Datasets Complexity using Network Science

Dieudonne Ouedraogo

10/04/2018

Abstract

In machine learning, the performance of a classifier is intrinsically related to the task. The structure of the dataset plays an important role. Even though new developments and improvements are done algorithmically, very little is done to understand the dataset structure. In many situations, it can take days even weeks to train and evaluate the models. If we have to try several models, this can be resource and time-consuming, and the current trial and error process for selecting the right algorithm is not efficient. We consider the case of a binary classification problem, but it is possible to generalize the findings into more than two classes problems. In this paper we use network metrics to describe the complexity of a data set relative to a classification task. A dataset is transformed into a graph representation based on the ϵNN algorithm. A data point is a node and an edge exist between two points i,j if $d(i, j) < \epsilon$. A post-processing step is applied to the graph, pruning edges between examples of different classes. The structural information such as density, clustering coefficient, and hubs are extracted. Various data sets are collected, and their metrics are computed and used as a training dataset to build a predictive model where the outputs are the algorithms competing to be used as the best classifier on a dataset. The algorithms used in this study are Decision Trees, Naive Bayes, SVM, Logistic regression, Artificial Neural Networks,K-nearest neighbors

keywords:

Meta-Learning, Machine Learning, Network Science, Dataset complexity, Classification, Algorithms Selection.

Introduction

In Machine learning a binary classification task difficulty can arise from the following reason: class ambiguity, the shape of the boundary, sample sparsity, feature space dimensionality and the network characteristics that define the dataset Papers published on the subject don't tackle the issue from a network science point of view. This paper is intended to show how vital the characteristics of a network formed by a dataset could be used to explain the performance of a classification task and to determine the competent algorithm to be assigned to the job.

We use binary classification here, but for multi-classes, the work could be easily extended by using one against all approach which will lead to binary classification. We use classification datasets available on UCI machine learning repository and dataset available on Weka.

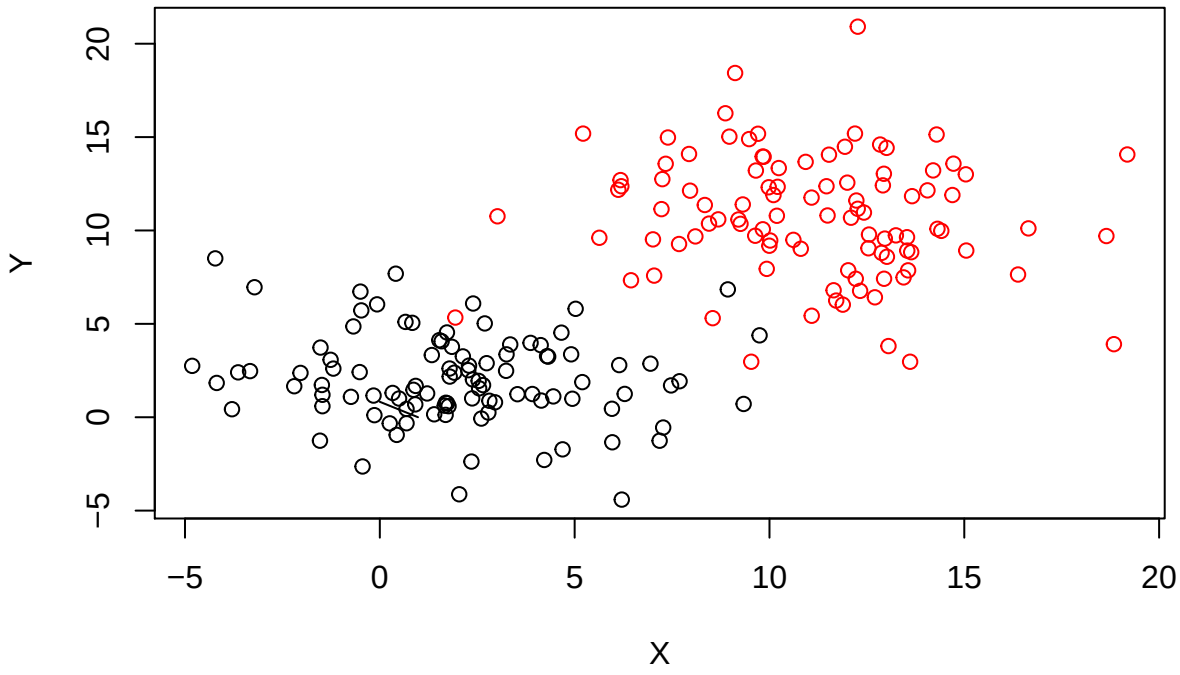
We explore the performance of five of the mainly used algorithms, Decision Trees, Naive Bayes, SVM, Logistic regression, Artificial Neural Networks,K-nearest neighbors.

The two plots below show two binary classifications one with two features (PredictorA and PredictorB) and another with features X and Y.

A task to determine the class for a data point could be relatively easy in one case and more complex in another

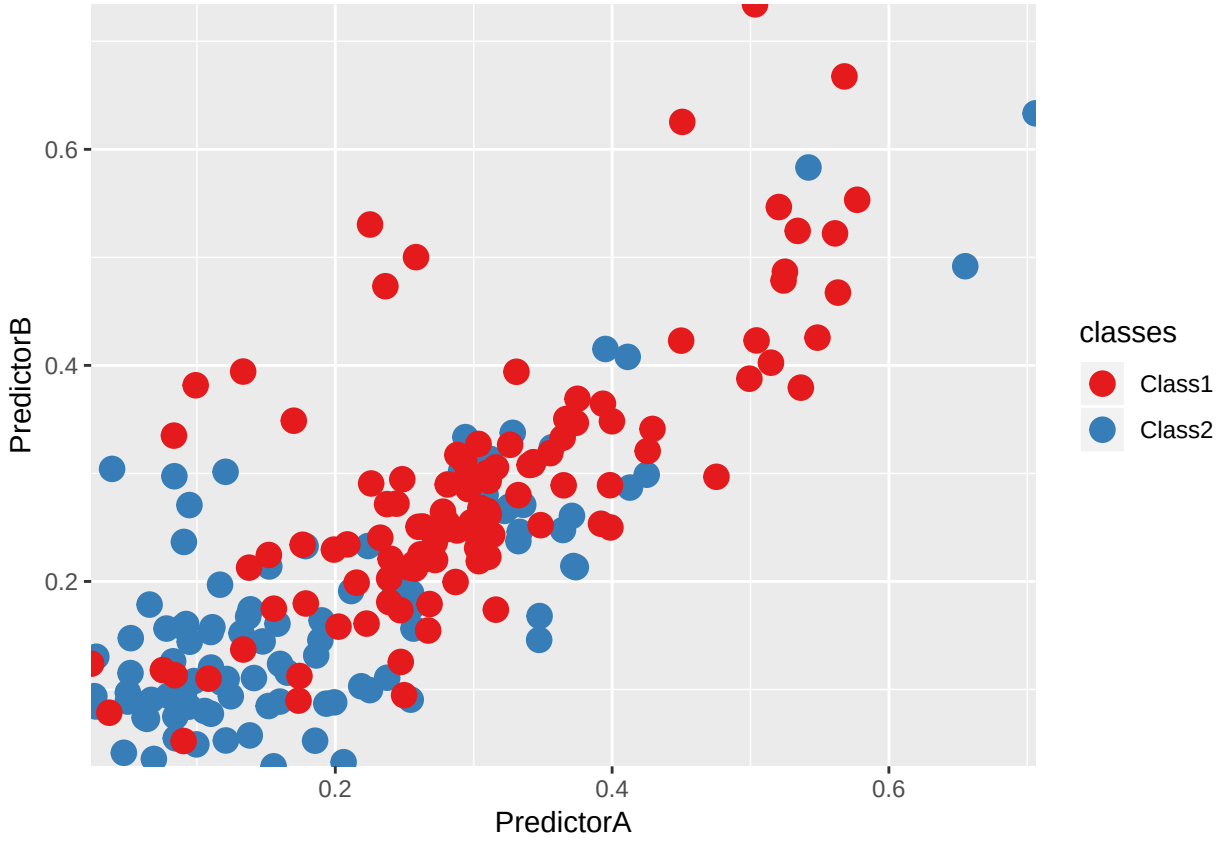
Low Complexity dataset for classification

