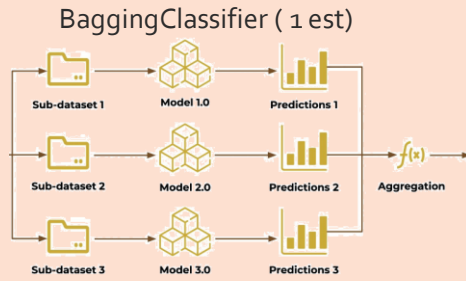
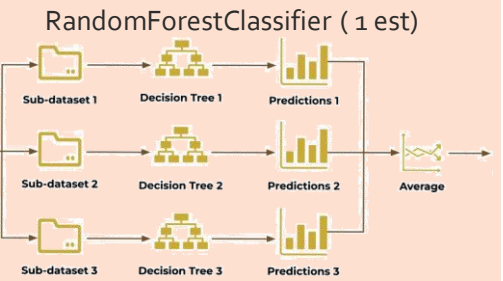
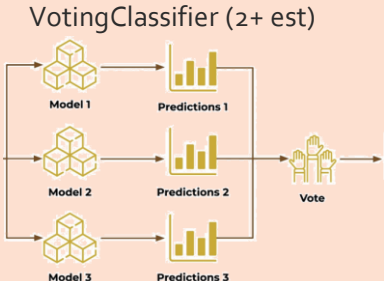
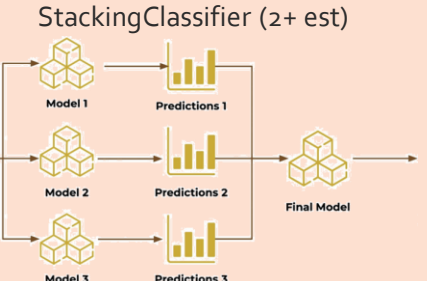


		NUMPY & SCIPY		SCIKIT LEARN		
	Steps	code	args	code	args	illustration
1	Import & export	load from file np.loadtxt() scipy.io.loadmat() np.genfromtxt() create array np.zeros() np.arange() np.linspace()	file.txt matlab file file.csv delimiter skip_header			
2	Describe & Subset	data types np.int64-float32-bool-... subset -> see Pandas from lists or tuples np.asarray() from a dataframe df.values (pandas method)		separate nums from categoricals dfnum = df.select_dtypes() dfobj = df.select_dtypes() dnum/dobj.describe() plot previews >sklearn.preprocessing data = MinMaxScaler().fit_transform(dfnum.values) pd.DataFrame(data, columns = dfnum.columns).plot.box() dfobj.nunique().plot.bar() plt.xlabel/ylabel("label_name")	include=np.number include=object	
3	Filter & Clean	change data type a.astype(int)		split train & test data >sklearn.model_selection train_test_split() StratifiedKFold() StratifiedShuffleSplit() GridSearchCV() / RandomizedSearchCV() select features >sklearn.feature_selection	X, y, test_size, train_size, stratify n_splits, test_size, train_size n_splits, test_size, train_size # see section 'Fine-tune model' # see section 'Customize & Transform'	
4	Impute (num & categ)	impute values np.insert() np.delete()	array, indexpos, value array, indexpos, axis	impute with a strategy >sklearn.impute SimpleImputer() KNNImputer()	on training set only strategy = {'mean','median', 'most_frequent'} missing_values, n_neighbors, weights	

		NUMPY & SCIPY		SCIKIT LEARN		
	Steps	code	args	code	args	illustration
4bis	Impute (dates & text)			concatenate or impute text data >sklearn.preprocessing FunctionTransformer() # custom function		
5	Summarize & Scale	arithmetic np.divide/multiply() np.sqrt/exp() aggregate functions np.sum() np.cumsum() np.median() np.corrcoef() np.std()	array1, array2 array array array, axis=0/1 array array array	Scale (num) >sklearn.preprocessing PolynomialFeatures() # vectorize feat to poly MinMaxScaler() # normalization StandardScaler() # standardization Visualize with lower dimensionality >sklearn.manifold TSNE() Transform	# on training + test set n_components, learning_rate, ... # see section 7. Customize & Transform	
6	Reshape & Combine	transpose np.transpose() reshape a.sort() a.ravel() a.reshape(3,2) combine np.concatenate() np.vstack() split np.hsplit()	array # sort the array # flatten the array # reshape rows, cols (a,b), axis=0 (a,b) array, nb_of_arrays	Encode (categ) >sklearn.preprocessing (features) OneHotEncoder() (features) OrdinalEncoder() (label) LabelEncoder() (text) >sklearn.feature_extraction.text CountVectorizer() # count tuples TfidfVectorizer() # transform corpus using Tfidf	categories, sparse=True, max_categories -> infrequent_categories_ -> a single column of integers -> classes_ # on training + test set input, encoding, lowercase, stop_words, token_pattern... input, encoding, lowercase, stop_words, token_pattern...	
7	Customize & Transform			Dimensionality reduction >sklearn.decomposition PCA() # num, categ TruncatedSVD() # num NMF() # text >sklearn.feature_selection chi2() # categ SelectKBest() # categ VarianceThreshold() # categ Transformers >sklearn.preprocessing PowerTransformer() KBinsDiscretizer()	n_components, svd_solver='randomized'... -> n_components_ # num # text : Non-Negative Matrix Factorization X, y score_func, k threshold method, standardize=True n_bins, encode, strategy={'uniform', 'quantile', 'kmeans'}	

