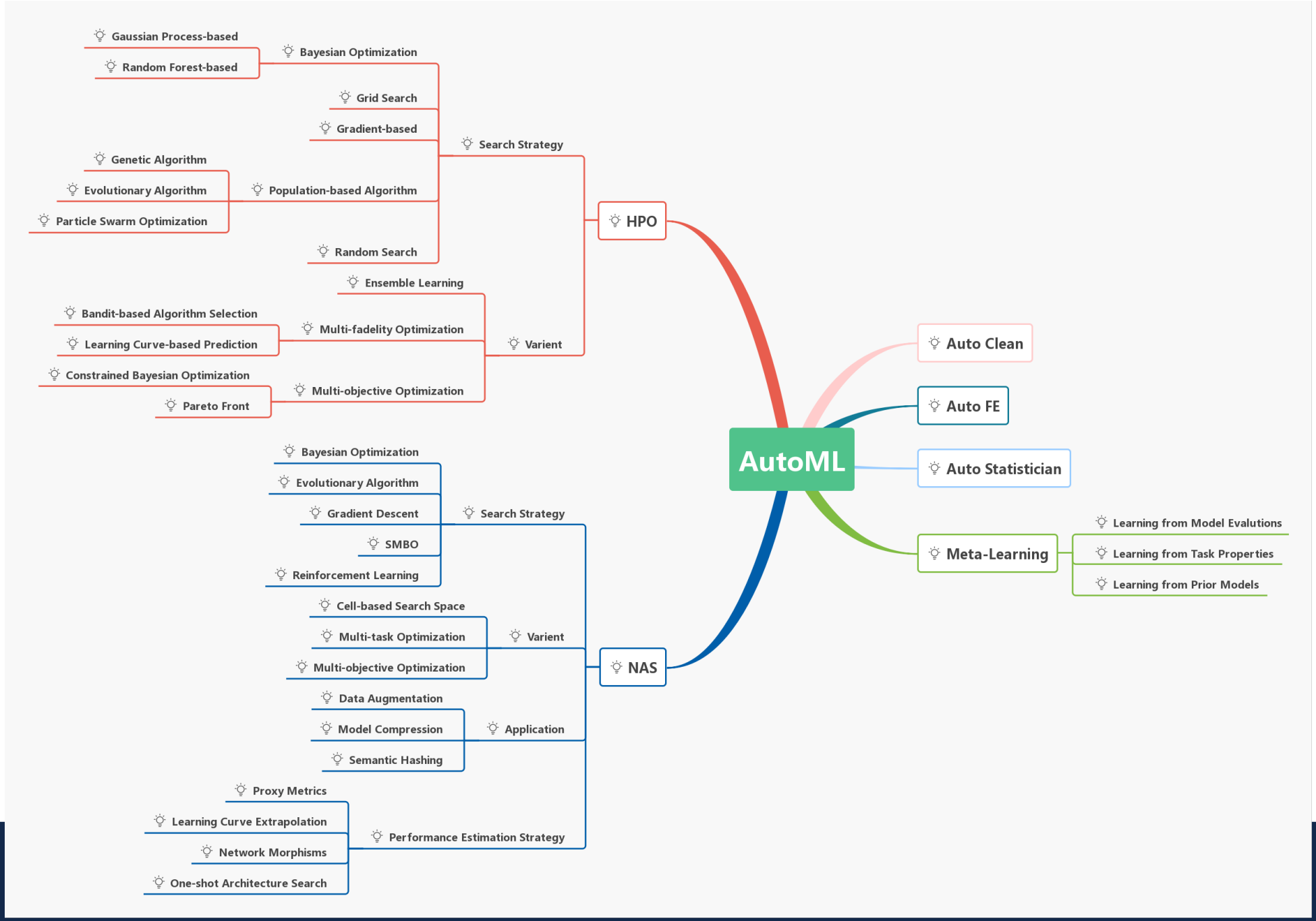


Fig. AutoML mainstream [29]

1. Introduction to AutoML

Keys of AutoML:

Better Performance,
What's the best preprocessing processes; How to build a proper search space;
Provide better model search algorithms, hyperparameter optimization algorithms; …

Higher Efficiency,
AutoML usually estimates multiple processes, how to efficiently complete computation
(or sometimes with limited computation resources)

Robustness,
The goal is to ease expertise required, a robust AutoML for all possible scenarios

This Project

Build a AutoML pipeline

Automated Data Preprocessing, Model Selection & Hyperparameter Optimization

Special focus of Insurance Unbalanced Datasets

Considering applications, aim for tabular datasets, classification/regression tasks (not for computer vision/CV, natural language processing/NLP)

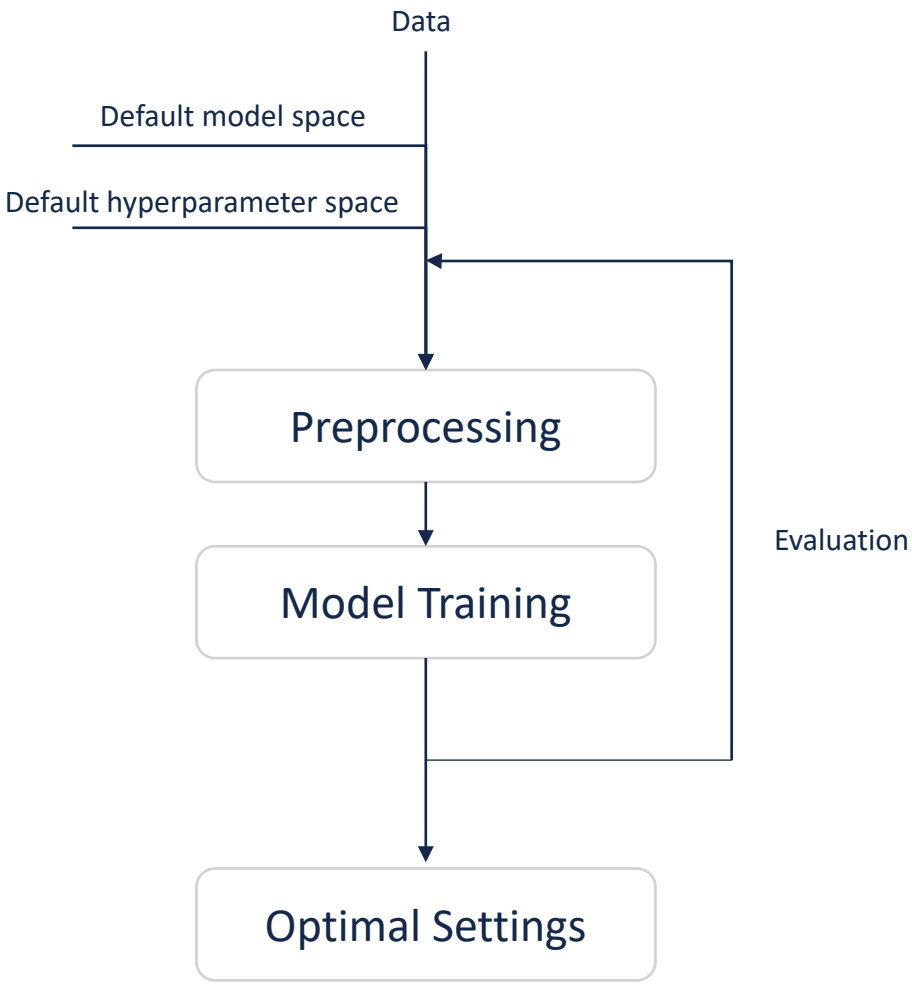
Considering difficulties, use traditional ML models (tree models, SVM, etc.)