Here the majority of the points in each class is easily separable by a straight line



High Complexity dataset for classification

Here, it is difficult to separate the two classes by a straight line



Literature review

A classification difficulty dramatically depends on the dataset. Understanding the characteristics associated with the dataset can help define and recommend the right algorithm to improve the performance of the task and to save time.

(Domingos, 2000) explained that the error of a predictor arises from three sources: a bias, from the difficulty of an algorithm to accurately model the relationship present in data; a variance, from the estimation of the correct parameters from the model due to imperfections from the sample used; a fundamental error referred to as noise.

For Ho & Basu (2002), classification difficulty comes from three components: the complexity of the decision boundary, the sample size and dimensionality induced sparsity and the ambiguity of the classes.

Ho (2008), believed that data complexity analysis is essential when comparing algorithms performance in machine learning. Usage of dataset complexity can also be found in combinatorial optimization, Smith-Miles & Lopes, (2012).

Choosing a sufficiently diverse set of problems to explain both strengths and weaknesses of the analyzed algorithms is essential in determining the domain of competence of the algorithm.

Macià et al. (2013), who described how algorithm comparison might be biased by benchmark dataset selection, and showed how complexity measures might guide the choice. Characterizing problem space with some metrics makes it possible to estimate regions in which specific algorithms perform well as detailed by Luengo & Herrera, (2013). This leads to possibilities of meta-learning as described by Smith-Miles et al.,(2014).

Complexity measures could then be used not only as predictors of classifier performance but also as diversity measures capturing various properties of the datasets.

Ho & Basu (2002) introduced complexity measures which were also extended by Ho, Basu & Law (2006) and Orriols-Puig, Macià & Ho (2010). Those measures are often used for algorithm's evaluation as described by Macià et al., (2013) and Luengo & Herrera, (2013), they are also used in meta-learning (Diez-Pastor et al., 2015; Mantovani et al., 2015). Part of these measures focuses on the overlap of values of specific attributes: Fisher's discriminant ratio, the volume of the overlap region, the attribute efficiency, etc.

Another part is toward the class separability; in this section, we have measures such as the fraction of points on the decision boundary, the linear separability, the ratio of intra/interclass distance. Those measures focus on specific properties of the classification problem, measuring the shape of the decision boundary and the amount class overlap. We also have topological measures concerned with data sparsity, such as the ratio of attributes to observations.

Li & Abu-Mostafa (2006) defined dataset complexity using the general concept of Kolmogorov complexity. The measures proposed to use the number of support vectors in the support vector machine (SVM) classifier. They analyzed the problems of data decomposition and data pruning using the above methodology. They defined the representation of the dataset complexity called the complexity-error plot.

Smith, Martinez & Giraud-CARRIER(2013) tackled the problem of complexity in the dataset from a single instance point of view where they analyzed misclassified instances by various algorithms approach to data complexity is to explain. They devised local complexity measures calculated concerning the individual case and explored the correlations of those measures with the global data complexity measures of Ho & Basu (2002). They concluded that they are mainly related to class overlap.

Yin et al. (2013) used a feature selection based on Hellinger distance to described complexity by measuring the similarity between probability distributions. They chose features, which conditional distributions (depending on the class) have a minimal affinity. The authors demonstrated experimentally that, for the high-dimensional imbalanced data sets, their method is superior to popular feature selection methods using the Fisher criterion, or mutual information.

Dataset complexity measures

A-Measure of overlapping

The feature overlapping measures characterize how informative the available fea- tures are to separate the classes

F1 - Maximum Fisher discriminant ratio.

The measure gives the effectiveness of a single feature in separating the classes This measure computes the maximum discriminative power (Fisher ratio) of the attributes. The ratio is defined as

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

where μ_1,μ_2,σ_1^2, σ_2^2 are the means and variances of each class, respectively, in that feature. f is computed for each feature and maximum is taken as F1. A high value of F1 indicates that at least one of the attributes enables the learner to separate the examples of different classes with partitions that are parallel to an axis of the feature space. A low value of this measure does not imply that the classes are not linearly separable, but that they cannot be discriminated by hyperplanes parallel to one of the axis of the feature space.

F1v

Directional-vector maximum Fisher's discriminant ratio (F1v) complements F1 by searching for a vector able to separate two classes after the training examples have been projected into it.

F2

Volume of overlap region,this measure computes the overlap of the tails of distributions defined by the instances of each class. Let $min(f_i, c_j)$ and $max(f_i, c_j)$ be, respectively, the minimum and maximum values of the feature f_i for class c_j. Then, the overlap measure is defined as

$$F2 = \prod \frac{MINMAX_i - MAXMIN_i}{MAXMAX_i - MINMIN_i}$$

where $MINMAX_i = min(max(f_i, c_1), max(f_i, c_2))$ $MAXMIN_i = max(min(f_i, c_1), min(f_i, c_2))$ $MAXMAX_i = max(max(f_i, c_1), max(f_i, c_2))$ $MINMin_i = min(min(f_i, c_1), min(f_i, c_2))$

A low value states that the attributes can discriminate the examples of different classes.

F4

Collective feature efficiency(F4) get an overview on how various features may work together in data separation. First the most discriminative feature according to F3 is selected and all examples that can be separated by this feature are removed from the dataset. The previous step is repeated on the remaining dataset until all the features have been considered or no example remains. F4 returns the ratio of examples that have been discriminated.

B- measures of neighborhood

Neighborhood measures characterize the presence and density of same or different classes in local neighborhoods. The Neighborhood measures analyze the neighborhoods of the data items and try to capture class overlapping and the shape of the decision boundary. They work over a distance matrix storing the distances between all pairs of data points in the dataset. To deal with both symbolic and numerical features, we adopt a heterogeneous distance measure named Gower distance.

N1

Fraction of borderline points (N1) computes the percentage of vertexes incident to edges connecting examples of opposite classes in a Minimum Spanning Tree (MST).

N2

Ratio of intra/extra class nearest neighbor distance (N2) computes the ratio of two sums: intra-class and inter-class. The former corresponds to the sum of the distances between each example and its closest neighbor from the same class. The later is the sum of the distances between each example and its closest neighbor from another class (nearest enemy).

N3

Error rate of the nearest neighbor(N3)classifier corresponds to the error rate of a one Nearest Neighbor (1NN) classifier, estimated using a leave-one-out procedure in dataset.

