UNIVERSITY OF LJUBLJANA
FACULTY OF COMPUTER AND INFORMATION SCIENCE

Marija Radović

Reliability Estimation in Regression Supported with Meta-learning and Principal Component Analysis

DIPLOMA THESIS
FOR THE INTERDISCIPLINARY UNIVERSITY PROGRAM

Mentor: doc. dr. Zoran Bosnić

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00022/2010
Datum: 06. 09. 2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdajata naslednjo nalogo:

Kandidatka: **MARIJA RADOVIĆ**

Naslov: **OCENJEVANJE ZANESLJIVOSTI REGRESIJSKIH NAPOVEDI, PODPRTO Z METAUČENJEM IN METODO POGLAVITNIH KOMPONENT**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Za ocenjevanje zanesljivosti regresijskih napovedi obstaja v strojnem učenju več ocen zanesljivosti. Pri njihovi uporabi se srečujemo s težavo, da so različne ocene zanesljivosti različno uspešne v različnih problemskih domenah in z različnimi regresijskimi modeli. Za te namene je bila predlagana uporaba metaklasifikatorja, ki pomaga pri izbiri najbolj ustrezne ocene zanesljivosti za dani regresijski model in problemsko domeno.

Kandidatka naj v svojem diplomskem delu analizira obstoječi metaklasifikator in naj predlaga njegovo nadgradnjo (dodatne atribute). Testira in ovrednoti naj delovanje tudi množice drugih klasifikacijskih modelov za te potrebe. V diplomskem delu naj kandidatka zasnuje tudi novo oceno zanesljivosti, ki naj bo izpeljana iz obstoječih z uporabo metode poglavitnih komponent. Ocene, izbrane z novim metamodelom, in nove ocene, izpeljane z metodo poglavitnih komponent, naj kandidatka testira po vzoru testiranja dosedanjih ocen ter naj ovrednoti njihove uspešnosti.

Mentor:

Doc. dr. Zoran Bosnić

Dekan Fakultete za računalništvo in informatiko:

Prof. dr. Franc Solina

Dekan Fakultete za matematiko in fiziko:

Prof. dr. Andrej Likar

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marija Radović

Ocenjevanje zanesljivosti regresijskih napovedi, podprto z metaučenjem in metodo poglavitnih komponent

DIPLOMSKO DELO
NA INTERDISCIPLINARNEM UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Zoran Bosnić

Ljubljana, 2010

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisana Marija Radović
z vpisno številko 63080377,

sem avtorica diplomskega dela z naslovom:
Ocenjevanje zanesljivosti regresijskih napovedi, podprto z metaučenjem in metodo poglavitnih komponent (angl. Reliability Estimation in Regression Supported with Meta-learning and Principal Component Analysis)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Zorana Bosnića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 6.9.2010

Podpis avtorice:

Rada bi se zahvalila svojemu mentorju, Zoranu Bosniću, za vse uporabne nasvete in za pomoč, ki mi jo je nudil pri pisanju tega diplomskega dela.
Prav tako bi se zahvalila mojim staršem, bratu in mojem Alešu, za podporo in razumevanje na vsakemu koraku.
Pripomogla sta tudi Stripi in Hanzi, ki sta me z mahanjem repkov vsak dan razveselila in dala motivacijo za naprej.

Božidaru, Zori, Predragu,
Alešu, Ivi in Dušanu

Index

Povzetek

Za ocenjevanje točnosti napovednih modelov se v strojnem učenju običajno uporabljata povprečna kvadratna napaka (MSE) in relativna povprečna kvadratna napaka (RMSE). Čeprav te ocene ocenjujejo točnost modela s seštevanjem doprinosov napak testnih primerov, pa ne nudijo nobene informacije o pričakovani napaki posameznih za posamezne neznane primere.
Ocene zanesljivosti posameznih napovedi lahko nudijo pomembno odločitveno informacijo o napakah posameznih napovedi. Pomembna potencialna področja uporabe takšnih ocen zanesljivosti posameznih napovedi so v tveganih aplikacijah strojnega učenja (npr. medicina, finance, nadzorne aplikacije itn.). Tam se lahko ocene zanesljivosti uporabijo za odločitev, ali bodo uporabniki teh sistemov sprejeli napoved sistema in izvedli ustrezne ukrepe v resničnem svetu ali ne (npr. ali bo zdravnik predpisal zdravila, ali bo direktor sprejel poslovne odločitve, kapitan zapovedal spremembe smeri plovbe in tako naprej).
Ocene zanesljivosti omogočajo uporabnikom tudi, da razlikujejo med boljšimi in slabšimi napovedmi, kar je zelo pomembno, če je odločitev tvegana ali so ogrožena človeška življenja. Na primer, zdravniki ne potrebujejo samo napovedi, ampak prav tako potrebujejo zanesljivost te napovedi. Če serviser pralnih strojev naredi napako pri odločitvi in ne popravi stroja pravilno, bo lastnik napako odkril v kratkem in zahteval, da serviser pride ponovno; če pa zdravnik narobe določi diagnozo in bolnik umre, drugega poskusa ni. V tem primeru je zelo pomembno za zdravnika, da ve, kakšna je zanesljivost napovedi avtomatskega sistema, ki podpira njegovo odločitev. Če je napoved nezanesljiva, ne bo tvegal življenja.

Ocene zanesljivosti, obravnavane v tej diplomski nalogi, temeljijo na različnih pristopih: analiza občutljivosti, varianca modela bagging, lokalno prečno preverjanje, zanesljivost kot gostota učnih primerov in ocena lokalne napake. V izvirnih člankih so bile te ocene ocenjevane z analizo njihove korelacije z absolutno vrednostjo napake posameznih napovedi, pri čemer so bile ocene zasnovane tako, da je pričakovana njihova pozitivna korelacija z napako. Dve od devetih obravnavanih ocen sta predznačeni, kar pomeni, da hranita tudi informacijo o smeri napake (in s tem potrebnega popravka). Posledično sta bili ti dve oceni evalvirani tudi z analizo korelacije z neabsolutno vrednostjo napake napovedi. Oznake teh devetih uporabljenih ocen so: SAvar, SAbias-s (neabsolutna), SAbias-a (absolutna), BAGV, LCV, DENS, CNK-s (neabsolutna), CNK-a (absolutna) in BVCK.

V praksi se dogaja, da ocene zanesljivosti na različnih domenah in modelih dosegajo različne rezultate uspešnosti (višine korelacije z dejansko napako primera). To je razlog, zakaj ne vemo, katera ocena bo najbolje delovala z določenim parom domena/model. V diplomi obravnavamo enega od dosedanjih (Bosnić in Kononenko) pristopov za avtomatsko izbiro najbolj ustrezne ocene, t.j. metaučenje (kjer so odločitvena drevesa uporabljena kot metaklasifikator). S primerjavo rezultatov ugotovimo, da je alternativni pristop z notranjim prečnim preverjanjem sicer bolj uspešen kot prvi, vendar pa je metaučenje časovno manj zahtevno, v praksi torej bolj uporabno, in je doseglo boljše rezultate kot vse posamezne ocene zanesljivosti. Zaradi tega smo se odločili dodatno raziskati ta pristop.
Glavni namen metaučenja je razumevanje interakcij med mehanizmom učenja in konkretnih kontekstov, v katerih se ta mehanizem uporablja. S praktičnega vidika ima metaučenje več ciljev. Po eni strani želimo premagati nekatere izzive, s katerimi se soočajo uporabniki pri uporabi sedanjih orodjih za analizo podatkov. Po drugi strani pa želimo obravnavati probleme, ki so pogosto opaženi pri praktični uporabi orodij za analizo podatkov, in sicer,

kako nekaj pridobiti iz ponavljajoče uporabe modela uporabljenih na podobnimi nalogami. Zadnji cilj je pripraviti učinkovite modele s predpostavko, da je izbira predsodkov izboljšana, če se ravna po izkušnjah pridobljenih iz preteklih poskusov. Metaučenje zahteva metapodatke za učenje, in sicer podatke o baznih učnih problemih in nalogah. Učni metapodatki vsebujejo informacije o uspešnosti skupine algoritmov na skupini domen. Te informacije so predstavljene v obliki metalastnosti (atributov), ki so pridobljeni z uporabo karakterizacije podatkov. Obstajajo trije različni pristopi za karakterizacijo podatkov, in sicer "enostavne, statistične in informacijsko teoretične mere", "mere zasnovane na modelih" in "markerji".

V procesu ustvarjanja metapodatkov smo se odločili za metalastnosti, za katere smo menili, da so najbolj primerni za naš problem. Sedem jih je bilo že uporabljenih za problem metaučenja: *regresijski model, število učnih primerov v domeni, število atributov v domeni, povprečna relativna kvadratna napaka, povprečna gostota, povprečna oddaljenost od pet najbližjih sosedov* in *povprečna razdalja med napovedjo za določen primer in napovedjo za pet najbližjih sosedov*. Dodali smo tri nove atribute, da vidimo, če bodo izboljšali natančnost metaklasifikatorjev: *razmerje med številom atributov in število učnih primerov, povprečno korelacijo atributov* in *povprečno entropijo atributov*. Ocenili smo vse te atribute z uporabo štirih mer: informacijski prispevek, gini-indeks, minimalna dolžina opisa (MDL) in ReliefF.
Atribut *relativna povprečna kvadratna napaka* je bil uvrščena v najboljše tri s strani vseh štiri mer. Atribut *model* je imel dobre rezultate pri merjenju s informacijski prispevkom in gini-indeksom, MDL pa ga je ocenil kot enega izmed najslabših. Atributa *povprečna razdalja do pet najbližjih sosedov* in *povprečna gostota* sta imela nizko kakovost in rang, pri merjenju s informacijskim prispevkom in ocenama MDL in Gini indeks.
Trije novi atributi so v tej raziskavi dosegli povprečne rezultate: *poprečna entropija atributov* je bil v najslabših treh pri merjenju z ocenama ReliefF in Gini indeks. Druga dva atributa pa nista izstopala.

Pri problemu metaučenja smo poleg odločitvenega drevesa kot metaklasifikatorja testirali še 6 drugih klasifikacijskih modelov, to so: naključni gozdovi, naivni Bayes, metoda podpornih vektorjev, nevronske mreže, k-najbližjih sosedov in metoda boosting. Vsak metaklasifikator je namenjen napovedovanju optimalne ocene zanesljivosti za posamezen par domena/regresijski model. Ocene zanesljivosti tako predstavljajo vrednosti razreda. Učna množica je sestavljena iz optimalne izbire ocene za znane pare domen/modelov. Atributi, navedeni zgoraj, opisujejo značilnosti domene in modela za poseben primer.
Vsakemu meta-učnemu primeru je bila dodeljena tista vrednost razreda, ki predstavlja oceno zanesljivosti, ki je dosegla najvišjo pozitivno korelacijo z napako napovedi (ne glede na to, ali je korelacija statistično pomembna ali ne) za določen regresijski model. Testiranje učinkovitosti metaklasifikatorja je opravljeno z uporabo prečnega preverjanja po principu »izpusti-enega«. Višina korelacijskega koeficienta avtomatsko izbrane ocene zanesljivosti je bila ovrednotena z uporabo t-testa.

Za testiranje smo uporabili 8 regresijskih modelov: regresijska drevesa, linearna regresija, nevronske mreže, bagging, metoda podpornih vektorjev, lokalno utežena regresija, naključni gozdovi in posplošeni aditivni model. Uporabili smo tudi 28 standardnih in javnodostopnih domen. Vsaka domena je bila uporabljena kot učna množica. Področja uporabe teh domen varirajo od zdravstvenih, ekoloških in tehničnih do matematičnih in fizikalnih domen. Večina domen je na voljo pri UCI Machine Learning Repository in StatLib DataSets Archive. Ker smo eksperimentirali z 28 domenami in 8 regresijskimi modeli, smo torej oblikovali učno množico za metaučenje, ki sestoji iz 224 (28×8) učnih primerov. Testiranje se je izvajalo v statističnem paketu R, ki je jezik in okolje za analizo podatkov in grafične prikaze. Zagotavlja

široko paleto statističnih (linearno in nelinearno modeliranje, klasične statistične teste, časovne analize, klasifikacijo, itd) in grafičnih tehnik ter je zelo razširljiv.

Metaklasifikator, ki uporablja model naključnega gozda je dosegel najboljše rezultate (najvišji odstotek pozitivnih korelacij med ocenami zanesljivosti in napakami napovedi in brez negativnih). Dobre rezultate je dosegel tudi model boosting, vendar je dosegel tudi 2% negativnih korelacij z napako (nezaželeno). Nevronske mreže so dosegle najslabši rezultat s samo 42% pozitivnih korelacij in 2% negativnih korelacij. Preostalih pet klasifikatorjev je imelo povprečne rezultate med 52% -58% pozitivnih korelacij.
Regresijska drevesa so se izkazala kot daleč najboljši regresijski model v kombinaciji z vsemi metaklasifikatorji. Lokalno uteženi regresijski model je imel najslabši rezultat, s povprečnih le 42% pozitivnih korelacij. Vsi drugi regresijski modeli so imeli podobne rezultate za vse klasifikatorje.

V diplomski nalogi smo testirali tudi uporabo metode poglavitnih komponent (PCA) za izdelavo nove ocene zanesljivosti z uporabo prispevkov obstoječih ocen in preoblikovanjem njih v novo oceno da bi videli, če bo uspešnejša nova ocena kot izvirne ocene. PCA je način določitve vzorcev v podatkih in izražanja podatkov s poudarkom na njihove podobnosti in razlike. PCA predstavlja matematični postopek, ki pretvori število morebitno koreliranih vektorjev (v našem primeru, vektorjev ocen zanesljivosti) v manjše število nekoreliranih vektorjev, imenovanih poglavitne komponente. To je ortogonalna linearna transformacija, ki transformira podatke v nov koordinatni sistem tako, da največja varianca dobljena s pomočjo katere koli projekcije podatkov na koncu leži na prvi koordinati (imenovani prva poglavitna komponenta), druga največja varianca na drugi koordinati in tako naprej. Torej, prva poglavitna komponenta predstavlja največ variabilnosti v podatkih, vsaka naslednja komponenta pa predstavlja največ od preostale variabilnosti.

Nove ocene zanesljivosti smo definirali tako, da imajo vrednost enako prvi poglavitni komponenti, ki smo jo nato testirali in primerjali z drugimi izvornimi ocenami kot novo samostojno oceno.
Glede na to, da je naša nova ocena kombinacija vseh ocen, vključno s SAbias in CNK, ki imata lahko pozitivne in tudi negativne vrednosti, moramo upoštevati, da bo tudi nova ocena lahko zavzemala tako pozitivne kot negativne vrednosti. Zato, smo v diplomi zasnovali dve različici ocen: PCA-a, kjer smo korelirali vrednosti ocene z absolutno napako napovedi (ocene, ki so bile uporabljene so SAvar, SAbias-a, CNK-a, LCV, BAGV, DENS in COMB) in PCA-s, kjer smo korelirali vrednosti ocene z neabsolutno napako napovedi (z ocenami SAbias-p in CNK-p).
Uspešnost novih ocen zanesljivosti smo ocenili na enak način kot je bilo testirano devet izvirnih ocen, torej z uporabo Pearsonovega korelacijskega koeficienta med ocenami zanesljivosti in napakami napovedi. Značilnost korelacije je statistično ovrednotena s t-testom.

Uspešnosti obeh ocen zanesljivosti smo primerjali z rezultati najvišjega uvrščenega metaklasifikatorja, naključnih gozdov, ter devet izvirnih ocen zanesljivosti. Metamodel je še vedno imel najboljše rezultate, boljše tudi od najbolj uspešne ocene zanesljivosti, BVCK. Oceni, pridobljeni z metodo poglavitnih komponent (PCA) pa nista predstavljali bistvenega izboljšanja v primerjavi z drugimi ocenami zanesljivosti. PCA-a, ki je dosegla 37% pozitivnih korelacij, je imela razmeroma povprečne rezultate v primerjavi z drugimi ocenami. Uvrstila se je na osmo mesto od dvanajstih. Imela je podobne rezultate kot ocena CNK-s. PCA-s je bila še šibkejša. Imela je samo 1% pozitivnih korelacij več kot najslabša ocena, SAbias-a, temveč

tudi za 2% več negativnih korelacij. Vse druge ocene in metamodeli so dosegli boljše rezultate.
Ocena PCA-a je dosegla najboljše rezultate z regresijskim modelom metode podpornih vektorjev. Dosegla je enak procent pozitivnih korelacij kot model bagging. Ocena PCA-s je dosegla najboljše rezultate v kombinaciji z linearno regresijo in posplošenim aditivnim modelom, kjer je imela 29% pozitivnih korelacij in 21% negativnih.

Ključne besede: ocene zanesljivosti, zanesljivost, napaka napovedi, metaučenje, metoda poglavitnih komponent, napovedi, strojno učenje

Abstract

Reliability estimates for individual predictions can provide important risk-sensitive information. They enable users to distinguish between better and worse predictions which are very important when dealing with decision-critical prediction problems. Unfortunately, when used on different domains and models, reliability estimates perform differently. That is the reason why we require an approach that can foretell which estimate will work best with a specific domain/model pair.

In our thesis we have experimented with meta-learning approach to automatically select the best estimate for a specific domain and regression model. We used seven different meta-classifiers on eight regression models and with nine reliability estimates. These reliability estimates represented the class values of the meta-classification process. During the creation of the metadata for meta-learning, we have chosen meta-features that we considered to be most appropriate for this problem. The results showed that the best performing meta-model was random forest. Meta-model neural networks gave the worst results.

Additionally, we proposed principal component analysis approach for creation of two new reliability estimates as combinations of existing ones to see if these new estimates would perform. These results were also compared with the results of the existing nine estimates and the best performing meta-classifier. The results showed that the meta-classifier achieved the best results and the estimate BVCK the second best.

Keywords: reliability estimates, reliability, prediction error, meta-learning, principal component analysis, predictions, machine learning

Chapter 1: Introduction

Reliability estimates for individual predictions provide important information about individual prediction error. This information cannot be obtained by just calculating the average accuracy of a predictive model. There are applications of machine learning (for example in medicine), where reliability estimates may represent decisive information. They enable the users to distinguish between better and worse predictions which are very important if decision risk is involved or human lives are at stake. Physicians can use the reliability estimates to decide whether they will accept the system's prediction and perform a corresponding action in the real world (e.g. prescribe a medicine) or not.
This thesis is motivated by the previous work in this field [2], where research was done using the meta-learning approach with decision trees as meta-classifier to predict the most appropriate reliability estimate. In our thesis we expand this approach by testing 6 additional meta-classifiers to see if any of them could improve the results achieved with the meta-decision tree. These classifiers are: random forest, naive Bayes, support vector machines, neural networks, k-nearest neighbors and boosting.
We use these meta-classifiers to predict the optimal reliability estimate for a given problem domain/regression model pair. The class values are represented by the reliability estimates. In the process of creating metadata for meta-learning, we have chosen meta-features (attributes) that we considered to be the most appropriate for this problem. We also evaluated these attributes using four measures: information gain, gini-index, MDL and ReliefF.
We have also proposed a new estimate based on the Principal Component Analysis (PCA) of the existing reliability estimates. Using PCA we transformed a set of reliability estimates into a linear combination of them and evaluated the success of the achieved values as new independent reliability estimates.

The thesis is structured as follows. In Chapter 2 we describe the importance of reliability estimation of individual predictions. That Chapter is divided into two parts: first, in Section 2.1 we present nine reliability estimates that will be used in thesis, then, in Section 2.2, we briefly describe two approaches for automatic selection of reliability estimates. In Chapter 3 is the theoretical part of the thesis. First, in Section 3.1, we give a brief overview of the classification methods and in Section 3.2 of the regression methods used in our research. Then in Section 3.3, you can read about statistical comparison of accuracies of the models. In

Section 3.4 we presented four measures that we used for attribute evaluation and in Section 3.5 we explained the main principles of meta-learning. In Chapter 4, we presented ways we used to try to improve the meta-models: additional attributes (Section 4.1) and additional meta-classifiers (Section 4.2). In Chapter 5, we showed how we used Principal Component Analysis for reliability estimation. Chapter 6 consists of our experimental results of: the attribute evaluation (Section 6.1), meta-learning (Section 6.2) and reliability estimation using PCA (Section 6.3). Chapter 7 concludes this thesis.

Chapter 2: Reliability estimation of individual predictions

In experimental sciences, reliability is the extent to which the measurements of a test remain consistent over repeated tests of the same subject under identical conditions. An experiment is reliable if it yields consistent results of the same measure.

In machine learning, examples of these measures are the mean squared error (MSE) and the relative mean squared error (RMSE). They are most commonly used for the evaluation of prediction accuracies, like it is shown in Figure 1(a). Although these estimates evaluate model performance by summarizing the error contributions of all test examples, they provide no information about the expected error of individual prediction for a given **unseen example**, like it is shown in Figure 1(b).

The availability of additional information about prediction reliability can enable users of the decision-making applications to decide to what degree they can trust the prediction. We use reliability estimates to provide that information (Figure 1(c)). Important potential application areas of the individual prediction reliability estimates are the risk-sensitive applications of machine learning (e.g. medicine, financial, control applications, and so on). They can use the reliability estimates to decide whether they will accept the system's prediction and perform a corresponding action in the real world (e.g. prescribe a medicine, make a business decision, change navigation direction, and so on) or not. For example, physicians do not just need the prediction; they also need the reliability of that prediction. If a washing machine repairman makes an error in judgment and does not fix the machine properly, the owner will discover this shortly and demand he come again. But if a doctor makes the wrong diagnosis and the patient dies, there is no second try. In this case, it is very important for a doctor to know what the reliability of his prediction is. If it is unreliable, he will not risk someone's life.