2. AutoML Pipeline

1. Data Encoding
2. Data Imputation
3. Data Balancing
4. Data Scaling
5. Data Feature Selection

Preprocessing

6. Classification/Regression Models
7. Model Selection and Hyperparameter Optimization



2.1 Data Encoding

Real-life datasets can contain features of string type.

age	sex	bmi	children	smoker	region	expenses
19	female	27.9	0	yes	southwest	16884.92
18	male	33.8	1	no	southeast	1725.55
28	male	33	3	no	southeast	4449.46
33	male	22.7	0	no	northwest	21984.47
32	male	28.9	0	no	northwest	3866.86
31	female	25.7	0	no	southeast	3756.62
46	female	33.4	1	no	southeast	8240.59
37	female	27.7	3	no	northwest	7281.51
37	male	29.8	2	no	northeast	6406.41

Convert unique strings to numerical types

age	sex	bmi	children	smoker	region	expenses
19	0	27.9	0	1	0	16884.92
18	1	33.8	1	0	1	1725.55
28	1	33	3	0	1	4449.46
33	1	22.7	0	0	2	21984.47
32	1	28.9	0	0	2	3866.86
31	0	25.7	0	0	1	3756.62
46	0	33.4	1	0	1	8240.59
37	0	27.7	3	0	2	7281.51
37	1	29.8	2	0	3	6406.41

String types (sex, smoker and region above) sometimes can not be dealt with easily.

2.2 Data Imputation

Some of datasets contains missing values.

Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Commision (in value)	Gender	Age
CBH	Travel Agency	Offline	Comprehensive Plan	No	186	MALAYSIA	-29	9.57	F	81
CBH	Travel Agency	Offline	Comprehensive Plan	No	186	MALAYSIA	-29	9.57	F	71
CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	65	AUSTRALIA	-49.5	29.7		32
CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	60	AUSTRALIA	-39.6	23.76		32
CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	79	ITALY	-19.8	11.88		41
JZI	Airlines	Online	Value Plan	No	66	UNITED STATES	-121	42.35	F	44
CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	47	UNITED STATES	-39.6	23.76		32
CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	63	AUSTRALIA	-108.9	65.34		29

Usually,

(1) Dataset contains missing values will return values not expected

(2) models can not handle datasets with missing values

Practice

(1) Delete entries with missing values

(2) Common solution: fill these missing values

2.2 Data Imputation

2.2.1 Simple Imputation

To fill the missing values using mean/zero/median/etc.

Efficient but not consider the spatial distribution of each observation.

May not accurately represent the missing values.

2.2.2 Joint Imputation

Consider the entire dataset as multivariate distribution.

Calculate distribution parameters using non-missing observations and fill missing values.

Observations may not accurately described by one distribution.

2.2 Data Imputation

2.2.3 Expectation Maximization (EM)

Use EM algorithm to maximize the likelihood of distribution.

Use the distribution parameters from EM algorithm to generate the missing values

2.2.4 KNN Imputation

Use K Nearest Neighbor (kNN) to find most suitable imputation for missing values.

In algorithm, use Simple Imputation as initial guess and fill missing values iteratively.

2.2.5 Miss Forest Imputation

Similar to KNN Imputation but use Random Forest as imputation method.

Order missing columns by amount of missing.

2.2 Data Imputation

2.2.6 MICE

Multiple Imputation by Chained Equation (MICE) runs Linear Regression/Logistic Regression/Lasso on the non-missing values and use the prediction to fill the missing values.

As multiple imputation method, MICE will impute multiple values.

2.2.7 GAIN

Generative Adversarial Imputation Nets (GAIN) is a variation of GAN where to train a Generator (G) and Discriminator (D) to impute missing values.

Generator tends to create more imputation while Discriminator minimize the loss, a min-max problem