N4

Non-linearity of the nearest neighbor classifier (N4) creates a new dataset randomly interpolating pairs of training examples of the same class and then induce a the 1NN classifier on the original data and measure the error rate in the new data points.

T1

Fraction of hyper-spheres covering data (T1) builds hyper-spheres centered at each one of the training examples, which have their radios growth until the hyper-sphere reaches an example of another class. Afterwards, smaller hyper-spheres contained in larger hyper-spheres are eliminated. T1 is finally defined as the ratio between the number of the remaining hyper-spheres and the total number of examples in the dataset.

LSCAvg

Local Set Average Cardinality (LSCAvg) is based on Local Set (LS) and defined as the set of points from the dataset whose distance of each example is smaller than the distance from the examples of the different class. LSCAvg is the average of the LS.

C-Measures of linearity

The linearity measures try to quantify if it is possible to separate the classes by a hyper-plane. The underlying assumption is that a linearly separable problem can be considered simpler than a problem requiring a non-linear decision boundary.

L1

Sum of the error distance by linear programming (L1) computes the sum of the distances of incorrectly classified examples to a linear boundary used in their classification.

L2

Error rate of linear classifier(L2)computes the error rate of the linear SVM classifier induced from dataset.

L3

Non-linearity of a linear classifier (L3) creates a new dataset randomly interpolating pairs of training examples of the same class and then induce a linear SVM on the original data and measure the error rate in the new data points.

D- Measures of dimensionality

These measures give an indicative of data sparsity. They capture how sparse a datasets tend to have regions of low density. These regions are know to be more difficult to extract good classification models.

T2

Average number of points per dimension (T2) is given by the ratio between the number of examples and dimensionality of the dataset.

T3

Average number of points per PCA (T3) is similar to T2, but uses the number of PCA components needed to represent 95 variability as the base of data sparsity assessment.

T4

Ratio of the PCA Dimension to the Original (T4) it estimates the proportion of relevant and the original dimensions for a dataset.

E-Measures of class balance

These measures capture the differences in the number of examples per class in the dataset. When these differences are severe, problems related to generalization of the ML classification techniques could happen because of the imbalance ratio.

C1

The entropy of class proportions (C1) measure the imbalance in a dataset based on the pro- portions of examples per class.

C2

The imbalance ratio (C2) is an index computed for measuring class balance. This is a version of the measure that is also suited for multiclass classification problems.

Measures of Network

The network measures represent the dataset as a graph and extract structural information from it. The transformation between raw data and the graph representation is based on the epsilon-NN algorithm. Next, a post-processing step is applied to the graph, pruning edges between examples of opposite classes.

Density

Average Density of the network (Density) represents the number of edges in the graph, divided by the maximum number of edges between pairs of data points.

ClsCoef

Clustering coefficient (ClsCoef) averages the clustering tendency of the vertexes by the ratio of existent edges between its neighbors and the total number of edges that could possibly exist between them.

Hubs

Hubs score (Hubs) is given by the number of connections it has to other nodes, weighted by the number of connections these neighbors have.

3-Methodology

Part1

Creation of the graph from a dataset and extraction of metrics. each data point from the dataset is a node. We transform the dataset into a graph and we extract structural information from it. The transformation between raw data and the graph representation is based on the ϵNN algorithm.nodes i and j are connected by an edge, if the distance $d(i, j) < \epsilon$. The hyper-parameter ϵ controls the neighborhood radius. Next, a post-processing step is applied to the graph, pruning edges between examples of opposite classes.

Dataset1

```
library(pander)
library(RDocumentation)
library(ECoL)
dataset1=twoClass
kable(head(dataset1))
```

PredictorA	PredictorB	classes
0.1582	0.1609	Class2
0.6552	0.4918	Class2
0.7060	0.6333	Class2
0.1992	0.0881	Class2
0.3952	0.4152	Class2
0.4250	0.2988	Class2

```
net1_Metrics=network(classes ~.,dataset1)
pander(net1_Metrics)
```

Density	ClsCoef	Hubs
0.1227	0.6626	0.1902

Dataset2 IRIS dataset

```
dataset2=iris
kable(head(iris))
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

```
net2_Metrics=network(Species ~ ., dataset2)
pander(net2_Metrics)
```

Density	ClsCoef	Hubs
0.1669	0.7326	0.2173

Dataset3

Breast Cancer Winsconsin

NB

```
library(readr)
BreastCancer<- read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wiscons
```

```
colnames(BreastCancer) <- c("SampleCodeNumber",
                            "ClumpThickness",
                            "UniformityOfCellSize",
                            "UniformityOfCellShape",
                            "MarginalAdhesion",
                            "SingleEpithelialCellSize",
                            "BareNuclei",
                            "BlandChromatin",
                            "NormalNucleoli",
                            "Mitosis",
                            "Classes")

BreastCancer$Classes <- ifelse(BreastCancer$Classes == "2", "Benign",
                               ifelse(BreastCancer$Classes == "4", "Malignant", NA))

#Data cleaning
BreastCancer[BreastCancer == "?"] <- NA
#length(which(is.na(BreastCancer)))
#Comparing the effect of the removal of NA
#str(BreastCancer)
#BreastCancer
nrow(BreastCancer[is.na(BreastCancer), ])
```

```
## [1] 16
```

```
#nrow(BreastCancer)
df=BreastCancer[,-c(1,7,11)]
#df
dataset3=BreastCancer[,-c(1,7)]
#str(df2)
#kable(head(dataset3))
net3_Metrics=network (Classes ~ ., dataset3)
pander(net3_Metrics)
```

Density	ClsCoef	Hubs
0.1877	0.615	0.3178

dataset4

Abalone dataset

```
abalone_data <- read_csv("~/Downloads/abalone.data.txt", col_names = FALSE)
dataset4=abalone_data
net4_Metrics=network (X1 ~ ., dataset4)
pander(net4_Metrics)
```

Density	ClsCoef	Hubs
0.08261	0.7537	0.1408

dataset5

Breast Cancer Data set

Decision Trees

```
breast.cancer.data <- read.csv("~/Downloads/breast-cancer.data.txt", header=FALSE)
dataset5=breast.cancer.data
net5_Metrics=complexity(V1 ~ ., dataset5)
net5_Metrics
```

```
##      overlapping.F1     overlapping.F1v      overlapping.F2
##                 Inf          1.20498637          0.00000000
##      overlapping.F3      overlapping.F4     neighborhood.N1
##          0.01398601          0.02097902          0.47202797
##     neighborhood.N2     neighborhood.N3     neighborhood.N4
##          0.77201705          0.27622378          0.16433566
##     neighborhood.T1 neighborhood.LSCAvg        linearity.L1
##          0.57342657          0.01135752          0.19354737
##        linearity.L2        linearity.L3   dimensionality.T2
##          0.25524476          0.23426573          8.41176471
##   dimensionality.T3   dimensionality.T4          balance.C1
##         14.30000000          0.58823529          0.87784470
##          balance.C2     network.Density     network.ClsCoef
##          1.39379573          0.12462275          0.46805337
##        network.Hubs
##          0.42546745
```

dataset6

Appendicitis dataset

SVM

```
Appendicitis<- read.csv("~/Downloads/php8KBo4A.csv")
dataset6=Appendicitis
net6_Metrics=network (Class ~ ., dataset6)
pander(net6_Metrics)
```

Density	ClsCoef	Hubs
0.1549	0.5503	0.3776

Hepatitis

Knn(21-nn)