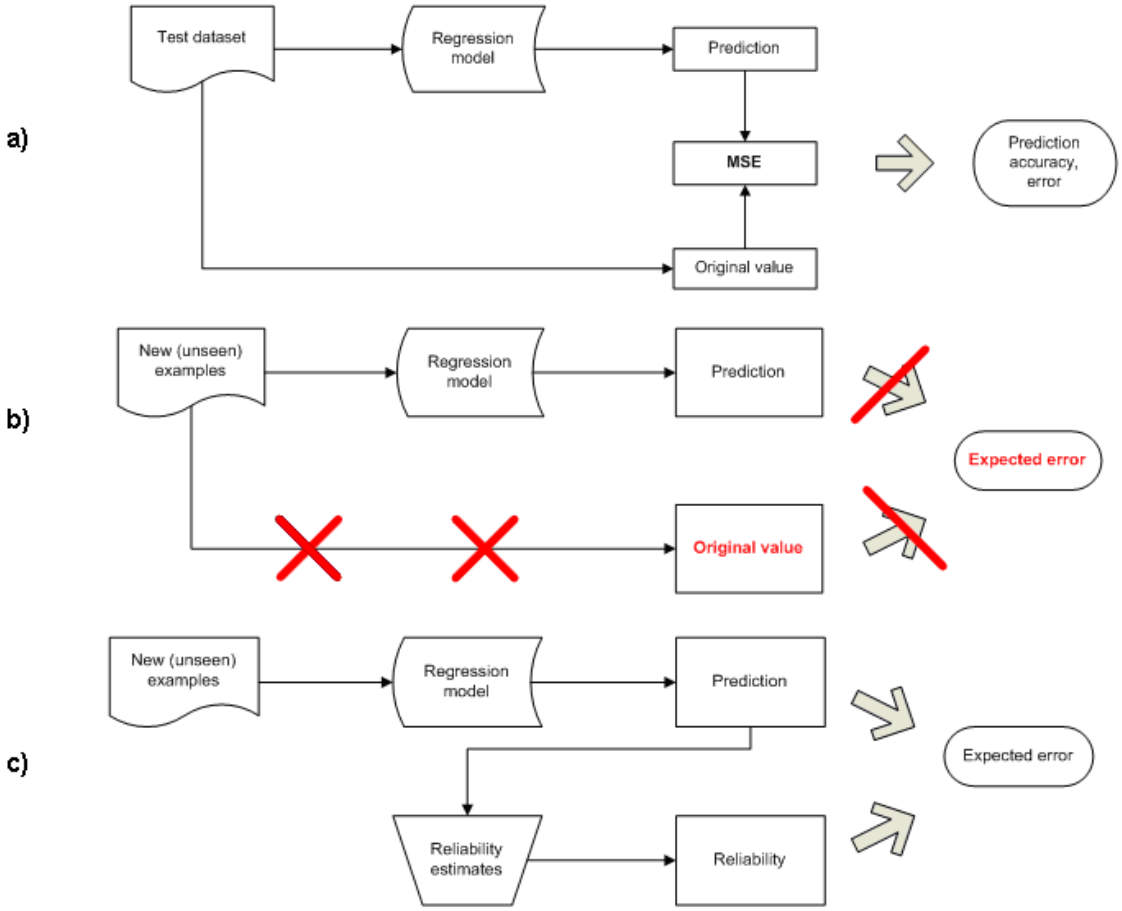


Figure 1. Reliability evaluation scenarios: a) obtaining the prediction accuracy using MSE; b) for the new (unseen) example we can not know the expected error; c) estimating the expected error using reliability estimates

In this chapter, we will first present nine reliability estimates [3], that we have used for our research in Section 2.1. Then, in Section 2.2, we will describe two approaches for automatic selection of reliability estimates [2]: meta-learning and internal cross-validation approach.

2.1 Reliability estimates

Previous work of Bosnić and Kononenko [2,3] presented a set of reliability estimates for individual predictions. In this section, we briefly present these estimates and their results on ranking by the average percentage of significant positive and negative correlations with the prediction error using different regression models.

All the estimates are expected to correlate positively with the prediction error. This means that all the estimates are founded so that higher absolute values represent less reliable predictions and lower absolute values represent more reliable predictions. In the previous work, the success of the estimates was therefore evaluated by correlating the magnitudes of estimates to the absolute prediction error of test examples. However, since estimates named SAbias and CNK can take also negative values, these two estimates were also tested by correlating their

signed values to the signed prediction error of test examples. In this way, nine estimates were proposed and tested with which we work in our thesis: **SAvar, SAbias-s** (signed)**, SAbias-a** (absolute)**, BAGV, LCV, DENS, CNK-s** (signed), **CNK-a** (absolute) and **BVCK**. These estimates are described in the following.

2.1.1 SAvar

Reliability estimate SAvar (Sensitivity Analysis – variance) [3,4] is based on the sensitivity analysis approach. Sensitivity analysis aims to determine how much the output of a system is affected by variations in input. It estimates the local variance for a given unlabeled example. We calculate SAvar using the following equation:

$$SAvar = \frac{\sum_{\varepsilon \in E} K_{\varepsilon} - K_{-\varepsilon}}{|E|}$$

where K is the prediction of the initial regression model (the initial prediction), K_{ε} and $K_{-\varepsilon}$ denote the sensitivity predictions, and E denotes a set of used sensitivity parameters ε. This parameter ε influences the label value of additional learning example with which we expand the original learning set in order to facilitate sensitivity analysis procedure. By doing so, ε defines the magnitude of the induced change in the initial learning set. To make the observation windows in local problem space wider and to make the measures robust to local anomalies, SAvar uses predictions from sensitivity models gained and averaged using different values of the parameter $\varepsilon \in E$. For our experiments, $E = \{0.01, 0.1, 0.5, 1.0, 2.0\}$.

Ranked by the average percentage of significant positive and negative correlations with the prediction error, SAvar gave average results. It worked best with generalized additive models and linear regression and worst with random forests.

2.1.2 SAbias

SAbias (Sensitivity Analysis – bias) [3,4] is another estimate, used in this thesis, which works by the principle of the sensitivity analysis. It estimates the local bias for a given example. SAbias is calculated in a similar way as SAvar:

$$SAbias = \frac{\sum_{\varepsilon \in E} (K_{\varepsilon} - K) + (K_{-\varepsilon} - K)}{2|E|}$$

Both SAbias-a and SAbias-s gave the worst averaged results across all used regression models. However, SAbias-s did work incredibly well with regression trees.

2.1.3 BAGV

Given a bagged aggregate of m predictive models, where each of the models yields a prediction K_i, $i = 1, ..., m$, the label of an example is predicted by averaging the individual predictions:

$$K = \frac{\sum_{i=1}^{m} K_i}{m}$$

and reliability estimate BAGV is defined as the prediction variance:

$$BAGV = \frac{\sum_{i=1}^{m} (K_i - K)^2}{m}$$

This estimate gave good results in terms of correlation to the prediction error. Considering the average across all the regression models, the only estimate with better results than BAGV was BVCK, which is actually a combination of BAGV and another estimate with good results, CNK-a. BAGV worked best with regression trees where it achieved positive correlations to the prediction error in 64% of experiments, and worst with the support vector machines and the locally weighted regression.

2.1.4 LCV

LCV (Local cross-validation) reliability estimate is computed using the local leave-one-out procedure on the *k* local models generated from the subspace defined by *k* nearest neighbors. First, the absolute local leave-one-out prediction errors $E_i = |C_i - K_i|$ are computed. The LCV estimate is then computed as the average of the nearest neighbors' local errors E_i, weighted by the neighbors' distance.

LCV did not stand out as best or worst compared to other reliability estimates. It worked the best with support vector machines and random forest and the worst with linear regression and generalized additive models.

2.1.5 DENS

DENS (Density-based) reliability estimate assumes that error is lower for predictions which are made in denser problem subspaces (a portion of the input space with a more learning examples), and higher for predictions which are made in sparser subspaces (a portion of the input space with fewer learning examples). This means that we trust the prediction with respect to the quantity of information that is available for its computation in the local region of the problem space.

Given the learning set $L = ((x_1,y_1),\ldots, (x_n,y_n))$, the density for unlabeled example $(x,\ldots)$ is defined as:

$$p(x) = \frac{\sum_{i=1}^{n} k(D(x, x_i))}{n}$$

where *D* denotes a distance function, *k* denotes a kernel function (in our case the Gaussian), *x* denotes an example for which we are estimating reliability and x_i, $i = 1, \ldots, n$ denotes the learning examples. The reliability estimate is defined as:

$$DENS(x) = \max_{i=1,\ldots,n}(p(x_i)) - p(x)$$

where $\max_{i=1,\ldots,n}(p(x_i))$ denotes the maximum value of estimated density over all learning examples in the learning set *L*.
DENS achieved fairly poor results, which were ranked 7th out of nine estimates. It worked the best with locally weighted regression and the worst with neural networks.

2.1.6 CNK

CNK ($C_{N\text{eighbors}}$ - *K*) works by the principle of local modeling of prediction error. Given a set of nearest neighbors *N*, $N = ((x_1,C_1),\ldots, (x_k,C_k))$, CNK is for an unlabeled example defined as the difference between the average label of the nearest neighbors and the example's prediction *K*:

$$CNK = \frac{\sum_{i=1}^{k} C_i}{k} - K$$

where *k* denotes the number of neighbors, C_i denotes neighbors' labels and *K* denotes the example's prediction.
As we explained in the beginning of this section, we used two versions of CNK estimates: CNK-a (absolute) and CNK-s (signed) that were tested for correlation with absolute and signed prediction error, respectively. CNK-a performed incredible well, being ranked as 3rd after BAGV and the BVCK. CNK-s worked very well with regression trees, having 86% of positive correlations which was the best model/estimate combination, but in combination with other models it achieved poor results.

2.1.7 BVCK

BVCK (Bagging Variance – ($C_{N\text{eighbors}} - K$)) is a linear combination of estimates BAGV and CNK-a. It is defined as follows:

$$BVCK(x) = \frac{BAGV(x) + CNK(x)}{2}$$

where *x* denotes the example for which we are estimating reliability and BAGV(*x*) and CNK(*x*) denote the corresponding estimates' values for that particular example.
This estimate consists of the two best ranked estimates, making it expectable that it also achieves good results.

2.2 Automatic selection of reliability estimates

In the following we present two approaches for automatic selection of the best performing reliability estimate for a given problem domain and regression model: meta-learning and internal cross-validation approach. These two approaches were proposed and tested by Bosnić and Kononenko in [2].
Reliability estimates, described above, perform differently when used on different domains and regression models. This is why we wish to create a mechanism which would provide us

with the information which reliability estimate would be the best to use on a specific domain/regression model pair.

2.2.1 Meta-Learning Approach

One way to relate the performance of the algorithms to the characteristics of the datasets is using the meta-level learning. In the research by Bosnić and Kononenko, a meta-classifier was proposed. This classifier was a decision tree, which was intended to predict the best reliability estimate for a given problem domain/regression model pair. The reliability estimates therefore represented class values to be predicted, the learning set consisted of the optimal selections for known domain/model pairs, and the attributes described the domain/model characteristics that are related to a particular example. To each meta-learning example they assigned a class value, representing a reliability estimate that achieved maximum positive correlation with the prediction error (irrespective of whether the correlation was statistically significant or not) for a given regression model. The correlation coefficients between estimate values and prediction errors were computed using a leave-one-out scenario on each particular data set.

The testing results of the automatic selection of the best performing estimate using the meta-learning approach are shown in Table 1. From comparison with the most successful individual reliability estimate BVCK we can see, that meta-classifier achieved better average results, and better individual results as well with all except one regression model (neural network).
The whole meta-learning process in more detail is described in Section 3.5.

2.2.2 Internal Cross-Validation

Similarly to meta-learning, if we lack relevant problem-specific knowledge, cross-validation methods may be used to empirically select a learner. If faced with a number of possible learning strategies but do not have any prior knowledge about the data, a natural idea is to allow the data itself to indicate which method will work best. Like in the standard cross-validation approach, the internal cross-validation approach divides the learning examples into *n* equally sized subsets. Each subset is used for performance evaluation (correlation to the prediction error) of all testing reliability estimates. Based on acquired *n* correlation coefficients for each reliability estimate, the final (most appropriate) reliability estimate is selected as the one with the highest average correlation. This estimate is then used to estimate the reliability of all testing examples in that particular model and domain.

As we can see in Table 1, which summarizes the achieved results of both approaches for automatic selection of reliability estimates, the internal cross-validation achieved better performance than the best reliability estimate BVCK. We can see that on average, with the internal cross-validation the 73% of estimates significantly positively correlated with the prediction error, while 0% of estimates significantly negatively correlated with the prediction error.

Model	BVCK	Meta-learning	Internal cross-validation
RT	71/4	79/0	87/0
LR	50/0	54/0	73/0
NN	54/0	46/0	73/0
BAG	61/0	61/0	67/0
SVM	46/0	54/0	67/0
LWR	43/0	43/4	73/0
RF	50/0	64/0	60/0
GAM	54/0	54/0	80/0
Average	**54/1**	**57/1**	**73/0**

Table 1. The performance comparison of the most successful individual estimate BVCK, the meta-predicted optimal estimate and the estimate, selected with the internal cross-validation

Comparing the results of the meta-learning approach and internal cross-validation approach, we can see that the second approach outperforms the meta-learning approach. However, considering that the meta-learning approach is less time-demanding and that it had achieved better results than the most successful individual estimate BVCK, this approach still shows the potential for predicting the optimal reliability estimate for a given domain/model pair. That is why we decided to explore this potential in this thesis. Instead of just using decision trees as meta-classifiers, we added 6 more classification models to see if they will improve the results of the meta-model.

Chapter 3: Applied machine learning methods

Machine Learning (ML) [1,12,18,19] is the study of computer algorithms that improve their performance automatically through experience. Applications of ML range from data mining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests.
When faced with the decision, for example, when trying to establish relationships between multiple features, people are prone to making mistakes. This makes it difficult for them to find solutions to certain problems. Machine learning can often be successfully applied to these problems, improving the efficiency of systems and the designs of machines.
Every instance in any dataset used by machine learning algorithms is represented using the same set of features (attributes). The features may be continuous, categorical or binary. If instances are given with known labels (the corresponding correct outputs) then the learning is called *supervised*, in contrast to *unsupervised* learning, where instances are unlabeled. By applying these unsupervised (clustering) algorithms, researchers hope to discover unknown, but useful, classes of items.
In our work we use the supervised learning algorithms for building a meta-learner. Supervised learning (SL) can be formalized as the problem of inferring a function $y=f(x)$, based on a training set $D = \{(x_1,y_1), \ldots,(x_n,y_n)\}$. When the output, y, is continuous, we are in the context of *regression*, whereas in *classification* problems, y is of categorical nature.
The obtained function $f(x)$ is evaluated by how well it generalizes, i.e., how accurately it performs on new data assumed to follow the same distribution as the training data. To achieve good generalization, it is necessary to control the complexity of the learned function. If it is too complex, it may follow irrelevant properties of the particular data set on which it is trained (what is usually called *overfitting*). But an overly simple function may not be rich enough to capture the true underlying relationship (*underfitting*).

We describe various algorithms in the following. In Section 3.1 we describe classification algorithms and in Section 3.2 regression algorithms that we used for our research.

3.1 Classification

Classification is a supervised machine learning procedure in which individual items are placed (classified) into groups based on one or more characteristics (attributes) inherent in the items and on a training set of previously classified items. We use the items in a training set and their characteristics to classify new items using various techniques. There are many classification methods in use today; here, we will present only those that were used for research in this thesis.

3.1.1 Decision trees

Decision trees [8,18], like all classification methods; classify instances based on feature values. Each node in a decision tree represents a feature and each branch represents a value that the feature can assume. Instances are classified starting at the root node and sorted down the tree based on their feature values. Figure 2 is an example of a decision tree.
Using the decision tree depicted in Figure 2 as an example, the instance [at1 = a1, at2 = b2, at3 = a3, at4 = b4] would sort to the nodes: at1, at2, and finally at3, which would classify the instance as being positive (represented by the value “Yes”).
The problem of constructing an optimal binary decision trees is an NP-complete problem and thus theoreticians have searched for efficient heuristics for constructing near-optimal decision trees.

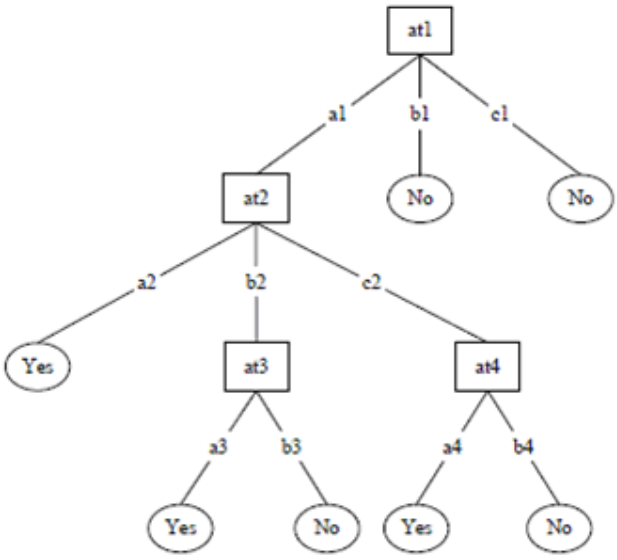


Figure 2. An example of the decision tree

When the tree is constructed, the feature that best divides the training data will be the root node of the tree. There are many methods for finding the feature that best divides the training data. To find such a feature, we use measures to evaluate the features, such as *information gain* and *gini index*. Most of the measures estimate each attribute independently; ReliefF algorithm estimates them in the context of other attributes. These three measures will be

presented in Section 4.1, along with MDL measure. Comparison of individual methods may still be important when deciding which metric should be used in a particular dataset. The same procedure is then repeated on each partition of the divided data, creating sub-trees until the training data is divided into subsets of the same class.

There are two common approaches that decision tree induction algorithms can use to avoid overfitting training data:

- stop the training algorithm before it reaches a point at which it perfectly fits the training data,
- prune the induced decision tree. If the two trees employ the same kind of tests and have the same prediction accuracy, the one with fewer leaves is usually preferred.

One of the most useful characteristics of decision trees is their comprehensibility. People can easily understand why a decision tree classifies an instance to a specific class. Since a decision tree constitutes a hierarchy of tests, an unknown feature value during classification is usually dealt with by passing the example down all branches of the node where the unknown feature value was detected, and each branch outputs a class distribution. Decision trees usually perform better when dealing with discrete/categorical features.

3.1.2 Random forest

Random forests [7,9] are a combination of tree structured classifiers where each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.
During the building of each tree, only a random subset of all attributes is chosen for candidate splits in each iteration. Considering that obtained trees are very specific for a part of the problem space, we calculate the final result by taking the average of the predictions of all the trees in the forest. The result calculated that way stabilizes the bias and the variance.
The generalization error for forests converges to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them.
Random forests do not overfit. Using the right kind of randomness makes them accurate classifiers and regressors. Forests give results competitive with boosting and adaptive bagging, yet do not progressively change the training set.

3.1.3 Naive Bayes

From the probabilistic perspective, according to Bayes rule, the probability of an example $x = (x_1, x_2, ..., x_n)$ being class c is:

$$p(c|x) = \frac{p(x|c)\,p(c)}{p(x)}.$$

x is classified as the class $c = +$ if and only if:

$$f_b(x) = \frac{p(c = +|x)}{p(c = -|x)} \geq 1,$$

where $f_b(x)$ is called a Bayesian classifier [26].
Let us assume that all attributes are independent given the value of the class variable; that is,

$$p(x|c) = p(x_1, x_2, ..., x_n|c) = \prod_{i=1}^{n} p(x_i|c),$$

The resulting classifier is then:

$$f_{nb}(x) = \frac{p(c=+)}{p(c=-)} \prod_{i=1}^{n} \frac{p(x_i|c=+)}{p(x_i|c=-)}.$$

The function $f_{nb}(x)$ is called a naive Bayesian classifier, or simply Naive Bayes (NB). Naive Bayes is the simplest form of Bayesian network, in which all attributes are independent given the value of the class variable. This is called conditional independence. It is obvious that the conditional independence assumption is rarely true in most real-world applications. A straightforward approach to overcome the limitation of naive Bayes is to extend its structure to represent explicitly the dependencies among attributes.

3.1.4 Support Vector Machines

Support Vector Machines (SVM) [10,18] revolve around the notion of a "margin". A margin is defined as a side of a hyperplane that separates two data classes. Maximizing the margin and thereby creating the largest possible distance between the separating hyperplane and the instances on either side of it has been proven to reduce an upper bound on the expected generalization error.
If it is possible to linearly separate two classes, we can find an optimally separating hyperplane by minimizing the squared norm of the separating hyperplane. For the minimization we can use convex quadratic programming (QP). In the case of linearly separable data, once the optimal separating hyperplane is found, data points that lie on its margin are known as support vector points and the solution is represented as a linear combination of only these points (see Figure 3). Other data points are ignored.

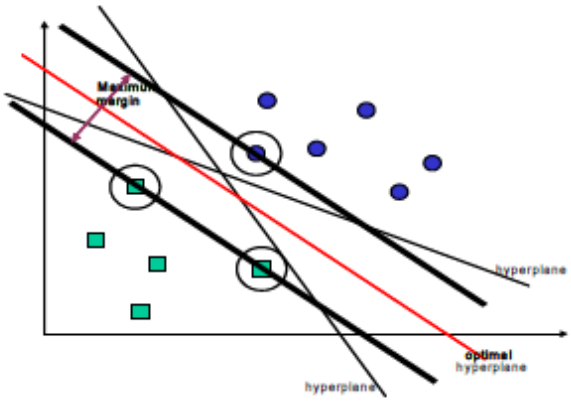


Figure 3. Example of binary SVM classifier, showing candidate separating hyperplanes (thin lines), the optimal hyperplane (red line), maximal margin (bold lines) and support vectors (circled examples)

The advantage of SVM is that its model complexity is unaffected by the number of features encountered in the training data. So that is why SVM is highly recommended for learning tasks where the number of features is large with respect to the number of training instances.

