Efficient tuning of automated machine learning pipelines

Nguyen, D.A.

APPENDIX A

Appendix

A.1 Additional information for the first experiment

A.1.1 Parameter setting

In this section, we present detailed information of the hyperparameters used in the classification algorithms[1] and resampling algorithms[2] that we used in our experiment. The entire Python code project can be found at https://github.com/ECOLE-ITN/CASH4IMBALANCE.

The detailed information on hyperparameters is listed in Table A.1, Table A.2, and Table A.3.

A.2 Imbalance datasets

In this section, we present 44 examined datasets taken from the KEEL repository [186] in Table A.4. For each dataset, we include the *Imbalance Ratio* (IR), which is the ratio of the number of majority class instances to that of minority class instances.

[1] All classification algorithms are implemented in the Python package SCIKIT-LEARN(version 0.23.2) [151].

[2] All resampling algorithms are implemented in the Python package IMBALANCED-LEARN(version 0.7.0) [48]

Table A.1: Hyperparameters of Classification Algorithms

Algorithm	Hyperparameter	Range
Support Vector Machines	max_iter	10000
	cache_size	700 (Megabyte)
	probability	[True, False]
	C	[0.5^5,100]
	kernel	[linear, rbf, poly, sigmoid]
	gamma	[auto, value, scale]
	gamma_value	[3.1e-05,8]
	coef0	[-1.0, 1.0]
	degree	[2, 5]
	shrinking	[True, False]
	tol	[1e-05, 1e-01]
Random Forest	n_estimators	[1, 150]
	criterion	[gini, entropy]
	max_features	[1, sqrt, log2, None]
	min_samples_split	[2, 20]
	min_samples_leaf	[1, 20]
	class_weight	[balanced, balanced_subsample, None]
	bootstrap	[True, False]
K-Nearest Neighbors	n_neighbors	[1, 51]
	weights	[uniform, distance]
	algorithm	[auto, ball_tree, kd_tree, brute]
	p	[0, 20]
	metric	· p = 0 → metric = chebyshev · p = 1 → metric = manhattan · p = 2 → metric = euclidean · p > 2 → metric = minkowski
Decision Tree	criterion	[gini, entropy]
	max_depth	[2,20]
	max_features	[1, sqrt, log2, None]
	min_samples_split	[2, 20]
	min_samples_leaf	[1, 20]
Logistic Regression	C	[1, 150]
	criterion	[0.5^5, 100]
	tol	[1e-05, 1e-01]
	l1_ratio	[1e-09, 1]
	(penalty, solver)	[(l1, liblinear), (l1, saga), (l2, lbfgs), (l2, newton-cg), (l2, liblinear), (l2, sag), (l2, saga), (elasticnet, saga), (none, newton-cg), (none, lbfgs), (none, sag), (none, saga)]

Table A.2: Hyperparameters of Resampling techniques (part I)

Group.	Hyperparameter	Range
Under resampling	**CondensedNearestNeighbour**	
	sampling_strategy	*default*
	n_neighbors	[1, 50]
	n_seeds_S	[1, 50]
	EditedNearestNeighbours	
	sampling_strategy	*default*
	n_neighbors	[1, 20]
	kind_sel	[all, mode]
	RepeatedEditedNearestNeighbours	
	sampling_strategy	*default*
	n_neighbors	[1, 20]
	kind_sel	[all, mode]
	AllKNN	
	sampling_strategy	*default*
	n_neighbors	[1, 20]
	kind_sel	[all, mode]
	allow_minority	[True, False]
	InstanceHardnessThreshold	
	sampling_strategy	*default*
	estimator	none, decision-tree, adaboost knn, linear-svm, gradient-boosting
	cv	[2, 10]
	OneSidedSelection	
	sampling_strategy	*default*
	n_neighbors	[1, 20]
	n_seeds_S	[1, 20]
	RandomUnderSampler	
	sampling_strategy	*default*
	replacement	[True, False]
	TomekLinks	
	sampling_strategy	*default*
	NearMiss	
	sampling_strategy	*default*
	version	[1,3]
	n_neighbors	[1, 20]
	n_neighbors_ver3	[1, 20]
	NeighbourhoodCleaningRule	
	sampling_strategy	*default*
	n_neighbors	[1, 20]
	threshold_cleaning	[0.0, 1.0]
	ClusterCentroids	
	sampling_strategy	*default*
	estimator	[KMeans, MiniBatchKMeans]
	voting	[hard, soft]

Table A.3: Hyperparameters of Resampling techniques (part II)

Group.	Hyperparameter	Range
Combine resampling	**SMOTENN**	
	sampling_strategy	*default*
	SMOTETomek	
	sampling_strategy	*default*
Over resampling	**SMOTE**	
	k_neighbors	[1, 10]
	sampling_strategy	*default*
	BorderlineSMOTE	
	sampling_strategy	*default*
	k_neighbors	[1, 10]
	m_neighbors	[1, 10]
	kind	[borderline1, borderline2]
	SMOTENC	
	sampling_strategy	*default*
	categorical_features	True
	k_neighbors	[1, 10]
	SVMSMOTE	
	sampling_strategy	*default*
	k_neighbors	[1, 10]
	m_neighbors	[1, 10]
	out_step	[0.0, 1.0]
	KMeansSMOTE	
	sampling_strategy	*default*
	k_neighbors	[1, 10]
	cluster_balance_threshold	[1e-2, 1]
	ADASYN	
	sampling_strategy	*default*
	n_neighbors	[1, 10]
	RandomOverSampler	
	sampling_strategy	*default*

Table A.4: The number of positive, negative classes, attributes (#Att) and the imbalance ratio (IR) of the KEEL Datasets, ordered by increasing IR value.

Data Sets	# Negative	# Positive	#Att	**IR**
glass1	138	76	9	1.82
ecoli-0_vs_1	77	143	7	1.86
wisconsin	444	239	9	1.86
pima	500	268	8	1.87
iris0	100	50	4	2
glass0	144	70	9	2.06
yeast1	1055	429	8	2.46
haberman	225	81	3	2.78
vehicle2	628	218	18	2.88
vehicle1	629	217	18	2.9
vehicle3	634	212	18	2.99
glass-0-1-2-3_vs_4-5-6	163	51	9	3.2
vehicle0	647	199	18	3.25
ecoli1	259	77	7	3.36
new-thyroid1	180	35	5	5.14
new-thyroid2	180	35	5	5.14
ecoli2	284	52	7	5.46
segment0	1979	329	19	6.02
glass6	185	29	9	6.38
yeast3	1321	163	8	8.1
ecoli3	301	35	7	8.6
page-blocks0	4913	559	10	8.79
yeast-2_vs_4	463	51	8	9.08
yeast-0-5-6-7-9_vs_4	477	51	8	9.35
vowel0	898	90	13	9.98
glass-0-1-6_vs_2	175	17	9	10.29
glass2	197	17	9	11.59
shuttle-c0-vs-c4	1706	123	9	13.87
yeast-1_vs_7	429	30	7	14.3
glass4	201	13	9	15.46
ecoli4	316	20	7	15.8
page-blocks-1-3_vs_4	444	28	10	15.86
abalone9-18	689	42	8	16.4
glass-0-1-6_vs_5	175	9	9	19.44
shuttle-c2-vs-c4	123	6	9	20.5
yeast-1-4-5-8_vs_7	663	30	8	22.1
glass5	205	9	9	22.78
yeast-2_vs_8	462	20	8	23.1
yeast4	1433	51	8	28.1
yeast-1-2-8-9_vs_7	917	30	8	30.57
yeast5	1440	44	8	32.73
ecoli-0-1-3-7_vs_2-6	274	7	7	39.14
yeast6	1449	35	8	41.4
abalone19	4142	32	8	129.44

A.3 Additional information for the second experiment

A.3.1 Datasets used in the second experiment

In this section, we present 73 examined datasets taken from OpenML repository [204] in Table A.5. For each task, we include the *OpenML ID* (#task id) and the corresponding dataset (#ID, Name) together with the number of classes (#Class), the number of instances(#Sample), the number of features for one instance (Total features, number of numeric and categorical features), the number of missing values (#Missing values), the number of instances with missing value (#Incomplete sample).

