Machine Learning

Lesson 11

Experimental evaluation of learning algorithms

Lecturer: Dr. Msagha J Mbogholi, PhD

Flashback from Lesson 10

- In machine learning projects we often run into the curse of dimensionality problem where the number of records of data are not a substantial factor of the number of features. This often leads to a problems since it means training a lot of parameters using a scarce data set, which can easily lead to overfitting and poor generalization.
- For dimensionality reduction to be effective, there needs to be underlying low dimensional structure in the feature space. I.e the features should have some relationship with each other.
- If there is non-linearity or curvature in low dim structure then autoencoders can encode more information using less dimensions. So they are a better dimensionality reduction technique in these scenarios.
- Autoencoders are neural networks that can be used to reduce the data into a low dimensional latent space by stacking multiple non-linear transformations(layers).
- PCA essentially learns a linear transformation that projects the data into another space, where vectors of projections are defined by variance of the data

Content

- Introduction to evaluation
- Evaluation types and methods
- Evaluation tests
- Evaluating model performance in SKlearn

Part 1

Introduction to Evaluation

Introduction

- Throughout this course we have learnt about the different types of machine learning algorithms.
- We have gone further to categorize them as supervised, unsupervised, semi supervised and reinforcement learning.
- We have further learnt that there are different ways in which algorithms associated with the different categories learn.
- In supervised learning we have seen that classification is the method used whereby a classifier (learner) learns to classify some given data using provided training examples. It is then provided with unseen data that is used as test data to see how well the classifier has learnt.
- In the case of unsupervised learning we learnt that there are no labels provided and thus the learner must look for similarities and cluster the data accordingly.

Introduction (cont'd)

- We have also seen that in the real world different algorithms are used in different domains.
- The choice of algorithm is usually quite a challenge.
- The scientist can't say that they will use 3 different algorithms for different reasons: first it is pretty resource consuming and secondly it is also time consuming.
- So what options are there?
- There are a certain number of factors that can be used in determining which algorithm to use; for example a study of literature can point you in the general direction.
- There are publications where authors have explained in details which ML algorithm they used and WHY. The latter can provide a better understanding of the application of the algorithm in that domain.

Introduction (cont'd)

- Unfortunately even within the same domain you may have to use a different algorithm than the one favored in the literature. Tough.
- Consequently there is a need to find ways in which to evaluate the algorithm and its performance.
- There is a lot of literature on evaluating different algorithms in machine learning.
- In this lesson we shall examine different ways of evaluating algorithms; some new terms are introduced and explained for you to follow the different methods and tests.
- Table 1 introduces the steps to take in order to perform a proper evaluation.

1. Identify the "interesting" properties of the classifier.
2. Choose an **evaluation metric** accordingly.
3. Choose the **learning algorithms** to involve in the study along with the **domain(s)** on which the various systems will be pitted against.
4. Choose a **confidence estimation** method.
5. Check that all the assumptions made by the evaluation metric and confidence estimator are verified.
6. Run the evaluation method with the chosen metric and confidence estimator, and analyze the results.
7. Interpret the results with respect to the domain(s).

Table 1. Recommended steps for proper evaluation (Japokwicz, 2009)

Introduction (cont'd)

- From table 1 the question of "interesting" properties arises; what are they? These are those that will contribute to a reasonable evaluation (since evaluations are usually pretty subjective despite the many available tests).
- The table also places emphasis on the domain involved, as well as the learning algorithms that will be evaluated.
- The metrics as well as the confidence estimation method are also emphasized. These will be discussed in detail later in the lesson.
- Fig 1 shows the steps taken in this process.