3.1.5 Neural Networks

A neural network (NN) [16,25] is a model inspired by the structure of biological neural networks. In most cases a NN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.
It consists of an interconnected group of artificial neurons. These neurons are input, hidden or output neurons. Input neurons represent the attributes of the given example, and the output neurons represent the classes (in classification). In one NN, we can have one or more hidden levels of hidden neurons.
During the classification of the given example, we provide the input neurons with the values of its attributes. After that, every other neuron (in the hidden levels and the output neurons) calculates its result by using the weighted sum of their input signals. The output value of the last level of the NN specifies the class for the given example.
The weight used in each neuron is calculated using a backpropagation algorithm. When we start the learning process, all the weights are set randomly. The classification of an example from the training set is then performed using these weights. After we obtain the final result, we calculate the difference between the obtained and the desired result. Using this difference, we correct the weights on links between neurons in the direction from the output to the input layer. We then repeat this for all the examples in the training set.
Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data.

3.1.6 k-Nearest Neighbors

K-nearest neighbors (kNN) [18] is an instance-based learning algorithm, which means that it is a lazy-learning algorithm as it delays the induction or generalization process until classification is performed. kNN and all lazy-learning algorithms require less computation time during the training phase than eager-learning algorithms (such as decision trees, neural and Bayes networks) but more computation time during the classification process.
kNN has a very simple principle. It assumes that the instances within a dataset will generally exist in close proximity to other instances that have similar properties. If the instances are tagged with a classification label, then the value of the label of an unclassified instance can be determined by observing the class of its nearest neighbors. The kNN classifier locates the k nearest instances to the query instance and determines its class by identifying the single most frequent class label.
Let us presume we have n features that describe the instances in a dataset. Then we can an instance as a point within an n-dimensional space where each dimension corresponds to one of the features. The most important information then is the relative distance between instances. It is determined by using a distance metric. Ideally, the distance metric must minimize the distance between two similarly classified instances, while maximizing the distance between instances of different classes.

kNN has been proved to be very powerful in many real domains, but there are some disadvantages, such as:

i) they have large storage requirements,
ii) they are sensitive to the choice of the similarity function that is used to compare instances,
iii) they lack a principled way to choose *k*, except through cross-validation or similar, computationally-expensive technique.

We say that the classifier is *unstable* if small changes in the training-test set split can result in large changes in the resulting classifier. kNNs stability distinguishes it from decision trees and some kinds of neural networks.

3.1.7 Boosting

In boosting [1,13], we try to generate complementary base-learners by training the next learner on the mistakes of the previous learners. Every learner should have the error probability less than ½ so that it would be better than random guessing on a two-class problem. We say that a learner is *weak* if it has error probability close to ½. A *strong* learner on the other hand has arbitrarily small error probability, much smaller than ½. The original *boosting* algorithm [1] combines three weak learners to generate one strong learner.

The idea of boosting is to calculate weights for the examples of the training set considering how difficult they are. In the first iteration, the algorithm builds a model on the original learning examples (the weight is 1.0). This model classifies all learning examples. After that, we decrease the weights for the examples that were classified properly, and increase the weights for the ones that were not. So the next iteration of learning will work mostly with examples that were more difficult to classify. We repeat this process until the classification error becomes significantly small, or it becomes too high (the remaining examples are too hard to classify properly). The result of all these iterations is the generated series of models. For prediction, we use all these models weighted by their accuracy on the training set they were generated from.

Though it is quite successful, the disadvantage of the boosting method is that it requires a very large training sample. The sample should be divided into three and furthermore, the second and third classifiers are only trained on a subset on which the previous ones were not.

The most appealing feature of boosting is its empirical resistance to overfitting for various classification tasks.

3.2 Regression

In classification, given an input, the output that is generated is categorical. When the output is a continuous function, then we are working on problem in regression [1].

At the beginning, we have a training set of examples:

$X = \{x^t, r^t\}_{t=1}^N$

where $r^t \in R$ and x^t is the vector of attributes for example *x*. We would like to find the function *f*(*x*) that passes through these points such that we have $r^t = f(x^t)$.

In regression, we would like to write the numeric output, called the dependent variable, as a function of the input, called the independent variable. We assume that the numeric output is the sum of a deterministic function of the input and random noise

$$r = f(x) + \varepsilon$$

where $f(x)$ is the unknown function and ε is the noise.
In the following, we make a brief overview of the regression models which are relevant for the work in our thesis.

3.2.1 Linear Regression

Using linear regression [1], the problem can be represented as a linear model, for example:

$$g(x^t | w_1, w_0) = w_1 x^t + w_0$$

where $g(x|\theta)$ is an *estimator* and θ is a *parameter vector*. Its elements are also called parameters, effects, or regression coefficients. In the example above, θ has two elements, w_0 and w_1.
By taking the derivative of the sum of squared errors, we get two equations and two unknowns:

$$\sum_t r^t = N w_0 + w_1 \sum_t x^t$$

$$\sum_t r^t x^t = w_0 \sum_t x^t + w_1 \sum_t (x^t)^2$$

which can be written in vector-matrix form as $\mathbf{A}\boldsymbol{w} = \boldsymbol{y}$ where

$$A = \begin{bmatrix} N & \sum_t x^t \\ \sum_t x^t & \sum_t (x^t)^2 \end{bmatrix}, w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, y = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \end{bmatrix}$$

and can be solved as $\boldsymbol{w} = \mathbf{A}^{-1}\boldsymbol{y}$.

3.2.2 Regression trees

A regression tree [1,8] is constructed in almost the same manner as a decision tree (Section 3.1.1). It has inner nodes that represent attributes, branches that are values of the attributes and leaves that represent the function which are used for calculation of the dependent variable. The function in leaves can be a constant, a linear or any other function.
The feature that best divides the training data would be the root node of the tree. There are numerous methods for finding the feature that best divides the training data. The same procedure is then repeated on each partition of the divided data, creating subtrees until the training data is divided into subsets. For the evaluation of attributes when building regression trees, we use these two measures: the difference in variance and a regression ReliefF.
To predict a target value for new instances, we start at the root node and move down the branches of the tree based on the feature values of the instance. We calculate the dependent variable for the instance using the function in leaf we reached moving down the tree. This function can be calculated in many different ways. For example we can use an average of the values of all the training instances in a leaf.

Frequently, a node is not split further if the number of training instances reaching a node is smaller than a certain percentage of the training set, for example, 5 percent, regardless of the impurity or error. The idea is that any decision based on too few instances causes variance and thus generalization error. Stopping tree construction early on, before it is full, is called prepruning the tree.
Another possibility to get simpler trees is postpruning, which in practice works better than prepruning. We saw before that tree growing is greedy where at each step we make a decision, namely, generate a decision node, and continue further on, never backtracking and trying out an alternative. The only exception is postpruning where we try to find and prune unnecessary subtrees.

3.2.3 Random Forest

Random forest [9,21] in regression works in a similar way as random forest in classification (see Section 3.1.2). Main difference to classification is that random forest for regression has output values which are numerical.

For regression, the random forest prediction is the unweighted average over the collection:

$$h(x) = \frac{1}{K}\sum_{k=1}^{K} h(x;\theta_k)$$

where $h(x;\theta_k), k = 1,...,K$ are tree predictors, x represents the observed input (covariate) vector and θ_k are independent and identically distributed random vectors. This is because we assume that the training set is independently drawn from the distribution of a random vector.

3.2.4 Neural Networks

We have already discussed neural networks [24] in classification in Section 3.1.5. Neural networks in regression are almost the same. The main difference is the following: while in classification, we have a separate neuron for each class at the output level, in regression, we only have one neuron which outputs a continuous value of the dependent variable.
The most common neural network architectures have outputs in a limited range. The output therefore often needs to be scaled to the target interval of the particular prediction problem.

3.2.5 Bagging

Bagging [1,6] is a voting method where the base-learners are transformed by training them over slightly different training sets. Given a sample, we generate L slightly different samples using a bootstrap method where given a training set X of size N, we choose N instances randomly from X with replacement. It is possible that some instances are drawn more than once and that certain instances are not drawn at all. Then the base-learners are trained with these L samples. During testing, bagging takes an average of all base-learners' predictions.
Bagging can be used both for classification and regression. In the case of regression, to be more robust, one can take the median instead of the average when combining predictions.

3.2.6 Support Vector Machines

We have already discussed SVM in the previous section. Here we will briefly discuss how support vector machines can be generalized to regression.
In support vector regression, we use the ε-sensitive loss function to determine the error of prediction:

$$e_2(r^t, f(x^t)) = \begin{cases} 0 & \text{if } \left|r^t - f(x^t)\right| < \varepsilon \\ \left|r^t - f(x^t)\right| - \varepsilon & \text{otherwise} \end{cases}$$

which means that we tolerate errors up to ε and also that errors beyond have a linear effect and not quadratic. This error function is therefore more tolerant to noise and is thus more robust. The target SVM hyperplane, achieved through optimizing this criterion, therefore represents a best fit through the given data points in the problem space.

3.2.7 Locally Weighted Regression

The nearest-neighbor approaches described in the previous section 3.1.6 can be thought of as approximating the target function $f(x)$ at the single query point $x = x_q$.
Locally weighted regression (LWR) [19] is a generalization of this approach. It constructs an explicit approximation to f over a local region surrounding x_q. Locally weighted regression uses nearby or distance-weighted training examples to form this local approximation to f. For example, we might approximate the target function in the neighborhood surrounding x, using a linear function, a quadratic function, a multilayer neural network, or some other functional form.
The phrase "locally weighted regression" is called *local* because the function is approximated based a only on data near the query point, *weighted* because the contribution of each training example is weighted by its distance from the query point, and *regression* because this is the term used widely in the statistical learning community for the problem of approximating real-valued functions.

3.2.8 Generalized Additive Models

The methods of generalized additive models [15,23] represent a generalization of multiple regression. Multiple regression is a special case of general linear models. In linear regression, if we have k predictors the equation we use is:
$y = b_0 + b_1*x_1 + ... + b_k*x_k$
where y stands for the (predicted values of the) dependent variable, x_1 through x_k represent the k values for the predictor variables, and b_0, and b_1 through b_k are the regression coefficients estimated by multiple regression.
A generalization of the multiple regression model would be to maintain the additive nature of the model and to replace the simple terms of the linear equation b_i*x_i with $f_i(x_i)$ where f_i is a non-parametric function of the predictor x_i. In other words, instead of a single coefficient for each variable (additive term) in the model, in additive models an unspecified (non-parametric) function is estimated for each predictor, to achieve the best prediction of the dependent variable values.

3.3 Statistical comparison of models' accuracies

When we wish to compare supervised ML algorithms, usually we perform statistical comparisons of the accuracies of trained classifiers on specific datasets [11]. If we have sufficient supply of data, we can sample a number of training sets of size *N*, run the two learning algorithms on each of them, and estimate the difference in accuracy for each pair of classifiers on a large test set. The average of these differences is an estimate of the expected difference in generalization error across all possible training sets of size *N*, and their variance is an estimate of the variance of the classifier in the total set.
The next step would be to perform paired t-test to check the null hypothesis that the mean difference between the classifiers is zero. This test can produce two types of errors:

- Type I error is the probability that the test rejects the null hypothesis incorrectly (i.e. it finds a "significant" difference although there is none).
- Type II error is the probability that the null hypothesis is not rejected, when there actually is a difference.

In practice, however, we often have only one dataset of size *N* and all estimates must be obtained from this sole dataset. We create different training sets by subsampling. The instances that are not used for training are used for testing. Unfortunately, this violates the independence assumption necessary for proper significance testing. The consequence of this is that Type I errors exceed the significance level. This is problematic because it is important for the researcher to be able to control Type I errors and know the probability of incorrectly rejecting the null hypothesis. Several heuristic versions of the t-test have been developed to alleviate this problem. Ideally, we would like the test's outcome to be independent of the particular partitioning resulting from the randomization process, because this would make it much easier to replicate experimental results published in the literature. However, in practice there is always certain sensitivity to the partitioning used. To measure replicability we need to repeat the same test several times on the same data with different random partitionings — usually ten repetitions— and count how often the outcome is the same.

3.4 Estimation of attribute quality

In our experimental work we evaluate the quality of existing and novel meta-attributes with the following quality measures [17]: information gain, minimum description length, gini-index and ReliefF.

3.4.1 Information gain

Most measures used in machine learning are based on information content. The amount of information, necessary to determine the outcome X_j of an experiment, is defined as follows:
$I(X_j) = -\log_2 P(X_j)$

The average amount of information, necessary to determine the outcome among *m* disjoint possible outcomes X_j; $j = 1,\ldots,m$, $\sum P(X_j) = 1$, is called *entropy* of an outcome. We calculate entropy using the formula:

$$H(X) = -\sum_{j}^{m} P(X_j) \log_2 P(X_j)$$

Information gain is defined as the amount of information provided by one attribute for determining the class:
$Gain(A) = H_C - H_{C|A}$.
where H_C denotes class entropy and $H_{C|A}$ conditional class entropy given the value of attribute A.

Information gain is also referred to as *mutual information* due to its symmetry:
$H_C - H_{C|A} = H_C - H_A - H_{CA} = I(A;C) = H_A - H_{A\backslash C} = I(C; A)$

3.4.2 Minimum Description Length

Minimum description length (MDL) principle states that the best hypothesis for a given set of data is the one that leads to the best compression of the data. Using that principle we can define the quality of attribute as its compressivity.
First, we need to define the problem of data transmission through the communication channel. We have a sender and a receiver and they both know the number of values m of the given attribute A and the number of classes m_0. They also have the access to the value $v^{(l)}$ of attribute A for each training instance t_l. However, only the sender knows the correct classes of instances. He needs to send the shortest possible message containing the classes of all instances to the receiver.
The sender has two options:

- he can explicitly code the classes,
- he can use the attribute A and code each class separately for each value of the attribute.

The first option requires that the message contains also the class probability distribution, so that receiver can decode the classification. In the second option, however, each value of the attribute corresponds to different class distribution. Therefore, in that case we have to put a class probability distribution for each attribute value in the message, besides classification.
The quality $MDL'(A)$ of attribute A is defined as the compressivity of attribute, i.e. as the length difference of code without using attribute values and the code with using them. The quality is normalized with the number of training instances:
$MDL'(A) = (Prior\_MDL' - Post\_MDL'(A)) / n =$

$$= Gain(A) + \frac{1}{n}\left(\log\binom{n_j + m_0 - 1}{m_0 - 1} - \sum_j \log\binom{n_{.j} + m_0 - 1}{m_0 - 1} \right)$$

where $Gain(A)$ is the measure information gain described above, n is the number of instances and m_0 is the number of classes.
The above quality measure uses coding that is optimal only for arbitrary number of codewords (in our case one codeword corresponds to one training instance). If, however, the number of codewords (training instances) is known in advance (as is the case with the number of training instances), we can apply more optimal coding. The number of all possible classifications of n training examples is equal to:

$$\binom{n}{n_{1.}, \ldots, n_{m_0.}}$$

Therefore, the quality of the attribute is defined with:

$$MDL(A) = \frac{1}{n}\left(\log\binom{n}{n_{1.},...,n_{m_0.}} - \sum_j \log\binom{n_{.j}}{n_{1j},...,n_{m_0 j}} + \log\binom{n+m_0-1}{m_0-1} - \log\binom{n_{.j}+m_0-1}{m_0-1} \right)$$

This measure is most appropriate for estimating the quality of multivalued attributes. Its advantage is also the detection of useless attributes: if $MDL(A) < 0$ then the attribute A is non-compressive and therefore useless for theory construction (unless it is useful in combination with other attributes, take a look at ReliefF algorithm in Section 3.4.4).

3.4.3 Gini-index

Prior Gini-index measure is defined as follows:

$$Gini\_prior = \sum_k \sum_{l \neq k} p_{k.} p_{l.} = 1 - \sum_k p_{k.}^2$$

It is an impurity measure. The quality $Gini(A)$ of attribute A is defined with a difference between prior and expected posterior Gini-index:

$$Gini(A) = \sum_j p_{.j} \sum_k p_{k|j}^2 - \sum_k p_{k.}^2$$

$Gini(A)$ has the following properties:

- **Nonnegativity:** $Gini(A) \geq 0$
- **Maximum:** $Gini(A) = Gini\_prior \Leftrightarrow \forall j : \exists! k : p_{k|j} = 1$
- **Increasing the number of attribute values increases *Gini*(*A*)**. This property is undesired because it makes $Gini(A)$ overestimate multivalued attributes (just like information gain).

3.4.4 ReliefF

All measures described so far evaluate the quality of an attribute independently of the context of other attributes. Equivalently, they assume independence of attributes with respect to class. The context of other attributes can be efficiently taken into account with algorithm ReliefF. This algorithm has a simpler version, called *RELIEF*, which is designed for two-class problems. It calculates a nearest instance from the same class (nearest hit H) and a nearest instance from the opposite class (nearest miss M) for each instance from a random subset of m ($m \leq n$) training instances. Then it updates the quality of (each) attribute with respect to weather the attribute differentiates two instances from the same class and weather it differentiates two instances from the opposite class.

A more realistic variant of *RELIEF* is its extension, called *ReliefF*. It is able to deal with incomplete and noisy data and can be used for evaluating the attribute quality in multi-class problems. The most important part of algorithm *RELIEF* is searching for the nearest hit and miss. Noise (mistakes) in class and/or attribute value significantly affects the selection of nearest hits and misses. In order to make this process more reliable in the presence of noise, ReliefF uses k nearest hits and k nearest misses and averages their contributions to attributes' quality estimates. That is how it works with noisy data. As for the multi-class problems,

instead of *k* nearest hits and misses, *ReliefF* searches for *k* nearest instances from each class. The contributions of different classes are weighted with their prior probabilities.

3.4.5 Principal Component Analysis

Principal Components Analysis (PCA) [22] is a way of identifying patterns in data and expressing the data in such a way as to highlight their similarities and differences. We mostly use PCA when we have data with high dimension, where the luxury of graphical representation is not available.

In this section, we will take you through the example of steps one needs to perform a Principal Components Analysis on a set of data.

Step 1: Acquire some data
We are going to present the process of PCA using a simple 2 dimensional data set because it makes it easier to provide plots of the data and show what the PCA analysis is doing at each step. The data is shown in Table 2(a). and the plot of that data can be seen in Figure 4(a).

Step 2: Subtract the mean
For PCA to work properly, you have to subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension. So, all the *x* values have *x*' (the mean of the *x* values of all the data points) subtracted, and all the *y* values have *y*' subtracted from them. This produces a data set whose mean is zero. The adjusted data set with mean subtracted is shown in Table 2(b).

Original data (a)

x	*y*
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.1	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Adjusted data (b)

x	*y*
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

Transformed Data (c)

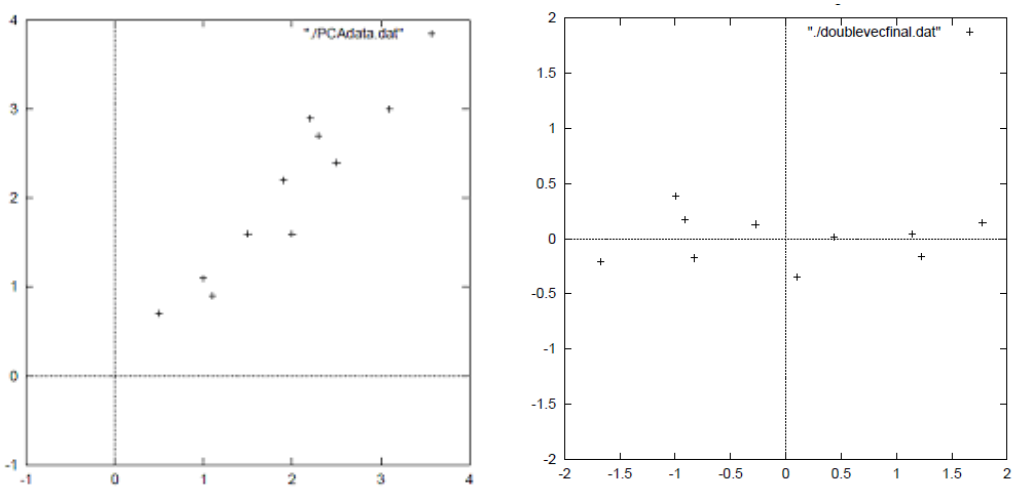
x	*y*
-0.827970186	-0.175115307
177758033	0.142857227
-0.992197494	0.384374989
-0.274210416	0.130417207
-167580142	-0.209498461
-0.912949103	0.175282444
-0.0991094375	-0.349824698
114457216	0.0464172582
0.438046137	0.0177646297
122382056	-0.162675287

Table 2. PCA example data a) original data (on the left); b) data with the means subtracted (in the middle); c) data by applying the PCA analysis using both eigenvectors (on the right)

Step 3: Calculate the covariance matrix
The covariance matrix for our example is:

$$\text{cov} = \begin{pmatrix} 0.616555556 & 0.615444444 \\ 0.615444444 & 0.716555556 \end{pmatrix}$$

So, since the non-diagonal elements in this covariance matrix are positive, we should expect that both the *x*and *y* variable increase together.



a) **b)**

Figure 4. A plot of example data: a) original data (on the left); b) data by applying the PCA analysis using both eigenvectors (on the right)

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix
Here are the eigenvectors and eigenvalues:

$$eigenvalues = \begin{pmatrix} 0.0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} 0.735178656 & 0.677873399 \\ 0.677873399 & 0.735178656 \end{pmatrix}$$

A very important thing for PCA is to have the eigenvectors in a *unit* form, i.e. their lengths need to be 1.
By the process of taking the eigenvectors of the covariance matrix, we have been able to extract lines that characterize the data. Now we want to transform the data so that it is expressed in terms of those lines.