A.3.2 Search space

Detailed information on the operators, algorithms and hyperparameters used in the second experiment is given in Table A.6. The search space was extracted from Auto-sklearn [45].

Table A.5: List of 73 datasets used in our second experiment, ordered by increasing OpenML ID.

Task ID	Data set		#Class	#Sample	Features			#Missing	#Incomplete
	#ID	Name			# Total	#Numeric	#Cate	Values	sample
3	3	kr-vs-kp	2	37	3196	0	37	0	0
12	12	mfeat-factors	10	217	2000	216	1	0	0
15	15	breast-w	2	10	699	9	1	16	16
23	23	cmc	3	10	1473	2	8	0	0
24	24	mushroom	2	23	8124	0	23	2480	2480
29	29	credit-approval	2	16	690	6	10	67	37
31	31	credit-g	2	21	1000	7	14	0	0
3021	38	sick	2	30	3772	7	23	6064	3772
41	42	soybean	19	36	683	0	36	2337	121
53	54	vehicle	4	19	846	18	1	0	0
2079	188	eucalyptus	5	20	736	14	6	448	95
3543	451	irish	2	6	500	2	4	32	32
3560	469	analcatdata_dmf	6	5	797	0	5	0	0
3561	470	profb	2	10	672	5	5	1200	666
3904	1053	jm1	2	22	10885	21	1	25	5
3917	1067	kc1	2	22	2109	21	1	0	0
3945	1111	KDDCup09_appete	2	231	50000	192	39	8024152	50000
3946	1112	KDDCup09_churn	2	231	50000	192	39	8024152	50000
3948	1114	KDDCup09_upsell	2	231	50000	192	39	8024152	50000
189354	1169	airlines	2	8	539383	3	5	0	0
14965	1461	bank-marketing	2	17	45211	7	10	0	0
10101	1464	blood-transfusi	2	5	748	4	1	0	0
9981	1468	cnae-9	9	857	1080	856	1	0	0
9985	1475	first-order-the	6	52	6118	51	1	0	0
9977	1486	nomao	2	119	34465	89	30	0	0
9952	1489	phoneme	2	6	5404	5	1	0	0
9955	1492	one-hundred-pla	100	65	1600	64	1	0	0
7592	1590	adult	2	15	48842	6	9	6465	3620
7593	1596	covertype	7	55	581012	10	45	0	0
9910	4134	Bioresponse	2	1777	3751	1776	1	0	0
34539	4135	Amazon_employee	2	10	32769	0	10	0	0
14952	4534	PhishingWebsite	2	31	11055	0	31	0	0
14969	4538	GesturePhaseSeg	5	33	9873	32	1	0	0
34538	4550	MiceProtein	8	82	1080	77	5	1396	528
14954	6332	cylinder-bands	2	40	540	18	22	999	263
14968	6332	cylinder-bands	2	40	540	18	22	999	263
14967	23380	cjs	6	35	2796	32	3	68100	2795
125920	23381	dresses-sales	2	13	500	1	12	835	401
146606	23512	higgs	2	29	98050	28	1	9	1

continued on the next page

Table A.5: List of 73 datasets used in our second experiment, ordered by increasing OpenML ID. – continued from the previous page

Task ID	Data set		#Class	#Sample	Features			#Missing	#Incomplete
	#ID	Name			# Total	#Numeric	#Cate	Values	sample
167120	23517	numerai28.6	2	22	96320	21	1	0	0
146607	40536	SpeedDating	2	123	8378	59	64	18372	7330
146195	40668	connect-4	3	43	67557	0	43	0	0
167140	40670	dna	3	181	3186	0	181	0	0
146212	40685	shuttle	7	10	58000	9	1	0	0
167141	40701	churn	2	21	5000	16	5	0	0
167121	40923	Devnagari-Scrip	46	1025	92000	1024	1	0	0
167124	40927	CIFAR_10	10	3073	60000	3072	1	0	0
146800	40966	MiceProtein	8	82	1080	77	5	1396	528
146821	40975	car	4	7	1728	0	7	0	0
167125	40978	Internet-Advert	2	1559	3279	3	1556	0	0
146824	40979	mfeat-pixel	10	241	2000	240	1	0	0
146818	40981	Australian	2	15	690	6	9	0	0
146817	40982	steel-plates-fa	7	28	1941	27	1	0	0
146820	40983	wilt	2	6	4839	5	1	0	0
146822	40984	segment	7	20	2310	19	1	0	0
146819	40994	climate-model-s	2	21	540	20	1	0	0
146825	40996	Fashion-MNIST	10	785	70000	784	1	0	0
167119	41027	jungle_chess_2p	3	7	44819	6	1	0	0
168868	41138	APSFailure	2	171	76000	170	1	1078695	75244
168908	41142	christine	2	1637	5418	1599	38	0	0
168911	41143	jasmine	2	145	2984	8	137	0	0
168912	41146	sylvine	2	21	5124	20	1	0	0
189356	41147	albert	2	79	425240	26	53	2734000	425159
168335	41150	MiniBooNE	2	51	130064	50	1	0	0
168337	41159	guillermo	2	4297	20000	4296	1	0	0
168338	41161	riccardo	2	4297	20000	4296	1	0	0
168909	41163	dilbert	5	2001	10000	2000	1	0	0
168910	41164	fabert	7	801	8237	800	1	0	0
168332	41165	robert	10	7201	10000	7200	1	0	0
168331	41166	volkert	10	181	58310	180	1	0	0
189355	41167	dionis	355	61	416188	60	1	0	0
168330	41168	jannis	4	55	83733	54	1	0	0
168329	41169	helena	100	28	65196	27	1	0	0

Table A.6: List of hyperparameters extracted from Auto-sklearn [39] for Classification problems

Operator	Hyperparameter	Type	Range
Classification	**AdaboostClassifier** (*sklearn.ensemble.AdaBoostClassifier*)		
	n_estimators	Integer	[50,500]
	learning_rate	Float	[0.01,2]
	algorithm	Categorical	[SAMME.R, SAMME]
	max_depth	Integer	[1,10]
	BernoulliNB (*sklearn.naive_bayes.BernoulliNB*)		
	alpha	Float	[0.01,100]
	fit_prior	Categorical	[True, False]
	DecisionTree (*sklearn.tree.DecisionTreeClassifier*)		
	criterion	Categorical	[gini, entropy]
	max_depth_factor	Float	[0,2]
	min_samples_split	Integer	[2,20]
	min_samples_leaf	Integer	[1,20]
	min_weight_fraction_le	Constant	[0]
	max_features	Constant	[1]
	max_leaf_nodes	Constant	[None]
	min_impurity_decrease	Constant	[0]
	ExtraTreesClassifier (*sklearn.ensemble.ExtraTreesClassifier*)		
	criterion	Categorical	[gini, entropy]
	max_features	Float	[0,1]
	max_depth_factor	Constant	[None]
	min_samples_split	Integer	[2,20]
	min_samples_leaf	Integer	[1,20]
	min_weight_fraction_le	Constant	[0]
	max_leaf_nodes	Constant	[None]
	min_impurity_decrease	Constant	[0]
	bootstrap	Categorical	[True, False]
	GaussianNB (*sklearn.naive_bayes.GaussianNB*)		
	Default hyperparameters		
	GradientBoostingClassifier (*sklearn.ensemble.HistGradientBoostingClassifier*)		
	loss	Constant	[auto]
	learning_rate	Float	[0.01,1]
	min_samples_leaf	Integer	[1,200]
	max_depth	Constant	[None]
	max_leaf_nodes	Integer	[3,2047]
	l2_regularization	Float	[0.0000000001,1]
	early_stop	Categorical	[off, train, valid]
	tol	Constant	[0.0000001]
	scoring	Constant	[loss]
	n_iter_no_change	Integer	[1,20]
	validation_fraction	Float	[0.01,0.4]
	KNearestNeighborsClassifier (*sklearn.neighbors.KNeighborsClassifier*)		
	n_neighbors	Integer	[1,100]
	weights	Categorical	[uniform, distance]
	p	Categorical	[1, 2]
	LDA (*sklearn.discriminant_analysis.LinearDiscriminantAnalysis*)		
	shrinkage	Categorical	[None, auto, manual]
	shrinkage_factor	Float	[0,1]
	n_components	Integer	[1,250]
	tol	Float	[0.00001,0.1]

continued on the next page

Table A.6: List of hyperparameters extracted from Auto-sklearn [39] for Classification problems – continued from the previous page