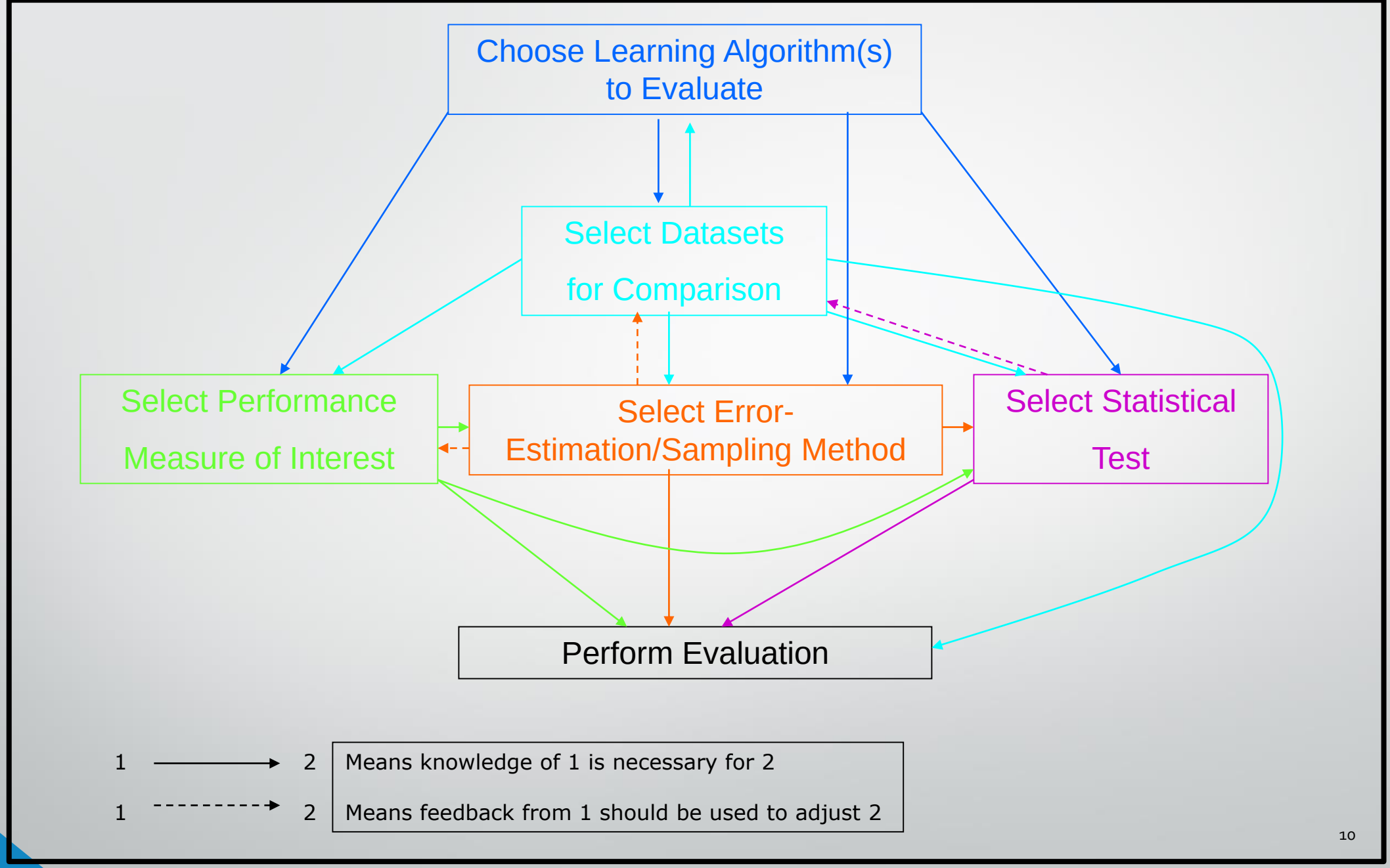

Fig 1. Classifier evaluation procedure (Japokwicz, 2009)

Introduction (cont'd)

- From fig 1 it can be seen that the choice of learning algorithm affects four other actions:
- Selection of performance measure of interest… (1);
- Selection of error estimation/sampling method …(2);
- selection of statistical tests… (3)
- Selection of data sets for comparison… (4)
- (2) provides feedback to (1) and (4); big errors might require a change in the data set (or choice of training data) or selected performance measure.
- (1), (2) and (4) are needed in order to perform (3)
- Finally performance evaluation is dependent on all 4 activities.

Part 2

Evaluation types and methods

Introduction

- Fig 1 presented a framework of sorts in aiding to evaluate a given classifier.
- However, this is not enough. We have interest in knowing not just how evaluate a classifier but also a given algorithm.
- This opens a whole Pandora box. This is because there are many unknowns and many factors to take into consideration. These are discussed in this part.
- There are many authors who have written in this area and Japokwicz and Shah (2014) is one good book that examines this domain in detail.
- However, fig 2 depicts the 3 areas that need to be analysed in algorithm and classifier evaluation.