Step 5: Choosing components and forming a feature vector
Here is where the notion of data compression and reduced dimensionality comes into place. The eigenvector with the *highest* eigenvalue is the *principle component* of the data set. It represents the most significant relationship between the data dimensions.
In general, once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance. Now, if you like, you can decide to *ignore* the components of lesser significance. You do lose some information, but if the eigenvalues are small, you do not lose much. If you leave out some components, the final data set will have fewer dimensions than the original. Now we need to form a *feature vector*, which is a matrix of vectors. This is constructed by

taking the eigenvectors that we want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.
$FeatureVector = (eig_1\, eig_2\, eig_3 \ldots eig_n)$

Step 5: Deriving the new data set
Once we have chosen the components (eigenvectors) that we wish to keep in our data and formed a feature vector, we simply take the transpose of the vector and multiply it on the left of the original data set, transposed.
FinalData = RowFeatureVector x *RowDataAdjust*
where *RowFeatureVector* is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and *RowDataAdjust* is the mean-adjusted data *transposed*, i.e. the data items are in each column, with each row holding a separate dimension.
This will give us the original data *solely in terms of the vectors we chose*. Our original data set had two axes, *x* and *y*, so our data was in terms of them. It is possible to express data in terms of any two axes that you like. If these axes are perpendicular, then the expression is the most efficient. This was why it was important that eigenvectors are always perpendicular to each other.
To show this on our data, we do the final transformation with each of the possible feature vectors. We get the data from Table 2(c) and the plot is shown in Figure 4(b). This plot is basically the original data, rotated so that the eigenvectors are the axes. This is understandable since we have lost no information in this decomposition.
Basically we have transformed our data so that is expressed in terms of the patterns between them, where the patterns are the lines that most closely describe the relationships between the data. This is helpful because we have now classified our data point as a combination of the contributions from each of those lines. This is the way we will use PCA for the creation of new reliability estimates. We will use contributions of all the existing estimates to try to create new estimates that will perform better that the original ones.

3.5 Meta-learning

In the book by Brazdil et al. [5], meta-learning [5,14] was defined as follows:
"Meta-learning is the study of principled methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes."

Meta-knowledge can be defined as any kind of knowledge that is derived in the course of employing a given learning system. Previous definition emphasizes the notion of meta-knowledge. This claims that a unifying point in meta-learning lies in how to exploit such knowledge acquired on past learning tasks to improve the performance of learning algorithms. The answer to this question is a key to the advancement of the field and continues being the subject of intensive research.
The definition also mentions machine learning processes; each process can be understood as a set of operations that form a learning mechanism. In this sense, a process can be a preprocessing step to learning (e.g., feature selection, dimensionality reduction, etc.), an entire learning algorithm, or a component of it (e.g., parameter adjustment, data splitting, etc.). The process of adaptation takes place when we replace, add, select, remove or change an existing operation (e.g., selecting a learning algorithm, combining learning algorithms, changing the

value for a capacity control parameter, adding a data preprocessing step, etc.). The definition is then broad enough to capture a large set of possible ways to adapt existing approaches to machine learning.

The primary purpose of meta-learning is the understanding of the interaction between the mechanism of learning and the concrete contexts in which that mechanism is applicable. From a practical standpoint, meta-learning has several goals. On one hand, we wish to overcome some of the challenges faced by users with current data analysis tools. On the other hand, we wish to address a problem commonly observed in the practical use of data analysis tools, namely how to profit from the repetitive use of a predictive model over similar tasks.

The last goal is to produce efficient models under the assumption that bias selection is improved when guided by experience gained from past performance. A model will often be predictive in that it will be used to predict the class of new data instances, but other types of models (e.g., descriptive ones) will also be considered.

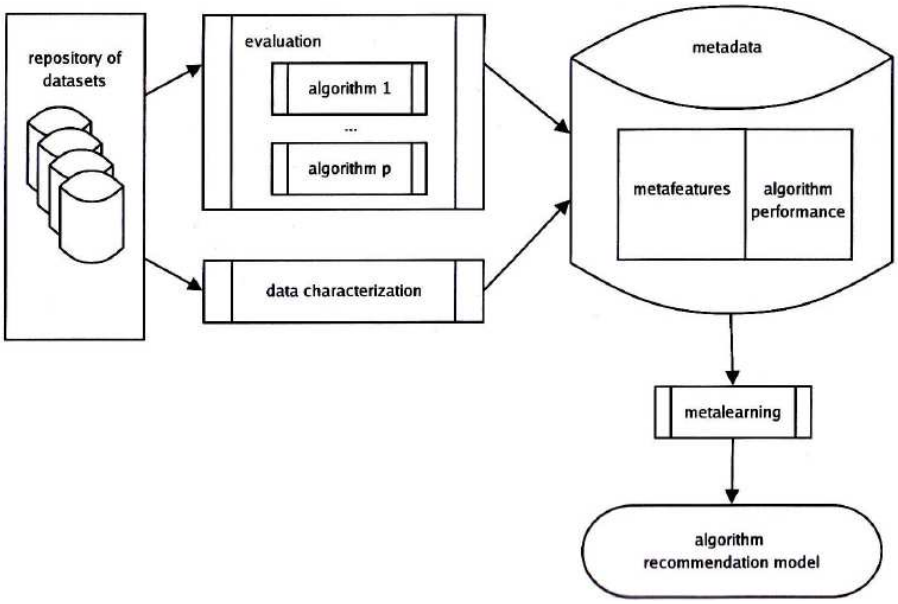


Figure 5. The creation of metadata, for the meta-learning process, using data characterization of the input datasets and the performance of the algorithms on those datasets

Meta-learning requires training metadata, i.e., data about base level learning problems or tasks. The training metadata is a database containing information about the performance of a set of algorithms on a set of datasets. This information is presented in the form of meta-features (attributes) which are obtained using data characterization. This is shown in Figure 5. Three different approaches to data characterization can be identified, namely "*simple, statistical and information-theoretic measures*", "*model-based measures*" and "*landmarkers"*.

The most common approach to data characterization consists of the use of descriptive statistics or information-theoretic measures to summarize the info dataset. Typically, it includes very simple descriptive measures such as the **number of examples** and the **number of features**, which were first used in the earliest meta-learning approaches and are still among the most commonly used meta-features. Most meta-features are based on measures used in statistics (e.g., **mean skewness of numeric features**) and information theory (e.g., **class entropy**). A different approach is *model-based* data characterization. In this approach, a

model is induced from the data and the meta-features are based on properties (e.g., morphological) of that model. An example of a model-based data characteristic is the **number of leaf nodes** in a decision tree. Yet another approach to data characterization is the use of *landmarkers.* Landmarkers are quick estimates of algorithm performance on a given dataset.

For our research, we used meta-learning for the automatic selection of the reliability estimate. When used on different domain and models, reliability estimates also perform differently. Because we do not know which estimate will work best with a specific domain/model pair, we have used meta-learning to help us answer this question. We also wanted to find the rules and patterns which can lead us to the optimal estimate selection by working with all the models.

Chapter 4: Extension and evaluation of the meta-learning model

Motivated by results from previous work in this field [2], where research was done using the meta-learning approach with decision trees as meta-classifier to predict the most appropriate reliability estimate, we decided to:

- evaluate existing and propose novel attributes that can be used for meta-classification task and
- test 6 other meta-classifiers (random forest, naive Bayes, support vector machines, neural networks, k-nearest neighbors, boosting) in addition to the decision tree, to see if any of them perform better than the decision trees.

Every meta-classifier is intended to predict the optimal reliability estimate for a given problem domain/regression model pair. The reliability estimates therefore represent the class values to be predicted, the learning set consists of optimal selections for known domain/model pairs, and the attributes (described in Section 4.1) describe the domain/model characteristics that are related to a particular example.
To each meta-learning example we assigned a class value, representing a reliability estimate which achieved maximum positive correlation with the prediction error (irrespective of whether the correlation was statistically significant or not) for a given regression model. The possible class values therefore were: SAvar, SAbias-s, SAbias-a, CNK-s, CNK-a, LCV, BAGV, DENS, and BVCK (described in Section 2.1). Testing of the meta-classifier's performance was performed using the leave-one-out cross-validation procedure. For each domain/regression model pair, the reliability estimate that was automatically selected was correlated with the prediction error and then the significance of that correlation was evaluated.

4.1 Additional attributes

For our meta-learning problem of choosing the best reliability estimate, we have employed the same meta-features that were used in the previous research [2] and we have added 3 new ones.

Existing meta-features [2]:

- **regression model** (discrete nominal attribute), used on a problem domain (we will refer to it as *model*),
- **number of learning examples** in a given domain (*no.examples*),
- **number of attributes** in a given domain (*no.attr*),
- **relative mean squared error**, achieved by the given regression model on a domain using ten-fold cross-validation (*cv.rmse*),
- **average density** of the problem space, estimated by Parzen windows and sampled in points, given by learning examples (*avg.dens*),
- **average distance to the 5 nearest neighbors**, averaged across all learning examples (*avg.DA*),
- **average distance between the prediction for a given example and the predictions for the 5 nearest neighbors**, averaged across all learning examples (*avg.DK*)

Three additional meta-features:

- **number of attributes and number of learning examples ratio** (*attr.example.ratio*), defined as:

$$attr.example.ratio = \frac{no.attr}{no.examples}$$

- **mean correlation of attributes** in a given domain (*mean.corr*), calculated as:

$$mean.corr = \frac{1}{n^2}\sum_i \sum_j \rho_{A_i,A_j}$$

  where ρ_{A_i,A_j} denotes Pearson's correlation coefficient between attributes A_1 and A_2.

- **mean entropy of attributes** in a given domain (*mean.entr*), calculated as

$$mean.entr = \frac{1}{n}\sum_i H(A_i)$$

  where *H(A)* denotes the entropy of the attribute *A*.

The attribute evaluation results are shown in Section 6.1.

4.2 Alternative meta-classification models

We used seven (7) classification models for construction of the meta-classifier. Parameters of each model are described in the following:

1. Decision trees (DT): for DT we have used **rpart** function implemented in the *rpart* package of R. Gini-index was used to evaluate the attributes during the building of the tree. The stopping criterions were: the minimal number of examples in a node which equaled 20, the minimum number of examples in leaves with the value 7 and the cost-complexity parameter, *cp* (which tells the algorithm not to do the split if that split would not improve the fit by factor *cp*) with the value 0.01.

2. Neural networks (NN): for NN we have used **nnet** function implemented in the *nnet* package of R. We used a neural network with one hidden layer, which had 5 units (neurons). For maximum number of iterations we used the default value of 100.
3. Support vector machines (SVM): for SVM we have used **svm** function implemented in the *e1071* package of R. For the kernel used in training and predicting, we left used the radial basis kernel. The value for the tolerance of termination criterion was 0.001 and the precision parameter was $\varepsilon = 0.1$.
4. Random forests (RF): for RF we have used **randomForest** function implemented in the *randomForest* package of R, using 500 trees. For the attribute evaluation, gini-index was used. We could have defined the maximum number of leaves that the trees in the forest can have. We decided not to define this value, so the trees are grown to the maximum possible.
5. Naive Bayes (NB): for NB we have used **naiveBayes** function implemented in the *e1071* package of R. For NB we defined no explicit parameters.
6. k-nearest neighbors (kNN): for kNN we have used **ipredknn** function implemented in the *ipred* package of R. The number of neighbors considered in the prediction (*k*) was 5.
7. Boosting: for boosting we have used **adaboost.M1** function implemented in the *adabag* package of R. For base-classifiers, we used 10 decision trees. To stop the splitting of the decision trees, we used a complexity parameter with the value 0.01 and a minimum number of instances that must exist in a node in order for a split to be attempted with the value 5.

The performance results of the evaluated models are shown in Section 6.2.

Chapter 5: Estimation of reliability with principal components

We talked about Principal Component Analysis (PCA) in Section 3.4.5. In this Chapter we will present our work with PCA.

We used PCA for the creation of the reliability estimate for a domain/model pair by using contributions of the existing estimates (described in Section 2.1) and transforming them into a new estimate to see if it will perform better that the original ones.
For testing we used 28 domains and 8 regression models. Domains are presented in Table 3 and regression models were covered in Section 3.2.
In the previous work by Bosnić and Kononenko [3], these estimates were ranked by the average percentage of significant positive and negative correlations with the prediction error used with all regression models. As we mentioned before, they worked with nine reliability estimates: SAvar, SAbias-s, SAbias-a, BAGV, LCV, DENS, CNK-s, CNK-a and BVCK.

PCA is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. It involves a mathematical procedure that transforms a number of possibly correlated vectors (in our case, vectors of reliability estimates) into a smaller number of uncorrelated vectors called *principal components*.
It is an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the *first principal component*), the second greatest variance on the second coordinate, and so on. So the first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.
The first principle component is what we are looking for. It will represent a new reliability estimate that we wish to test and compare with other estimates.
Considering that our new estimate is a combination of all estimates, including SAbias and CNK, which can have positive and also negative values, we have to take into account that the new estimate will also have both types of values.

Therefore, we will actually have two new estimates:

- **PCA-a**, where we correlated the values to the absolute prediction error (estimates that were used are SAvar, SAbias-a, CNK-a, LCV, BAGV, DENS, COMB)
- **PCA-s**, where we correlated the values to the signed prediction error (used estimates SAbias-p, CNK-p)

We evaluated the performances of new reliability estimates the same way nine original estimates were tested in [3], using the Pearson correlation coefficient between the reliability estimates and the prediction errors. The significance of the correlation was statistically evaluated using the t-test. Then we compared these results with the results of the other estimates. The results are shown in Section 6.3. This whole process is presented in Figure 6.

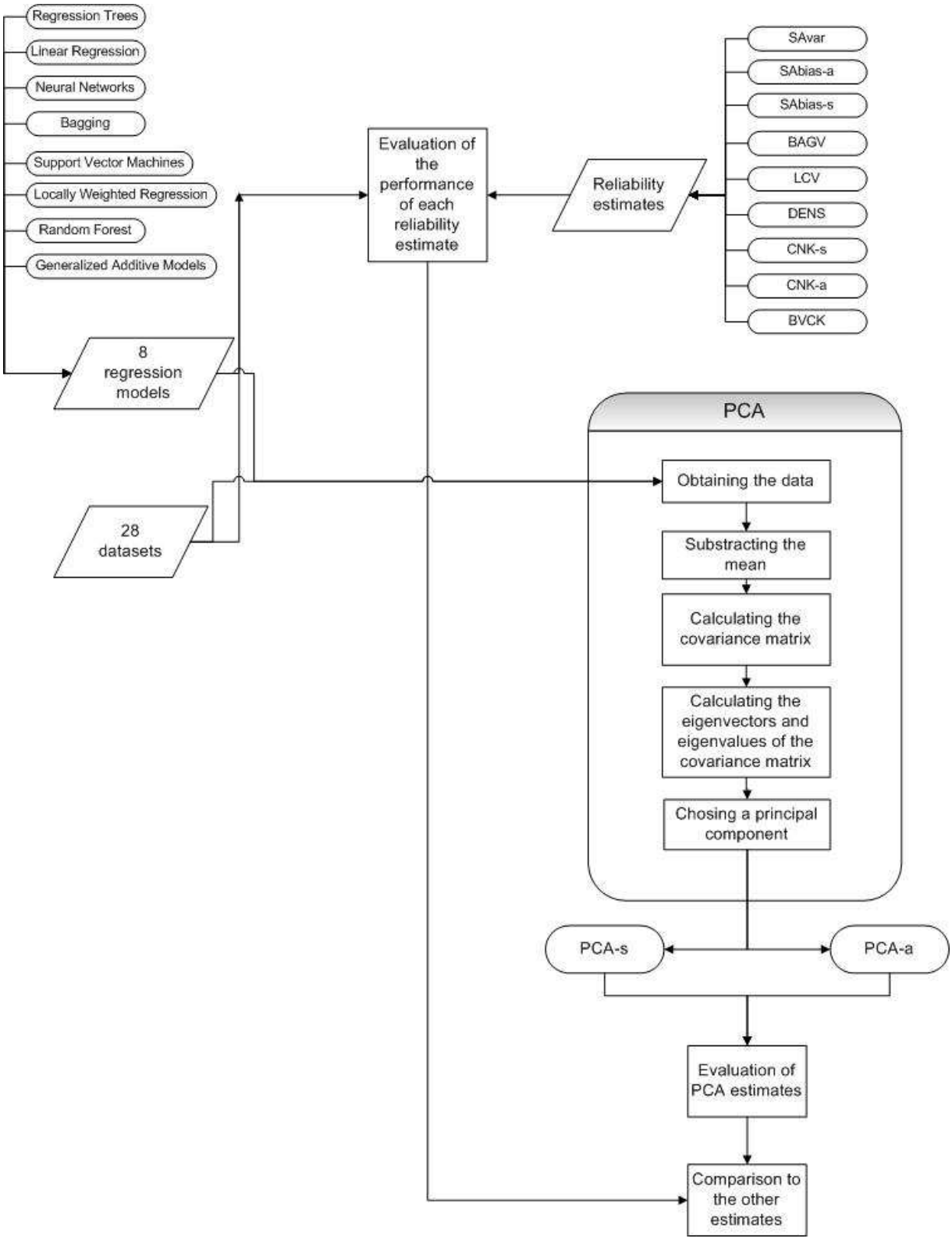


Figure 6. Transformation of existing estimates using principal component analysis and evaluation of the new estimates

Chapter 6: Experimental results

For testing we used 28 standard benchmark data sets, well-known across the whole machine learning community. Each data set is a regression problem. The application domains vary from medical, ecological and technical to mathematical and physical domains. Most of the data sets are available from UCI Machine Learning Repository and from StatLib DataSets Archive. The brief description of data sets is given in Table 3.

Data set	no. of examples	no. of discr. attr.	no. of cont. attr.
autoprice	159	1	14
auto93	93	6	16
autohorse	203	8	17
baskball	96	0	4
bodyfat	252	0	14
brainsize	20	0	8
breasttumor	286	1	8
cloud	108	2	4
cpu	209	0	6
diabetes	43	0	2
echomonths	130	3	6
elusage	55	1	1
fishcatch	158	2	5
fruitfly	125	2	2
grv	123	0	3
hungarian	294	7	6
lowbwt	189	7	2
mbagrade	61	1	1
pharynx	195	4	7
pollution	60	0	15
pwlinear	200	0	10
pyrim	74	0	27
servo	167	2	2
sleep	58	0	7
transplant	131	0	2
triazines	186	0	60
tumor	86	0	4
wpbc	198	0	32

Table 3. Basic characteristics of testing data sets.

Since we experimented with 28 domains and 8 regression models, we therefore formed the meta-learning set consisting of 224 (28×8) learning examples. To assure the unbiasedness of the testing procedure, we removed the example from the meta-learning set which represented the domain/model combination, for which we were meta-predicting the optimal reliability estimate.

The testing has been implemented in statistical package R [20]. R is a language and environment for statistical computing and graphics. It provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering etc.) and graphical techniques, and is highly extensible.

The whole process of meta-learning is shown in the Figure 7.

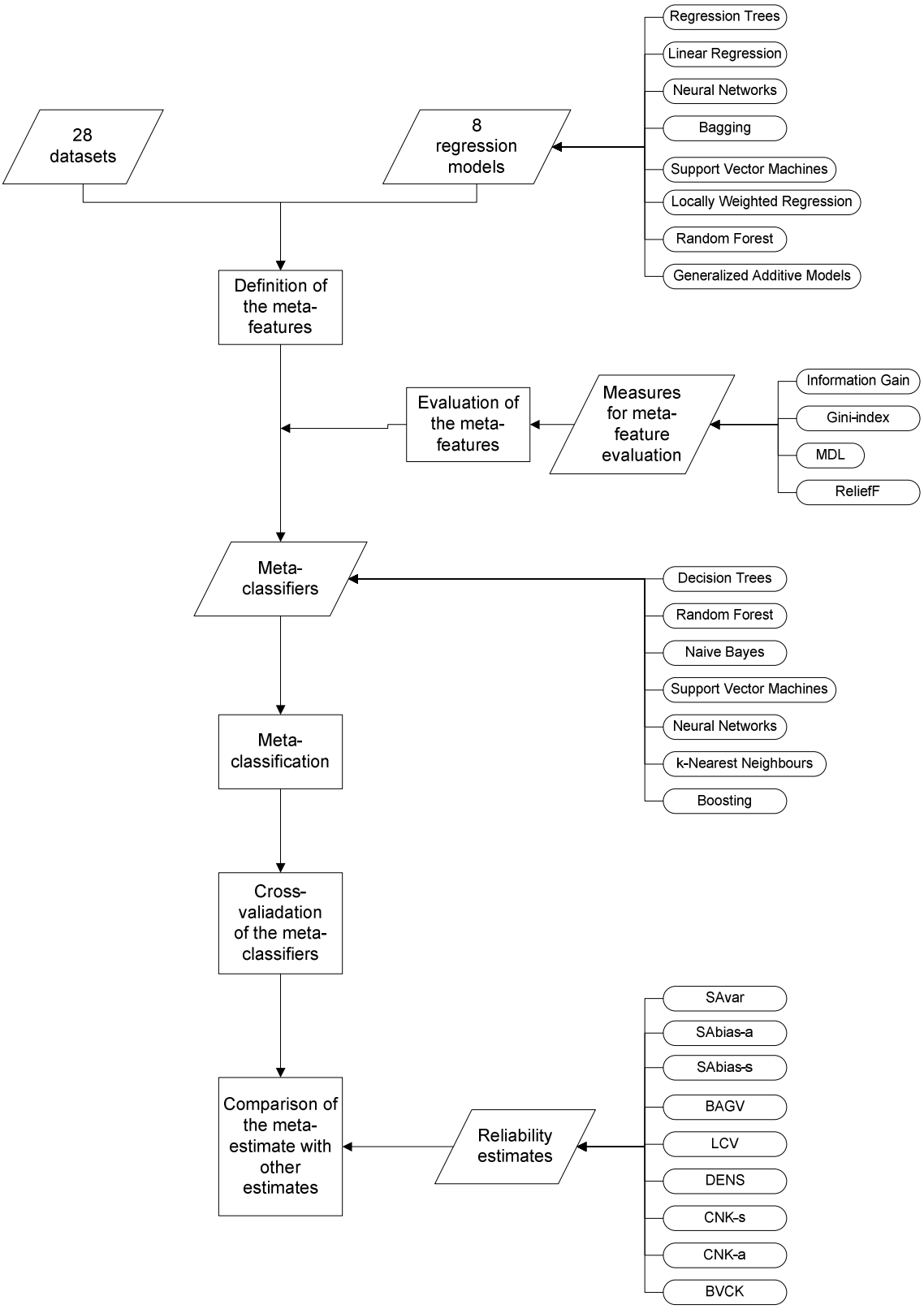


Figure 7. Optimal reliability estimate selection using meta-learning

6.1 Attribute evaluation

In the following, we present the results of our work. In Section 3.4, we described the measures for evaluating the attributes that we used for meta-classification. The description of these attributes can be found in the same Section 4.1
Table 4 shows the results of the evaluation of attributes using measures: information gain, MDL, gini-index and ReliefF. For each measure, we highlighted with green color the best three attributes and with red color the worst three attributes.