Operator	Hyperparameter	Type	Range
Classification	**LinearSVC** (*sklearn.svm.LinearSVC*)		
	penalty	Categorical	[l1, l2]
	loss	Categorical	[hinge, squared_hinge]
	dual	Constant	[False]
	tol	Float	[0.00001,0.1]
	multi_class	Constant	[ovr]
	fit_intercept	Constant	[True]
	LibSVM_SVC (*sklearn.svm.SVC*)		
	kernel	Categorical	[rbf, poly, sigmoid]
	degree	Integer	[2,5]
	gamma	Float	[0.000031,8]
	coef0	Float	[-1,1]
	shrinking	Categorical	[True, False]
	tol	Float	[0.00001,0.1]
	max_iter	Constant	[-1]
	MultinomialNB(*sklearn.naive_bayes.MultinomialNB*)		
	alpha	Float	[0.01,100]
	fit_prior	Categorical	[True, False]
	PassiveAggressive (*from sklearn.linear_model import PassiveAggressiveClassifier*)		
	loss	Categorical	[hinge, squared_hinge]
	tol	Float	[0.00001,0.1]
	average	Categorical	[True, False]
	QDA (*sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis*)		
	reg_param	Float	[0,1]
	RandomForest (*sklearn.ensemble.RandomForestClassifier*)		
	criterion	Categorical	[gini, entropy]
	max_features	Float	[0,1]
	max_depth	Constant	[None]
	min_samples_split	Integer	[2,20]
	min_samples_leaf	Integer	[1,20]
	min_weight_fraction_le	Constant	[0]
	max_leaf_nodes	Constant	[None]
	min_impurity_decrease	Constant	[0]
	bootstrap	Categorical	[True, False]
	SGD (*sklearn.linear_model.SGDClassifier*)		
	loss	Categorical	[hinge, log, modified_huber]
	penalty	Categorical	[l1, l2, elasticnet]
	alpha	Float	[0.0000001,0.1]
	l1_ratio	Float	[0.000000001,1]
	fit_intercept	Constant	[True]
	tol	Float	[0.00001,0.1]
	epsilon	Float	[0.00001,0.1]
	learning_rate	Categorical	[optimal, invscaling, constant,]
	eta0	Float	[0.0000001,0.1]
	power_t	Float	[0.00001,1]
	average	Categorical	[True, False]

continued on the next page

Table A.6: List of hyperparameters extracted from Auto-sklearn [39] for Classification problems – continued from the previous page

Operator	Hyperparameter	Type	Range
Feature Preprocessing	**ExtraTreesClassifier** (*sklearn.ensemble import ExtraTreesClassifier*)		
	n_estimators	Constant	[100]
	criterion	Categorical	[gini, entropy]
	max_features	Float	[0,1]
	max_depth	Constant	[None]
	min_samples_split	Integer	[2,20]
	min_samples_leaf	Integer	[1,20]
	min_weight_fraction_leaf	Constant	[0]
	max_leaf_nodes	Constant	[None]
	min_impurity_decrease	Constant	[0]
	bootstrap	Categorical	[True, False]
	FastICA (*sklearn.decomposition.FastICA*)		
	n_components	Categorical	[10,2000]
	algorithm	Categorical	[parallel, deflation]
	whiten	Categorical	[True, False]
	fun	Categorical	[logcosh, exp, cube]
	FeatureAgglomeration (*sklearn.cluster.FeatureAgglomeration*)		
	n_clusters	Integer	[2, 400]
	affinity	Categorical	[euclidean, manhattan, cosine]
	linkage	Categorical	[ward, complete, average]
	pooling_func	Categorical	[mean, median, max]
	KernelPCA (*sklearn.decomposition.KernelPCA*)		
	n_components	Integer	[10,2000]
	kernel	Categorical	[poly, rbf, sigmoid, cosine]
	gamma	Float	[3.05E-05,8]
	degree	Float	[-1,1]
	RandomKitchenSinks (*sklearn.kernel_approximation.RBFSampler*)		
	gamma	Float	[3.05E-05,8]
	n_components	Float	[50,10000]
	LibLinear (*sklearn.svm.LinearSVC*)		
	penalty	Constant	[l1]
	loss	Categorical	[hinge, squared_hinge]
	dual	Constant	[False]
	tol	Float	[1E-05 1E-01]
	C	Float	[0.03125, 32768]
	multi_class	Constant	[ovr]
	fit_intercept	Constant	[True]
	intercept_scaling	Constant	[1]
	No Preprocessing		

continued on the next page

Table A.6: List of hyperparameters extracted from Auto-sklearn [39] for Classification problems – continued from the previous page

Operator	Hyperparameter	Type	Range
Feature Preprocessing	**Nystroem**(*sklearn.kernel_approximation.Nystroem*)		
	kernel	Categorical	[poly, rbf, sigmoid, cosine]
	gamma	Float	[3.05E-05,8]
	n_components	Integer	[50, 10000]
	degree	Integer	[2,5]
	coef0	Float	[-1,1]
	PCA(*sklearn.decomposition.PCA*)		
	keep_variance	Float	[0.5, 0.9999]
	whiten	Categorical	[True, False]
	Polynomial(*sklearn.preprocessing.PolynomialFeatures*)		
	degree	Integer	[2,3]
	interaction_only	Categorical	[True, False]
	include_bias	Categorical	[True, False]
	RandomTreesEmbedding(*sklearn.ensemble.RandomTreesEmbedding*)		
	n_estimators	Integer	[10,100]
	max_depth	Constant	[2,10]
	min_samples_split	Integer	[2,20]
	min_samples_leaf	Integer	[1,20]
	min_weight_fraction_leaf	Constant	[0]
	max_leaf_nodes	Constant	[None]
	bootstrap	Categorical	[True, False]
	SelectPercentile(*sklearn.feature_selection.SelectPercentile*)		
	percentile	Float	[1,99]
	score_func	Categorical	[chi2, f_classif, mutual_info]
	score_func	Constant	[chi2]
	SelectRates(*sklearn.feature_selection.GenericUnivariateSelect*)		
	alpha	Float	[0.01,0.5]
	score_func	Constant	[chi2]
	score_func	Categorical	[chi2, f_classif]
	mode	Categorical	[for, fdr, fwe]
	TruncatedSVD(*sklearn.decomposition.TruncatedSVD(algorithm='randomized')*)		
	target_dim	Integer	[10,256]

continued on the next page

Table A.6: List of hyperparameters extracted from Auto-sklearn [39] for Classification problems – continued from the previous page