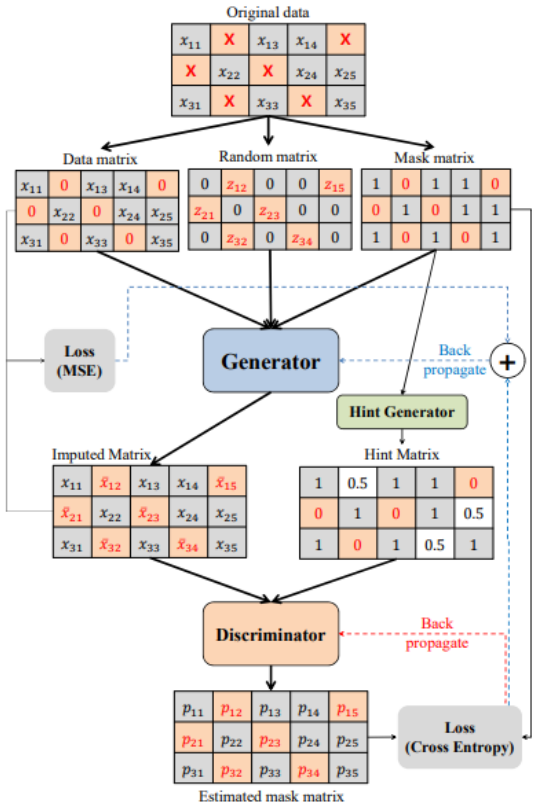
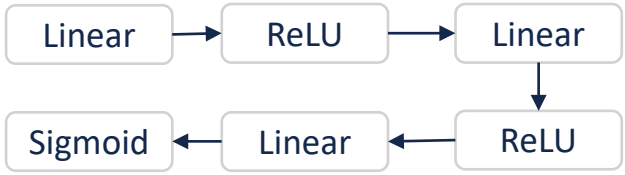


Fig. Architecture of GAIN [5]

Generator/Discriminator Net Structure



2.3 Data Balancing

Sometimes, the dataset is unbalanced where one class of observations take majority of the entire datasets.

The phenomenon is common in insurance datasets. E.g., claims only consist of small number of total policyholders.

The unbalanced nature may impact the accuracy of the models.

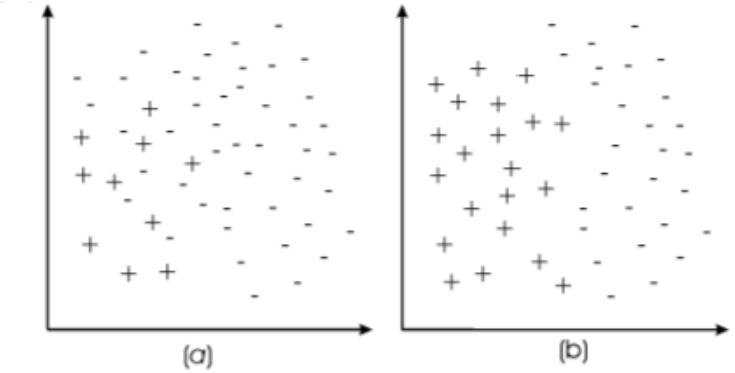


Fig. Spatial illustration of unbalanced datasets (a) and balanced datasets (b). [6]

A balancing method is critically important for the data preprocessing in unbalanced datasets.

In the pipeline, all methods belong to over-sampling/under-sampling class.

2.3 Data Balancing

2.3.1 Simple Random Over-Sampling

Randomly select one observations from minority class as new observation.

Percentage of minority class increases.

2.3.2 Simple Random Under-Sampling

Randomly eliminate observations from majority class.

Disadvantages: (1) importance of the observations
(2) copying/eliminating exact observations, no information learnt

2.3 Data Balancing

2.3.3 Tomek Link

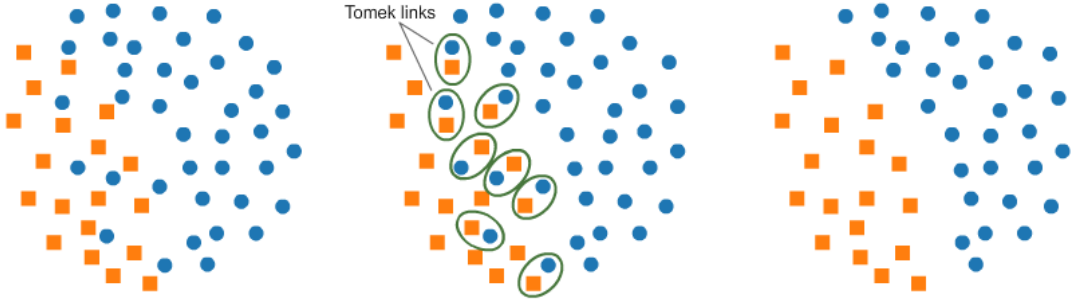


Fig. Tomek Link [31]

An Under-Sampling method.

Tomek Link define as E_i, E_j belong to different class and $d(E_i, E_j)$ is smallest.

Remove noise observations and decision significant observations.

2.3.4 Edited Nearest Neighbor

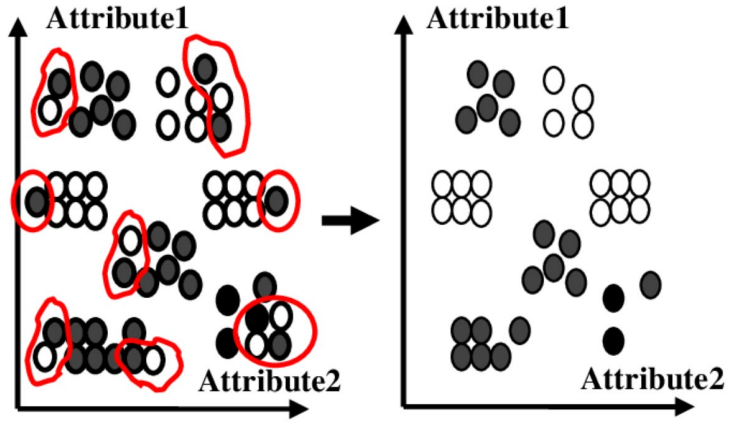


Fig. Edited Nearest Neighbor [32]

Edited Nearest Neighbor (ENN) is an Under-Sampling method.

If the observation belongs to minority class and prediction by kNN disagrees, remove k neighbors; if observation belongs to majority class and prediction by kNN disagrees, remove the observation.

2.3 Data Balancing

2.3.5 Condensed Nearest Neighbor

Condensed Nearest Neighbor (CNN) is an Under-Sampling method.

Select a subset $\hat{E} \subseteq E$ (contains all minority class and some of the majority class) where $\hat{E}$ can predict all observations correctly using 1-NN.

2.3.6 One Sided Selection

One Sided Selection (OSS) is an Under-Sampling method with combining Tomek Link and CNN.

OSS take advantages from both methods. First remove noise and boundary significant observations (Tomek Link), then remove observations distant from decision boundary (CNN).