	ReliefF	Information Gain	MDL	Gini-index
model	0.03674669	**0.25169580**	**-0.04736606**	**0.046157526**
no.examples	**0.01481601**	**0.08563378**	0.02643294	0.015313890
no.attr	**0.04666153**	0.13874646	**0.07110737**	0.017857609
attr.example.ratio	0.03521726	0.11363837	0.04989815	0.018488255
mean.corr	0.02794570	0.09675295	0.03371233	0.015937745
mean.entr	**0.01842227**	0.09712562	0.03495932	**0.015255318**
cv.rmse	**0.05045326**	**0.21002378**	**0.13929851**	**0.033727301**
avg.dens	**0.04646499**	**0.07625064**	**0.01905591**	**0.014816418**
avg.DA	0.04342832	**0.05967544**	**0.01748912**	**0.009714073**
avg.DK	**0.02396952**	**0.15271199**	**0.08400351**	**0.026589294**

Table 4. Evaluation of attributes

As we can see in Table 4 and in Figure 8, attribute *cv.rmse* was ranked in the top three by all four measures. Attribute *avg.DK* had also very good results. Attribute *model* definitely stands out when measured with information gain and Gini-index, however MDL has evaluated it as one of the worst three attributes. Attribute *avg.DA* and *avg.dens* had sufficiently low quality values and ranks when measured with Information Gain, MDL and Gini-index.
Three new attributes that we added for this research achieved average results: *mean.entr* was in the bottom three when measured with ReliefF and Gini-index. The other two attributes, however, did not stand out in any way.

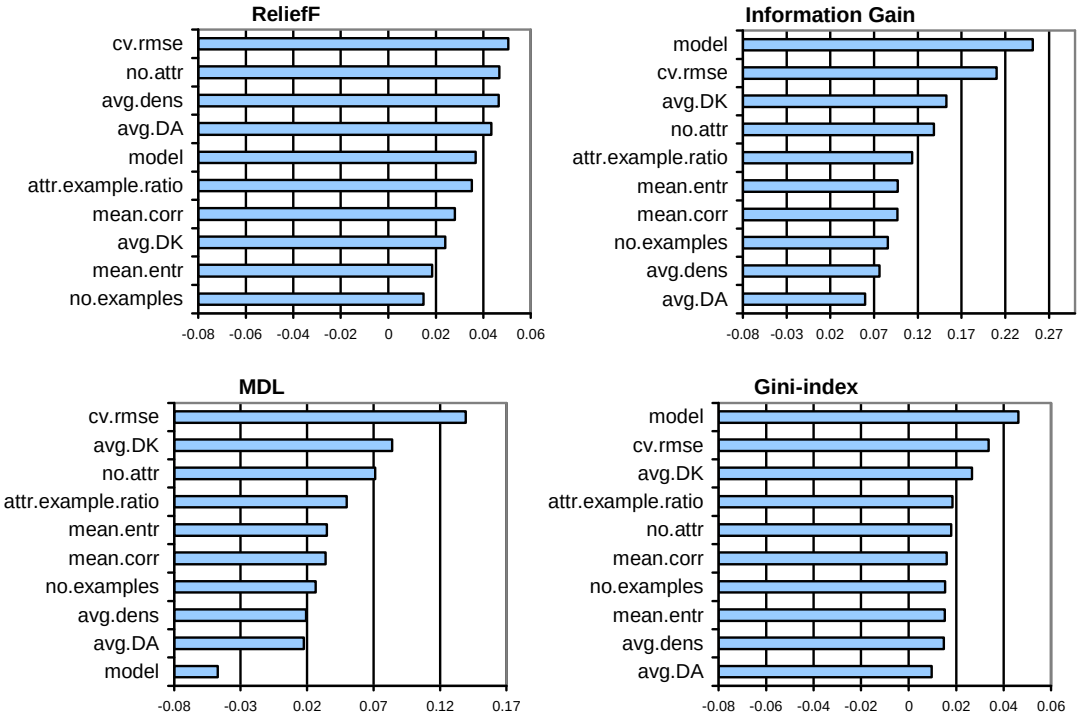


Figure 8. Results of the evaluation of the attributes separately for each measure

6.2 Evaluation of different meta-classifiers

Testing of each meta-classifier was performed using the leave-one-out cross-validation, giving the leave-one-out error and reliability estimate. The performance of the automatically selected reliability estimate was measured using the Pearson correlation coefficient between that reliability estimate and the prediction error, before statistically evaluating its significance using t-test.

	META- – CLASSIFIER MODELS							
model	**DT**	**NN**	**SVM**	**RF**	**NB**	**KNN**	**BOOST**	**Average**
rt	86/0	71/4	89/0	89/0	86/0	86/0	86/0	85/1
lr	64/0	46/0	54/0	64/0	54/0	54/0	61/0	57/0
nn	43/0	36/4	46/0	54/0	46/0	50/0	46/4	46/1
bag50	61/0	43/0	57/0	54/0	57/0	54/0	57/0	55/0
svm	50/4	36/0	54/4	54/0	39/0	57/0	54/4	49/2
lwr	50/0	25/7	43/0	50/0	39/4	46/0	43/4	42/2
rf	43/0	36/0	50/0	50/0	43/0	54/0	57/0	48/0
gam	64/0	43/0	54/0	61/0	54/0	61/0	64/0	57/0
Average	58/1	42/2	56/1	60/0	52/1	58/0	59/2	

Table 5. The percentage of experiments exhibiting significant positive/negative correlations between the reliability estimates and the prediction error for different meta-models

Table 5 shows the results achieved with each meta-classifier. The values in the table cells are comprised of a couple, denoting *positive correlation percentage/negative correlation percentage*. Figure 9 shows the average results across all regression models and Figure 10 shows these results separately for each regression model.

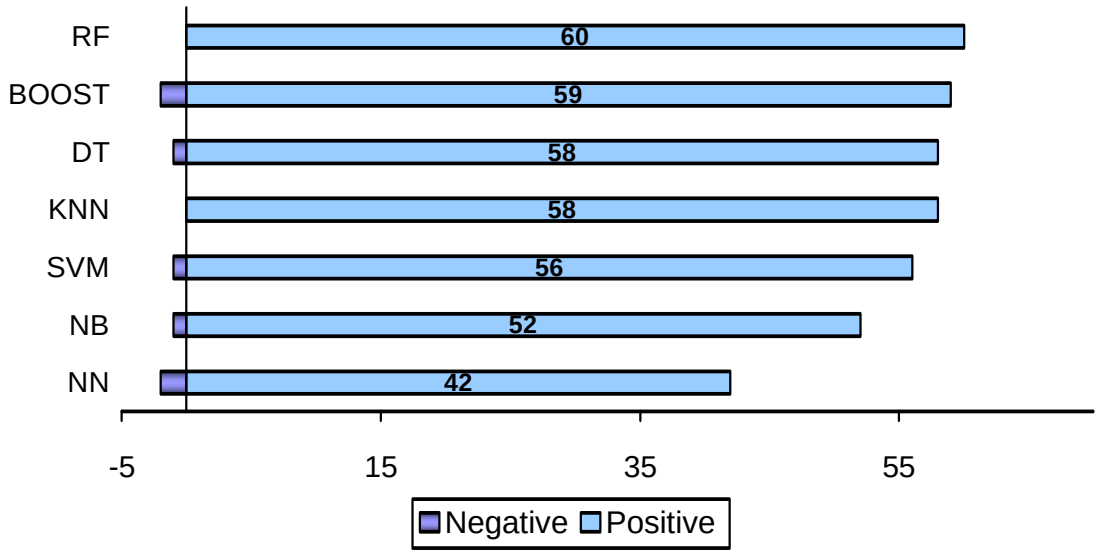


Figure 9. Ranking of the meta-models by the average percentage of significant positive and negative correlations with the prediction error.

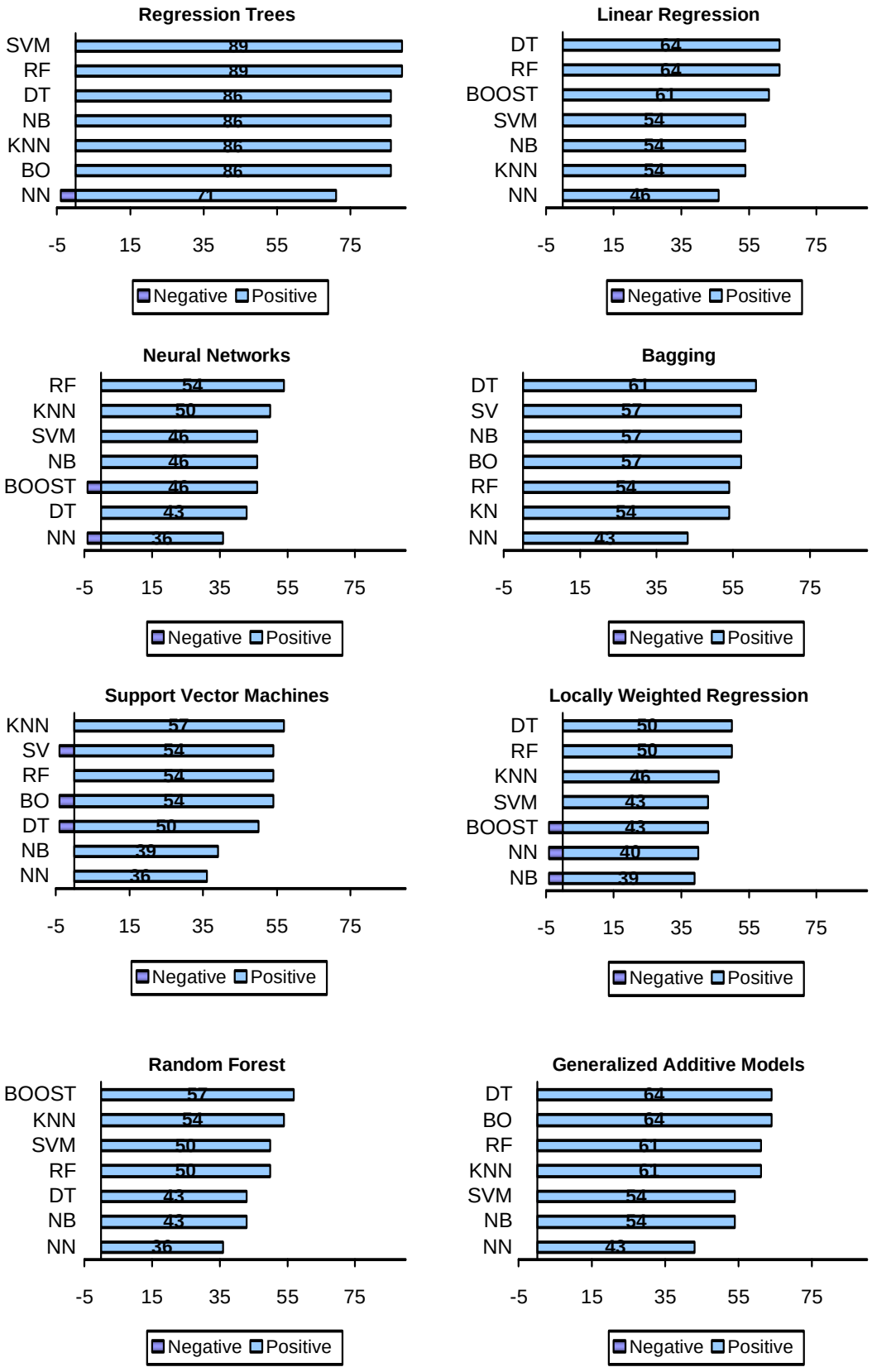


Figure 10. Meta-learning results separately for each regression model

From the summarized results in Table 5 we can see that **random forest** meta-model achieved the best results (highest percentage of positive correlations between reliability estimates and the prediction error and no negative ones). Boosting meta-model did achieve a high percentage of positive correlations as well, but it also achieved 2% of negative ones. Neural networks achieved the worst results with only 42% positive correlations and 2% of negative correlations. The remaining five classifiers all had average results between 52%-58% of the positive correlations.
Regression trees were shown to be the best regression model by far in combination with all of the meta-classifiers. The Locally Weighted Regression (lwr) model had worst results, with an average of only 42% of positive correlations. All other regression models had similar results for all of the classifiers. The detailed results are shown in Tables 7-13 for each classifier separately in the Appendix.

6.3 PCA-based reliability estimate

The performances of both reliability estimates were measured using the Pearson correlation coefficient between the reliability estimate and the prediction error, before statistically evaluating its significance using t-test. Their results are presented in Table 6 in the form *positive correlation percentage/negative correlation percentage* along with the results of the best ranked meta-classifier, random forest, and the original reliability estimates. Results of the evaluation of performance of the original nine estimates were taken from [3].

Model	SAvar	SAbias-s	SAbias-a	BAGV	LCV	DENS	CNK-s	CNK-a	BVCK	META-RF	PCA-a	PCA-s
RT	46/0	82/0	50/0	64/0	36/0	36/4	86/0	68/0	71/4	89/0	36/0	21/64
LR	54/0	7/0	7/4	54/0	32/0	32/4	50/0	57/0	50/0	64/0	36/4	29/21
NN	39/4	18/4	29/4	50/0	36/0	25/4	36/4	39/4	54/0	54/0	36/4	21/18
BAG	46/4	21/0	11/0	57/0	50/0	36/7	25/0	46/0	61/0	54/0	43/7	4/21
SVM	46/4	36/7	25/0	46/0	61/0	39/4	29/11	36/0	46/0	54/0	43/0	14/25
LWR	39/7	4/7	11/7	46/0	46/0	43/7	25/11	32/0	43/0	50/0	32/4	21/11
RF	25/7	14/0	11/0	57/0	61/0	32/11	11/25	46/4	50/0	50/0	32/7	21/18
GAM	54/0	7/0	7/4	50/0	32/0	32/4	50/0	57/0	54/0	61/0	39/4	29/21
Avg	**44/3**	**24/2**	**19/2**	**53/0**	**44/0**	**34/6**	**39/6**	**48/1**	**54/1**	**60/0**	**37/4**	**20/4**

Table 6. Comparison of results for all original reliability estimates, best meta- model (Random Forest), and for PCA-a and PCA-s

From the summarized results in Table 6 we can see that the meta-model still has best results, better even that the best reliability estimate BVCK. Principal Component Analysis (PCA) however, gave no improvement compared to the other reliability estimates. PCA with absolute values, with 37% of positive correlations, had relatively average results when compared to other estimates. When compared to other average results shown in this table, it was ranked 8th out of 12. It was similar results as the estimate CNK-s. PCA-s was even weaker. It had 1% of positive correlations more than the worst estimate, SAbias-a, but also 2% more of negative correlations. All the other estimates and the meta-model had better results.
PCA-a worked best with regression model SVM (support vector machines). It had the same amount of positive correlations as with the bagging model, but no negative correlations.

PCA-s on the other side had the lowest percentage of positive correlations with those two regression models. PCA-s worked best in combination with linear regression and generalized additive models, where it had 29% of positive correlations and 21% of negative ones.
The detailed results for PCA-a are shown in Table 14 and for PCA-s in Table 15 in the Appendix. In Figure 11, we can see the results averaged across all regression models and in Figure 12, separately for each regression model.

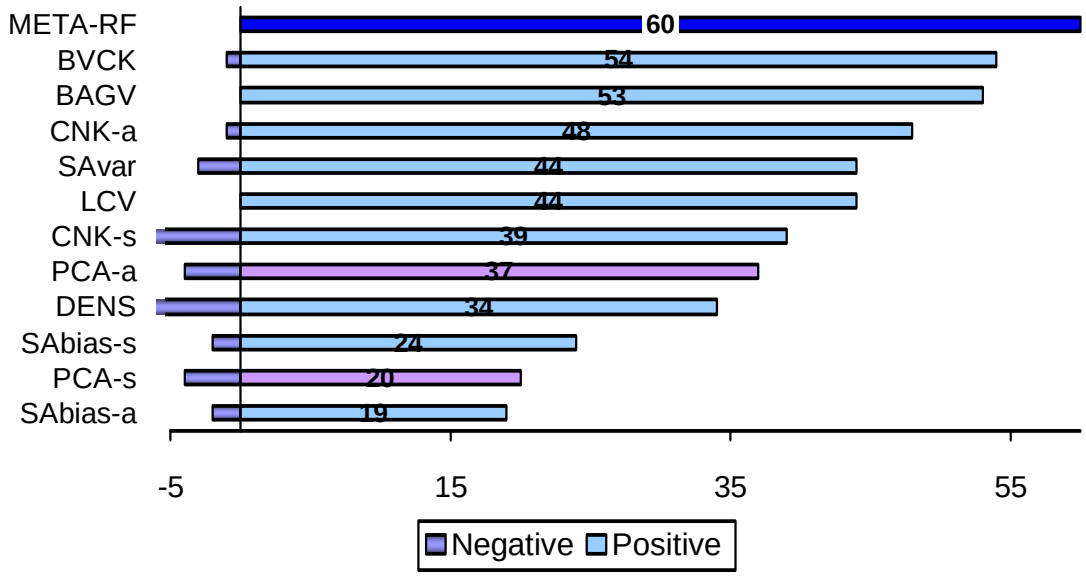


Figure 11. Ranking of the reliability estimates, meta-model and PCAs by the average percentage of significant positive and negative correlations with the prediction error.

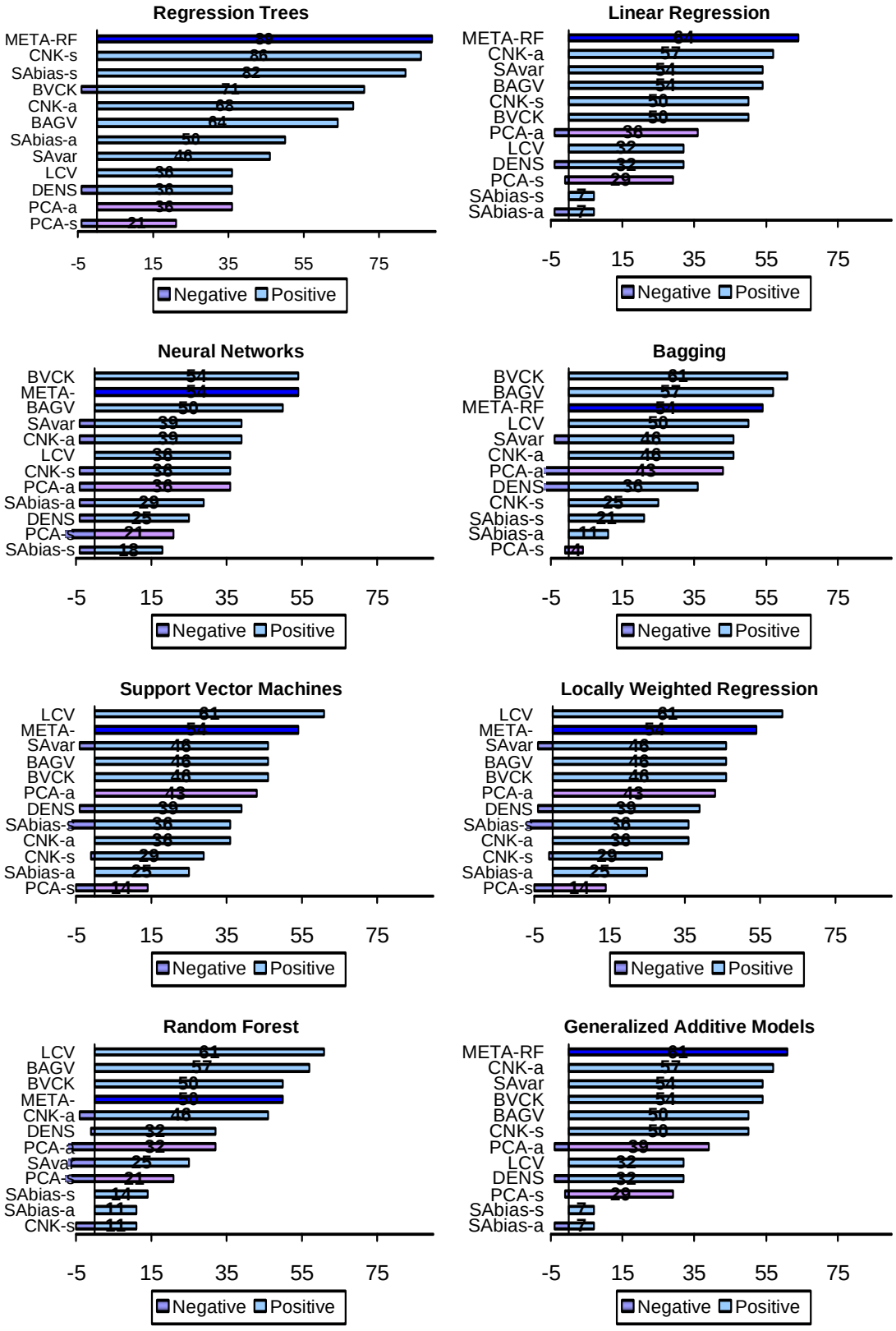


Figure 12. Comparison of estimates separately for each regression model

Chapter 7: Conclusion

The testing of the individual reliability estimates, developed in the previous work by Bosnić and Kononenko [2], exhibited different potentials for the usage of different estimates with particular regression models. In addition, the results also showed that the success of the chosen reliability estimate may also be domain-dependent.
To deal with this problem, in this thesis we explored further an approach for automatic selection of the most appropriate estimate for a given domain/regression model pair: the meta-learning approach. We also used principal component analysis to create new reliability estimates.
For the meta-learning approach, we first created the metadata (meta-examples) that the meta-model would learn from. For this purpose, we chose meta-features (attributes) that, we considered, best described the given domain and regression model. To make sure that the attributes were helpful enough for meta-learning, that they gave enough information about the domain and model, we evaluated the using four different measures.
Attribute ***cv.rmse*** was one of the best three evaluated with all four measures. Attributes ***avg.DA*** and ***avg.dens*** had sufficiently low quality values and ranks when measured with three out of four measures.