Operator	Hyperparameter	Type	Range
Balancing(*Additional hyperparameter to classification algorithms*)			
	strategy	Categorical	[none, weighting]
Categorical encoding			
	NoEncoding (*No hyperparameter*)		
	OneHotEncoder use *autosklearn.pipeline.implementations.SparseOneHotEncoder* for spare data and *sklearn.preprocessing.OneHotEncoder*for dense data		
Categorical Imputation (*sklearn.impute.SimpleImputer*)			
	strategy	Constant	[constant]
	fill_value	Constant	[2]
	copy	Constant	[False]
Numerical Imputation (*sklearn.impute.SimpleImputer*)			
	copy	Constant	[False]
	strategy	Categorical	[mean, median, most_frequent]
Minority Coalescense	**NoCoalescence** (*No hyperparameter*)		
	MinorityCoalescer (*autosklearn.pipeline.implementations.MinorityCoalescer*)		
	minimum_fraction	Float	[0.0001,0.5]
Rescaling	**MinMaxScaler** (*sklearn.preprocessing.MinMaxScaler*)		
	copy	Constant	[False]
	None (*No hyperparameter*)		
	QuantileTransformer (*sklearn.preprocessing.QuantileTransformer*)		
	n_quantiles	Integer	[10, 2000]
	output_distribution	Categorical	[uniform, normal]
	RobustScaler (*sklearn.preprocessing.RobustScaler*)		
	q_min	Float	[0.001, 0.3]
	q_max	Float	[0.7, 0.999]
	StandardScaler (*sklearn.preprocessing.StandardScaler*)		
	copy	Constant	[False]
Variance Threshold (*sklearn.feature_selection.VarianceThreshold*)			
	threshold	Constant	[0.0]

Bibliography

[1] S. Perla, N. N. K, and S. Potta, "Implementation of autonomous cars using machine learning," in *International Conference on Edge Computing and Applications (ICECAA)*, 2022, pp. 1444–1451.

[2] I. Kononenko, "Machine learning for medical diagnosis: History, state of the art and perspective," *Artificial Intelligence in Medicine*, vol. 23, no. 1, pp. 89–109, 2001.

[3] L. Galway, D. Charles, and M. Black, "Machine learning in digital games: A survey," *Artificial Intelligence Review*, vol. 29, pp. 123–161, 2008.

[4] P. Jain, S. Gupta, and K. Ramkumar, "Review on face recognition by machine learning and deep learning approaches," in *International Conference on Computing for Sustainable Global Development*, 2019, pp. 528–534.

[5] Y. A. Mohamed, A. Khanan, M. Bashir, A. H. H. M. Mohamed, M. A. E. Adiel, and M. A. Elsadig, "The impact of artificial intelligence on language translation: A review," *IEEE Access*, vol. 12, pp. 25 553–25 579, 2024.

[6] A. H. Sabry and U. A. B. Ungku Amirulddin, "A review on fault detection and diagnosis of industrial robots and multi-axis machines," *Results in Engineering*, vol. 23, p. 102 397, 2024.

[7] V. Duc Nguyen, E. Zwanenburg, S. Limmer, W. Luijben, T. Bäck, and M. Olhofer, "A combination of fourier transform and machine learning for fault detection and diagnosis of induction motors," in *International Conference on Dependable Systems and Their Applications (DSA)*, 2021, pp. 344–351.

[8] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automatic machine learning: methods, systems, challenges* (Challenges in Machine Learning). Germany: Springer, 2019.

[9] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[10] C. Giraud-Carrier and F. Provost, "Toward a justification of meta-learning : Is the no free lunch theorem a showstopper ?" In *Proceedings of the ICML-2005 Workshop on Meta-learning*, 2005, pp. 12–19.

[11] M.-A. Zöller, W. Titov, T. Schlegel, and M. F. Huber, "XAutoML: A visual analytics tool for understanding and validating automated machine learning," *ACM Transactions on Interactive Intelligent Systems*, vol. 13, no. 4, pp. 1–39, 2023.

[12] J. Drozdal, J. Weisz, D. Wang, *et al.*, "Trust in automl: Exploring information needs for establishing trust in automated machine learning systems," in *Proceedings of the International Conference on Intelligent User Interfaces*, ser. IUI '20, Association for Computing Machinery, 2020, pp. 297–307.

[13] B. Bischl, M. Binder, M. Lang, *et al.*, *Hyperparameter optimization: Foundations, algorithms, best practices and open challenges*, 2021. arXiv: 2107.05847.

[14] D. Vermetten, H. Wang, C. Doerr, and T. Bäck, "Integrated vs. sequential approaches for selecting and tuning CMA-ES variants," in *The Genetic and Evolutionary Computation Conference*, 2020, pp. 903–912.

[15] D. A. Nguyen, A. V. Kononova, S. Menzel, B. Sendhoff, and T. Bäck, "Efficient AutoML via combinational sampling," in *IEEE Symposium Series on Computational Intelligence*, 2021, pp. 01–10.

[16] D. Peng, X. Dong, E. Real, *et al.*, "PyGlove: Symbolic programming for automated machine learning," in *Conference on Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 96–108.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[18] G. Batista, A. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: A case study.," *II Brazilian Workshop on Bioinformatics*, pp. 10–18, 2003.

[19] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 3, no. 1, pp. 4–21, 2011.

[20] W. Zuo, D. Zhang, and K. Wang, "On kernel difference-weighted k-nearest neighbor classification," *Pattern Analysis and Applications*, vol. 11, pp. 247–257, 2007.

[21] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Conference on Neural Information Processing Systems (NIPS)*, vol. 9, MIT Press, 1996.

[22] M. Zöller and M. Huber, "Benchmark and survey of automated machine learning frameworks," *Journal of Artificial Intelligence Research*, vol. 70, pp. 409–472, 2021.

[23] J. Lévesque, A. Durand, C. Gagné, and R. Sabourin, "Bayesian optimization for conditional hyperparameter spaces," in *International Joint Conference on Neural Networks*, 2017, pp. 286–293.

[24] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Conference on Neural Information Processing Systems (NIPS)*, vol. 24, 2011, pp. 2546–2554.

[25] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization (LION)*, 2011, pp. 507–523.

[26] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Parallel algorithm configuration," in *Learning and Intelligent Optimization (LION)*, 2012, pp. 55–70.

[27] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss, "Sequential parameter optimization," *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 773–780, 2005.

[28] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Conference on Neural Information Processing Systems (NIPS)*, vol. 25, Curran Associates, Inc., 2012.

[29] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[30] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012.

[31] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. De Freitas, "Bayesian optimization in high dimensions via random embeddings," in *International Joint Conference on Artificial Intelligence*, 2013.

[32] O. Maron and A. W. Moore, "The racing algorithm: Model selection for lazy learners," *Artificial Intelligence Review*, vol. 11, pp. 193–225, 1997.

[33] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[34] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *The International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016, pp. 240–248.

[35] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, 2017.

[36] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *International Conference on Machine Learning*, 2018, pp. 1437–1445.

[37] D. A. Nguyen, A. V. Kononova, S. Menzel, B. Sendhoff, and T. Bäck, "An efficient contesting procedure for automl optimization," in *IEEE Access*, vol. 10, 2022, pp. 75 754–75 771.

[38] B. van Stein, H. Wang, and T. Bäck, "Automatic configuration of deep neural networks with parallel efficient global optimization," in *International Joint Conference on Neural Networks*, 2019, pp. 1–7.

[39] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Conference on Neural Information Processing Systems (NIPS)*, 2015.

[40] C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, 2012, pp. 847–855.

[41] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *Journal of Machine Learning Research*, vol. 18, no. 25, pp. 1–5, 2017.

[42] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-Sklearn: Automatic hyperparameter configuration for scikit-learn," in *Proceedings of the Python in Science Conferences (SciPy)*, 2014.

[43] R. S. Olson and J. H. Moore, "TPOT: A tree-based pipeline optimization tool for automating machine learning," in *Automated Machine Learning: Methods, Systems, Challenges*. Cham: Springer International Publishing, 2019, pp. 151–160.