```
hepatitis.data <- read.csv("~/Downloads/hepatitis.data.txt", header=FALSE)
dataset7=hepatitis.data
net7_Metrics=network (V1 ~ ., dataset7)
pander(net7_Metrics)
```

Density	ClsCoef	Hubs
0.163	0.4624	0.327

Cleveland heart dataset

28-nn

SVM

```
cleveland.data <- read.csv("~/Downloads/processed.cleveland.data.txt", header=FALSE)
dataset8=cleveland.data
net8_Metrics=network (V14 ~ ., dataset8)
pander(net8_Metrics)
```

Density	ClsCoef	Hubs
0.1003	0.5413	0.2798

Diabetes

logdisc

lda

SVM

```
diabetes <- read.csv("~/Downloads/diabetes.csv")
dataset9=diabetes
net9_Metrics=network (Outcome ~ ., dataset9)
pander(net9_Metrics)
```

Density	ClsCoef	Hubs
0.131	0.5238	0.2583

Hypotyroid

Decision trees(CART)

```
ann.train.data <- read.table("~/Downloads/ann-train.data.txt", quote="\"", comment.char="")
dataset10=ann.train.data
net10_Metrics=network (V22 ~ ., dataset10)
pander(net10_Metrics)
```

Density	ClsCoef	Hubs
0.1894	0.6415	0.3496

Hepatobiliary disorders

Neural Network

Landsat Satellite image datase

MLP,Neural Network

```
sat.trn <- read.table("~/Downloads/sat.trn.txt", quote="\"", comment.char="")
dataset11=sat.trn
net11_Metrics=network (V37 ~ ., dataset11)
pander(net11_Metrics)
```

Density	ClsCoef	Hubs
0.1147	0.8305	0.1613

Ionosphere

knn

(3-nn)

With CV

SVM

```
ionosphere.data <- read.csv("~/Downloads/ionosphere.data.txt", header=FALSE)
dataset12=ionosphere.data
net12_Metrics=network (V1 ~ ., dataset12)
pander(net12_Metrics)
```

Density	ClsCoef	Hubs
0.1911	0.6085	0.3759

Sonar: Mines vs Rocks

1-nn

With CV

MLP Neural Network

```
sonar.all.data <- read.csv("~/Downloads/sonar.all-data.txt", header=FALSE)
dataset13=sonar.all.data
net13_Metrics=network (V61 ~ ., dataset13)
pander(net13_Metrics)
```

Density	ClsCoef	Hubs
0.1311	0.56	0.2608

Vovel

DT

Test over 90%

1-NN

for best training

```
vowel.context.data <- read.table("~/Downloads/vowel-context.data.txt",quote="\"",comment.char="")
dataset14=vowel.context.data
net14_Metrics=network(V14 ~ ., dataset14)
```

```
## Warning in cluster::daisy(x, metric = "gower", stand = TRUE): binary
## variable(s) 1, 3 treated as interval scaled
```

```
pander(net14_Metrics)
```

Density	ClsCoef	Hubs
0.02253	0.8426	0.03761

Wine

RDA,QDA,LDA

kNN

With CV

kNN

```
wine.data <- read.csv("~/Downloads/wine.data.txt", header=FALSE)
dataset15=wine.data
net15_Metrics=network(V1 ~ ., dataset15)
pander(net15_Metrics)
```

Density	ClsCoef	Hubs
0.1689	0.6495	0.2124

Glass identification

kNN

```
glass.data <- read.csv("~/Downloads/glass.data.txt", header=FALSE)
dataset16=glass.data
net16_Metrics=network(V11 ~ ., dataset16)
pander(net16_Metrics)
```

Density	ClsCoef	Hubs
0.1271	0.7043	0.2188

DNA-Primate splice-junction gene sequences

RBF Neural Network 720 nodes

```
splice.data <- read.csv("~/Downloads/splice.data.txt", header=FALSE)
dataset17=splice.data
net17_Metrics=network(V1 ~ ., dataset17)
pander(net17_Metrics)
```

Density	ClsCoef	Hubs
0.04958	0.885	0.2129

Adult

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/

NB and Dt

```
#adult.data <- read.csv("~/Downloads/adult.data.txt", header=FALSE)
#dataset18=adult.data
#net18_Metrics=network(V15 ~ ., dataset18)
#pander(net18_Metrics)
```

Part2

Creation of the meta Dataset

We pick datasets from UCI and Weka and Kaggle which cover a broad spectrum of characteristics for datasets. We create a meta feature target which value represents the best classifier reported so far by previous studies on that particular dataset. We use the characteristics of our networks as features for the meta-dataset (the dataset formed by the collection of datasets). The set of values of the characteristics of each dataset is an instance(observation) in our meta-dataset.

Meta-Dataset metrics

Dummy Characteristics

```
DummyChar <- read.csv("~/DummyChar.csv")
#My Meta Dataset
dataset <- DummyChar
dataset
```

```
##       X  Al     Density     ClsCoef        Hubs
## 1     1 Knn 0.434580803 0.505828630 0.834303992
## 2     2 ANN 0.090080537 0.224253573 0.436768870
## 3     3 Knn 0.419699779 0.155886438 0.273755538
## 4     4 ANN 0.052599533 0.369392853 0.007740361
## 5     5 ANN 0.256935979 0.969474012 0.294008983
## 6     6 Knn 0.368179576 0.110480352 0.825518414
## 7     7 ANN 0.670263665 0.134672191 0.188341918
## 8     8 SVM 0.327016433 0.848711706 0.972758641
## 9     9 SVM 0.934182715 0.660916655 0.727416854
## 10   10  DT 0.793256408 0.397110511 0.998485462
## 11   11 ANN 0.832247808 0.139800921 0.567296213
## 12   12 ANN 0.955299830 0.582108269 0.027239352
## 13   13  DT 0.751189198 0.441838027 0.636733853
## 14   14  DT 0.247397340 0.274850797 0.467445820
## 15   15 ANN 0.635009364 0.096827013 0.212732401
## 16   16 GLM 0.477511239 0.127507338 0.912666553
## 17   17 GLM 0.075253368 0.555477221 0.131263494
## 18   18 Knn 0.714617108 0.749057634 0.292804381
## 19   19 GLM 0.279828392 0.433155332 0.310376519
## 20   20  DT 0.083322857 0.614668286 0.229193623
## 21   21 SVM 0.774320417 0.394164665 0.459603766
## 22   22 SVM 0.742131842 0.296841853 0.609014487
## 23   23  DT 0.917339455 0.066138336 0.830549632
## 24   24  DT 0.706701624 0.557290955 0.594914233
## 25   25 ANN 0.531069254 0.627113671 0.246578350
## 26   26 GLM 0.660572416 0.348658987 0.635287225
## 27   27 GLM 0.505592469 0.325275371 0.344468823
## 28   28  DT 0.067123532 0.771095380 0.061027562
## 29   29 ANN 0.839457619 0.998739472 0.378727464
## 30   30  DT 0.534837588 0.838920244 0.377442583
## 31   31 SVM 0.294662886 0.328155127 0.282598115
## 32   32 Knn 0.647637736 0.346642039 0.279111577
## 33   33 Knn 0.892987225 0.268089701 0.303393231
## 34   34 Knn 0.789889335 0.372382391 0.374572926
## 35   35  DT 0.327302751 0.923850395 0.554780233
```