2.3 Data Balancing

2.3.7 CNN-Tomek Link

Compared with OSS (Tomek Link-CNN), CNN-Tomek Link can process more efficiently.

Tomek Link need the calculation for distance between every observations, removing some of the observations using CNN can reduce the computation.

2.3.8 Smote

Synthetic Minority Over-Sampling Technique (Smote) is an Over-Sampling method.

Smote intends to generate more realistic observations using spatial interpolation.

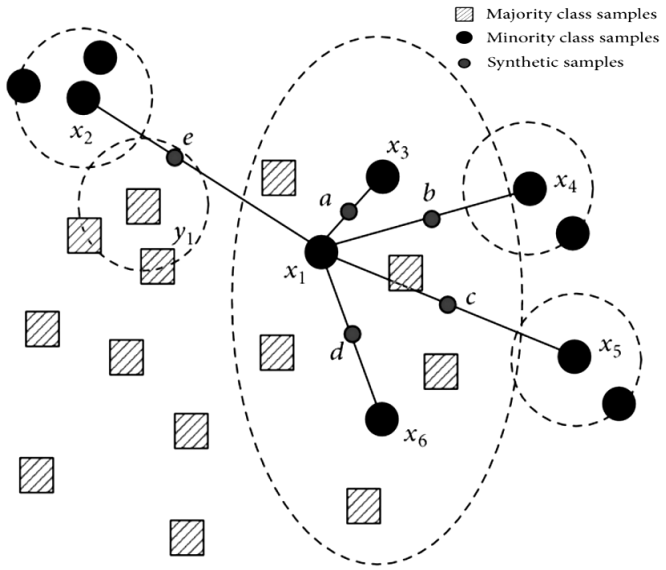


Fig. Illustration of Smote [30]

2.3 Data Balancing

2.3.9 Smote-Tomek Link

Smote-Tomek Link is an Over-Sampling, Under-Sampling combined method.

Sometimes, synthetic observations generated are invading into wrong class space.

Tomek Link can solve the problem of data skewness.

2.3.10 Smote-ENN

Similar idea with Smote-Tomek Link, but ENN removes more observations.

Smote-ENN provides more in depth data cleaning.

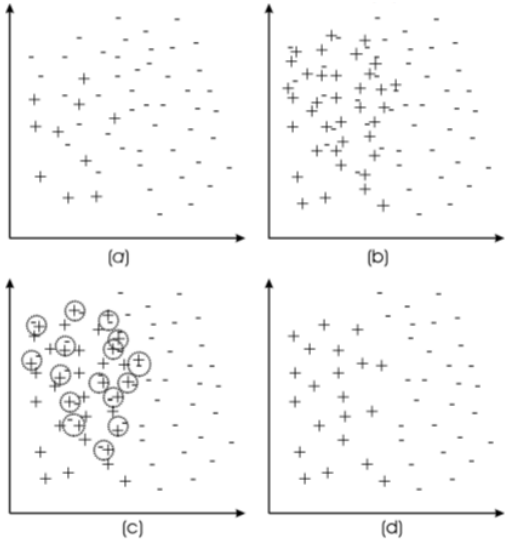


Fig. Original unbalanced datasets (a); Over-Sampling datasets (b); Tomek Link Identification (c); Removal of noise and boundary observations (d). [6]

2.4 Data Scaling

Some of the features in datasets varies. E.g., Price of houses can be 100 thousands, or can be 10 millions.

The situation can cause convergence issues on some algorithms.

However, scaling can avoid the convergence problem. Sometimes, scaling can provide a better model accuracy.

2.4 Data Scaling

2.4.1 Standardization

Feature are standardized using $(x - \bar{x})/\sigma_x$

2.4.2 Normalization

Feature are normalized using x/x_{max} into unit scale

2.4.3 Robust Scaling

Feature are scaled by range of two feature percentiles

2.4.4 Min-Max Scaling

Feature are scaled by min/max values

2.4.5 Winsorization

Removes extreme observations (outliers)

2.5 Data Feature Selection

Some datasets contains numerous features.

However, not all of them are relevant for the response. E.g., Age and medical history/birthing moth to whether a person will die from heart attack

Feature Selection intends to select only relevant features for the model to reduce computation required while still maintains performance.

Some common methods like: SVM (Support Vector Machine), Percentile for feature selection, etc.

2.5 Data Feature Selection

2.5.1 Feature Filter

Feature filter select relevant features by scoring features and selecting highest score features.

(1) Pearson Correlation Coefficient

$$R(i) = \frac{Cov(X_i, Y)}{\sqrt{Var(X_i)Var(Y)}}$$

(2) Mutual Information using Shannon Entropy

$$H(Y) = -\sum_{y} p(y)\log(p(y))$$

$$H(Y|X) = -\sum_{x}\sum_{y} p(x,y)\log(p(y|x))$$

$$I(X,Y) = H(Y) - H(Y|X)$$

2.5 Data Feature Selection

2.5.2 ASFFS

Adaptive Sequential Forward Floating Search (ASFFS) is a variation of sequential feature selection method.

ASFFS iteratively performs forward phase and backward until target reached. Forward phase adds features, backward phase removes features.

Methods like Sequential Forward Selection (SFS) only consider adding/removing one feature at a time, without correlation between features.

ASFFS will adaptively change number of features consider at a time.