[44] H. J. Escalante, M. Montes, and L. E. Sucar, "Particle swarm model selection," *Journal of Machine Learning Research*, vol. 10, no. 15, pp. 405–440, 2009.

[45] M. Feurer, K. Eggensperger, S. Falkner, M. L., and F. Hutter, *Auto-sklearn 2.0: The next generation*, 2020. arXiv: `2007.04074 [cs.LG]`.

[46] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[47] D. A. Nguyen, J. Kong, H. Wang, *et al.*, "Improved automated cash optimization with tree parzen estimators for class imbalance problems," in *IEEE International Conference on Data Science and Advanced Analytics*, 2021, pp. 1–9.

[48] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[49] X. Wen, J. Shan, Y. He, and K. Song, "Steel surface defect recognition: A survey," *Coatings*, vol. 13, no. 1, p. 17, 2022.

[50] D. Jackson, "Towards a theory of conceptual design for software," in *2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, Association for Computing Machinery, 2015, pp. 282–296.

[51] C. Molnar, *Interprtable machine learning: A guide for making black box models explainable*, 2018.

[52] D. Conway and J. M. White, "Machine learning for hackers - case studies and algorithms to get you started," " O'Reilly Media, Inc.", 2012.

[53] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[54] I. V. Tetko, R. van Deursen, and G. Godin, *Be aware of overfitting by hyperparameter optimization!* 2024. arXiv: 2407.20786 [cs.LG].

[55] O. Falana, O. Durodola, and O. Oladipupo, "Implementing kaizen/continuous improvement in manufacturing industry," 2022.

[56] J. Brownlee, *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery, 2020.

[57] D. Pyle, *Data preparation for data mining*. Morgan Kaufmann Publishers Inc., 1999.

[58] R. Egele, J. C. S. J. Junior, J. N. van Rijn, *et al.*, *Ai competitions and benchmarks: Dataset development*, 2024. arXiv: 2404.09703 [cs.LG].

[59] R. Barga, V. Fontama, W. H. Tok, R. Barga, V. Fontama, and W. H. Tok, "Data preparation," *Predictive Analytics with Microsoft Azure Machine Learning*, pp. 45–79, 2015.

[60] X. Chu, J. Morcos, I. F. Ilyas, *et al.*, "KATARA: A data cleaning system powered by knowledge bases and crowdsourcing," in *Proceedings of the ACM SIGMOD international conference on management of data*, Association for Computing Machinery, 2015, pp. 1247–1261.

[61] O. Deshpande, D. S. Lamba, M. Tourn, *et al.*, "Building, maintaining, and using knowledge bases: A report from the trenches," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2013, pp. 1209–1220.

[62] S. Buchholz and J. Latorre, "Crowdsourcing preference tests, and how to detect cheating," in *Annual Conference of the International Speech Communication Association*, 2011.

[63] S. Krishnan, J. Wang, M. J. Franklin, *et al.*, "Sampleclean: Fast and reliable analytics on dirty data," *IEEE Data Engineering Bulletin*, vol. 38, pp. 59–75, 2015.

[64] S. Krishnan, M. J. Franklin, K. Goldberg, J. Wang, and E. Wu, "Activeclean: An interactive data cleaning framework for modern machine learning," in *Proceedings of the International Conference on Management of Data*, ser. SIGMOD '16, Association for Computing Machinery, 2016, pp. 2117–2120.

[65] D. Haas, S. Krishnan, J. Wang, M. J. Franklin, and E. Wu, "Wisteria: Nurturing scalable data cleaning infrastructure," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 2004–2007, 2015.

[66] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106 622, 2021.

[67] R. E. Shawi, M. Maher, and S. Sakr, "Automated machine learning: State-of-the-art and open challenges," *ArXiv*, 2019. arXiv: 1906.02287.

[68] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[69] A. E. Hoerl and R. W. Kennard, "Ridge regression: Applications to nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 69–82, 1970.

[70] L. Melkumova and S. Shatskikh, "Comparing ridge and LASSO estimators for data analysis," *Procedia Engineering*, vol. 201, pp. 746–755, 2017.

[71] S. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[72] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[73] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[74] J. Kong, W. Kowalczyk, D. A. Nguyen, S. Menzel, and T. Bäck, "Hyperparameter optimisation for improving classification under class imbalance," in *IEEE Symposium Series on Computational Intelligence*, IEEE, 2019, pp. 3072–3078.

[75] A. Ali, S. M. Shamsuddin, and A. Ralescu, "Classification with class imbalance problem: A review," *Soft Computing Models in Industrial and Environmental Applications*, vol. 7, pp. 176–204, 2015.

[76] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005.

[77] J.-O. Palacio-Niño and F. Berzal, "Evaluation metrics for unsupervised learning algorithms," arXiv, 2019. arXiv: 1905.05667.

[78] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: Review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, e01802, 2019.

[79] Y. Liu and S. Yang, "Application of decision tree-based classification algorithm on content marketing," *Journal of Mathematics*, vol. 2022, Mar. 2022.

[80] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," *ArXiv*, 2020. arXiv: 2008.05756.

[81] M. Kubat, "Addressing the curse of imbalanced training sets: One-sided selection," *International Conference on Machine Learning*, 2000.

[82] A. G. Salman, B. Kanigoro, and Y. Heryadi, "Weather forecasting using deep learning techniques," in *2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2015, pp. 281–285.

[83] T. D. Phan, "Housing price prediction using machine learning algorithms: The case of melbourne city, australia," in *International Conference on Machine Learning and Data Engineering (iCMLDE)*, 2018, pp. 35–42.

[84] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[85] O. A. Montesinos López, A. Montesinos López, and J. Crossa, "Overfitting, model tuning, and evaluation of prediction performance," in *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Cham: Springer International Publishing, 2022, pp. 109–139.

[86] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010.

[87] A. Krogh and J. Hertz, "A simple weight decay can improve generalization," in *Conference on Neural Information Processing Systems (NIPS)*, vol. 4, Morgan-Kaufmann, 1991.

[88] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[89] E. LeDell and S. Poirier, "H2O AutoML: Scalable automatic machine learning," *ICML Workshop on Automated Machine Learning*, 2020.

[90] T. Swearingen, W. Drevo, B. Cyphers, A. Cuesta-Infante, A. Ross, and K. Veeramachaneni, "ATM: A distributed, collaborative, scalable system for automated machine learning," in *IEEE International Conference on Big Data (Big Data)*, 2017, pp. 151–162.

[91] R. Fakoor, J. W. Mueller, N. Erickson, P. Chaudhari, and A. J. Smola, "Fast, accurate, and simple models for tabular data via augmented distillation," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[92] X. Shi, J. Mueller, N. Erickson, M. Li, and A. Smola, "Multimodal automl on structured tables with text fields," in *ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

[93] R. Lopez, R. Lourenco, R. Rampin, *et al.*, "AlphaD3M: An open-source automl library for multiple ml tasks," in *Proceedings of the International Conference on Automated Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 224, PMLR, 2023, pp. 22/1–22.

[94] I. Drori, Y. Krishnamurthy, R. Lourenço, *et al.*, "Automatic machine learning by pipeline synthesis using model-based reinforcement learning and a grammar," *CoRR*, vol. abs/1905.10345, 2019. arXiv: 1905.10345.

[95] F. Mohr, M. Wever, and E. Hüllermeier, "ML-Plan: Automated machine learning via hierarchical planning," *Machine Learning*, vol. 107, pp. 1495–1515, 2018.

[96] M. Wever, F. Mohr, and E. Hüllermeier, "ML-Plan for unlimited-length machine learning pipelines," in *International Conference on Machine Learning*, 2018.

[97] Y. Gil, K.-T. Yao, V. Ratnakar, *et al.*, "P4ML: A phased performance-based pipeline planner for automated machine learning," in *Proceedings of Machine Learning Research, ICML 2018 AutoML Workshop*, 2018.