2.1 Algorithm and classifier evaluation

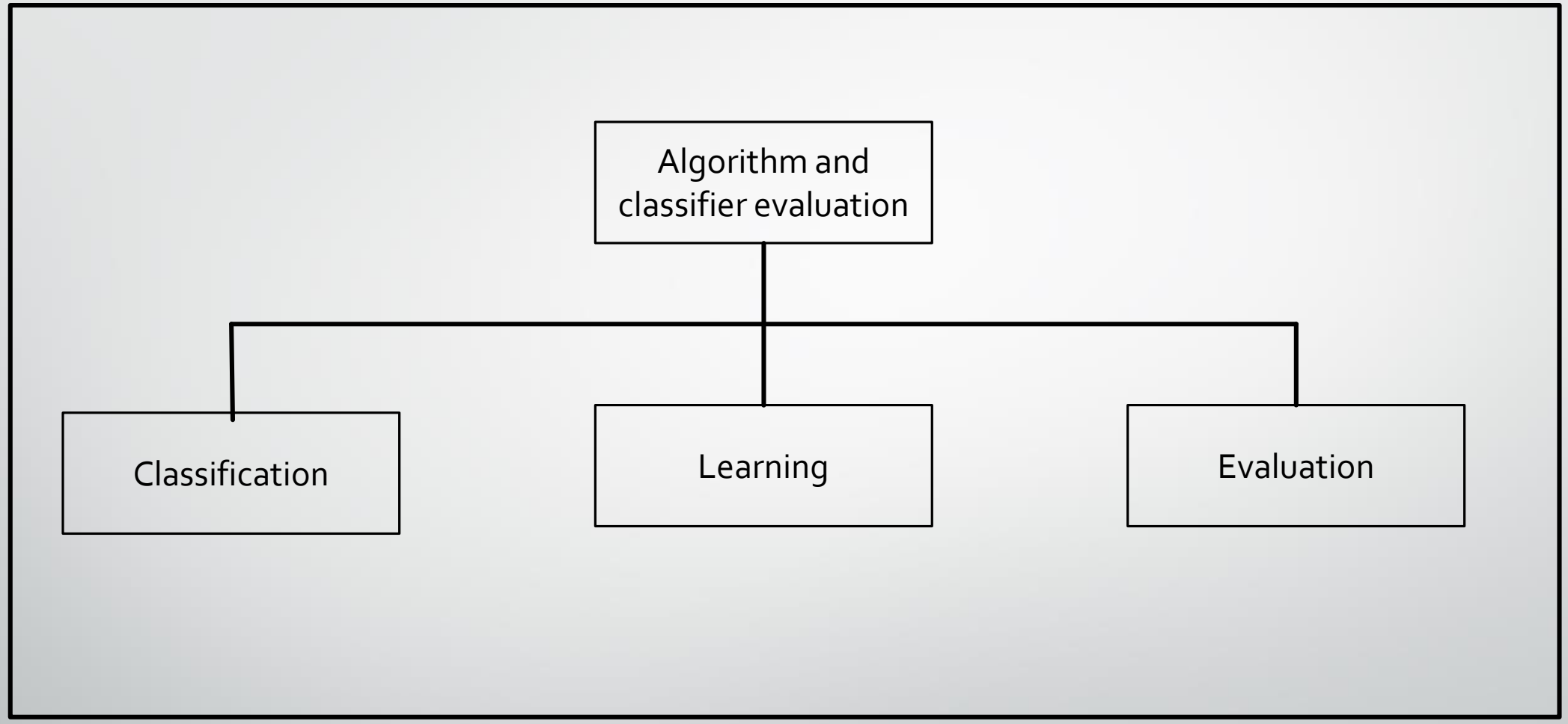

Fig 2. Algorithm and classifier evaluation (adapted from Lavesson, 2006)

2.1 Algorithm and classifier evaluation

- Fig 2 depicts the algorithm and classifier evaluation problem.
- The problem can be broken down into 3 sub domains namely classification, learning and evaluation.
- These 3 are briefly described; our emphasis for this lesson though, is on the third sub domain namely evaluation.
- However, this can not be discussed without introducing the other two.

2.1.1 Classification

- Let I be the set of all instances; that is to say the instance space. Let n be the number of features in this space, meaning n can be denoted as F = (f_1, f_2, …f_n) where f denotes a given feature.
- Features are assigned values and we assign each feature a value. Since we are working with a supervised environment each instance will belong to a certain class k (label); if c is a function that maps I to K(the space of all classes) we can write this mathematically as c: I → K.
- Since c is just one classifier from many say, c_1, c_2,….c_n, then we can generalize all classifiers as belonging to a set denoted by C.
- Consequently the classifier space consists of all instances mapped (paired) to their classes (labels).
- The classification problem therefore lies in ensuring that c will map all instances to their predicted classes.