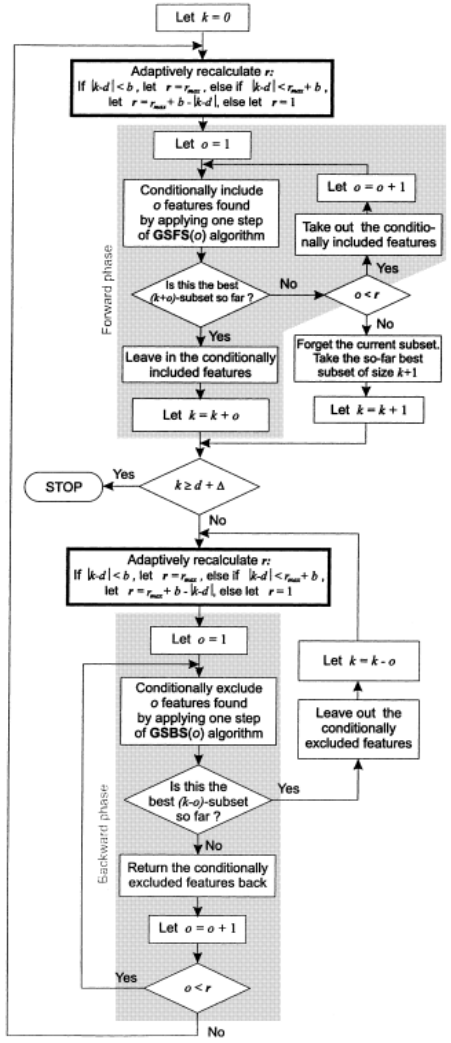
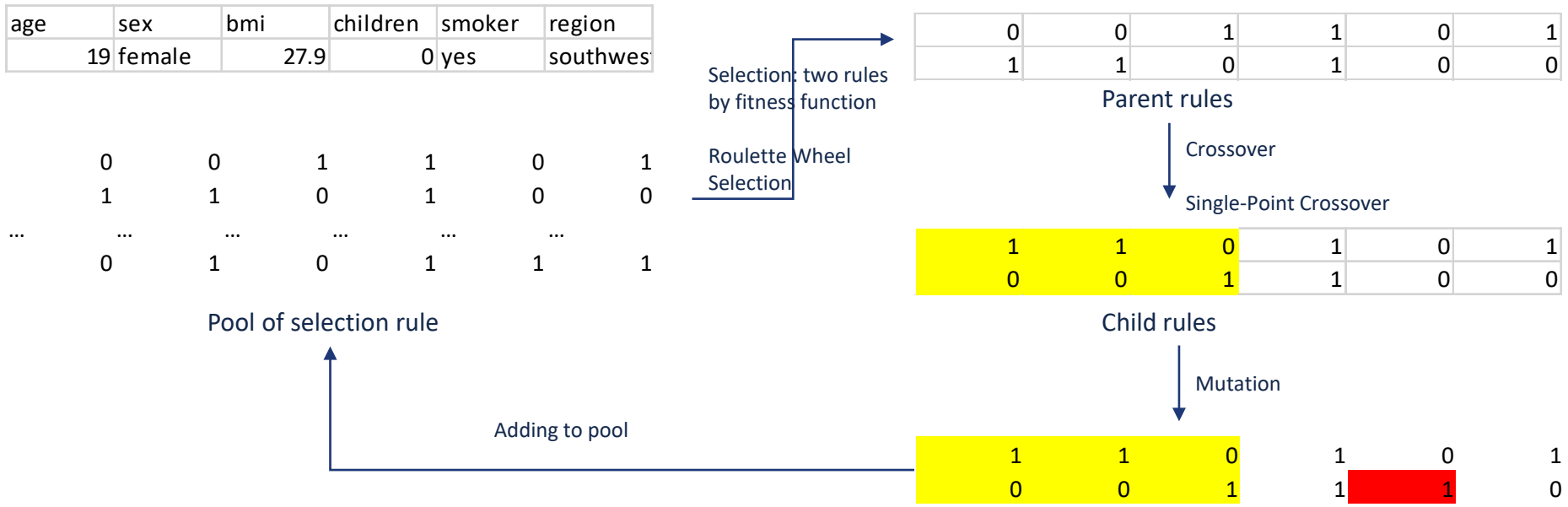


Fig. Workflow of ASFFS [12]

2.5 Data Feature Selection

2.5.3 GA

Genetic Algorithm (GA) is a popular algorithm inspired by natural DNA replication/mutation.



During the process, all rules are evaluated by fitness function with their performance and assigned a possibility of selection.

After generations, best performing selection rule will be selected.

2.6 Classification/Regression Models

2.6.1 Classification Models

(1) Adaboost Classifier: Combine many week classifiers. During training, all misclassified observations will be increased at each step.

(2) Bernoulli Naïve Bayes: Using Bernoulli kernel for Bayesian Theorem to calculate probability of belonging certain class.

(3) Gaussian Naïve Bayes: Using Gaussian kernel for Bayesian Theorem to calculate probability of belonging certain class.

(4) Multinomial Naïve Bayes: Using Multinomial kernel for Bayesian Theorem to calculate probability of belonging certain class.

2.6 Classification/Regression Models

2.6.1 Classification Models

(5) Decision Tree: Grow a decision tree to assign each observations to a leaf node.

(6) Random Forest: Grow multiple trees, where randomly select subset of features at each split.

(7) Extra Trees Classifier: Use subsamples of datasets to train multiple decision trees and average results.

(8) Gradient Boosting Classifier: Iteratively build decision trees based on residuals from previous trees. Create an ensemble of trees.

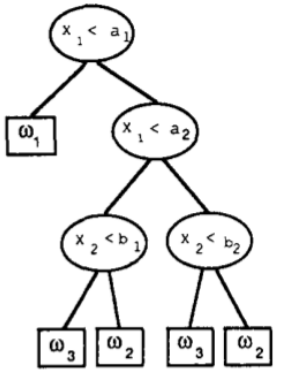


Fig. Decision Tree Split [17]

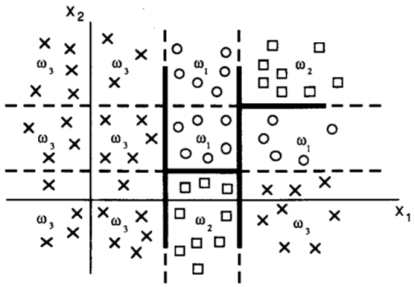


Fig. Split On Data Points [17]

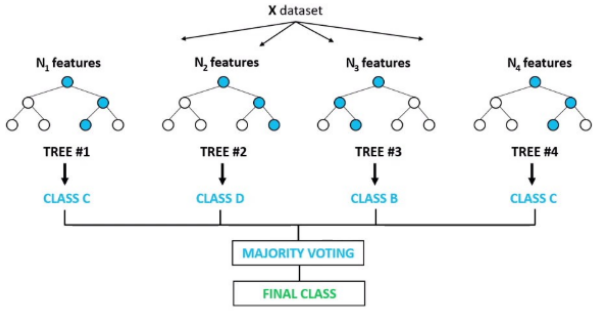


Fig. Random Forest [36]

2.6 Classification/Regression Models

2.6.1 Classification Models

(9) K Nearest Neighbor Classifier (KNN): Use k nearest neighbors with weighted average to predict the category.