[98] H. Rakotoarison, M. Schoenauer, and M. Sebag, "Automated machine learning with monte-carlo tree search," in *International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 3296–3303.

[99] N. O. Nikitin, P. Vychuzhanin, M. Sarafanov, *et al.*, "Automated evolutionary approach for the design of composite machine learning pipelines," *Future Generation Computer Systems*, vol. 127, pp. 109–125, 2022.

[100] T.-D. Nguyen, K. Musial, and B. Gabrys, "Autoweka4mcps-avatar: Accelerating automated machine learning pipeline composition and optimisation," *Expert Systems with Applications*, vol. 185, p. 115 643, 2021.

[101] M. Zöller, T. Nguyen, and M. F. Huber, "Incremental search space construction for machine learning pipeline synthesis," *CoRR*, vol. abs/2101.10951, 2021. arXiv: 2101.10951.

[102] J. R. Koza and R. Poli, "Genetic programming," in *Search methodologies*, Springer, 2005, pp. 127–164.

[103] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[104] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, vol. 40, e253, 2017.

[105] M. Ghallab, D. Nau, and P. Traverso, *Automated planning. Theory & practice*. Morgan Kaufmann Publishers, 2004.

[106] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293.

[107] C. Browne, E. Powley, D. Whitehouse, *et al.*, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4:1, pp. 1–43, 2012.

[108] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[109] N. Hansen, "The CMA evolution strategy: A comparing review," *Towards a new evolutionary computation*, pp. 75–102, 2006.

[110] A. Słowik and H. Kwasnicka, "Evolutionary algorithms and their applications to engineering problems," *Neural Computing and Applications*, pp. 1–17, 2020.

[111] M. J. Post, P. van der Putten, and J. N. van Rijn, "Does feature selection improve classification? a large scale experiment in openml," in *International Symposium on Intelligent Data Analysis*, 2016.

[112] B. Pfahringer and C. Giraud-Carrier, "Meta-learning by landmarking various learning algorithms," in *International Conference on Machine Learning*, 2000, pp. 743–750.

[113] J. N. van Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren, "Fast algorithm selection using learning curves," in *Advances in Intelligent Data Analysis XIV*, Cham: Springer International Publishing, 2015, pp. 298–309.

[114] J. Vanschoren, "Meta-learning," F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., ser. Challenges in Machine Learning, Germany: Springer, 2019, ch. 2, pp. 39–68.

[115] K. Li and J. Malik, "Learning to optimize," in *International Conference on Learning Representations*, 2017.

[116] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 32, PMLR, 2014, pp. 754–762.

[117] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Hyperparameter search space pruning – a new component for sequential model-based hyperparameter optimization," in *Machine Learning and Knowledge Discovery in Databases*, Cham: Springer International Publishing, 2015, pp. 104–119.

[118] J. N. van Rijn and F. Hutter, "Hyperparameter importance across datasets," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2367–2376.

[119] P. Probst, A.-L. Boulesteix, and B. Bischl, "Tunability: Importance of hyperparameters of machine learning algorithms," *Journal of Machine Learning Research (JMLR)*, vol. 20, no. 1, pp. 1934–1965, 2019.

[120] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Learning hyperparameter optimization initializations," in *IEEE International Conference on Data Science and Advanced Analytics*, 2015, pp. 1–10.

[121] T.-D. Nguyen, D. J. Kedziora, K. Musial, and B. Gabrys, "Exploring opportunistic meta-knowledge to reduce search spaces for automated machine learning," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[122] V. Perrone, H. Shen, M. Seeger, C. Archambeau, and R. Jenatton, "Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning," in *Proceedings of the International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2019.

[123] B. Bilalli, A. Abelló, and T. Aluja-Banet, "On the predictive power of meta-features in openml," *International Journal of Applied Mathematics and Computer Science*, vol. 27, no. 4, pp. 697–712, 2017.

[124] B. Schoenfeld, C. Giraud-Carrier, M. Poggemann, J. Christensen, and K. Seppi, *Preprocessor selection for machine learning pipelines*, 2018.

[125] M. Feurer, J. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[126] M. Lindauer and F. Hutter, "Warmstarting of model-based algorithm configuration," in *Proceedings of the AAAI Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence Conference and AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18, AAAI Press, 2018.

[127] T. A. F. Gomes, R. B. C. Prudêncio, C. Soares, A. L. D. Rossi, and A. Carvalho, "Combining meta-learning and search techniques to svm parameter selection," in *2010 Eleventh Brazilian Symposium on Neural Networks*, 2010, pp. 79–84.

[128] S. Ullah, Z. Xu, H. Wang, S. Menzel, B. Sendhoff, and T. Bäck, "Exploring clinical time series forecasting with meta-features in variational recurrent models," in *International Joint Conference on Neural Networks*, 2020, pp. 1–9.

[129] C. Wang, T. Bäck, H. H. Hoos, M. Baratchi, S. Limmer, and M. Olhofer, "Automated machine learning for short-term electric load forecasting," in *IEEE Symposium Series on Computational Intelligence*, 2019, pp. 314–321.

[130] M. Kefalas, M. Baratchi, A. Apostolidis, D. van den Herik, and T. Bäck, "Automated machine learning for remaining useful life estimation of aircraft engines," in *International Conference on Prognostics and Health Management*, 2021, pp. 1–9.

[131] D. Wang, J. D. Weisz, M. Muller, *et al.*, "Human-ai collaboration in data science: Exploring data scientists' perceptions of automated ai," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, 2019.

[132] A. Crisan and B. Fiore-Gartland, "Fits and starts: Enterprise use of automl and the role of humans in the loop," *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2021.

[133] D. Wang, J. Andres, J. D. Weisz, E. Oduor, and C. Dugan, "Autods: Towards human-centered automation of data science," *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2021.

[134] Q. Wang, Y. Ming, Z. Jin, *et al.*, "Atmseer: Increasing transparency and controllability in automated machine learning," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI '19, Association for Computing Machinery, 2019, pp. 1–12.

[135] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *Journal of Artificial Intelligence Research (JAIR)*, vol. 70, pp. 245–317, 2021.

[136] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[137] E. Kazim and A. Koshiyama, *Explaining decisions made with AI: a review of the co-badged guidance by the ICO and the Turing Institute*. 2020.

[138] I. van der Linden, H. Haned, and E. Kanoulas, "Global aggregations of local explanations for black box models," *ArXiv*, vol. abs/1907.03039, 2019.

[139] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf, "Principles of explanatory debugging to personalize interactive machine learning," in *Proceedings of the International Conference on Intelligent User Interfaces*, ser. IUI '15, Association for Computing Machinery, 2015, pp. 126–137.

[140] J. Fox, D. Glasspool, D. Grecu, S. Modgil, M. South, and V. Patkar, "Argumentation-based inference and decision making–a medical perspective," *IEEE Intelligent Systems*, vol. 22, no. 6, pp. 34–41, 2007.

[141] M. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, San Diego, California: Association for Computational Linguistics, 2016, pp. 97–101.

[142] B. Hayes and J. A. Shah, "Improving robot controller transparency through autonomous policy explanation," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI*, 2017, pp. 303–312.

[143] J. E. Mercado, M. A. Rupp, J. Y. C. Chen, M. J. Barnes, D. Barber, and K. Procci, "Intelligent agent transparency in human–agent teaming for multi-uxv management," *Human Factors*, vol. 58, no. 3, pp. 401–415, 2016, PMID: 26867556.

[144] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[145] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. E. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1487–1495, 2017.

[146] J. Liu, S. Tripathi, U. Kurup, and M. Shah, "Auptimizer - an extensible, open-source framework for hyperparameter tuning," *IEEE International Conference on Big Data (Big Data)*, pp. 339–348, 2019.