2.2.1 Classification

- Using an unclassified instance u, we assign a class k from K. Using the same notation from 2.2.1 we can write this as c(u) = k
- When u is assigned a class k, we call this the classified instance of u, denoted as u^k
- Let P denote a classified set of instances (in pairs), such that each pair consists of an instance u and and assigned class k, with an instance index j.
- Thus we have $P = \{< u_j , k_j > | u_j \in I, k_j \in K, j = 1 \ldots n\}$
- This formula, however, assumes that there are no duplicate pairs (which in a real world example will exist), and does not take noise into account

2.2.2 Learning

- This problem is about selecting the best classifier for the instances.
- This because as discussed earlier the classifier space C is too big to choose from and certain criteria needs to be adopted in order to make this practical.
- Our training set T of course is much smaller than I.
- The process of inductive bias or prior assumptions of the learning target is used in this instance. It is a method makes the learner choose one algorithm over another.
- It is divided into two categories: representational and procedural.
- Representational – this defines which part of C the learner considers.
- Procedural – this defines how the learner will move through the part defined above (in the representational phase).

2.2.2 Learning

- Consequently the representational bias will influence the subspace that the learner gets to pick while the procedural bias will influence the selection process.
- By definition we say that a learning problem is realizable if there exists a true classifier (assumed known) in the classifier space C, and unrealizable if not. The classifier in this case will classify all instances of the learning problem correctly.
- Further in some literature the phrase realizable is replaced with faithful, that is to say, the classifier is referred to as being faithful (instead of realizable) or unfaithful (unrealizable).

2.2.3 Evaluation

- We choose to assign a value related to performance in order to evaluate the classifier chosen. Let us call this value x.
- Let us also have a learner – independent function (to enable proper evaluation) that will evaluate each classifier. Let us call this value γ. We then have γ(c, T) = x…(1); where c is a classifier, and T is the training set.
- On the same note and using the same notation let us have a learner dependent function from γ and call it γ^d; we then have γ^d(c, T) = x….(2)
- In most literature γ is usually error rate or accuracy rate (these were visited when discussing confusion matrix).
- Recall that in lesson 1 we described the concepts of overfitting and generalization. It was stated then that a good model is one that generalizes well. So in our formula instead of using x let us use g, which is a symbol for generalization. Thus (1) can be re-written as γ(c, T) = g….(3).
- This means that by measuring the level of generalization we can have a good indicator of how well a given classifier performs.