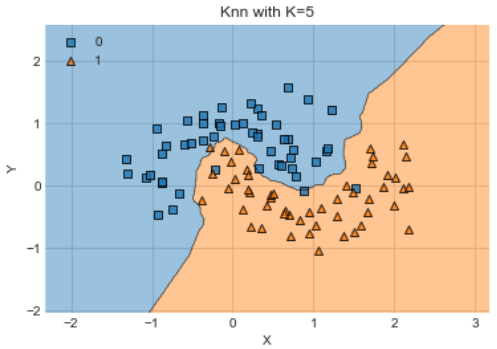


Fig. kNN [33]

(10) Linear Discriminant Analysis (LDA): Assume observations belong to different multivariate normal distributions with same covariance matrix.

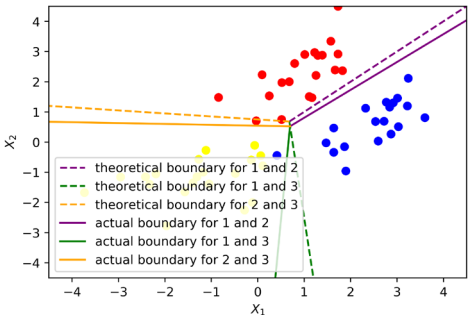


Fig. Linear Discriminant Analysis

(11) Quadratic Discriminant Analysis (QDA): Assume observations belong to different multivariate normal distribution with different mean vector and covariance matrix.

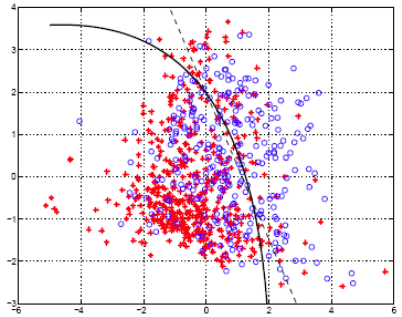


Fig. Quadratic Discriminant Analysis [35]

2.6 Classification/Regression Models

2.6.1 Classification Models

(12) LibLinear_SVC: Use LibLinear library to solve Support Vector Machine with linear kernel.

(13) LibSVM_SVC: Use LibSVM library to sovle Support Vector Machine with non-linear kernel.

(14) MultiLayer Perceptron (MLP): A Feedforward Neural Network structure, consists of multiple fully connected layers and non-linear activation functions.

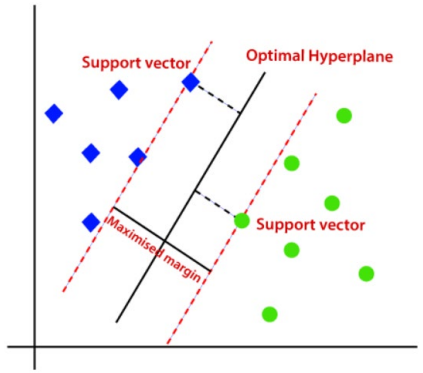


Fig. Support Vector Machine [34]

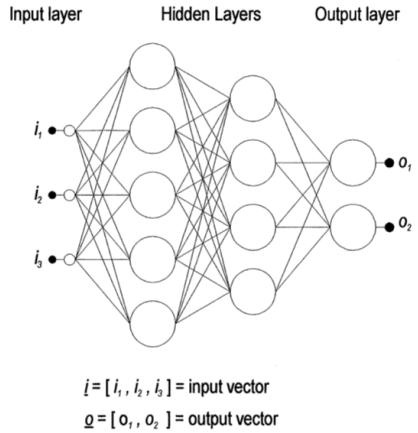


Fig. MultiLayer Perceptron [20]

2.6 Classification/Regression Models

2.6.1 Classification Models

(15) Passive Aggressive: A online learning algorithm. Feed model with small batch of observations, if predict correctly, retain the model (passive); if not, update to adjust the misclassified observations (aggressive).

(16) Stochastic Gradient Descent (SGD): Use random observation to calculate gradient and minimize the loss function.

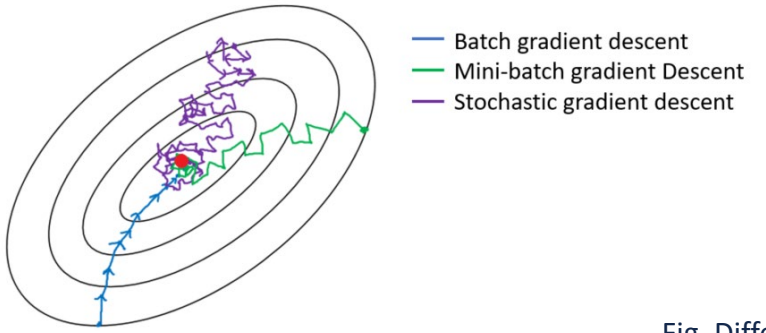


Fig. Different Gradient Descent [37]

2.6 Classification/Regression Models

2.6.2 Regression Models

(1) Adaboost Regressor

(2) ARD Regressor: Bayes ARD Regression, weights of regression assumed to Gaussian distributed.

(3) Decision Tree Regressor

(4) Extra Trees Regressor

(5) Gaussian Process

(6) Gradient Boosting

2.6 Classification/Regression Models

2.6.2 Regression Models

(7) K Nearest Neighbors

(8) LibLinear_SVR

(9) LibSVM_SVR

(10) MultiLayer Perceptron (MLP)

(11) Random Forest

(12) Stochastic Gradient Descent

2.7 Model Selection & Hyperparameter Optimization

In the pipeline, Hyperopt is used to help achieve the goal of model selection and Hyperparameter optimization.