```
## 36   36  DT 0.385755476 0.549616409 0.649107078
## 37   37 ANN 0.844058162 0.778503740 0.161517241
## 38   38 SVM 0.149129537 0.400834924 0.434228894
## 39   39 Knn 0.905944139 0.488055433 0.931709651
## 40   40 Knn 0.490494096 0.846814046 0.920801261
## 41   41 SVM 0.849580212 0.565812823 0.076611118
## 42   42 ANN 0.524455148 0.608506304 0.829641144
## 43   43 ANN 0.435558229 0.686716855 0.890104957
## 44   44  DT 0.377797641 0.717299161 0.678433914
## 45   45 ANN 0.842881087 0.309635599 0.120701248
## 46   46 SVM 0.396095385 0.717672178 0.521147619
## 47   47 ANN 0.433861117 0.652951320 0.783085904
## 48   48 ANN 0.619266049 0.241513318 0.112547658
## 49   49 GLM 0.747957997 0.822065714 0.203100086
## 50   50 SVM 0.635721548 0.010247614 0.981186080
## 51   51  DT 0.915156104 0.672113417 0.188252902
## 52   52 ANN 0.381690593 0.063075250 0.492589360
## 53   53  DT 0.136195937 0.315327777 0.499114726
## 54   54 Knn 0.394717490 0.947672809 0.518582083
## 55   55 GLM 0.452000631 0.925481438 0.121044706
## 56   56 GLM 0.955698144 0.278794164 0.629601354
## 57   57 Knn 0.809108085 0.160447626 0.367712016
## 58   58 SVM 0.677199753 0.828859241 0.855054024
## 59   59 ANN 0.942715781 0.297373000 0.386013612
## 60   60 ANN 0.321297069 0.194484663 0.540044380
## 61   61 GLM 0.840824933 0.358876864 0.053578788
## 62   62 GLM 0.761308599 0.025766444 0.963520196
## 63   63 ANN 0.333991461 0.536774381 0.217451517
## 64   64  DT 0.383448350 0.727733783 0.347446445
## 65   65 ANN 0.520947080 0.470624811 0.232816439
## 66   66 GLM 0.932378297 0.989035743 0.763457848
## 67   67 ANN 0.908609169 0.689000950 0.336789932
## 68   68 SVM 0.237595514 0.947060585 0.101169857
## 69   69 GLM 0.568510177 0.803745979 0.852907582
## 70   70 Knn 0.121289274 0.369490425 0.735202145
## 71   71 Knn 0.464104182 0.054309928 0.114017973
## 72   72 GLM 0.001329487 0.497923074 0.853308586
## 73   73 Knn 0.785753832 0.006241411 0.089414136
## 74   74 ANN 0.271361142 0.791904239 0.340363621
## 75   75 ANN 0.395177545 0.179661431 0.305195317
## 76   76 Knn 0.571995028 0.368167469 0.155040714
## 77   77 GLM 0.396877033 0.142592615 0.640644673
## 78   78  DT 0.214104882 0.383716457 0.493126134
## 79   79 SVM 0.266732447 0.327382949 0.722287842
## 80   80 ANN 0.921491834 0.343418207 0.183504572
## 81   81 Knn 0.257919340 0.544612807 0.475219886
## 82   82 Knn 0.818323778 0.673514441 0.496796929
## 83   83 ANN 0.011563461 0.742580002 0.406963136
## 84   84 SVM 0.431548402 0.441676070 0.837973147
## 85   85 GLM 0.961294862 0.308448579 0.166292246
## 86   86 ANN 0.173327772 0.763190150 0.343508566
## 87   87 Knn 0.236495495 0.920607767 0.386814666
## 88   88 Knn 0.150972271 0.709880081 0.668038144
## 89   89 SVM 0.118577880 0.908701573 0.011638352
```

```
## 90   90 GLM 0.326196855 0.013256095 0.751364159
## 91   91 SVM 0.041436470 0.825745783 0.860852013
## 92   92  DT 0.799129178 0.355889887 0.582804612
## 93   93 SVM 0.199266033 0.251417209 0.375890657
## 94   94  DT 0.321273886 0.342287786 0.845589792
## 95   95 ANN 0.636599988 0.398546886 0.332593503
## 96   96 Knn 0.659727328 0.808344178 0.669480772
## 97   97 GLM 0.259194372 0.473933065 0.777932419
## 98   98 GLM 0.722354265 0.921819190 0.944273145
## 99   99 SVM 0.813069060 0.880828673 0.986080467
## 100 100 GLM 0.603607563 0.028597273 0.966878956
```

Part3

We run multiclass classification algorithms of our meta-dataset. From this classification, we extract the most predictive features from the network characteristics. We then build a predictive model based on the network characteristics related to datasets

```
library(caret)
```

```
## Loading required package: lattice
```

```
validation_index <- createDataPartition(dataset$Al, p=0.80, list=FALSE)
# select 20% of the data for validation
validation <- dataset[-validation_index,]
# use the remaining 80% of data to training and testing the models
dataset <- dataset[validation_index,]

# dimensions of dataset
dim(dataset)
```

```
## [1] 82  5
```

```
# list types for each attribute
sapply(dataset, class)
```

```
##         X        Al   Density   ClsCoef      Hubs
## "integer"  "factor" "numeric" "numeric" "numeric"
```

```
# list the levels for the class
levels(dataset$Al)
```

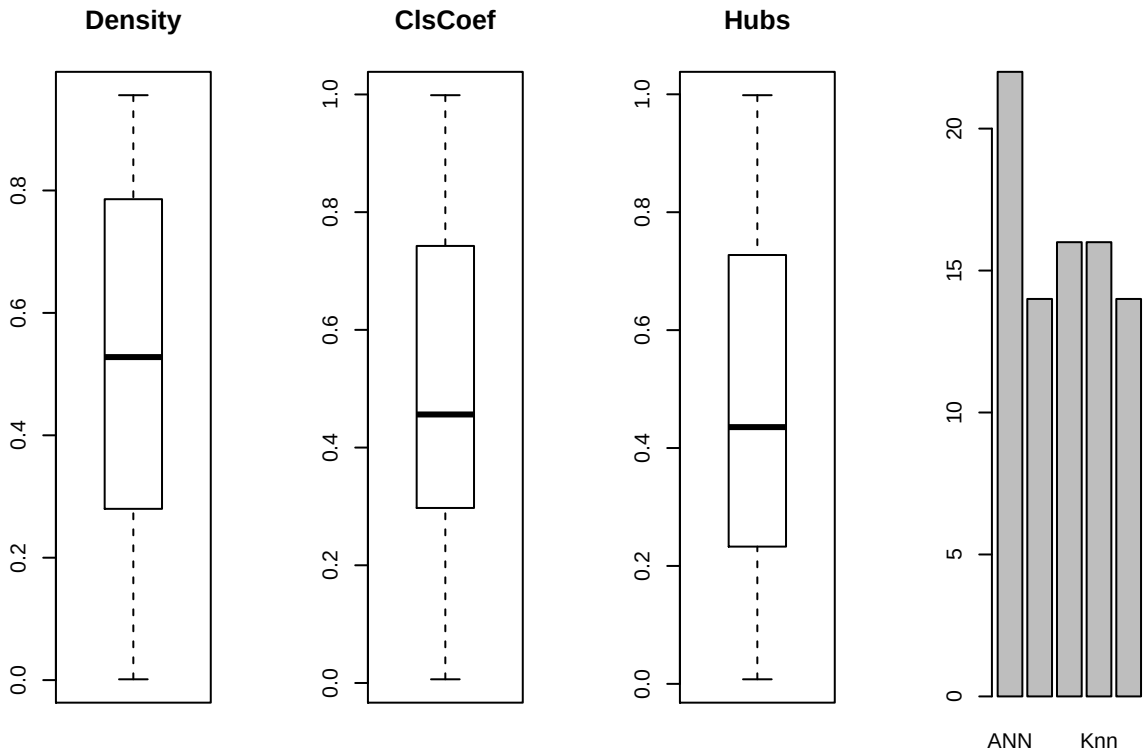
```
## [1] "ANN" "DT"  "GLM" "Knn" "SVM"
```