2.2.3 Evaluation

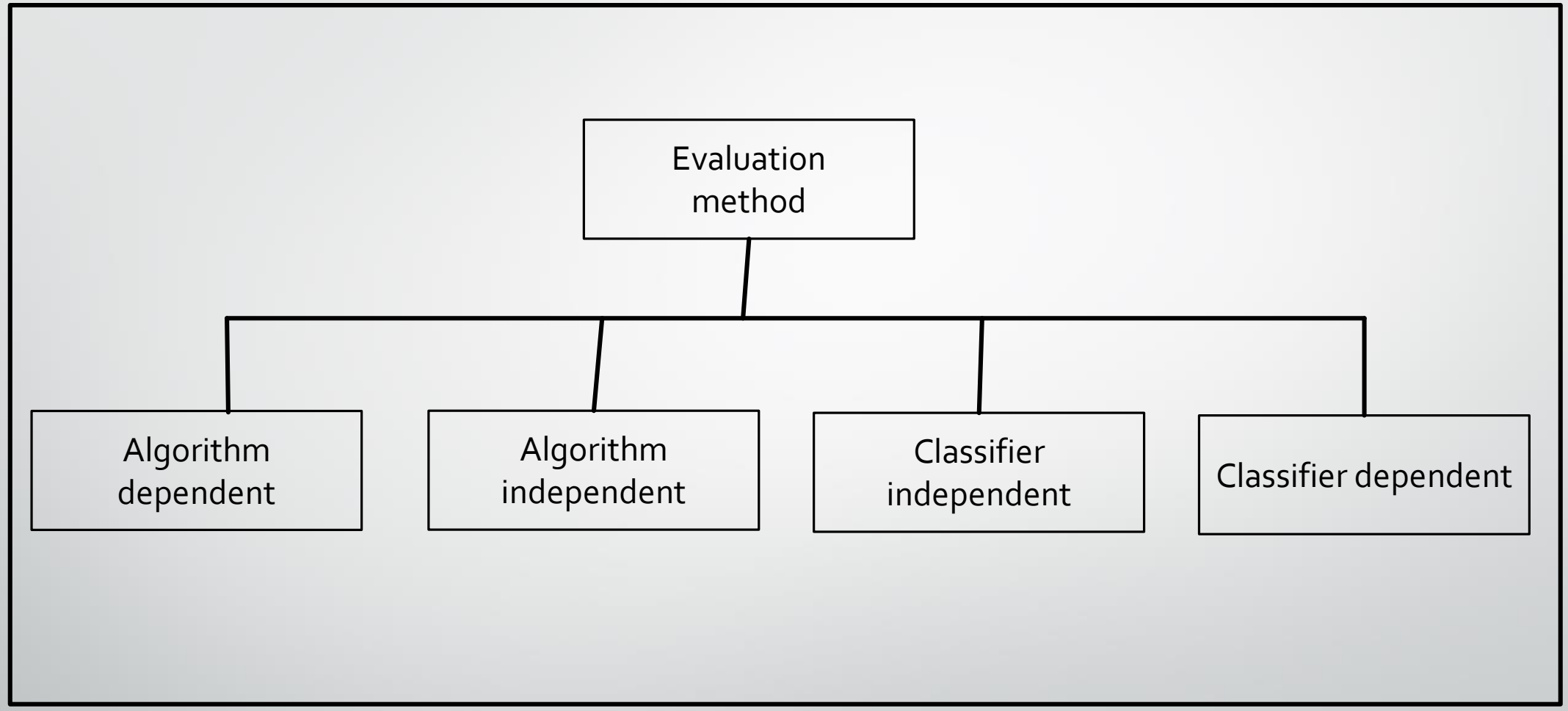

Fig 3. Taxonomy of algorithm evaluation (adapted from Lavesson, 2006)

2.2.3 Evaluation

- Levasson (2006) proposed a framework by which evaluation can be done.
- The study focused more in the domain of supervised learning, which is where classification is to be found.
- Fig 3 shows his framework (adapted from the study) and it proposes the 4 methods as:
  - Algorithm dependent
  - Algorithm independent
  - Classifier independent
  - Classifier dependent.

Algorithm dependent

- Algorithm dependent classifier evaluation depends on either a specific algorithm or a specific configuration.
- Two common methods used are:
- Vapnik-Chervonenkis (VC) bound – this is a measure based on the algorithmic capacity of a given algorithm, and the empirical risk associated with any classifier chosen by that algorithm. It is a theoretical measure.
- Minimum description length principle (MDL) – it borrows principles from compression algorithm methods in order to determine the complexity of an ML algorithm. Simply, any compression in data can be used to compress that data; thus MDL compares the size of a classifier with others to determine the best one to use. It is said to be not very accurate in choice of classifier in some instances.

Classifier independent

- There are several methods that can be used here: these include cross validation, jack-knife and bootstrap to mention just a few.
- Cross validation – this is done by partitioning a training set S into a number of subsets, say n. Training is performed on all the subsets except one. Each subset is called a fold, and the remaining subset is used to evaluate the chosen classifier. This process is repeated until each subset has been used for evaluation once. Care should be taken to ensure balance in the folds from S; that is, if there are 20% of class 1 in one fold, it should be the same in all the other folds.
- Bootstrap – Brownlee (2018) describes the process succinctly: "The bootstrap method is a resampling technique used to estimate statistics on a population by sampling a dataset with replacement. It can be used to estimate summary statistics such as the mean or standard deviation. It is used in applied machine learning to estimate the skill of machine learning models when making predictions on data not included in the training data."

Classifier dependent

- These methods depend on structure of the classifiers that need evaluation.
- For clarity in this naming classifiers refer to the general body of known classifiers while the algorithm is the specific one being referred to for this learning problem.
- Structural risk minimization (SRM) – this method works like VC method. The classifiers are arranged in subsets in a hierarchical structure and trained. One classifier is picked from each subset in order to minimize training error. The classifier among these with the lowest error and VC confidence is chosen as the best classifier for the problem.