For all the pipeline components, we pre-define the hyperparameter space and allow Hyperopt to do rest automatically.

```
{
    "model": "AdaboostClassifier",
    "n_estimators": scope.int(
        hp.quniform("AdaboostClassifier_n_estimators", 10, 500, 1)
    ),
    "learning_rate": hp.uniform("AdaboostClassifier_learning_rate", 0.01, 2),
    "algorithm": hp.choice("AdaboostClassifier_algorithm", ["SAMME", "SAMME.R"]),
    # for base_estimator of Decision Tree
    "max_depth": scope.int(hp.quniform("AdaboostClassifier_max_depth", 1, 10, 1)),
},
```

Fig. Hyperparameter space of Adaboost Classifier

```
Encoding method: DataEncoding
Encoding Hyperparameters:{}

Imputation method: no_processing
Imputation Hyperparameters:{}

Balancing method: no_processing
Balancing Hyperparameters:{}

Scaling method: MinMaxScale
Scaling Hyperparameters:{}

Feature Selection method: select_percentile_classification
Feature Selection Hyperparameters:{'percentile': 23, 'score_func': 'mutual_info'}

Classification model: KNearestNeighborsClassifier
Classifier Hyperparameters:{'n_neighbors': 98, 'p': 1, 'weights': 'uniform'}
```

Fig. Hyperparameter selected for a task

2.7 Model Selection & Hyperparameter Optimization

Some features of the pipeline:

(1) Classification/Regression task can be selected automatically by the response.

(2) Temporary hyperparameter settings, preprocessed datasets can be stored for later examination.

(3) The model can be saved after training for easier reproduction.

3 AutoML Test

3.1 Heart Failure Prediction (Classification)

Kaggle Heart Failure Prediction dataset

https://www.kaggle.com/fedesoriano/heart-failure-prediction

In [2]:
```
# Load data
database = load_data(data_type = '.csv').load('example_data', ['heart'])
database_names = [*database]
```

In [3]:
```
database['heart'].head(5)
```

Out[3]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Features like Age, Sex, Resting Blood Pressure, Fasting Blood Sugar, …

Fig. Load dataset

3 AutoML Test

3.1 Heart Failure Prediction (Classification)

```
Encoding method: DataEncoding
Encoding Hyperparameters:{}

Imputation method: no_processing
Imputation Hyperparameters:{}

Balancing method: OneSidedSelection
Balancing Hyperparameters:{'imbalance_threshold': 0.8953006180841536}

Scaling method: Normalize
Scaling Hyperparameters:{}

Feature Selection method: GeneticAlgorithm
Feature Selection Hyperparameters:{}

Classification model: AdaboostClassifier
Classifier Hyperparameters:{'algorithm': 'SAMME', 'learning_rate': 1.1121565970253824, 'max_depth': 4,
'n_estimators': 56}
```

Fig. Optimal Hyperparameter setting

781 train observations reduce to 664

Fig. Preprocessed datasets

Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0.818181818	1	0	0.5	0	1	0.5	0.53960396	0	-0.14516129	0.5	1
0.454545455	0	0	0.7	0.27694859	0	0.5	0.742574257	0	0	1	0
0.792207792	1	0	0.55	0	1	0.5	0.534653465	1	0.322580645	0	1
0.545454545	1	0.666666667	0.8	0.243781095	0	0.5	0.722772277	0	0	1	0
0.363636364	1	0.333333333	0.65	0.218905473	0	0	0.915841584	0	0	1	0

3 AutoML Test

3.1 Heart Failure Prediction (Classification)

In [5]:
```
# train/test split
train_X, test_X, train_y, test_y = train_test_split(
    database['heart'][features], database['heart'][[response]]
)
```

In [6]:
```
# fit AutoML model
mol = AutoML(seed = 1)
mol.fit(train_X, train_y)
```

Out[6]:
```
<My_AutoML._model_selection.AutoML at 0x7f8c36beaf70>
```

In [7]:
```
from sklearn.metrics import accuracy_score
y_pred = mol.predict(test_X)
accuracy_score(y_pred, test_y)
```

Out[7]:
```
0.8686131386861314
```

Fig. Train/Test AutoML pipeline

3 AutoML Test

3.1 Insurance Premium Prediction (Regression)

Insurance premium Prediction

https://www.kaggle.com/noordeen/insurance-premium-prediction

In [2]:
```
# Load data
database = load_data(data_type = '.csv').load('example_data', ['insurance'])
database_names = [*database]
```

In [3]:
```
database['insurance'].head(5)
```

Out[3]:

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86

Fig. Load dataset

3 AutoML Test

3.1 Insurance Premium Prediction (Regression)

In [5]:
```
# train/test split
train_X, test_X, train_y, test_y = train_test_split(
    database['insurance'][features], database['insurance'][[response]]
)
```

In [6]:
```
# fit AutoML model
mol = AutoML(seed = 1)
mol.fit(train_X, train_y)
```

Out[6]:
```
<My_AutoML._model_selection.AutoML at 0x7f4e386e77f0>
```

In [7]:
```
# predict using AutoML model
from sklearn.metrics import mean_squared_error
y_pred = mol.predict(test_X)
mean_squared_error(y_pred, test_y)
```

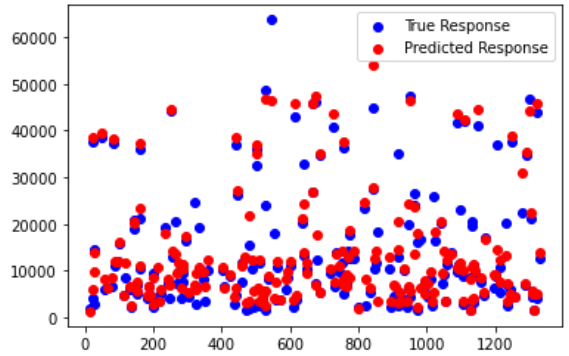
Out[7]:
```
21309279.613129355
```

Fig. Train/Test AutoML pipeline

3 AutoML Test

3.1 Insurance Premium Prediction (Regression)

In [8]:
```
import matplotlib.pyplot as plt
plt.figure()
plt.scatter(list(test_y.index), test_y.values, color = 'blue', label = 'True Response')
plt.scatter(list(test_y.index), y_pred, color = 'red', label = 'Predicted Response')
plt.legend()
plt.show()
```