[147] J. Ono, S. Castelo, R. Lopez, E. Bertini, J. Freire, and C. Silva, "Pipelineprofiler: A visual analytics tool for the exploration of automl pipelines," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, pp. 1–1, 2020.

[148] T. Li and T. Zajonc, "HyperTuner: Visual analytics for hyperparameter tuning by professionals," in *IEEE Workshop on Machine Learning from User Interaction for Visualization and Analytics (MLUI)*, 2018, pp. 1–11.

[149] H. Park, Y. J. Nam, J.-H. Kim, and J. Choo, "HyperTendril: Visual analytics for user-driven hyperparameter optimization of deep neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, pp. 1407–1416, 2021.

[150] D. K. I. Weidele, J. D. Weisz, E. Oduor, *et al.*, "AutoAIViz: Opening the blackbox of automated artificial intelligence with conditional parallel coordinates," *Proceedings of the International Conference on Intelligent User Interfaces*, 2020.

[151] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[152] C. Wang, Q. Wu, M. Weimer, and E. Zhu, "FLAML: A fast and lightweight automl library," in *The Fourth Conference on Machine Learning and Systems (MLSys 2021)*, 2021.

[153] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. Cox, "Hyperopt: A Python library for model selection and hyperparameter optimization," *Computational Science & Discovery*, vol. 8, no. 1, p. 014 008, 2015.

[154] P. Das, N. Ivkin, T. Bansal, *et al.*, "Amazon SageMaker autopilot: A white box automl solution at scale," in *SIGMOD/PODS 2020*, 2020.

[155] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[156] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization Techniques IFIP Technical Conference Novosibirsk*, Springer Berlin Heidelberg, 1975, pp. 400–404.

[157] D. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, pp. 345–383, 2001.

[158] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International Conference on Machine Learning*, vol. 28, 2013, pp. 115–123.

[159] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning: ML Summer Schools*. Berlin, Heidelberg, 2004, pp. 63–71.

[160] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs.," *Journal of Machine Learning Research*, vol. 3, pp. 397–422, 2002.

[161] S. Kotz, *Continuous multivariate distributions. volume 1, models and applications* (Wiley series in probability and statistics. Applied probability and statistics.), Array. Wiley, 2000, p. 722, "A Wiley-Interscience publication.".

[162] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[163] X. Wang, P. Tino, M. A. Fardal, S. Raychaudhury, and A. Babul, "Fast parzen window density estimator," in *2009 International Joint Conference on Neural Networks*, 2009, pp. 3267–3274.

[164] S. Węglarczyk, "Kernel density estimation and its application," in *ITM web of conferences*, EDP Sciences, vol. 23, 2018, p. 00 037.

[165] D. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*. Boston, MA: Springer US, 2009, pp. 659–663.

[166] J. P. C. Kleijnen, W. C. M. V. Beers, and I. V. Nieuwenhuyse, "Expected improvement in efficient global optimization through bootstrapped kriging," *Journal of Global Optimization*, vol. 54, no. 1, pp. 59–73, 2012.

[167] A. Zilinskas, "A review of statistical models for global optimization," *Journal of Global Optimization*, vol. 2, no. 2, pp. 145–153, 1992.

[168] M. Schonlau, W. J. Welch, and D. R. Jones, "Global versus local search in constrained optimization of computer models," *Lecture Notes-Monograph Series*, vol. 34, pp. 11–25, 1998.

[169] M. Hoffman, B. Shahriari, and N. De Freitas, "On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning," *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 365–374, 2014.

[170] V. Gabillon, M. Ghavamzadeh, and A. Lazaric, "Best arm identification: A unified approach to fixed budget and fixed confidence," in *Conference on Neural Information Processing Systems (NIPS)*, 2012, pp. 3221–3229.

[171] M. Hoffman, E. Brochu, and N. de Freitas, "Portfolio allocation for bayesian optimization," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, ser. UAI'11, AUAI Press, 2011, pp. 327–336.

[172] R. K. Ursem, "From expected improvement to investment portfolio improvement: Spreading the risk in kriging-based optimization," in *The International Conference on Parallel Problem Solving From Nature (PPSN)*, ser. Lecture Notes in Computer Science, vol. 8672, Springer, 2014, pp. 362–372.

[173] O. Maron and A. W. Moore, "Hoeffding Races: Accelerating model selection search for classification and function approximation," in *Conference on Neural Information Processing Systems (NIPS)*, 1993.

[174] B. Samanta, "Gear fault detection using artificial neural networks and support vector machines with genetic algorithms," *Mechanical Systems and Signal Processing*, vol. 18, no. 3, pp. 625–644, 2004.

[175] A. V. D. Bosch, "Wrapped progressive sampling search for optimizing learning algorithm parameters," in *Proceedings of the Belgian-Dutch Conference on Artificial Intelligence*, 2004, pp. 219–226.

[176] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, pp. 675–701, 1937.

[177] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO'02, Morgan Kaufmann Publishers Inc., 2002, pp. 11–18.

[178] M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective*, 1st ed. 2005. 2nd printing. Springer Publishing Company, Incorporated, 2009.

[179] H. H. Hoos, "Automated algorithm configuration and parameter tuning," in *Autonomous Search*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–71.

[180] P. Balaprakash, M. Birattari, and T. Stützle, "Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement," in *Hybrid Metaheuristics*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 108–122.

[181] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "F-Race and Iterated F-Race: An overview," in *Experimental Methods for the Analysis of Optimization Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 311–336.

[182] C. Vieira, A. de Araújo, J. E. Andrade, and L. C. T. Bezerra, "iSklearn: Automated machine learning with irace," in *IEEE Congress on Evolutionary Computation*, 2021, pp. 2354–2361.

[183] W. J. Conover, *Practical nonparametric statistics*. john wiley & sons, 1999, vol. 350.

[184] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[185] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 3354–3361.

[186] J. Alcalá-Fdez, L. Sánchez, S. Garcia, *et al.*, "KEEL: A software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[187] H. L. Le, D. Landa-Silva, M. Galar, S. Garcia, and I. Triguero, "EUSC: A clustering-based surrogate model to accelerate evolutionary undersampling in imbalanced classification," *Applied Soft Computing*, vol. 101, p. 107 033, 2021.

[188] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1263–1284, 2008.

[189] P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren, "An open source automl benchmark," *ICML Workshop on Automated Machine Learning*, 2019.

[190] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1322–1328.

[191] H. Han, W. Wang, and B. H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," vol. 3644, 2005, pp. 878–887.

[192] F. Last, G. Douzas, and F. Bacao, *Oversampling for imbalanced learning based on k-means and smote*, 2017. arXiv: `1711.00837 [cs.LG]`.

[193] Imbalanced-learn, *Randomoversampler*, https://imbalanced-learn.org /stable/generated/imblearn.over_sampling. RandomOverSampler.html, Accessed: 2020-08-30.

[194] I. Tomek, "Two modifications of CNN," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, 1976.

[195] K. Gowda and G. Krishna, "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood (corresp.)," *IEEE Transactions on Information Theory*, vol. 25, no. 4, pp. 488–490, 1979.

[196] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-2, no. 3, pp. 408–421, 1972.

[197] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 6, pp. 448–452, 1976.

[198] M. R. Smith, T. Martinez, and C. Giraud-Carrier, "An instance level analysis of data complexity," *Machince Learning*, vol. 95, no. 2, pp. 225–256, 2014.

[199] J. Ziang, "KNN approach to unbalanced data distributions: A case study involving information extraction," *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*, 2003.

[200] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Artificial Intelligence in Medicine*, 2001, pp. 63–66.

[201] Imbalanced-learn, *Clustercentroids*, https://imbalanced-learn.org/ stable/ generated /imblearn.under_sampling.ClusterCentroids.html, Accessed: 2020-08-30.