Algorithm independent

- These evaluate performance as opposed to the properties of the algorithm, as seen previously.
- Sample error – this simply refers to the error fraction in the sample; that is, the fraction of the sample that has been misclassified.
- Receiver Operating Characteristics (ROC) – it uses the features in the contingency table to come up with a curve (true positive, false positive, true negative, false negative). The ROC curve is drawn based on the possibility that a instance belongs to a certain class. The probabilities are ranked and the curve drawn (up for each true positive and right for each false positive).
- Area under the ROC curve (AUC) – the area under the ROC curve can be used to measure how good a classifier is.
- Measure based evaluation – this makes use of calculating measures such as accuracy, recall, precision, and such like measures in evaluating the choice of classifier.

Bringing it together

- Levasson (2006) developed the final framework based on three parameters for evaluation of the algorithms. These were:
- Type: either empirical (E) or theoretical (T) (training and testing performed on same data) evaluation
- Target: algorithm (A) (those whose spaces can be arranged into subnets) or classifier (C)(from specific class space C)
- Metrics: Accuracy/Error/Subset fit (A), Complexity/Simplicity (C), Cost (M), Similarity (S).
- The resulting framework is shown in table 2.

Method	Type	Target	Metric
Sample Error (SE)	T,E	C	A
Cross-validation/Jackknife (CV/JK)	E	A	A
Bootstrap (BS)	E	A	A
Measure-based Evaluation (MF)	T	C	A,C,S
Structural Risk Minimisation (SRM)	T	A_n	A,C
Vapnik-Chervonenkis Bound (VCB)	T	C_n	A,C
ROC Plots (ROC)	T,E	A,C	M
Area Under the ROC Curve (AUC)	T,E	A,C	M
Minimum Description Length Principle (MDL)	T	C_n	C

Table 2. Evaluation methods framework (Levasson, 2006)

Part 3

Evaluation tests

Introduction

- So for we have examined different ways by which algorithms can be evaluated using performance.
- From fig 1 this roughly covers the box called ‘performance measures’.
- However, there are several tests that are also performed; the box named ‘statistical tests’.
- Whereas it is not our intention to delve into deep statistics, it is important to know what some of these tests that are used in performance evaluation.
- Fig 4 gives an overview of the tests depending on the number of domains and algorithms in question.
- The learner is encouraged to do some further reading on these tests.