In [9]:
```
from sklearn.metrics import r2_score
r2_score(y_pred, test_y)
```

Out[9]:
```
0.8551737495890323
```

Fig. Further on performance

3 AutoML Test

3.1 Test on some datasets

Datasets	Task Type	Runtime/s	Criterion	Performance
Employee Future	Classification	142.9	accuracy score	0.8508
Heart Failure	Classification	40.6	accuracy score	0.8686
Insurance Premium	Regression	229.7	mean squared error	21309279.6131
			r2 score	0.8552
Imbalanced Insurance	Classification	over 6000	accuracy score	0.8965
Travel Insurance	Classification	81.9	accuracy score	0.8121
Credit	Classification	256.3	accuracy score	0.982
Medical Premium	Regression	393.1	mean squared error	6927603.5311
			r2 score	0.7912

Summary

The AutoML pipeline developed has special concentration on unbalanced data.

For now, the AutoML pipeline can tune our model performance at an acceptable level.

For improving performance, setting more evaluation rounds is an option, or use the AutoML provided optimal hyperparameters as a baseline for further manual tuning.

Future Improvements

(1) Define a better correlated hyperparameter space

(2) The efficiency of the pipeline can be improved since it can not handle large datasets well. Perhaps a parallelism plan or GPU acceleration can be adopted to reduce the runtime.

(3) For now, all regression/classification models are selected from auto-sklearn, which mostly focus on traditional Machine Learning models (too many hyperparameters). But there are work focusing on automated neural network, perhaps it can help improve the performance.

Reference

[1] Müller, A. C. & Guido, S. *Introduction to machine learning with Python: a guide for data scientists* (" O'Reilly Media, Inc.", 2016).

[2] Feurer, M. & Hutter, F. Hyperparameter optimization. In *Automated machine learning*, 3–33 (Springer, Cham, 2019).

[3] Stekhoven, D. J. & B¨uhlmann, P. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics* **28**, 112–118 (2012).

[4] Azur, M. J., Stuart, E. A., Frangakis, C. & Leaf, P. J. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research* **20**, 40–49 (2011).

[5] Yoon, J., Jordon, J. & Schaar, M. Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, 5689–5698 (PMLR, 2018).

[6] Batista, G. E., Prati, R. C. & Monard, M. C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter* **6**, 20–29 (2004).

[7] Tomek, I. *et al.* Two modifications of cnn. (1976).

Reference

[8] Wilson, D. L. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics* 408–421 (1972).

[9] Hart, P. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory* **14**, 515–516 (1968).

[10] Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research* **16**, 321–357 (2002).

[11] Chandrashekar, G. & Sahin, F. A survey on feature selection methods. *Computers & Electrical Engineering* **40**, 16–28 (2014).

[12] Somol, P., Pudil, P., Novovi˘cov´a, J. & Paclık, P. Adaptive floating search methods in feature selection. *Pattern recognition letters* **20**, 1157–1163 (1999).

[13] Tan, F., Fu, X., Zhang, Y. & Bourgeois, A. G. A genetic algorithm-based method for feature subset selection. *Soft Computing* **12**, 111–120 (2008).

[14] R0oland. Onepointcrossover. *https://en.wikipedia.org/wiki/File:OnePointCrossover.svg* (2013).

[15] Feurer, M., Klein, A., Eggensperger, J., Katharina Springenberg, Blum, M. & Hutter, F. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28 (2015)*, 2962–2970 (2015).

Reference

[16] Hastie, T., Rosset, S., Zhu, J. & Zou, H. Multi-class adaboost. *Statistics and its Interface* **2**, 349–360 (2009).

[17] Safavian, S. R. & Landgrebe, D. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics* **21**, 660–674 (1991).

[18] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. & Lin, C.-J. Liblinear: A library for large linear classification. *the Journal of machine Learning research* **9**, 1871–1874 (2008).

[19] Chang, C.-C. & Lin, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* **2**, 1–27 (2011).

[20] Gardner, M. W. & Dorling, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* **32**, 2627–2636 (1998).

[21] Wilhelm, F. The idea behind automatic relevance determination and bayesian interpolation. *https://www.youtube.com/watch?v=2gT-Q0NZzoE* (2016).

[22] Wang, J. An intuitive tutorial to gaussian processes regression. *arXiv preprint arXiv:2009.10862* (2020).

Reference

[23] Bergstra, J., Yamins, D. & Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, 115–123 (PMLR, 2013).

[24] He, X., Zhao, K. & Chu, X. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems* **212**, 106622 (2021).

[25] https://towardsdatascience.com/stop-using-smote-to-handle-all-your-imbalanced-data-34403399d3be

[26] https://auto.gluon.ai/stable/index.html

[27] https://automl.github.io/auto-sklearn/master/

[28] https://automl.github.io/Auto-PyTorch/development/

[29] https://github.com/hibayesian/awesome-automl-papers/blob/master/resources/automl.png

[30] https://towardsdatascience.com/stop-using-smote-to-handle-all-your-imbalanced-data-34403399d3be

[31] https://mlwhiz.com/blog/2020/01/28/imbal/

[32] https://www.researchgate.net/figure/Wilson-Editing-for-a-1-NN-Classifier_fig1_4133603

[33] https://towardsdatascience.com/knn-visualization-in-just-13-lines-of-code-32820d72c6b6

[34] https://medium.com/@viveksalunkhe80/support-vector-machine-svm-88f360ff5f38

[35] https://online.stat.psu.edu/stat508/book/export/html/696

[36] https://www.freecodecamp.org/news/how-to-use-the-tree-based-algorithm-for-machine-learning/

[37] https://suniljangirblog.wordpress.com/2018/12/13/variants-of-gradient-descent/

Additional Resources

GitHub Repository: https://github.com/PanyiDong/My_AutoML

Report of the work: https://github.com/PanyiDong/My_AutoML/blob/master/report.pdf

Examples on how to use the pipeline: https://github.com/PanyiDong/My_AutoML/tree/master/example

Docker Environment files: https://github.com/PanyiDong/My_AutoML/tree/master/Dockerfiles

Thanks!

Questions