```
# summarize the class distribution
percentage <- prop.table(table(dataset$Al)) * 100
cbind(freq=table(dataset$Al), percentage=percentage)
```

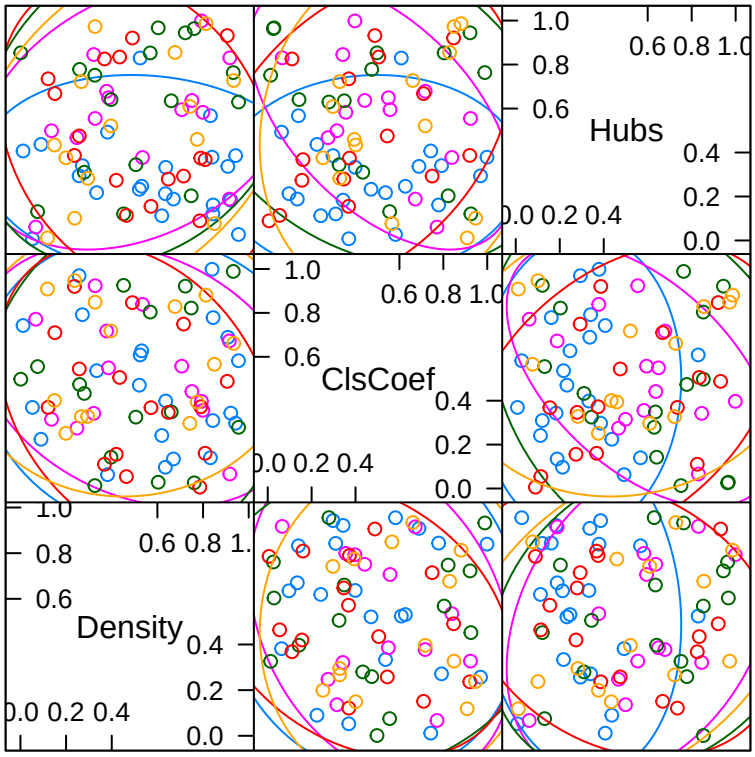
```
##     freq percentage
## ANN   22   26.82927
## DT    14   17.07317
## GLM   16   19.51220
## Knn   16   19.51220
## SVM   14   17.07317
```

```
# split input and output
x <- dataset[,3:5]
y <- dataset[,2]
# boxplot for each attribute on one image
par(mfrow=c(1,4))
```