Steps		NUMPY & SCIPY		SCIKIT LEARN		illustration
code	args	code	args			
8	Build model (supervised)			Regression >sklearn.linear_model LinearRegression() Lasso/ElasticNet/Ridge() (GLM) PoissonRegressor() >sklearn.svm LinearSVR() >sklearn.tree DecisionTreeRegressor()	fit_intercept, normalize alpha, fit_intercept, normalize, max_iter alpha, fit_intercept, max_iter # Support Vector Machine Regressor epsilon, loss, fit_intercept criterion, max_depth, max_leaf_nodes...	
			Classification >sklearn.linear_model LogisticRegression() >sklearn.svm LinearSVC() >sklearn.neighbors KNeighborsClassifier()	penalty, solver # Support Vector Machine Classifier penalty, loss, fit_intercept n_neighbors, weights, p...		
			Regression/Classification with polynomial model >from sklearn.preprocessing import PolynomialFeatures X_poly = PolynomialFeatures().fit_transform(X) LinearRegression().fit(X_poly, y)	degree={2,3...}, include_bias # X_poly contains 2 cols : X and X²		
			>sklearn.svm SVC()	# Support Vector Machine kernel='poly', degree=3		
8bis	Build model (ensemble models)		Single-class ensemble (unique estimator) >sklearn.ensemble RandomForestClassifier() # pre-defined DT est = DecisionTreeClassifier() >>BaggingClassifier() # fill-in 'base_estimator'	n_estimators, max_leaf_nodes, criterion... criterion, splitter... base_estimator=est, n_estimators		
			Single-class ensemble (several estimators) >sklearn.ensemble VotingClassifier() StackingClassifier()	# need to indicate 2+ 'estimators' estimators : [est1, est2...], voting : {'hard', 'soft'} estimators : [est1, est2...], final_estimator, stack_method		
			Multi-class Classifier >sklearn.multiclass OneVsRestClassifier() OnevsOneClassifier()	estimator, n_jobs estimator, n_jobs		
						
8ter	Build model (ensemble X boosting)		Boosting algorithms >sklearn.ensemble AdaBoostClassifier() GradientBoostingClassifier() HistGradientBoostingClassifier() >xgboost	max_depth, learning_rate, n_estimators max_depth, learning_rate, n_estimators max_depth, learning_rate, max_iter [equiv to n_estimators]		
8qua	Build model (unsupervised)		Unsupervised learning >sklearn.cluster Kmeans() DBSCAN() MeanShift()	n_clusters eps, min_samples bandwidth		

		NUMPY & SCIPY		SCIKIT LEARN	
Steps		code	args	code	args
					illustration
9	Fit & predict			Fit model model.fit() transformer.transform() transformer.fit_transform() clustermodel.fit_predict() clustermodel.predict() Predict y_pred = model.predict() y_pred = model.predict_proba()	X_train, y_train X X_new X_test X_test
10	Evaluate model (supervised)			Extract features from model RandomForestClassifier() LinearRegression() Evaluate model >sklearn.metrics mean_squared_error() r2_score() <hr/> Reg accuracy_score confusion_matrix() classification_report() roc_auc_score()	-> feature_importances_ -> reg.coef_ , reg.intercept_ y_true, y_pred, squared=True (False for RMSE) y_true, y_pred y_true, y_pred y_true, y_pred y_true, y_pred y_true, y_pred
10bis	Evaluate model (unsupervised)			Extract features from model KMeans() DBSCAN() Evaluate model KMeans().score() >sklearn.metrics silhouette_score() [=-Kmeans().score()] Build a silhouette diagram scikit-learn silhouette analysis page silhouette_samples()	-> labels_ , cluster_centers_ , inertia_ -> labels_ , components_ X X, labels # avg silhouette score X, labels # sample silhouette values to build diagram
11	Fine-tune model			Hyper-parameter tuning >sklearn.model_selection params=ParameterGrid() # set param grid GridSearchCV() RandomizedSearchCV() Evaluate a score by cross-validation >sklearn.model_selection cross_val_score() estimates	takes as input a dict of list , with key as param and value as list of vals estimator, param_grid=params, n_jobs, cv -> best_params_ estimator, n_iter, n_jobs estimator, X, scoring='accuracy'