With meta-learning, the selection of estimates was performed from the set of nine estimates. These estimates are based on various approaches, i.e. sensitivity analysis, variance of bagged models, local cross-validation, density-based estimation and local error estimation. We used meta-learning approach with seven different classifiers: decision trees, random forests, naive Bayes, support vector machines, neural networks, k-nearest neighbors and boosting. They all predicted a reliability estimate based on the attributes that describe the domain and regression model properties.
Looking at the results we see that the **random forest** meta-model had the highest average percentage of positive correlations between reliability estimates and the prediction error and no negative ones. **Neural networks** gave worst results with only 42% positive correlations and 2% of negative correlations.
Regression trees were shown to be the best regression model by far in combination with all of the meta-classifiers. The **locally weighted regression** model had worst results, with an average of only 42% of positive correlations.

For our second approach of automatically selecting the optimal reliability estimate we used Principal Component Analysis. PCA involves a mathematical procedure that transforms a number of possibly correlated vectors (in our case, vectors of reliability estimates) into a smaller number of uncorrelated vectors called *principal components*. The first (best) principal component accounts for as much of the variability in the data as possible. And that is exactly what we are looking for. We used that component as a new reliability estimate and we compared it with the nine other estimates.

Considering that our new estimate is a combination of all estimates, including SAbias and CNK, which can have positive and also negative values, we took into account that the new estimate will also have both types of values. So we created two new estimates: **PCA-a** (absolute) and **PCA-s** (signed).

We compared the testing results of these two estimates with the results of the original estimates (the results were obtained from [3]) and with the results of the best meta-model, random forest. The meta-model had best results, better even that the best reliability estimate BVCK. PCA however, gave no improvement compared to the other reliability estimates. PCA-a had relatively average results when compared to other estimates. PCA-s was even weaker.

PCA-a worked best with **support vector machines** and PCA-s worked best in combination with **linear regression** and **generalized additive models**.

Considering the results, we feel that we should work on the improvement of the meta-classifier. The purposed improvements could be:

- building a super-classifier with each sub-classifier contributing a vote to the final outcome, perhaps including only the better performing classification models.
- adding additional meta-attributes and evaluating them to ensure better description of the given domain and regression model and more valuable information for meta-learning.

Appendix

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	COMB 0.53427	**COMB** **0.4646**	**COMB** **0.41277**	**CNK-a** **0.61069**	**COMB** **0.2672**	**SAvar** **0.52299**	**COMB** **0.45955**	**COMB** **0.51264**
auto93	CNK-p 0.53887	**CNK-p** **0.38063**	**SAvar** **0.23585**	**COMB** **0.2757**	CNK-p -0.16151	**CNK-a** **0.23312**	SAvar -0.13565	**CNK-p** **0.38063**
autohorse	SAvar 0.27493	**COMB** **0.43584**	**COMB** **0.59094**	**SAvar** **0.26555**	**BAGV** **0.17789**	**SAvar** **0.47808**	**CNK-a** **0.18246**	**COMB** **0.45075**
baskball	BAGV 0.07917	BAGV 0.1022	CNK-p 0.07388	BAGV -0.05046	SAbias-p -0.10536	LCV -0.03887	BAGV -0.07535	BAGV -0.00144
bodyfat	BAGV 0.27545	**SAvar** **0.13982**	**CNK-a** **0.51487**	**COMB** **0.19186**	**BAGV** **0.52404**	COMB 0.08983	**COMB** **0.13887**	**SAvar** **0.13982**
brainsize	SAbias-p 0.29739	CNK-p 0.32309	CNK-p 0.01613	SAbias-p 0.18579	SAbias-p -0.32845	**CNK-p** **0.47202**	SAbias-p 0.10981	CNK-p 0.32309
breasttumor	SAbias-p 0.15076	LCV 0.08169	SAbias-p -0.01371	BAGV -0.01623	**SAbias-p** **0.27749**	LCV -0.05085	SAbias-p 0.1012	LCV 0.08169
cloud	COMB 0.5023	**SAvar** **0.36624**	**SAvar** **0.43477**	**COMB** **0.38232**	**SAvar** **0.66354**	CNK-a 0.12919	**COMB** **0.38733**	**SAvar** **0.36624**
cpu	CNK-a 0.67851	**COMB** **0.62541**	**COMB** **0.63352**	**CNK-a** **0.533**	**SAvar** **0.71583**	**CNK-a** **0.47194**	**COMB** **0.53218**	**COMB** **0.61502**
diabetes	SAbias-p 0.3895	**DENS** **0.41939**	CNK-p 0.22557	**DENS** **0.41375**	**DENS** **0.45087**	**CNK-p** **0.4478**	DENS 0.22544	**DENS** **0.41939**
echomonths	CNK-a 0.33875	CNK-a 0.13434	COMB 0.0842	CNK-a 0.15484	SAvar 0.07245	CNK-a 0.05641	CNK-p 0.10789	CNK-a 0.13434
elusage	COMB 0.38069	SAbias-p -0.19683	DENS 0.2049	**CNK-a** **0.31957**	DENS 0.26141	**COMB** **0.34495**	**CNK-a** **0.31757**	SAbias-p -0.18652
fishcatch	COMB 0.77291	**CNK-a** **0.4976**	**CNK-a** **0.50185**	**CNK-a** **0.59679**	**CNK-a** **0.26217**	**CNK-a** **0.64588**	**CNK-a** **0.32423**	**CNK-a** **0.4976**
fruitfly	SAbias-p 0.21937	SAbias-p 0.04472	LCV -0.11485	SAbias-p 0.07756	SAbias-p 0.10149	LCV -0.03311	**SAbias-p** **0.24583**	SAbias-p 0.04472
grv	CNK-p 0.37882	DENS 0.08467	DENS 0.02761	CNK-p 0.1369	CNK-p 0.03671	CNK-p -0.05567	CNK-p 0.01439	DENS 0.08467
hungarian	BAGV 0.52527	**COMB** **0.33124**	**CNK-p** **0.46864**	**BAGV** **0.62856**	**COMB** **0.40111**	**COMB** **0.40052**	**COMB** **0.43677**	**COMB** **0.32988**
lowbwt	BAGV 0.09024	COMB 0.03933	BAGV 0.02265	**BAGV** **0.27622**	COMB 0.09492	COMB 0.07618	COMB 0.12192	COMB 0.01127
mbagrade	SAbias-p 0.26329	CNK-a 0.0236	CNK-a 0.02416	SAbias-p 0.10203	SAbias-p -0.26381	LCV 0.11005	CNK-a 0.11016	CNK-a 0.0236
pharynx	BAGV 0.45637	**BAGV** **0.20443**	COMB 0.09912	**BAGV** **0.30932**	COMB 0.05888	**COMB** **0.20256**	**BAGV** **0.22485**	**BAGV** **0.33096**
pollution	CNK-p 0.48263	**CNK-p** **0.48793**	CNK-p 0.18868	CNK-p 0.19908	CNK-p 0.07106	BAGV -0.12953	CNK-p 0.09734	**CNK-p** **0.48793**
pwlinear	COMB 0.28619	**CNK-p** **0.31308**	**SAvar** **0.33762**	SAvar 0.10202	**SAvar** **0.2871**	**CNK-p** **0.27003**	SAvar 0.00045	**COMB** **0.21986**
pyrim	BAGV 0.25469	**CNK-p** **0.79437**	**BAGV** **0.63968**	**CNK-p** **0.22991**	CNK-p -0.17277	**COMB** **0.79151**	CNK-p 0.01688	**CNK-p** **0.79437**
servo	BAGV 0.63921	**BAGV** **0.27243**	SAvar 0.1246	**BAGV** **0.60785**	**BAGV** **0.50845**	**BAGV** **0.76048**	**BAGV** **0.76526**	**BAGV** **0.37202**
sleep	SAbias-p 0.50901	**CNK-p** **0.41196**	CNK-p 0.08953	CNK-p 0.1861	**SAbias-p** **0.2814**	COMB 0.21568	CNK-p 0.03716	**CNK-p** **0.41196**
transplant	CNK-p 0.44348	**CNK-p** **0.23291**	**CNK-p** **0.23214**	**CNK-p** **0.39049**	**CNK-p** **0.46497**	CNK-p 0.07256	CNK-p 0.17138	**CNK-p** **0.23291**
triazines	COMB 0.35974	**CNK-p** **0.51331**	**CNK-p** **0.27695**	**COMB** **0.32008**	CNK-a 0.0891	**CNK-p** **0.25893**	**BAGV** **0.469**	**CNK-p** **0.51331**
tumor	SAbias-p 0.09716	SAbias-p 0.03983	BAGV -0.04218	SAbias-p 0.10812	**SAbias-p** **0.34323**	CNK-p -0.02637	SAbias-p 0.15015	CNK-p 0.14711
wpbc	SAbias-p 0.49432	**CNK-p** **0.21094**	CNK-p 0.0978	**SAbias-p** **0.1442**	SAbias-p -0.07399	SAvar 0.01117	COMB 0.08689	**CNK-p** **0.21094**
+	86	64	43	61	50	50	43	64
-	0	0	0	0	4	0	0	0

Table 7. Automatically selected reliability estimates using decision trees and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	SAbias-p 0.43881	**SAvar** **0.40827**	**COMB** **0.41277**	**COMB** **0.64217**	SAbias-p 0.10599	SAbias-p 0.0603	**BAGV** **0.45925**	**COMB** **0.51264**
auto93	SAbias-p 0.34463	**CNK-p** **0.38063**	**CNK-a** **0.3845**	**COMB** **0.2757**	CNK-p -0.16151	**CNK-a** **0.23312**	**CNK-p** **0.28225**	COMB -0.01647
autohorse	COMB 0.29635	**COMB** **0.43584**	SAbias-p -0.00244	SAbias-p -0.0495	**COMB** **0.1901**	COMB 0.08921	**CNK-a** **0.18246**	**COMB** **0.45075**
baskball	SAbias-p 0.43344	SAbias-p 0.13391	CNK-a -0.06495	SAbias-p -0.00488	SAbias-p -0.10536	SAbias-p -0.02788	CNK-a 0.03434	COMB -0.09841
bodyfat	COMB 0.28929	**COMB** **0.5126**	SAbias-p -0.00516	**COMB** **0.19186**	**COMB** **0.48134**	COMB 0.08983	**COMB** **0.13887**	SAbias-p -0.00164
brainsize	COMB -0.44863	SAbias-p -0.17131	CNK-p 0.01613	CNK-p -0.18022	SAbias-p -0.32845	SAbias-p 0.22928	CNK-p -0.262	SAbias-p -0.17131
breasttumor	COMB -0.05201	SAbias-p 0.06077	SAbias-p -0.01371	SAbias-p 0.11442	COMB 0.00712	CNK-a -0.03975	COMB -0.04857	CNK-a 0.0093
cloud	SAbias-p 0.49279	**COMB** **0.22807**	**COMB** **0.2519**	**COMB** **0.38232**	**SAbias-p** **0.36586**	**COMB** **0.30231**	**CNK-a** **0.35965**	**COMB** **0.23066**
cpu	CNK-a 0.67851	**CNK-a** **0.75279**	**SAvar** **0.60414**	SAbias-p 0.00957	**COMB** **0.70706**	**COMB** **0.3752**	**COMB** **0.53218**	**COMB** **0.61502**
diabetes	COMB 0.09126	CNK-a -0.08637	COMB -0.02583	SAbias-p 0.127	SAbias-p -0.27488	SAbias-p 0.01604	SAbias-p 0.17113	**DENS** **0.41939**
echomonths	COMB 0.31377	SAbias-p 0.00093	COMB 0.0842	CNK-a 0.15484	**CNK-a** **0.18781**	SAbias-p -0.09502	SAvar 0.00061	COMB 0.05648
elusage	DENS 0.24812	CNK-a -0.03941	**COMB** **0.36236**	SAbias-p 0.12436	DENS 0.26141	SAbias-p -0.3676	**CNK-a** **0.31757**	SAbias-p -0.18652
fishcatch	BAGV 0.77195	SAvar 0.13012	**BAGV** **0.43351**	**COMB** **0.8209**	**DENS** **0.36597**	**CNK-a** **0.64588**	SAbias-p -0.05937	**COMB** **0.41717**
fruitfly	SAbias-p 0.21937	COMB -0.0476	COMB -0.03753	COMB -0.08393	SAbias-p 0.10149	COMB 0.09136	CNK-a 0.06393	SAbias-p 0.04472
grv	SAbias-p 0.41169	COMB 0.11181	COMB 0.1603	**SAvar** **0.19259**	COMB 0.01965	SAbias-p 0.0994	COMB 0.07699	COMB 0.11085
hungarian	COMB 0.47015	SAbias-p -0.06559	**COMB** **0.55259**	**COMB** **0.4058**	**COMB** **0.40111**	**CNK-a** **0.3701**	**CNK-a** **0.40298**	SAbias-p -0.06559
lowbwt	COMB 0.09121	COMB 0.03933	COMB 0.02354	**BAGV** **0.27622**	BAGV 0.09503	COMB 0.07618	COMB 0.12192	CNK-a 0.05212
mbagrade	CNK-a 0.05769	SAbias-p -0.19564	SAbias-p -0.00057	COMB -0.08325	COMB 0.13271	SAbias-p -0.0487	CNK-p 0.08267	COMB 0.01785
pharynx	BAGV 0.45637	**CNK-a** **0.26137**	COMB 0.09912	**BAGV** **0.30932**	COMB 0.05888	**COMB** **0.20256**	SAbias-p 0.09497	**CNK-a** **0.26137**
pollution	CNK-p 0.48263	**CNK-p** **0.48793**	COMB -0.05869	CNK-p 0.19908	CNK-p 0.07106	COMB -0.12725	SAbias-p -0.09445	**CNK-a** **0.41107**
pwlinear	COMB 0.28619	**COMB** **0.23355**	**COMB** **0.21469**	**COMB** **0.13996**	**COMB** **0.16323**	SAbias-p -0.02374	COMB 0.12567	**COMB** **0.21986**
pyrim	CNK-p 0.33106	**CNK-p** **0.79437**	**CNK-p** **0.52189**	**CNK-p** **0.22991**	CNK-p -0.17277	CNK-p 0.22602	COMB 0.04306	**CNK-p** **0.79437**
servo	COMB 0.53662	**COMB** **0.32036**	CNK-p -0.08082	CNK-a 0.14795	CNK-a -0.02488	SAbias-p -0.40191	SAbias-p -0.11768	SAbias-p 0.09471
sleep	SAbias-p 0.50901	**CNK-p** **0.41196**	**SAbias-p** **0.3575**	CNK-p 0.1861	COMB 0.16659	COMB 0.21568	CNK-p 0.03716	SAbias-p 0.05824
transplant	SAbias-p 0.13492	**CNK-a** **0.38692**	SAbias-p -0.20258	SAbias-p 0.16487	**CNK-a** **0.49518**	COMB 0.07478	**SAbias-p** **0.25701**	**SAbias-p** **0.22242**
triazines	CNK-a 0.31322	SAbias-p -0.01562	SAvar -0.05528	**CNK-a** **0.3013**	SAbias-p 0.04911	**CNK-p** **0.25893**	**CNK-a** **0.30786**	**CNK-p** **0.51331**
tumor	SAbias-p 0.09716	SAbias-p 0.03983	DENS -0.05626	SAbias-p 0.10812	**SAbias-p** **0.34323**	SAbias-p 0.03306	SAbias-p 0.15015	CNK-p 0.14711
wpbc	CNK-p 0.44144	COMB -0.04695	SAbias-p 0.09406	SAvar 0.07246	CNK-p 0.09372	SAbias-p 0.03245	CNK-p 0.06242	CNK-a 0.05306
+	71	46	36	43	36	25	36	43
-	4	0	4	0	0	7	0	0

Table 8. Automatically selected reliability estimates using neural networks and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	COMB 0.53427	**CNK-a** **0.41501**	**CNK-a** **0.31797**	**COMB** **0.64217**	**CNK-a** **0.21855**	**COMB** **0.38269**	**COMB** **0.45955**	**CNK-a** **0.41501**
auto93	COMB 0.37801	**CNK-p** **0.38063**	**CNK-p** **0.24603**	**COMB** **0.2757**	CNK-p -0.16151	**COMB** **0.31111**	**COMB** **0.45413**	**CNK-p** **0.38063**
autohorse	COMB 0.29635	**CNK-a** **0.35847**	**CNK-a** **0.45587**	**COMB** **0.29866**	**CNK-a** **0.18705**	COMB 0.08921	**COMB** **0.28718**	**CNK-a** **0.35847**
baskball	SAbias-p 0.43344	CNK-p 0.08535	SAbias-p -0.04537	SAbias-p -0.00488	SAbias-p -0.10536	SAbias-p -0.02788	SAbias-p 0.14108	CNK-p 0.08535
bodyfat	BAGV 0.27545	**CNK-a** **0.51776**	**CNK-a** **0.51487**	**CNK-a** **0.12371**	**CNK-a** **0.3177**	CNK-a 0.10871	CNK-a 0.02095	**CNK-a** **0.51776**
brainsize	CNK-p -0.17479	CNK-p 0.32309	CNK-p 0.01613	SAbias-p 0.18579	CNK-p -0.09579	SAbias-p 0.22928	CNK-p -0.262	CNK-p 0.32309
breasttumor	SAbias-p 0.15076	SAbias-p 0.06077	SAbias-p -0.01371	SAbias-p 0.11442	**SAbias-p** **0.27749**	SAbias-p 0.08328	SAbias-p 0.1012	SAbias-p 0.06077
cloud	COMB 0.5023	**CNK-a** **0.21216**	**CNK-a** **0.23988**	**COMB** **0.38232**	**CNK-a** **0.55244**	**COMB** **0.30231**	**COMB** **0.38733**	**CNK-a** **0.21216**
cpu	CNK-a 0.67851	**CNK-a** **0.75279**	**CNK-a** **0.75413**	**CNK-a** **0.533**	**CNK-a** **0.76226**	**CNK-a** **0.47194**	**CNK-a** **0.65552**	**CNK-a** **0.75279**
diabetes	DENS 0.37506	SAbias-p -0.03958	SAbias-p 0.09376	SAbias-p 0.127	SAbias-p -0.27488	SAbias-p 0.01604	SAbias-p 0.17113	SAbias-p -0.03958
echomonths	COMB 0.31377	CNK-a 0.13434	CNK-a 0.119	**COMB** **0.19676**	**CNK-a** **0.18781**	**COMB** **0.21363**	**COMB** **0.2366**	CNK-a 0.13434
elusage	COMB 0.38069	CNK-p 0.07081	SAbias-p -0.11094	**COMB** **0.32841**	**SAbias-p** **0.6628**	**COMB** **0.34495**	**COMB** **0.39937**	CNK-p 0.07081
fishcatch	COMB 0.77291	**CNK-a** **0.4976**	**CNK-a** **0.50185**	**COMB** **0.8209**	**CNK-a** **0.26217**	**COMB** **0.70069**	**COMB** **0.55138**	**CNK-a** **0.4976**
fruitfly	SAbias-p 0.21937	SAbias-p 0.04472	SAbias-p -0.03883	SAbias-p 0.07756	SAbias-p 0.10149	SAbias-p -0.05633	**SAbias-p** **0.24583**	SAbias-p 0.04472
grv	CNK-a 0.30343	CNK-a 0.10838	CNK-a 0.15813	**CNK-a** **0.22924**	CNK-a 0.01442	CNK-a 0.00533	CNK-a 0.06836	CNK-a 0.10838
hungarian	SAbias-p 0.1504	**CNK-a** **0.3247**	**SAbias-p** **0.27161**	**COMB** **0.4058**	**CNK-a** **0.36423**	**COMB** **0.40052**	**BAGV** **0.58801**	**CNK-a** **0.3247**
lowbwt	COMB 0.09121	SAbias-p 0.04871	BAGV 0.02265	**COMB** **0.27682**	SAbias-p 0.00218	COMB 0.07618	COMB 0.12192	BAGV 0.01035
mbagrade	SAbias-p 0.26329	SAbias-p -0.19564	SAbias-p -0.00057	SAbias-p 0.10203	<u>SAbias-p</u> <u>-0.26381</u>	SAbias-p -0.0487	**SAbias-p** **0.28305**	SAbias-p -0.19564
pharynx	BAGV 0.45637	**COMB** **0.21162**	COMB 0.09912	**COMB** **0.31065**	COMB 0.05888	**COMB** **0.20256**	**BAGV** **0.22485**	**CNK-a** **0.26137**
pollution	CNK-p 0.48263	**CNK-p** **0.48793**	CNK-p 0.18868	CNK-p 0.19908	CNK-p 0.07106	CNK-p 0.05327	CNK-p 0.09734	**CNK-p** **0.48793**
pwlinear	COMB 0.28619	**COMB** **0.23355**	**COMB** **0.21469**	BAGV 0.09683	**COMB** **0.16323**	**BAGV** **0.3056**	BAGV 0.06329	**COMB** **0.21986**
pyrim	CNK-p 0.33106	**CNK-p** **0.79437**	**CNK-p** **0.52189**	**CNK-p** **0.22991**	CNK-p -0.17277	CNK-p 0.22602	CNK-p 0.01688	**CNK-p** **0.79437**
servo	BAGV 0.63921	**BAGV** **0.27243**	**BAGV** **0.41849**	**BAGV** **0.60785**	**BAGV** **0.50845**	**BAGV** **0.76048**	**BAGV** **0.76526**	**BAGV** **0.37202**
sleep	SAbias-p 0.50901	SAbias-p 0.03144	DENS 0.03027	DENS -0.07456	**SAbias-p** **0.2814**	SAbias-p 0.20245	SAbias-p -0.00858	SAbias-p 0.05824
transplant	COMB 0.40614	**CNK-a** **0.38692**	**CNK-a** **0.35738**	**COMB** **0.52661**	**DENS** **0.47163**	COMB 0.07478	**COMB** **0.47138**	**CNK-a** **0.38692**
triazines	CNK-p 0.36178	**CNK-p** **0.51331**	**CNK-p** **0.27695**	CNK-p -0.0426	CNK-p 0.06417	**CNK-p** **0.25893**	CNK-p -0.12382	**CNK-p** **0.51331**
tumor	SAbias-p 0.09716	SAbias-p 0.03983	CNK-p 0.15827	SAbias-p 0.10812	**SAbias-p** **0.34323**	SAbias-p 0.03306	SAbias-p 0.15015	SAbias-p 0.03983
wpbc	CNK-p 0.44144	CNK-a 0.05306	CNK-a 0.01941	CNK-p 0.07364	CNK-p 0.09372	CNK-a 0.11089	CNK-p 0.06242	CNK-a 0.05306
+	89	54	46	57	54	43	50	54
-	0	0	0	0	4	0	0	0