```r
for(i in 1:3) {
  boxplot(x[i], main=names(x)[i])
}

# barplot for class breakdown
plot(y)
```

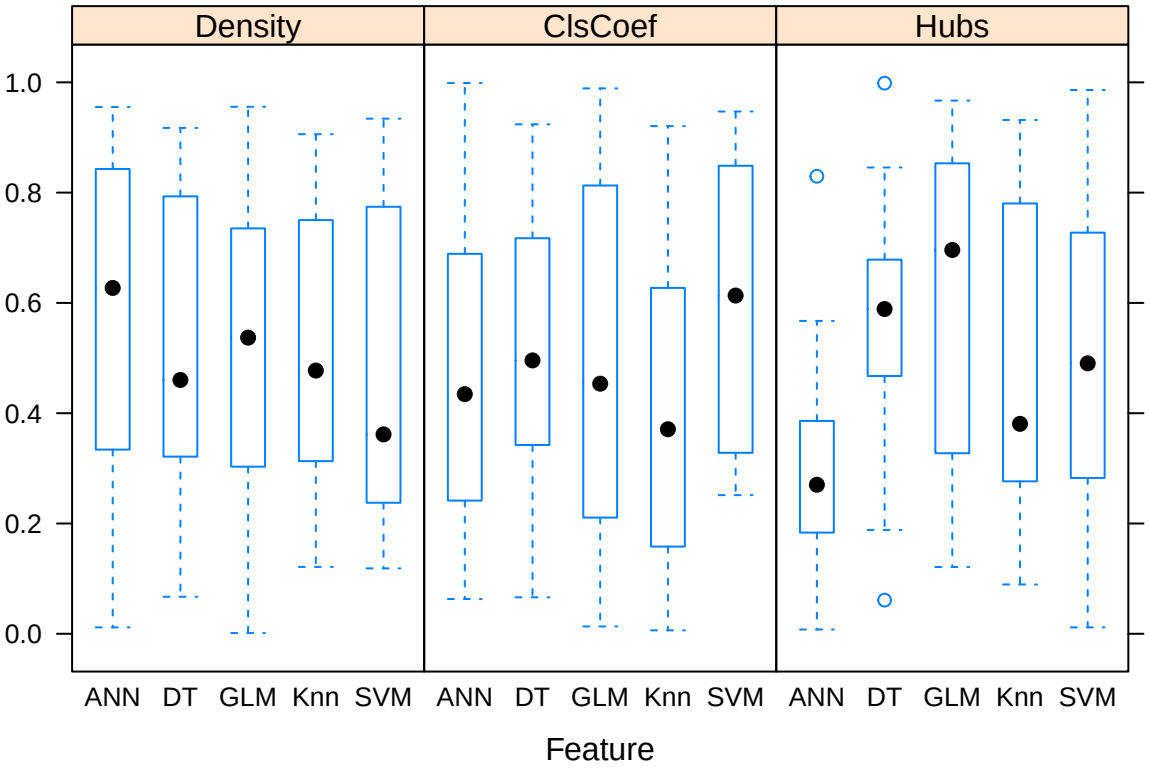



```r
# scatterplot matrix
featurePlot(x=x, y=y, plot="ellipse")
```

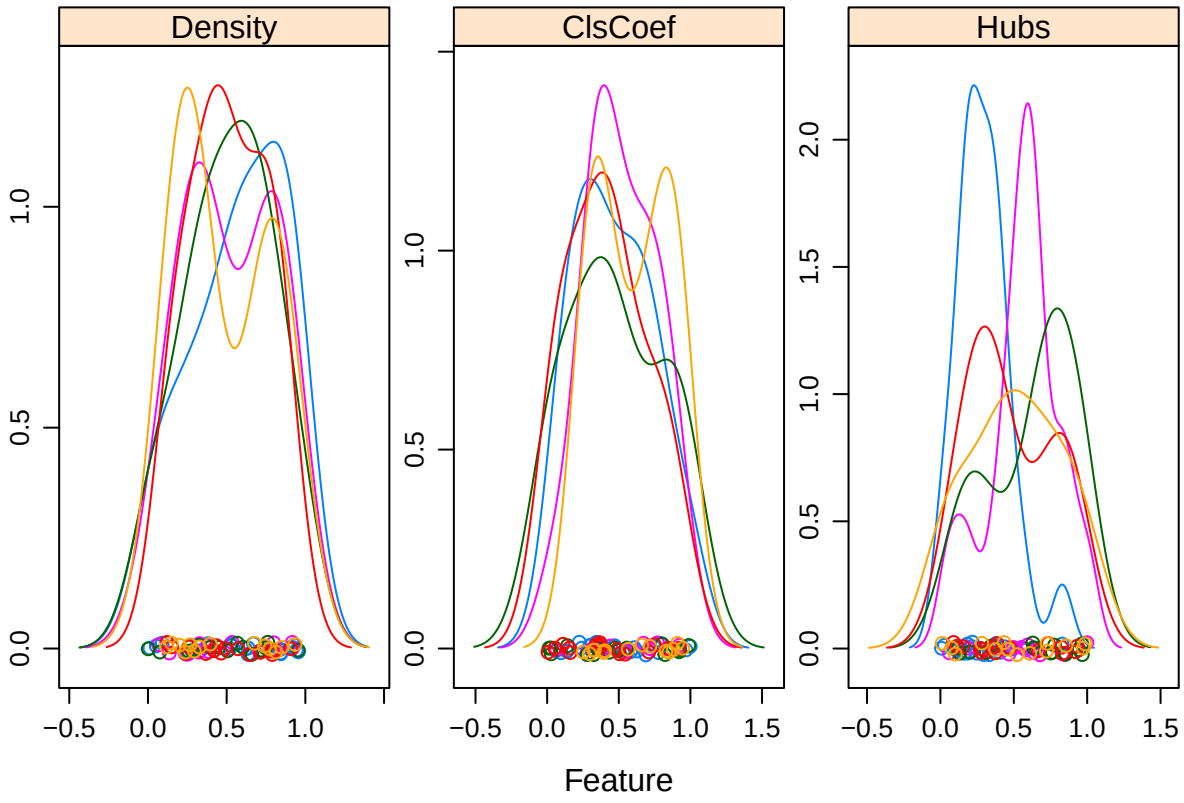


Scatter Plot Matrix

```
# box and whisker plots for each attribute
featurePlot(x=x, y=y, plot="box")
```



```
# density plots for each attribute by class value
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)
```

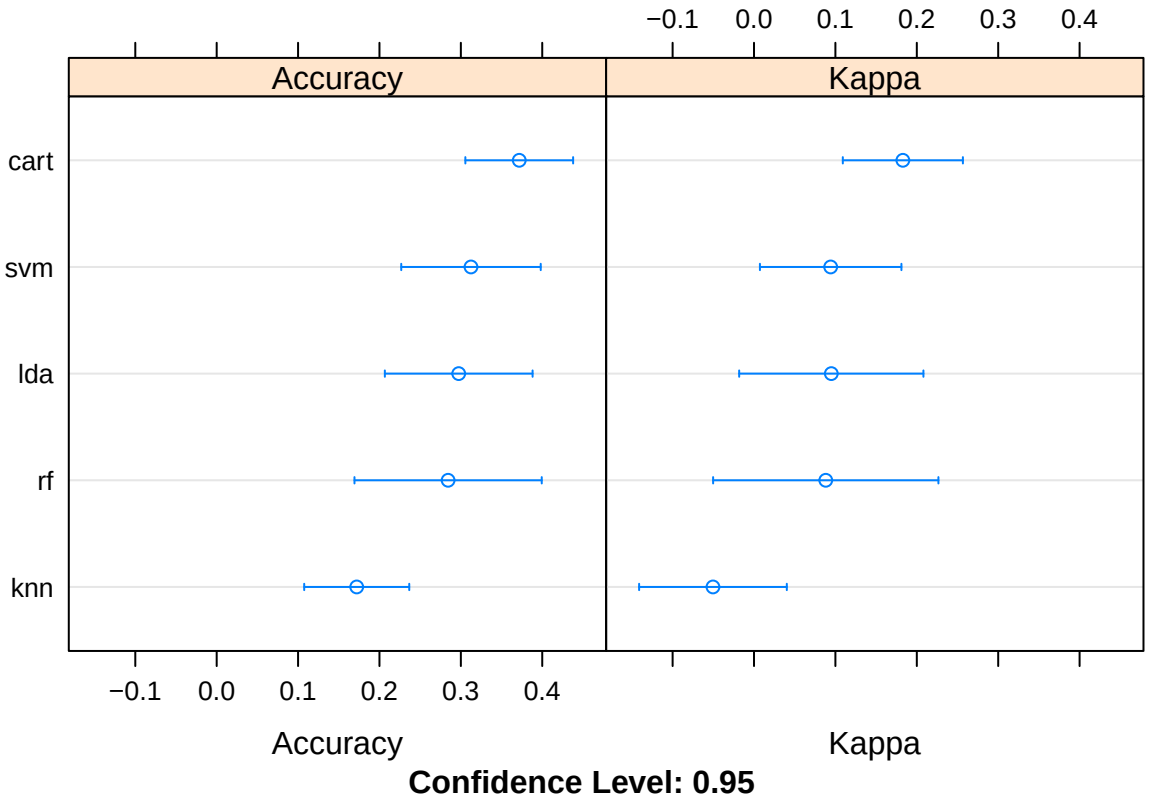



```
# Run algorithms using 10-fold cross validation
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"

# a) linear algorithms
set.seed(100)
fit.lda <- train(Al~., data=dataset, method="lda", metric=metric, trControl=control)
# b) nonlinear algorithms
# CART
fit.cart <- train(Al~., data=dataset, method="rpart", metric=metric, trControl=control)
# kNN
fit.knn <- train(Al~., data=dataset, method="knn", metric=metric, trControl=control)
# c) advanced algorithms
# SVM
fit.svm <- train(Al~., data=dataset, method="svmRadial", metric=metric, trControl=control)
# Random Forest
fit.rf <- train(Al~., data=dataset, method="rf", metric=metric, trControl=control)

# summarize accuracy of models
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
```

```
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda  0.1250000 0.2055556 0.3125000 0.2973810 0.4151786 0.4444444    0
## cart 0.2222222 0.3333333 0.3750000 0.3717460 0.4214286 0.5000000    0
## knn  0.0000000 0.1250000 0.1547619 0.1720635 0.2500000 0.3000000    0
## svm  0.2000000 0.2291667 0.2500000 0.3124603 0.4047619 0.5000000    0
## rf   0.0000000 0.1562500 0.3333333 0.2843254 0.4047619 0.5000000    0
##
## Kappa
##               Min.       1st Qu.      Median        Mean    3rd Qu.
## lda  -1.428571e-01  0.003906250  0.09191176  0.09496400 0.23396226
## cart  0.000000e+00  0.148809524  0.17424242  0.18286612 0.25986842
## knn  -2.500000e-01 -0.140926641 -0.07236467 -0.05044797 0.05411765
## svm  -3.469447e-17  0.005102041  0.03388889  0.09418836 0.17260062
## rf   -2.307692e-01 -0.080000000  0.13461538  0.08816809 0.22788698
##           Max. NA's
## lda  0.2857143    0
## cart 0.3076923    0
## knn  0.1463415    0
## svm  0.3043478    0
## rf   0.3600000    0
```