[202] Imbalanced-learn, *Randomundersampler*, https://imbalanced-learn.org /stable/generated/imblearn.under_sampling.RandomUnderSampler.html, Accessed: 2020-08-30.

[203] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[204] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[205] B. Bischl, G. Casalicchio, M. Feurer, *et al.*, "Openml benchmarking suites and the OpenML100," 2017. arXiv: `1708.03731 [stat.ML]`.

[206] B. Bischl, G. Casalicchio, M. Feurer, *et al.*, "Openml benchmarking suites," *Proceedings of the NeurIPS Datasets and Benchmarks Track*, 2021.

[207] M. Feurer, J. van Rijn, A. Kadra, *et al.*, "OpenML-Python: An extensible python api for OpenML," *Journal of Machine Learning Research*, vol. 22, 2021.

[208] H. Bal, D. Epema, C. de Laat, *et al.*, "A medium-scale distributed system for computer science research: Infrastructure for the long term," *Computer*, vol. 49, no. 05, pp. 54–63, 2016.

[209] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*. Springer, 2018, vol. 10.

[210] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 434–443, 2013.

[211] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering*, 2014, pp. 1–10.

[212] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6585–6608, 2012.

[213] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, *Automated algorithm selection: Survey and perspectives*, 2018. arXiv: `1811.11597 [cs.LG]`.

[214] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 4, pp. 42–47, 2012.

[215] J. Kong, T. Rios, W. Kowalczyk, S. Menzel, and T. Bäck, "On the performance of oversampling techniques for class imbalance problems," *Advances in Knowledge Discovery and Data Mining*, vol. 12085, p. 84, 2020.

[216] S. García and F. Herrera, "Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy," *Evolutionary Computation*, vol. 17, no. 3, pp. 275–306, 2009.

[217] J. Bacardit, D. Goldberg, M. Butz, X. Llora, and J. M. Garrell, "Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy," *The International Conference on Parallel Problem Solving From Nature (PPSN)*, vol. 3242, pp. 1021–1031, 2004.

[218] H. L. Le, D. Landa-Silva, M. Galar, S. Garcia, and I. Triguero, "A hybrid surrogate model for evolutionary undersampling in imbalanced classification," in *IEEE Congress on Evolutionary Computation*, 2020, pp. 1–8.

[219] J. Huang and C. X. Ling, "Using auc and accuracy in evaluating learning algorithms," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[220] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information sciences*, vol. 250, pp. 113–141, 2013.

[221] A. Tharwat, "Classification assessment methods: A detailed tutorial," *Applied Computing and Informatics*, vol. 17, no. 1, pp. 168–192, 2018.

[222] S. Wang, L. L. Minku, and X. Yao, "Dealing with multiple classes in online class imbalance learning.," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 2118–2124.

[223] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[224] J. Xu, X. Liu, Z. Huo, C. Deng, F. Nie, and H. Huang, "Multi-class support vector machine via maximizing multi-class margins," in *International Joint Conference on Artificial Intelligence*, 2017.

[225] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *The Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.

[226] J. Fürnkranz, "Round robin classification," *The Journal of Machine Learning Research*, vol. 2, pp. 721–747, 2002.

[227] A. C. Tan, D. Gilbert, and Y. Deville, "Multi-class protein fold classification using a new ensemble machine learning approach," *Genome Informatics*, vol. 14, pp. 206–217, 2003.

[228] D. Rathgamage Don and I. Iacob, "Dcsvm: Fast multi-class classification using support vector machines," *International Journal of Machine Learning and Cybernetics*, vol. 11, 2020.

[229] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[230] X. He, Z. Wang, C. Jin, Y. Zheng, and X. Xue, "A simplified multi-class support vector machine with reduced dual optimization," *Pattern Recognition Letters*, vol. 33, no. 1, pp. 71–82, 2012.

[231] B. Jijo and A. Mohsin Abdulazeez, "Classification based on decision tree algorithm for machine learning," *Journal of Applied Science and Technology Trends*, vol. 2, pp. 20–28, 2021.

[232] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of machine learning research*, vol. 2, no. Dec, pp. 265–292, 2001.

[233] P. Piro, R. Nock, F. Nielsen, and M. Barlaud, "Multi-class leveraged k-nn for image classification," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2010.

[234] Z. Younes, F. Abdallah, and T. Denœux, "An evidence-theoretic k-nearest neighbor rule for multi-label classification," in *Scalable Uncertainty Management (SUM)*, Springer, 2009, pp. 297–308.

[235] A. Kontorovich and R. Weiss, "Maximum margin multiclass nearest neighbors," in *International conference on machine learning*, PMLR, 2014, pp. 892–900.

[236] J. D. Conklin, "Applied logistic regression," *Technometrics*, vol. 44, no. 1, pp. 81–82, 2002.

[237] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.

[238] N. Neogi, D. K. Mohanta, and P. K. Dutta, "Review of vision-based steel surface inspection systems," *EURASIP Journal on Image and Video Processing*, vol. 2014, no. 1, pp. 1–19, 2014.

[239] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, pp. 239–245, 1979.

[240] G. Muniraju, B. Kailkhura, J. J. Thiagarajan, P.-T. Bremer, C. Tepedelenlioglu, and A. Spanias, "Coverage-based designs improve sample mining and hyperparameter optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 3, pp. 1241–1253, 2021.

[241] J. Li Y.and Jiang, J. Gao, Y. Shao, C. Zhang, and B. Cui, "Efficient automatic CASH via rising bandits," in *Association for the Advancement of Artificial Intelligence*, AAAI Press, 2020, pp. 4763–4771.

[242] K. Eggensperger, M. Feurer, F. Hutter, *et al.*, "Towards an empirical foundation for assessing bayesian optimization of hyperparameters," in *NIPS Workshop on Bayesian Optimization in Theory and Practice*, 2013.

[243] P. Yang, K. Tang, and X. Yao, "Turning high-dimensional optimization into computationally expensive optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 143–156, 2018.

[244] P. Yang, K. Tang, and X. Yao, "A parallel divide-and-conquer-based evolutionary algorithm for large-scale optimization," *IEEE Access*, vol. 7, pp. 163 105–163 118, 2019.

[245] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.

[246] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller, "Deep learning for time series classification: A review," *Data Mining and Knowledge Discovery*, vol. 33, pp. 917–963, 2019.

[247] I. Rodríguez-Fdez, A. Canosa, M. Mucientes, and A. Bugarín, "STAC: A web platform for the comparison of algorithms using statistical tests," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2015, pp. 1–8.

[248] D. Ginsbourger, R. Le Riche, and L. Carraro, "Kriging is well-suited to parallelize optimization," in *Computational Intelligence in Expensive Optimization Problems*, ser. Springer series in Evolutionary Learning and Optimization, springer, 2010, pp. 131–162.

[249] K. Eggensperger, F. Hutter, H. Hoos, and K. Leyton-Brown, "Efficient benchmarking of hyperparameter optimizers via surrogates," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[250] K. Eggensperger, M. Lindauer, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Efficient benchmarking of algorithm configurators via model-based surrogates," *Machine Learning*, vol. 107, pp. 15–41, 2018.

[251] K. Leyton-Brown, E. Nudelman, and Y. Shoham, "Empirical hardness models: Methodology and a case study on combinatorial auctions," *Journal of the ACM (JACM)*, vol. 56, no. 4, pp. 1–52, 2009.

[252] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, "Algorithm runtime prediction: Methods & evaluation," *Artificial Intelligence*, vol. 206, pp. 79–111, 2014.

[253] I. Guyon, L. Sun-Hosoya, M. Boullé, *et al.*, "Analysis of the automl challenge series 2015-2018," in *Automated Machine Learning*, ser. Springer series on Challenges in Machine Learning, 2019.

Index