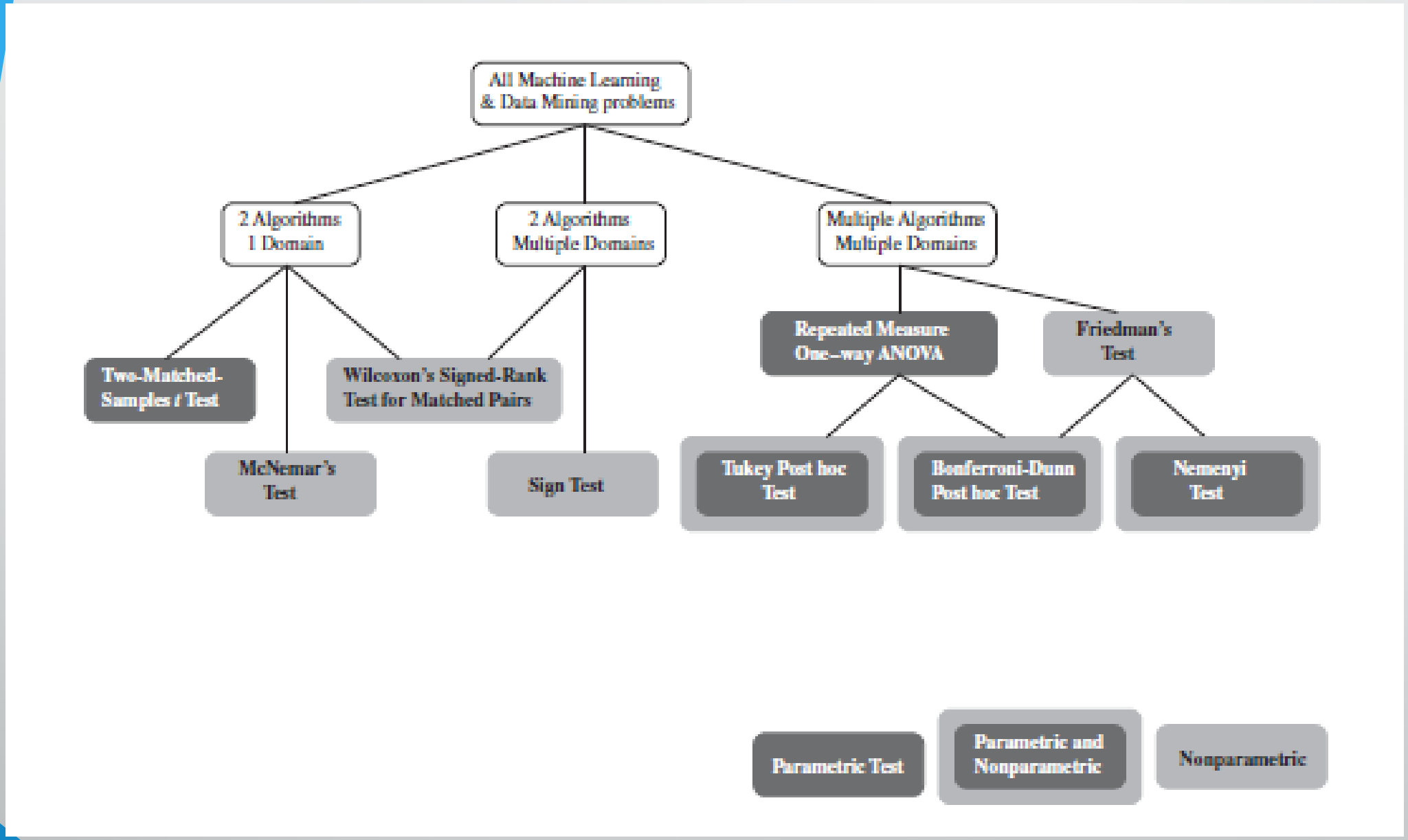

Fig 4 Statistical tests algorithms vs domains (Japokwicz and Shah, 2014)

Other tests

- Measurement errors – mean absolute error, mean square error, correlation coefficient of Pearson,
- Evaluation methods – done through regular assessment of the model, and through sequential evaluation.
- Benchmarking – Friedman's test, Nemenyi test.

Part 3

Evaluating model performance in Sklearn

Introduction

- Sklearn (scikit – learn) is "the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**." (tutorialspoint.com)
- The current version is scikit-learn 0.21.0 released in May 2019
- In classification it is used for applications such as spam detection and image recognition
- In regression it is used for applications such as drug response and stock prices
- In clustering it is used for applications such as customer segmentation and grouping experiment outcomes.
- In this final part we wish to see how it is used in the evaluation of algorithms.

Evaluating model performance

- When evaluating a model using sklearn there are different strategies that may be used.
- Inevitably it leads up to have some kind of scoring system such that some empirical values can reasonably guide in the evaluation.
- In sklearn there are three of such:
- Estimator score – it refers to the inbuilt score of the estimator via the score () method, and is specific to each estimator.
- Scoring parameters – these are the cross – validation tools that use an internal scoring strategy
- Metric functions – implemented vide metric module

Evaluating model performance (cont'd)

- Julian (2016) proposes that rather than use the estimator score () method it is more appropriate to use the cross_val_score object which is more specific scoring parameter.
- It has a cv parameter that controls how the data is split. it determines how many random consecutive splits are made on the data.
- Julian (2016) goes further to opine as follows: "Also important in cross_val_score is the scoring parameter. This is usually set by a string indicating a scoring strategy. For classification, the default is *accuracy*, and some common values are f1, precision, recall, as well as the micro-averaged, macro-averaged, and weighted versions of these. For regression estimators, the scoring values are mean_absolute_error, mean_squared error, median_ absolute_error, and r2. "

Evaluating model performance

- The literature by Julian (2016) also offers an example to demonstrate how this can be done:
- "The following code estimates the performance of three models on a dataset using 10 consecutive splits. Here, we print out the mean of each score, using several measures, for each of the four models. In a real-world situation, we will probably need to preprocess our data in one or more ways, and it is important to apply these data transformations to our test set as well as the training set. To make this easier, we can use the sklearn.pipeline module. This sequentially applies a list of transforms and a final estimator, and it allows us to assemble several steps that can be cross-validated together. Here, we also use the StandardScaler() class to scale the data. Scaling is applied to the logistic regression model and the decision tree by using two pipelines:

Evaluating model performance

- from sklearn import cross_validation
- from sklearn.tree import DecisionTreeClassifier
- from sklearn import svm
- from sklearn.linear_model import LogisticRegression
- from sklearn.datasets import samples_generator
- from sklearn.preprocessing import LabelEncoder
- from sklearn.preprocessing import StandardScaler
- from sklearn.cross_validation import cross_val_score
- from sklearn.pipeline import Pipeline
- X, y = samples_generator.make_classification(n_samples=1000,n_ informative=5, n_redundant=0,random_state=42)
- le=LabelEncoder()
- y=le.fit_transform(y)
- Xtrain, Xtest, ytrain, ytest = cross_validation.train_test_split(X, y, test_size=0.5, random_state=1)