```
# compare accuracy of models
dotplot(results)
```



```r
# summarize Best Model
print(fit.rf)
```

```
## Random Forest
##
## 82 samples
##  4 predictor
##  5 classes: 'ANN', 'DT', 'GLM', 'Knn', 'SVM'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 73, 75, 73, 73, 74, 73, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.2686508  0.07135151
##   3     0.2732143  0.08112002
##   4     0.2843254  0.08816809
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```r
# estimate skill of LDA on the validation dataset
predictions <- predict(fit.rf, validation)
confusionMatrix(predictions, validation$Al)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ANN DT GLM Knn SVM
##        ANN   2  1   2   1   0
##        DT    1  0   1   0   0
##        GLM   0  1   0   1   3
##        Knn   2  1   0   1   0
##        SVM   0  0   0   1   0
##
## Overall Statistics
##
##                Accuracy : 0.1667
##                  95% CI : (0.0358, 0.4142)
##     No Information Rate : 0.2778
##     P-Value [Acc > NIR] : 0.9127
##
##                   Kappa : -0.063
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: ANN Class: DT Class: GLM Class: Knn Class: SVM
## Sensitivity              0.4000    0.0000     0.0000    0.25000    0.00000
## Specificity              0.6923    0.8667     0.6667    0.78571    0.93333
## Pos Pred Value           0.3333    0.0000     0.0000    0.25000    0.00000
## Neg Pred Value           0.7500    0.8125     0.7692    0.78571    0.82353
## Prevalence               0.2778    0.1667     0.1667    0.22222    0.16667
## Detection Rate           0.1111    0.0000     0.0000    0.05556    0.00000
```

```
## Detection Prevalence     0.3333    0.1111     0.2778    0.22222    0.05556
## Balanced Accuracy        0.5462    0.4333     0.3333    0.51786    0.46667
```

Part4

Algorithm domain of competence With the previous model build, a new dataset could be handled by computing its characteristics and using the model to predict the suitable algorithm which is the class of the meta-data set.

The more datasets we use to train our model the more accurate the heuristic method will work. So finding more diverse classification datasets and feeding our system is essential.

REFERENCES

Lorena, A. C., Garcia, L. P. F., Lehmann, J., de Souto, M. C. P., and Ho, T. K. (2018). How Complex is your classification problem? A survey on measuring classification complexity. arXiv:1808.03591

Lorena, A. C., Maciel, A. I., de Miranda, P. B. C., Costa, I. G., and Prudêncio, R. B. C. (2018). Data complexity meta-features for regression problems. Machine Learning, 107(1):209-246.

Ana C Lorena, Ivan G Costa, Newton Spolaor and Marcilio C P Souto. (2012). Analysis of complexity indices for classification problems: Cancer gene expression data. Neurocomputing 75, 1, 33–42.

Gleison Morais and Ronaldo C Prati. (2013). Complex Network Measures for Data Set Characterization. In 2nd Brazilian Conference on Intelligent Systems (BRACIS). 12–18.

Luis P F Garcia, Andre C P L F de Carvalho and Ana C Lorena. (2015). Effect of label noise in the complexity of classification problems. Neurocomputing 160, 108–119.

Ajay K Tanwani and Muddassar Farooq. (2010). Classification potential vs. classification accuracy: a comprehensive study of evolutionary algorithms with biomedical datasets. Learning Classifier Systems 6471, 127–144.

Tin K Ho and Mitra Basu. (2002). Complexity measures of supervised classification problems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24, 3, 289–300.

Albert Orriols-Puig, Nuria Macia and Tin K Ho. (2010). Documentation for the data complexity library in C++. Technical Report. La Salle - Universitat Ramon Llull.

Enrique Leyva, Antonio Gonzalez and Raul Perez. (2014). A Set of Complexity Measures Designed for Applying Meta-Learning to Instance Selection. IEEE Transactions on Knowledge and Data Engineering 27, 2, 354–367.

R. Leite, P. Brazdil, and J. Vanschoren, "Selecting classification algorithms with active testing," in MLDM, ser. Lecture Notes in Computer Science, P. Perner, Ed., vol. 7376. Springer, 2012, pp. 117–131.

P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, Metalearning: applications to data mining. Springer, 2009.

T. K. Ho, M. Basu, and M. H. C. Law, Data Complexity in Pattern Recognition. Springer, 2005, ch. Measures of Geometrical Complexity in Classification Problems.

J. M. Sotoca, R. A. Mollineda, and J. S. Sanchez "A meta-learning framework for pattern classification by means of data complexity measures," Inteligencia Artificial, vol. 10, no. 29, pp. 31–38, 2006.

T. K. Ho and H. S. Baird, "Pattern classification with compact distribution maps," Computer Vision and Image Understanding, vol. 70, no. 1, pp. 101 – 110, 1998.

F. Smith, "Pattern classifier design by linear programming," Computers, IEEE Transactions on, vol. C-17, no. 4, pp. 367–372, 1968.

A. Hoekstra and R. Duin, "On the nonlinearity of pattern classifiers," in Proceedings of the 13th International Conference on Pattern Recognition, vol. 4, 1996, pp. 271–275.

L. Frank and E. Hubert, "Pretopological approach for supervised learn- ing," in Proceedings of the 13th International Conference on Pattern Recognition, vol. 4, 1996, pp. 256–260.

E. Mansilla and T. K. Ho, "On classifier domains of competence," in Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, vol. 1, 2004, pp. 136–139 Vol.1.

S.-W. Kim and B. J. Oommen, "On using prototype reduction schemes to enhance the computation of volume-based inter-class overlap measures," Pattern Recognition, vol. 42, no. 11, pp. 2695 – 2704, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320309001642

X. Zhu, "Semi-supervised learning with graphs,"Ph.D. thesis, Carnegie Mellon University, 2005, http: //pages.cs.wisc.edu/ jerryzhu/pub/thesis.pdf.

N. Ganguly, A. Deutsch, and A. Mukherjee, Eds., Dynamics On and Of Complex Networks Applications to Biology, Computer Science, and the Social Sciences. Springer, 2009.

E. D. Kolaczyk, Statistical Analysis of Network Data: Methods and Model. Springer, 2009.

A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," SIGKDD Explorations, vol. 11, no. 1, pp. 10–18, 2009.

R. C. Prati, "Combining feature ranking algorithms through rank aggre- gation," in IJCNN. IEEE, 2012, pp. 1–8.

D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," J. Artif. Intell. Res. (JAIR), vol. 6, pp. 1–34, 1997.

G. E. A. P. A. Batista and D. F. Silva, "How k-nearest neighbor parameters affect its performance," in Argentine Symposium on Artificial Intelligence, 2009, pp. 1–12.

C. Soares, "Uci++: Improved support for algorithm selection using datasetoids," in PAKDD, ser. Lecture Notes in Computer Science, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T. B. Ho, Eds., vol. 5476. Springer, 2009, pp. 499–506.

A. Ben-David, "Comparison of classification accuracy using Cohen's Weighted Kappa," Expert Syst. Appl., vol. 34, no. 2, pp. 825–832, 2008.