Table 9. Automatically selected reliability estimates using support vector machines and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	COMB 0.53427	**COMB** **0.4646**	**COMB** **0.41277**	**CNK-a** **0.61069**	**COMB** **0.2672**	**CNK-a** **0.22805**	**SAvar** **0.34788**	**COMB** **0.51264**
auto93	CNK-p 0.53887	**CNK-a** **0.50544**	**SAvar** **0.23585**	**COMB** **0.2757**	CNK-a 0.15672	**COMB** **0.31111**	**CNK-p** **0.28225**	**CNK-a** **0.50544**
autohorse	CNK-a 0.45654	**SAvar** **0.56372**	**SAvar** **0.61479**	**SAvar** **0.26555**	**SAvar** **0.54906**	**SAvar** **0.47808**	**CNK-a** **0.18246**	**SAvar** **0.56372**
baskball	CNK-p 0.43243	SAbias-p 0.13391	SAbias-p -0.04537	SAbias-p -0.00488	SAbias-p -0.10536	CNK-p 0.05828	BAGV -0.07535	SAbias-p 0.13391
bodyfat	BAGV 0.27545	**CNK-a** **0.51776**	**CNK-a** **0.51487**	SAbias-p 0.05371	**BAGV** **0.52404**	COMB 0.08983	SAvar -0.08509	**COMB** **0.51451**
brainsize	SAvar 0.26246	CNK-p 0.32309	CNK-p 0.01613	SAbias-p 0.18579	CNK-p -0.09579	SAbias-p 0.22928	CNK-p -0.262	CNK-p 0.32309
breasttumor	SAbias-p 0.15076	SAbias-p 0.06077	SAbias-p -0.01371	SAbias-p 0.11442	**SAbias-p** **0.27749**	SAbias-p 0.08328	SAbias-p 0.1012	SAbias-p 0.06077
cloud	DENS 0.52852	**SAvar** **0.36624**	**SAvar** **0.43477**	**DENS** **0.5643**	**SAvar** **0.66354**	CNK-a 0.12919	**DENS** **0.55607**	**SAvar** **0.36624**
cpu	CNK-a 0.67851	**CNK-a** **0.75279**	**CNK-a** **0.75413**	**CNK-a** **0.533**	**CNK-a** **0.76226**	**CNK-a** **0.47194**	**CNK-a** **0.65552**	**CNK-a** **0.75279**
diabetes	DENS 0.37506	**DENS** **0.41939**	**DENS** **0.42516**	**DENS** **0.41375**	**DENS** **0.45087**	**DENS** **0.43888**	DENS 0.22544	**DENS** **0.41939**
echomonths	COMB 0.31377	CNK-a 0.13434	CNK-a 0.119	**COMB** **0.19676**	**CNK-a** **0.18781**	**COMB** **0.21363**	**COMB** **0.2366**	CNK-a 0.13434
elusage	COMB 0.38069	DENS 0.16725	DENS 0.2049	**COMB** **0.32841**	DENS 0.26141	**COMB** **0.34495**	**COMB** **0.39937**	DENS 0.16725
fishcatch	COMB 0.77291	**CNK-a** **0.4976**	**CNK-a** **0.50185**	**COMB** **0.8209**	**CNK-a** **0.26217**	**COMB** **0.70069**	**COMB** **0.55138**	**CNK-a** **0.4976**
fruitfly	SAbias-p 0.21937	LCV -0.09187	SAbias-p -0.03883	SAbias-p 0.07756	LCV -0.07872	SAbias-p -0.05633	**SAbias-p** **0.24583**	LCV -0.09187
grv	CNK-p 0.37882	**CNK-p** **0.28526**	**CNK-p** **0.2782**	**BAGV** **0.27524**	CNK-p 0.03671	CNK-p -0.05567	SAbias-p -0.00932	**CNK-p** **0.28526**
hungarian	BAGV 0.52527	**COMB** **0.33124**	**CNK-p** **0.46864**	**BAGV** **0.62856**	**COMB** **0.40111**	**COMB** **0.40052**	**BAGV** **0.58801**	**COMB** **0.32988**
lowbwt	COMB 0.09121	CNK-p 0.06145	DENS 0.01125	CNK-a 0.10136	DENS 0.11735	SAvar 0.03428	COMB 0.12192	CNK-p 0.06145
mbagrade	SAbias-p 0.26329	CNK-a 0.0236	CNK-a 0.02416	SAbias-p 0.10203	CNK-a 0.13414	SAbias-p -0.0487	CNK-p 0.08267	CNK-a 0.0236
pharynx	BAGV 0.45637	**COMB** **0.21162**	COMB 0.09912	**COMB** **0.31065**	CNK-a 0.10916	**COMB** **0.20256**	**COMB** **0.22839**	**CNK-a** **0.26137**
pollution	CNK-p 0.48263	**CNK-p** **0.48793**	CNK-p 0.18868	CNK-p 0.19908	CNK-p 0.07106	CNK-p 0.05327	CNK-p 0.09734	**CNK-p** **0.48793**
pwlinear	COMB 0.28619	**CNK-p** **0.31308**	**COMB** **0.21469**	SAvar 0.10202	**SAvar** **0.2871**	**BAGV** **0.3056**	SAvar 0.00045	**BAGV** **0.31258**
pyrim	BAGV 0.25469	**CNK-a** **0.84896**	**BAGV** **0.63968**	COMB 0.15901	BAGV 0.15525	**CNK-a** **0.79749**	COMB 0.04306	**CNK-a** **0.84896**
servo	BAGV 0.63921	**BAGV** **0.27243**	**BAGV** **0.41849**	**BAGV** **0.60785**	**BAGV** **0.50845**	**BAGV** **0.76048**	**BAGV** **0.76526**	**LCV** **0.50024**
sleep	SAbias-p 0.50901	**BAGV** **0.5879**	LCV 0.06176	BAGV 0.01164	**SAbias-p** **0.2814**	BAGV 0.22435	BAGV 0.2239	SAbias-p 0.05824
transplant	LCV 0.68332	**DENS** **0.48231**	**DENS** **0.47251**	**LCV** **0.73861**	**DENS** **0.47163**	BAGV 0.05901	**LCV** **0.43956**	**DENS** **0.48231**
triazines	CNK-p 0.36178	**CNK-p** **0.51331**	**CNK-p** **0.27695**	**BAGV** **0.50219**	CNK-p 0.06417	**DENS** **0.18545**	**BAGV** **0.469**	**CNK-p** **0.51331**
tumor	SAbias-p 0.09716	SAbias-p 0.03983	CNK-p 0.15827	CNK-p 0.17772	**CNK-p** **0.23795**	BAGV -0.02141	CNK-p 0.06184	CNK-p 0.14711
wpbc	SAbias-p 0.49432	CNK-a 0.05306	CNK-a 0.01941	BAGV 0.02669	SAbias-a 0.04018	CNK-a 0.11089	SAbias-p 0.07376	CNK-a 0.05306
+	89	64	54	54	54	50	50	61
-	0	0	0	0	0	0	0	0

Table 10. Automatically selected reliability estimates using random forests and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	COMB 0.53427	**CNK-a** **0.41501**	**SAvar** **0.37431**	**COMB** **0.64217**	**SAvar** **0.51785**	**COMB** **0.38269**	**COMB** **0.45955**	**CNK-a** **0.41501**
auto93	BAGV 0.35035	**CNK-p** **0.38063**	**CNK-p** **0.24603**	**COMB** **0.2757**	CNK-p -0.16151	**SAvar** **0.2901**	**BAGV** **0.38612**	**CNK-p** **0.38063**
autohorse	COMB 0.29635	**CNK-a** **0.35847**	**CNK-a** **0.45587**	**COMB** **0.29866**	**SAvar** **0.54906**	**SAvar** **0.47808**	**COMB** **0.28718**	**CNK-a** **0.35847**
baskball	SAbias-p 0.43344	DENS 0.09129	SAbias-a -0.08005	DENS 0.06929	SAbias-a -0.05618	SAbias-a -0.09244	DENS 0.08774	DENS 0.09129
bodyfat	COMB 0.28929	**CNK-a** **0.51776**	**SAvar** **0.20086**	**COMB** **0.19186**	**SAvar** **0.76628**	SAvar 0.08167	**COMB** **0.13887**	**CNK-a** **0.51776**
brainsize	CNK-p -0.17479	CNK-p 0.32309	CNK-p 0.01613	CNK-p -0.18022	CNK-p -0.09579	SAbias-p 0.22928	CNK-p -0.262	CNK-p 0.32309
breasttumor	SAbias-p 0.15076	CNK-a 0.0093	CNK-a -0.03471	SAbias-p 0.11442	CNK-a -0.07793	SAbias-p 0.08328	SAbias-p 0.1012	CNK-a 0.0093
cloud	COMB 0.5023	**CNK-a** **0.21216**	**COMB** **0.2519**	**COMB** **0.38232**	**SAvar** **0.66354**	**COMB** **0.30231**	**COMB** **0.38733**	**CNK-a** **0.21216**
cpu	CNK-a 0.67851	**CNK-a** **0.75279**	**COMB** **0.63352**	**COMB** **0.78217**	**SAvar** **0.71583**	**CNK-a** **0.47194**	**COMB** **0.53218**	**CNK-a** **0.75279**
diabetes	SAbias-p 0.3895	**DENS** **0.41939**	**DENS** **0.42516**	**DENS** **0.41375**	LCV 0.15192	SAbias-p 0.01604	DENS 0.22544	**DENS** **0.41939**
echomonths	COMB 0.31377	CNK-a 0.13434	COMB 0.0842	**COMB** **0.19676**	SAbias-a 0.12709	**COMB** **0.21363**	**COMB** **0.2366**	CNK-a 0.13434
elusage	SAbias-p 0.42909	CNK-p 0.07081	DENS 0.2049	**DENS** **0.30584**	**LCV** **0.28993**	DENS 0.12981	DENS 0.15487	CNK-p 0.07081
fishcatch	COMB 0.77291	**CNK-a** **0.4976**	SAvar 0.11476	**COMB** **0.8209**	**SAvar** **0.63048**	**COMB** **0.70069**	**COMB** **0.55138**	**CNK-a** **0.4976**
fruitfly	SAbias-p 0.21937	SAbias-a -0.06012	CNK-a -0.00374	SAbias-p 0.07756	SAbias-a -0.00825	CNK-a 0.10714	CNK-a 0.06393	SAbias-a -0.06012
grv	CNK-a 0.30343	CNK-a 0.10838	CNK-a 0.15813	**COMB** **0.24211**	SAvar 0.13567	CNK-a 0.00533	COMB 0.07699	CNK-a 0.10838
hungarian	SAbias-p 0.1504	**CNK-a** **0.3247**	**SAbias-a** **0.21559**	**COMB** **0.4058**	**SAvar** **0.31883**	**COMB** **0.40052**	**COMB** **0.43677**	**CNK-a** **0.3247**
lowbwt	COMB 0.09121	SAbias-a -0.12723	COMB 0.02354	**COMB** **0.27682**	**SAvar** **0.20521**	COMB 0.07618	COMB 0.12192	COMB 0.01127
mbagrade	SAbias-p 0.26329	DENS -0.07807	DENS -0.07708	DENS -0.04427	LCV 0.01213	DENS -0.15127	DENS -0.04527	DENS -0.07807
pharynx	SAbias-p 0.33255	SAbias-a 0.11214	SAbias-a -0.00703	**COMB** **0.31065**	SAvar 0.12474	**COMB** **0.20256**	**COMB** **0.22839**	SAbias-a 0.11214
pollution	CNK-p 0.48263	**CNK-p** **0.48793**	CNK-p 0.18868	CNK-p 0.19908	SAvar 0.13898	CNK-p 0.05327	BAGV -0.04173	**CNK-p** **0.48793**
pwlinear	CNK-p 0.24613	**SAvar** **0.22893**	**COMB** **0.21469**	SAvar 0.10202	**SAvar** **0.2871**	SAbias-a -0.17757	SAvar 0.00045	**SAvar** **0.22893**
pyrim	CNK-p 0.33106	**CNK-p** **0.79437**	**CNK-p** **0.52189**	**CNK-p** **0.22991**	CNK-p -0.17277	CNK-p 0.22602	CNK-p 0.01688	**CNK-p** **0.79437**
servo	SAbias-p 0.57877	**COMB** **0.32036**	**COMB** **0.3202**	SAbias-p 0.01055	SAvar 0.10257	**COMB** **0.64616**	**BAGV** **0.76526**	**LCV** **0.50024**
sleep	SAbias-p 0.50901	DENS 0.12541	DENS 0.03027	DENS -0.07456	DENS -0.00022	DENS -0.15156	DENS -0.18518	DENS 0.12541
transplant	SAbias-p 0.13492	**LCV** **0.25598**	**DENS** **0.47251**	**DENS** **0.47425**	**LCV** **0.72511**	COMB 0.07478	**DENS** **0.50515**	**LCV** **0.25598**
triazines	CNK-p 0.36178	**CNK-p** **0.51331**	**CNK-p** **0.27695**	CNK-p -0.0426	CNK-p 0.06417	**CNK-p** **0.25893**	CNK-p -0.12382	**CNK-p** **0.51331**
tumor	SAbias-p 0.09716	DENS -0.05673	SAbias-a 0.15898	DENS -0.12207	SAbias-a 0.1029	DENS -0.20433	DENS -0.11546	DENS -0.05673
wpbc	CNK-p 0.44144	CNK-a 0.05306	SAbias-a 0.08231	BAGV 0.02669	SAbias-a 0.04018	SAbias-a -0.04367	BAGV 0.09578	CNK-a 0.05306
+	86	54	46	57	39	39	43	54
-	0	0	0	0	0	4	0	0

Table 11. Automatically selected reliability estimates using naive Bayes and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	SAvar 0.51754	**COMB** **0.4646**	**COMB** **0.41277**	**CNK-a** **0.61069**	**COMB** **0.2672**	**CNK-a** **0.22805**	**COMB** **0.45955**	**COMB** **0.51264**
auto93	LCV 0.29341	**CNK-a** **0.50544**	**SAvar** **0.23585**	**LCV** **0.30294**	**COMB** **0.31353**	CNK-p -0.01816	**CNK-p** **0.28225**	**SAvar** **0.25103**
autohorse	SAvar 0.27493	**SAvar** **0.56372**	**SAvar** **0.61479**	**SAvar** **0.26555**	**SAvar** **0.54906**	CNK-a 0.07699	**CNK-a** **0.18246**	**SAvar** **0.56372**
baskball	CNK-p 0.43243	SAbias-p 0.13391	SAbias-p -0.04537	CNK-p 0.10494	SAbias-p -0.10536	CNK-p 0.05828	DENS 0.08774	BAGV -0.00144
bodyfat	BAGV 0.27545	BAGV 0.08218	**CNK-a** **0.51487**	SAbias-p 0.05371	**BAGV** **0.52404**	BAGV 0.04399	**BAGV** **0.41533**	**CNK-a** **0.51776**
brainsize	SAvar 0.26246	BAGV -0.04515	CNK-p 0.01613	BAGV -0.27543	SAbias-p -0.32845	**CNK-p** **0.47202**	SAvar -0.13015	BAGV 0.16979
breasttumor	SAbias-p 0.15076	SAbias-p 0.06077	SAbias-p -0.01371	SAbias-p 0.11442	**SAbias-p** **0.27749**	SAbias-p 0.08328	SAbias-p 0.1012	SAbias-p 0.06077
cloud	DENS 0.52852	**SAvar** **0.36624**	**SAvar** **0.43477**	**DENS** **0.5643**	**SAvar** **0.66354**	**DENS** **0.5866**	**DENS** **0.55607**	**SAvar** **0.36624**
cpu	CNK-a 0.67851	**SAvar** **0.59386**	**CNK-a** **0.75413**	**CNK-a** **0.533**	**CNK-a** **0.76226**	**CNK-a** **0.47194**	**SAvar** **0.1912**	**CNK-a** **0.75279**
diabetes	DENS 0.37506	**DENS** **0.41939**	**DENS** **0.42516**	**DENS** **0.41375**	**DENS** **0.45087**	**DENS** **0.43888**	DENS 0.22544	**DENS** **0.41939**
echomonths	COMB 0.31377	CNK-a 0.13434	CNK-a 0.119	**COMB** **0.19676**	**CNK-a** **0.18781**	**COMB** **0.21363**	**COMB** **0.2366**	CNK-a 0.13434
elusage	COMB 0.38069	SAbias-p -0.19683	DENS 0.2049	SAbias-p 0.12436	DENS 0.26141	**COMB** **0.34495**	**COMB** **0.39937**	SAbias-p -0.18652
fishcatch	COMB 0.77291	**COMB** **0.43973**	**CNK-a** **0.50185**	**COMB** **0.8209**	**CNK-a** **0.26217**	**COMB** **0.70069**	**COMB** **0.55138**	**COMB** **0.41717**
fruitfly	SAbias-p 0.21937	SAbias-a -0.06012	LCV -0.11485	SAbias-p 0.07756	LCV -0.07872	SAbias-p -0.05633	**SAbias-p** **0.24583**	SAbias-p 0.04472
grv	SAbias-p 0.41169	SAbias-p 0.07664	**CNK-p** **0.2782**	**BAGV** **0.27524**	CNK-p 0.03671	CNK-p -0.05567	LCV 0.15605	BAGV 0.16258
hungarian	BAGV 0.52527	**BAGV** **0.24964**	BAGV -0.03189	**BAGV** **0.62856**	**COMB** **0.40111**	**BAGV** **0.41542**	**BAGV** **0.58801**	**BAGV** **0.2101**
lowbwt	COMB 0.09121	CNK-p 0.06145	SAvar -0.00047	SAbias-p 0.09246	DENS 0.11735	SAvar 0.03428	SAbias-p -0.02728	COMB 0.01127
mbagrade	SAbias-p 0.26329	CNK-a 0.0236	CNK-a 0.02416	BAGV -0.1082	CNK-a 0.13414	SAbias-p -0.0487	CNK-p 0.08267	CNK-a 0.0236
pharynx	COMB 0.45717	**COMB** **0.21162**	CNK-a -0.086	**COMB** **0.31065**	COMB 0.05888	**COMB** **0.20256**	**COMB** **0.22839**	**CNK-a** **0.26137**
pollution	CNK-p 0.48263	**CNK-p** **0.48793**	CNK-p 0.18868	CNK-p 0.19908	CNK-p 0.07106	CNK-p 0.05327	CNK-p 0.09734	**CNK-p** **0.48793**
pwlinear	COMB 0.28619	**COMB** **0.23355**	**COMB** **0.21469**	SAvar 0.10202	**COMB** **0.16323**	**BAGV** **0.3056**	SAvar 0.00045	**COMB** **0.21986**
pyrim	BAGV 0.25469	COMB -0.07027	**CNK-a** **0.63602**	CNK-a 0.15275	BAGV 0.15525	**BAGV** **0.70454**	CNK-a 0.03664	**CNK-p** **0.79437**
servo	BAGV 0.63921	**BAGV** **0.27243**	**BAGV** **0.41849**	**BAGV** **0.60785**	**BAGV** **0.50845**	**BAGV** **0.76048**	**BAGV** **0.76526**	**LCV** **0.50024**
sleep	BAGV 0.06549	**BAGV** **0.5879**	LCV 0.06176	**LCV** **0.38161**	**LCV** **0.2861**	BAGV 0.22435	BAGV 0.2239	**BAGV** **0.68534**
transplant	LCV 0.68332	**COMB** **0.45337**	**DENS** **0.47251**	**LCV** **0.73861**	**DENS** **0.47163**	BAGV 0.05901	**DENS** **0.50515**	**DENS** **0.48231**
triazines	BAGV 0.54746	**CNK-p** **0.51331**	**CNK-p** **0.27695**	**BAGV** **0.50219**	BAGV 0.07731	BAGV 0.05313	**BAGV** **0.469**	**CNK-p** **0.51331**
tumor	BAGV 0.19543	SAbias-p 0.03983	CNK-p 0.15827	SAbias-p 0.10812	**BAGV** **0.34204**	BAGV -0.02141	CNK-p 0.06184	BAGV 0.01621
wpbc	SAbias-p 0.49432	SAbias-p 0.02251	SAbias-p 0.09406	SAbias-a -0.05979	SAbias-a 0.04018	SAbias-a -0.04367	SAbias-p 0.07376	SAbias-p 0.02251
+	86	54	50	54	57	46	54	61
-	0	0	0	0	0	0	0	0