Evaluating model performance

- clf1=DecisionTreeClassifier(max_depth=2,criterion='gini'). fit(Xtrain,ytrain)
- clf2= svm.SVC(kernel='linear', probability=True, random_state=0). fit(Xtrain,ytrain)
- clf3=LogisticRegression(penalty='l2', C=0.001).fit(Xtrain,ytrain)
- pipe1=Pipeline([['sc',StandardScaler()],['mod',clf1]])
- mod_labels=['Decision Tree','SVM','Logistic Regression']
- print('10 fold cross validation: \n')
- for mod,label in zip([pipe1,clf2,clf3], mod_labels):
- #print(label)
- auc_scores= cross_val_score(estimator= mod, X=Xtrain, y=ytrain, cv=10, scoring ='roc_auc')
- p_scores= cross_val_score(estimator= mod, X=Xtrain, y=ytrain, cv=10, scoring ='precision_macro')
- r_scores= cross_val_score(estimator= mod, X=Xtrain, y=ytrain, cv=10, scoring ='recall_macro')
- f_scores= cross_val_score(estimator= mod, X=Xtrain, y=ytrain, cv=10, scoring ='f1_macro')

Evaluating model performance

- **print(label)**
- **print("auc scores %2f +/- %2f " % (auc_scores.mean(), auc_scores. std()))**
- **print("precision %2f +/- %2f " % (p_scores.mean(), p_scores.std()))**
- **print("recall %2f +/- %2f]" % (r_scores.mean(), r_scores.std()))**
- **print("f scores %2f +/- %2f " % (f_scores.mean(), f_scores.std()))**
- The output of the code is presented in fig 5.

```
10 fold cross validation:

Decision Tree
auc scores 0.692144 +/- 0.056865
precision 0.706912 +/- 0.065688
recall 0.648131 +/- 0.043604 ]
f scores 0.628455 +/- 0.051711
SVM
auc scores 0.768374 +/- 0.038460
precision 0.709994 +/- 0.058011
recall 0.707064 +/- 0.056323 ]
f scores 0.703605 +/- 0.055579
Logistic Regression
auc scores 0.754150 +/- 0.048137
precision 0.688979 +/- 0.077614
recall 0.686077 +/- 0.076052 ]
f scores 0.682859 +/- 0.075356
```

Fig 5 Output of evaluation code (Julian,)

Summary

- Within the same domain you may have to use a different algorithm than the one favored in the literature; this brings about the need to have a way by which an evaluation of a suitable algorithm among many can be done.
- The algorithm and classifier problem can be broken down into 3 sub domains namely classification, learning and evaluation
- Levasson (2006) proposed a framework by which evaluation can be done; the 4 methods proposed are : algorithm dependent, algorithm independent, classifier independent, and classifier dependent.
- Other evaluation tests include f test, accuracy, precision, recall, error rate, Nemenyi test and Friedman test.
- In Python sklearn provides a platform to test and evaluate algorithms; this can be done by determining a scoring strategy and implementing it in the kit.

References

- Brownlee, J. (2019, August 8). *A gentle introduction to the Bootstrap Method*. Machine Learning Mastery. Retrieved June 6, 2022, from https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/#:~:text=The%20bootstrap%20method%20is%20a,the%20mean%20or%20standard%20deviation.
- Japkowicz, N., & Shah, M. (2014). *Evaluating learning algorithms: A classification perspective*. Cambridge University Press.
- Japkowicz, N. (2022, June). *Experimental Evaluation of Learning Algorithms (2009)*. *Machine Learning*. Ottawa; Canada.
- Julian, D. (2016). *Designing machine learning systems with python*. Packt Publishing Limited.

References

- Lavesson, N. (2006). *Evaluation and analysis of supervised learning algorithms and classifiers*. Blekinge Institute of Technology.
- *Learn*. scikit. (n.d.). Retrieved June 6, 2022, from https://scikit-learn.org/stable/
- *Scikit learn - introduction*. Tutorials Point. (n.d.). Retrieved June 6, 2022, from https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm
- Stancu, A.- M. R., & Perez-Danielescu, D. P. (2014). *Indian Journal of Applied Research*, *4*(5), 141–147. https://doi.org/10.15373/2249555X