Table 12. Automatically selected reliability estimates using k-nearest neighbors and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	COMB 0.53427	**CNK-a** **0.41501**	**COMB** **0.41277**	**CNK-a** **0.61069**	**COMB** **0.2672**	**SAvar** **0.52299**	**SAvar** **0.34788**	**CNK-a** **0.41501**
auto93	BAGV 0.35035	**CNK-p** **0.38063**	**SAvar** **0.23585**	DENS 0.07593	**COMB** **0.31353**	**COMB** **0.31111**	**COMB** **0.45413**	**CNK-p** **0.38063**
autohorse	CNK-a 0.45654	**SAvar** **0.56372**	**COMB** **0.59094**	**CNK-a** **0.32319**	**SAvar** **0.54906**	**SAvar** **0.47808**	**CNK-a** **0.18246**	**CNK-a** **0.35847**
baskball	CNK-p 0.43243	COMB -0.09685	SAbias-p -0.04537	BAGV -0.05046	SAbias-p -0.10536	BAGV 0.04158	BAGV -0.07535	BAGV -0.00144
bodyfat	BAGV 0.27545	**CNK-a** **0.51776**	**CNK-a** **0.51487**	**BAGV** **0.32678**	**BAGV** **0.52404**	SAvar 0.08167	**BAGV** **0.41533**	**CNK-a** **0.51776**
brainsize	BAGV -0.12717	CNK-p 0.32309	CNK-p 0.01613	SAbias-p 0.18579	CNK-p -0.09579	**CNK-p** **0.47202**	SAbias-p 0.10981	CNK-p 0.32309
breasttumor	SAbias-p 0.15076	SAbias-p 0.06077	SAbias-p -0.01371	SAbias-p 0.11442	**SAbias-p** **0.27749**	SAbias-p 0.08328	SAbias-p 0.1012	SAbias-p 0.06077
cloud	DENS 0.52852	**SAvar** **0.36624**	**SAvar** **0.43477**	**BAGV** **0.45077**	**SAvar** **0.66354**	CNK-a 0.12919	**DENS** **0.55607**	**COMB** **0.23066**
cpu	BAGV 0.77521	**CNK-a** **0.75279**	**CNK-a** **0.75413**	**CNK-a** **0.533**	**COMB** **0.70706**	**CNK-a** **0.47194**	**CNK-a** **0.65552**	**CNK-a** **0.75279**
diabetes	DENS 0.37506	**DENS** **0.41939**	CNK-p 0.22557	**DENS** **0.41375**	SAbias-p -0.27488	**DENS** **0.43888**	DENS 0.22544	**DENS** **0.41939**
echomonths	BAGV 0.27798	CNK-a 0.13434	COMB 0.0842	SAbias-p 0.04743	**COMB** **0.17292**	CNK-a 0.05641	**BAGV** **0.18402**	CNK-a 0.13434
elusage	DENS 0.24812	SAbias-p -0.19683	SAbias-p -0.11094	**COMB** **0.32841**	DENS 0.26141	SAbias-p -0.3676	**COMB** **0.39937**	DENS 0.16725
fishcatch	COMB 0.77291	**COMB** **0.43973**	**CNK-a** **0.50185**	**COMB** **0.8209**	**SAvar** **0.63048**	**COMB** **0.70069**	**CNK-a** **0.32423**	**COMB** **0.41717**
fruitfly	SAbias-p 0.21937	SAbias-p 0.04472	CNK-p -0.18261	SAbias-p 0.07756	SAbias-p 0.10149	SAbias-p -0.05633	**SAbias-p** **0.24583**	SAbias-p 0.04472
grv	COMB 0.32729	CNK-a 0.10838	BAGV 0.14897	**BAGV** **0.27524**	COMB 0.01965	SAvar 0.07175	CNK-p 0.01439	**CNK-p** **0.28526**
hungarian	BAGV 0.52527	**COMB** **0.33124**	BAGV -0.03189	**BAGV** **0.62856**	**SAvar** **0.31883**	**COMB** **0.40052**	**BAGV** **0.58801**	**COMB** **0.32988**
lowbwt	COMB 0.09121	BAGV 0.03868	DENS 0.01125	**COMB** **0.27682**	DENS 0.11735	COMB 0.07618	COMB 0.12192	BAGV 0.01035
mbagrade	SAbias-p 0.26329	CNK-a 0.0236	CNK-a 0.02416	SAbias-p 0.10203	SAbias-p -0.26381	SAbias-p -0.0487	**BAGV** **0.26084**	CNK-a 0.0236
pharynx	BAGV 0.45637	DENS 0.09084	SAbias-a -0.00703	**COMB** **0.31065**	SAvar 0.12474	**COMB** **0.20256**	**COMB** **0.22839**	**CNK-a** **0.26137**
pollution	CNK-p 0.48263	**CNK-a** **0.41107**	SAbias-p -0.05924	CNK-p 0.19908	CNK-p 0.07106	BAGV -0.12953	CNK-p 0.09734	**CNK-p** **0.48793**
pwlinear	COMB 0.28619	**COMB** **0.23355**	**COMB** **0.21469**	SAvar 0.10202	**COMB** **0.16323**	**COMB** **0.30561**	BAGV 0.06329	**BAGV** **0.31258**
pyrim	CNK-p 0.33106	**CNK-p** **0.79437**	**CNK-p** **0.52189**	**CNK-p** **0.22991**	**CNK-a** **0.40083**	**COMB** **0.79151**	COMB 0.04306	**CNK-p** **0.79437**
servo	BAGV 0.63921	**BAGV** **0.27243**	**BAGV** **0.41849**	**BAGV** **0.60785**	**BAGV** **0.50845**	**BAGV** **0.76048**	**BAGV** **0.76526**	**LCV** **0.50024**
sleep	SAbias-p 0.50901	**BAGV** **0.5879**	**COMB** **0.36025**	BAGV 0.01164	COMB 0.16659	LCV 0.22182	BAGV 0.2239	**CNK-p** **0.41196**
transplant	LCV 0.68332	**COMB** **0.45337**	**CNK-a** **0.35738**	**COMB** **0.52661**	**LCV** **0.72511**	BAGV 0.05901	**DENS** **0.50515**	**BAGV** **0.39158**
triazines	CNK-p 0.36178	**CNK-p** **0.51331**	**LCV** **0.14698**	**BAGV** **0.50219**	BAGV 0.07731	CNK-a 0.07119	**SAvar** **0.30451**	**CNK-p** **0.51331**
tumor	SAbias-p 0.09716	DENS -0.05673	CNK-p 0.15827	SAbias-p 0.10812	**BAGV** **0.34204**	BAGV -0.02141	DENS -0.11546	SAbias-p 0.03983
wpbc	SAbias-p 0.49432	**CNK-p** **0.21094**	CNK-a 0.01941	CNK-p 0.07364	SAbias-a 0.04018	SAbias-p 0.03245	CNK-p 0.06242	CNK-a 0.05306
+	86	61	46	57	54	43	57	64
-	0	0	4	0	4	4	0	0

Table 13. Automatically selected reliability estimates using boosting and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	PCA 0.32323	**PCA 0.2297**	**PCA 0.24616**	**PCA 0.47914**	PCA 0.1464	**PCA 0.25579**	**PCA 0.253**	**PCA 0.29909**
auto93	PCA 0.18393	PCA 0.01924	**PCA 0.37698**	PCA 0.05966	PCA 0.12719	**PCA 0.22221**	PCA 0.1868	PCA -0.02502
autohorse	PCA 0.14174	**PCA 0.36377**	**PCA 0.4888**	PCA 0.06031	PCA 0.07451	PCA -0.03092	PCA 0.13721	**PCA 0.37258**
baskball	PCA -0.03819	**PCA 0.2345**	PCA 0.1418	PCA -0.0379	PCA 0.03674	PCA 0.05653	PCA 0.09156	**PCA 0.2345**
bodyfat	PCA 0.04607	PCA -0.02766	**PCA 0.24654**	PCA 0.10024	**PCA 0.28885**	PCA 0.02023	**PCA 0.15177**	PCA -0.02766
brainsize	PCA -0.37625	PCA 0.14684	PCA -0.33442	PCA -0.31169	PCA -0.26053	PCA -0.37434	PCA -0.45158	PCA 0.16108
breasttumor	PCA -0.00763	PCA 0.08429	PCA 0.02719	PCA 0.11486	PCA -0.01884	PCA -0.022	PCA -0.03583	PCA 0.08283
cloud	PCA 0.3015	PCA 0.10491	PCA 0.14718	**PCA 0.24125**	**PCA 0.5536**	**PCA 0.62182**	**PCA 0.23222**	PCA 0.10491
cpu	PCA 0.72934	**PCA 0.56457**	**PCA 0.57404**	**PCA 0.78981**	**PCA 0.6666**	**PCA 0.34885**	**PCA 0.48939**	**PCA 0.55358**
diabetes	PCA 0.05374	PCA -0.16908	PCA 0.18463	PCA 0.07739	PCA 0.20542	PCA 0.15098	PCA 0.00341	PCA -0.16908
echomonths	PCA -0.0743	PCA 0.06792	PCA 0.16061	PCA -0.1777	PCA 0.13465	PCA 0.10015	PCA -0.05203	PCA 0.06792
elusage	PCA -0.18828	PCA -0.31634	PCA 0.07337	PCA -0.22995	PCA -0.01466	PCA -0.10913	PCA 0.04039	PCA -0.36204
fishcatch	PCA 0.68395	**PCA 0.32609**	**PCA 0.3657**	**PCA 0.75825**	**PCA 0.68521**	**PCA 0.54452**	**PCA 0.46551**	**PCA 0.28913**
fruitfly	PCA 0.03479	PCA -0.09504	PCA -0.11581	PCA -0.00631	PCA -0.02138	PCA 0.04069	PCA -0.10438	PCA -0.09548
grv	PCA 0.23306	PCA 0.17693	PCA 0.11375	**PCA 0.28562**	**PCA 0.22276**	PCA 0.14099	PCA 0.16671	PCA 0.17693
hungarian	PCA 0.28527	PCA 0.01911	PCA -0.11647	**PCA 0.11492**	**PCA 0.26484**	**PCA 0.16161**	PCA 0.03573	PCA 0.01911
lowbwt	PCA -0.0589	PCA 0.03757	PCA 0.02085	**PCA 0.18361**	PCA 0.02739	PCA 0.07206	PCA 0.01627	PCA -0.03565
mbagrade	PCA -0.00368	PCA 0.0367	PCA 0.03698	PCA -0.22834	PCA 0.01021	PCA 0.09926	PCA 0.105	PCA 0.03669
pharynx	PCA 0.28283	**PCA 0.14246**	PCA 0.01046	**PCA 0.21967**	PCA -0.08259	**PCA 0.15325**	PCA 0.09263	**PCA 0.14246**
pollution	PCA 0.1718	**PCA 0.37259**	PCA -0.11034	PCA -0.12019	PCA -0.07123	PCA 0.02648	PCA -0.13783	**PCA 0.36991**
pwlinear	PCA 0.12846	PCA -0.05503	PCA -0.01514	PCA 0.09484	PCA 0.07888	PCA -0.01246	PCA 0.06027	PCA -0.05501
pyrim	PCA 0.19217	PCA -0.07046	**PCA 0.50306**	PCA -0.09829	**PCA 0.41449**	**PCA 0.41906**	PCA 0.17435	**PCA 0.48936**
servo	PCA 0.34256	**PCA 0.52153**	**PCA 0.5232**	PCA -0.04677	**PCA 0.77466**	PCA -0.192	PCA -0.23252	**PCA 0.52162**
sleep	PCA 0.24298	PCA -0.0343	**PCA 0.35708**	**PCA 0.3782**	**PCA 0.28203**	PCA 0.13556	**PCA 0.32989**	PCA -0.03675
transplant	PCA 0.31881	**PCA 0.32631**	**PCA 0.58078**	**PCA 0.74077**	**PCA 0.70935**	PCA 0.02162	**PCA 0.3485**	**PCA 0.31022**
triazines	PCA 0.12335	**PCA 0.24804**	PCA -0.12485	**PCA 0.24834**	**PCA 0.27078**	**PCA 0.23093**	**PCA 0.16237**	**PCA 0.1985**
tumor	PCA 0.10099	PCA 0.04189	PCA 0.12202	**PCA 0.21756**	**PCA 0.24969**	PCA 0.04519	**PCA 0.27752**	PCA 0.04189
wpbc	PCA -0.02167	PCA 0.06794	PCA 0.05665	PCA -0.23585	PCA 0.12449	PCA -0.01614	PCA -0.06588	PCA 0.06792
+	36	36	36	43	43	32	32	39
-	0	4	4	7	0	4	7	4

Table 14. Automatically selected reliability estimates using principal component analysis (with absolute values) and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

	rt	lr	nn	bag50	svm	lwr	rf	gam
auto_price	PCA -0.50898	**PCA 0.22268**	**PCA 0.21828**	PCA -0.16303	PCA 0.122	PCA 0.01762	**PCA 0.20295**	**PCA 0.22268**
auto93	PCA -0.53241	**PCA 0.38063**	**PCA 0.2463**	PCA -0.28741	PCA -0.15108	PCA 0.0181	PCA -0.28141	**PCA 0.38063**
autohorse	PCA -0.29922	**PCA 0.18638**	PCA 0.08918	PCA -0.10062	PCA -0.03448	PCA 0.09294	PCA 0.03606	**PCA 0.18638**
baskball	PCA -0.45164	PCA 0.08535	PCA -0.0739	PCA -0.10456	PCA 0.09736	PCA 0.05778	PCA -0.20753	PCA 0.08535
bodyfat	PCA -0.30493	PCA -0.18402	PCA -0.15866	PCA -0.13878	PCA -0.20669	**PCA 0.48262**	**PCA 0.26518**	PCA -0.18402
brainsize	PCA 0.24307	PCA -0.32309	PCA -0.01599	PCA 0.10224	PCA -0.0981	PCA -0.47202	PCA -0.25616	PCA -0.32309
breasttumor	PCA 0.15003	PCA 0.01123	PCA -0.0271	PCA 0.02293	**PCA 0.1371**	PCA -0.01841	PCA 0.03573	PCA 0.01123
cloud	PCA -0.26132	PCA -0.05119	PCA -0.00986	PCA 0.04897	PCA -0.36201	PCA 0.11498	PCA 0.11663	PCA -0.05119
cpu	PCA -0.47946	PCA 0.01865	PCA 0.03556	PCA -0.03715	PCA -0.35735	PCA -0.08913	**PCA 0.49216**	PCA 0.01865
diabetes	PCA 0.36166	PCA -0.2235	PCA -0.22557	PCA -0.0409	PCA 0.12933	**PCA 0.44561**	PCA 0.09901	PCA -0.2235
echomonths	PCA -0.35127	PCA -0.10731	PCA -0.09331	PCA -0.08736	PCA -0.29166	PCA -0.03723	PCA -0.1089	PCA -0.10731
elusage	PCA 0.37289	PCA -0.07081	PCA 0.08833	PCA -0.07444	PCA 0.24065	**PCA 0.43317**	PCA 0.01436	PCA -0.07081
fishcatch	PCA -0.75394	PCA -0.42173	**PCA 0.49297**	PCA -0.6585	PCA -0.00988	**PCA 0.47227**	PCA -0.14428	PCA -0.42173
fruitfly	PCA -0.03812	PCA 0.17035	**PCA 0.18262**	PCA 0.12447	PCA 0.11035	**PCA 0.24442**	**PCA 0.20257**	PCA 0.17035
grv	PCA -0.40241	PCA -0.28526	PCA -0.2782	PCA -0.14215	PCA -0.03696	PCA 0.05451	PCA -0.01406	PCA -0.28526
hungarian	PCA -0.30515	PCA -0.2342	PCA -0.47007	PCA -0.19708	PCA -0.20764	PCA -0.00708	PCA -0.12446	PCA -0.2342
lowbwt	PCA -0.03893	PCA 0.06145	**PCA 0.18535**	PCA -0.09827	PCA -0.02814	PCA -0.07722	PCA -0.07311	PCA 0.06145
mbagrade	PCA 0.23455	PCA 0.17122	PCA 0.16459	PCA -0.10544	PCA 0.18968	PCA -0.03064	PCA -0.08404	PCA 0.17122
pharynx	PCA -0.18319	PCA 0.03796	PCA 0.05509	PCA 0.02049	**PCA 0.15015**	PCA -0.00309	**PCA 0.14297**	PCA 0.03796
pollution	PCA 0.46541	PCA -0.48793	PCA -0.18876	PCA 0.19172	PCA -0.07079	PCA 0.05324	PCA 0.09235	PCA -0.48793
pwlinear	PCA -0.25703	**PCA 0.31308**	PCA 0.06359	PCA 0.04342	**PCA 0.23501**	PCA -0.27003	**PCA 0.19742**	**PCA 0.31308**
pyrim	PCA -0.32813	**PCA 0.79437**	PCA -0.52197	PCA -0.22987	PCA 0.17197	PCA 0.22614	PCA -0.01729	**PCA 0.79437**
servo	PCA 0.24199	PCA 0.00102	PCA 0.08067	PCA 0.09202	PCA -0.50651	PCA -0.26214	PCA -0.26434	PCA 0.00102
sleep	PCA -0.47429	**PCA 0.41196**	PCA -0.09175	PCA -0.19009	PCA 0.00096	PCA -0.20782	PCA -0.03696	**PCA 0.41196**
transplant	PCA -0.43091	PCA -0.23291	PCA -0.23217	**PCA 0.39263**	**PCA 0.46791**	PCA 0.07258	PCA -0.19304	PCA -0.23291
triazines	PCA -0.38581	**PCA 0.51331**	**PCA 0.27695**	PCA 0.04144	PCA 0.06432	**PCA 0.25887**	PCA 0.12346	**PCA 0.51331**
tumor	PCA -0.25953	PCA -0.14711	PCA -0.15828	PCA -0.17827	PCA -0.24027	PCA 0.02579	PCA -0.06153	PCA -0.14711
wpbc	PCA 0.45566	**PCA 0.21094**	PCA 0.09787	PCA 0.07485	PCA 0.0937	PCA -0.08164	PCA 0.06284	**PCA 0.21094**
+	21	29	21	4	14	21	21	29
-	64	21	18	21	25	11	18	21

Table 15. Automatically selected reliability estimates using principal component analysis (with signed values) and their correlation coefficients with the prediction error. Statistically significant values ($\alpha \leq 0.05$) are denoted with boldface. Significant negative values (undesired negative correlations) are additionally underlined.

Index of figures

Index of tables

References

[1] Alpaydin E., *Introduction to Machine learning*, MIT Press, 2004.

[2] Bosnić Z., Kononenko I., *Automatic Selection Of Reliability Estimates For Individual Regression Predictions*, The Knowledge Engineering Review, Vol. 25:1. Cambridge University Press, 2010, pages 27–47.

[3] Bosnić Z., Kononenko I., *Comparison of approaches for estimating reliability of individual regression predictions*, Data and Knowledge Enginnering, Volume 67, Issue 3, December 2008, pages 504-516.

[4] Bosnić Z., Kononenko I., *Estimation of individual prediction reliability using the local sensitivity analysis*, Applied Intelligence, Volume 29, no. 3, str. 187-203.

[5] Brazdil P., Giraud-Carrier C., Soares C., Vilalta R., *Meta-learning: Applications to Data Mining*, Springer-Verlag Berlin Heidelberg, 2009.

[6] Breiman L., *Bagging predictors*, Machine Learning 24, Number 2, 1996, Kluwer Academic Publishers, pages 123-140. Available from: http://www.springerlink.com/content/l4780124w2874025/

[7] Breiman L., Cutler A., *Random Forests*, Available from: http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

[8] Breiman L., Friedman J., Stone C., Olshen R.A., *Classification and Regression Trees*, Chapman and Hall/CRC, 1984.

[9] Breiman L., *Random Forests***,** Machine Learning, 45, 2001, Kluwer Academic Publishers, pages 5-32. Available from: http://www.springerlink.com/content/u0p06167n6173512/

[10] Christiannini N., Shawe-Taylor J., *Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[11] Demšar J., *Statistical Comparisons of Classifiers over Multiple Data Sets*, Journal of Machine Learning Research 7 , 2006, pages 1-30. Available from: http://jmlr.csail.mit.edu/papers/v7/demsar06a.htmlFigueiredo M., *Adaptive Sparseness for Supervised Learning*, IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 25, No. 9, September 2003, page 1.

[12] Figueiredo M., *Adaptive Sparseness for Supervised Learning*, IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 25, No. 9, September 2003, page 1.

[13] Freund Y., Schapire R., *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*, Proceedings of the Second European Conference on Computational Learning Theory, Springer-Verlag London, 1995, pages 23 – 37.

[14] Giraud-Carrier C., *Meta-learning - A Tutorial*, 2008, pages 4-25. Available from: http://www.icmla-conference.org/icmla08/slides2.pdf

[15] Hastie T.J., Tibshirani R.J., *Generalized Additive Models*, Chapman and Hall, 1990, pages 136-174.

[16] Kononenko I., *Evaluating the Quality of Attributes*, ACAI-05 summer school, IJS, Ljubljana, 2005, pages 1-14. Available from: http://carbon.videolectures.net/2005/ijs/acai05/kononenko_igor/allACAI2005.pdf

[17] Kononenko I., *Strojno učenje*, Fakulteta za računalništvo in matematiko, Ljubljana, 2005.

[18] Kotsiantis S. B., *Supervised Machine Learning: A Review of Classification Techniques*, Emerging Artificial Intelligence Applications in Computer Engineering, 2007, pages 1-16. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.9683&rep=rep1&type=pdf

[19] Mitchell T., *Machine Learning*, McGraw Hill Higher Education, 1997.

[20] R Development Core Team, *A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, 2006.

[21] Segal M., *Machine Learning Benchmarks and Random Forest Regression*, CBMB Working Paper, 2004, pages 1-4. Available from: http://www.epibiostat.ucsf.edu/biostat/cbmb/publications/bench.rf.regn.pdf

[22] Smith L., *A tutorial on Principal Components Analysis*, 2002, pages 12-20. Available from: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

[23] StatSoft, Electronic Statistics Textbook, *Generalized Additive Models (GAM)*, Available from: http://www.statsoft.com/textbook/generalized-additive-models/

[24] StatSoft, Electronic Statistics Textbook, *Neural Networks*, Available from: http://www.statsoft.com/textbook/neural-networks/

[25] Zhang G., *Neural Networks for Classification: A Survey*, IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews, Vol. 30, No. 4, November 2000, page 1. Available from: http://vis.lbl.gov/~romano/mlgroup/papers/neural-networks-survey.pdf

[26] Zhang H., *The Optimality of Naive Bayes*, FLAIRS2004 conference, 2004, pages 1-2. Available from: http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf