R Tutorial at the WZB

13 – Introduction to Machine Learning with R II

Markus Konrad

February 7, 2019

Processing math: 100%

Today's schedule

1. Classification and performance measures
2. Decision trees
3. Random forests
4. End of the tutorial

Review of last week's tasks

Solution for tasks #12

now online on
https://wzbsocialsciencecenter.github.io/wzb_r_tutorial/

Classification

Classification

In classification, the aim is to predict a **categorical response**. This can be a two-class (i.e. binary) response or multi-class response.

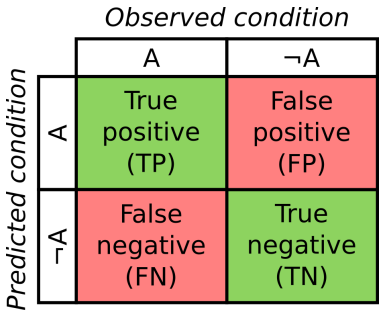
Although the outcome categories are discrete, the predictions itself are usually continuous class probabilities. Example: The predicted probability of an email to belong to the class "spam" is 0.7. Using a probability threshold of 0.5, this email would be classified as spam.

Classification models are also called classifiers.

Performance measures for classifiers

Confusion matrix

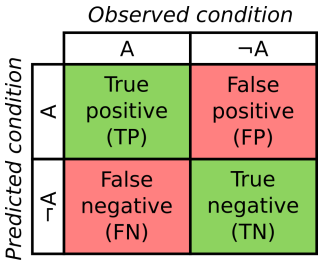
A confusion matrix is a cross-tabulation of observed vs. predicted classes:



Confusion matrix for a two-class problem

→ everything on the diagonal is correctly classified, everything else is misclassified

Accuracy



The Accuracy rate metric is the **overall rate of correctly classified samples** and can be calculated from a confusion matrix as:

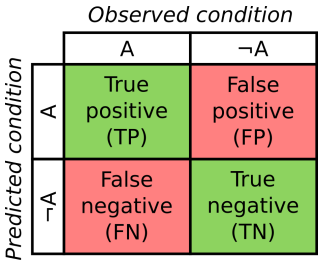
$$\text{Accuracy} = \frac{TP + TN}{N},$$

where $N = TP + TN + FP + FN$ (the number of observations).

Disadvantages:

- no distinction about type of errors (a false positive error may be more severe than a false negative)
- class imbalances are not considered (what if A occurred much more often "in nature" than ¬A?)

Misleading accuracy



Confusion matrix for a two-class problem

If we have high class imbalance, a simple model could achieve almost perfect accuracy by simply classifying each response as negative:

```
# "A" occurs only 0.1% of the time:
obs <- factor(rep(c('A', 'notA'), times = c(1, 999)))
# a model that classifies everything as "notA":
pred <- factor(rep('notA', times = 1000), levels = levels(obs))
(confmat <- table(pred, obs))
```

```
##       obs
## pred     A notA
##   A      0    0
##   notA   1  999
```

Misleading accuracy

```
##       obs
## pred     A notA
##   A      0    0
##   notA   1  999
```

Calculating the (impressive) accuracy of our "model":

```
(confmat[1,1] + confmat[2,2]) / 1000
```

```
## [1] 0.999
```

Metrics that take class imbalance into account:

- comparison with no-information-rate (how good is the model compared to random guessing?)
- Cohen's Kappa

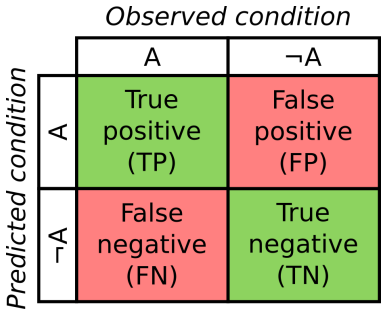
WZB Wissenschaftszentrum Berlin für Sozialforschung

Sensitivity, specificity, balanced acc.

Sensitivity / True positive rate: Rate that condition A is predicted correctly for all samples that actually have condition A.

Sensitivity = TPTP + FN

The false negative rate is defined as 1 − Sensitivity.



Confusion matrix for a two-class problem

Specificity / True negative rate: Rate that condition ¬A is predicted correctly for all samples that actually have condition ¬A.

Specificity = TNTN + FP

The false positive rate is defined as 1 − Specificity.

Example: Should a spam classifier rather focus on high sensitivity or high specificity?

Sensitivity, specificity, balanced acc.

What do the sensitivity and specificity measures tell us practically?

Take for example official test results of a face detection system that was tested by German Federal Police at station Südkreuz, Berlin:

- sensitivity (true positive rate): 80%
- specificity: 99.9% → false positive rate (FPR): 0.1%

We have 90, 000 passengers per day at Südkreuz, which means 90 "false alarms" → 90 innocent people stopped per day.

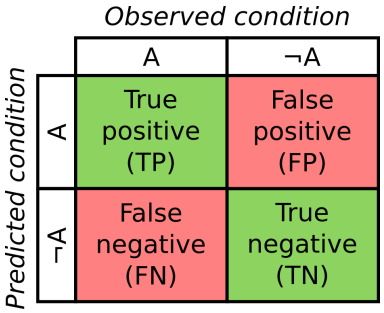
The CCC later critized the results as being "whitewashed": The actual average FPR was actually 0.67%.

Sensitivity, specificity, balanced acc.

Balanced accuracy takes the mean of both measures:

Balanced acc. = Sensitivity + Specificity2

What would be the balanced accuracy of the "always-predict-negative model"?



Confusion matrix for a two-class problem

```
##       obs
## pred     A notA
##   A      0    0
##   notA   1  999
```

```
(sens <- confmat[1,1] / (confmat[1,1] + confmat[2,1]))
```

```
## [1] 0
```

```
(spec <- confmat[2,2] / (confmat[1,2] + confmat[2,2]))
```

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

```
(sens + spec) / 2
```

```
## [1] 0.5
```

WZB Wissenschaftszentrum Berlin für Sozialforschung

Receiver-Operating-Characteristic (ROC) Curve

Some history:

- name stems from radar receiver operators that should detect enemy aircraft in radar signals in World War II
- US army did research on how to improve detections

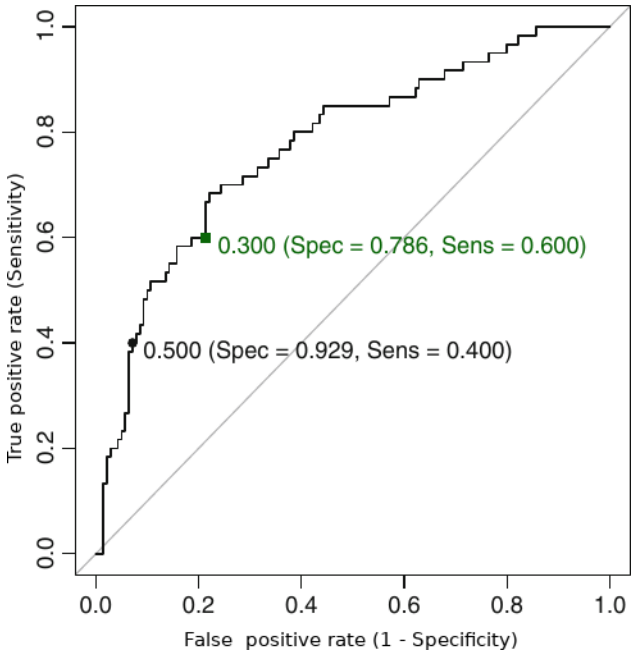
Idea:

- different thresholds can be used to determine a class from a predicted class probability
- example: classifier predicts that class probability for email being spam is 0.7
- threshold of 0.5: "spam" – however, this increases likelihood of false positives
- threshold of 0.8: "not spam" – however, this increases likelihood of false negatives

→ trade-off: True positive rate (sensitivity) vs. false positive rate (1 − Specificity)

ROC Curve

ROC curves visualize that trade-off for a given model by evaluating sensitivity and specificity for a range of thresholds.



adapted from Kuhn & Johnson 2016

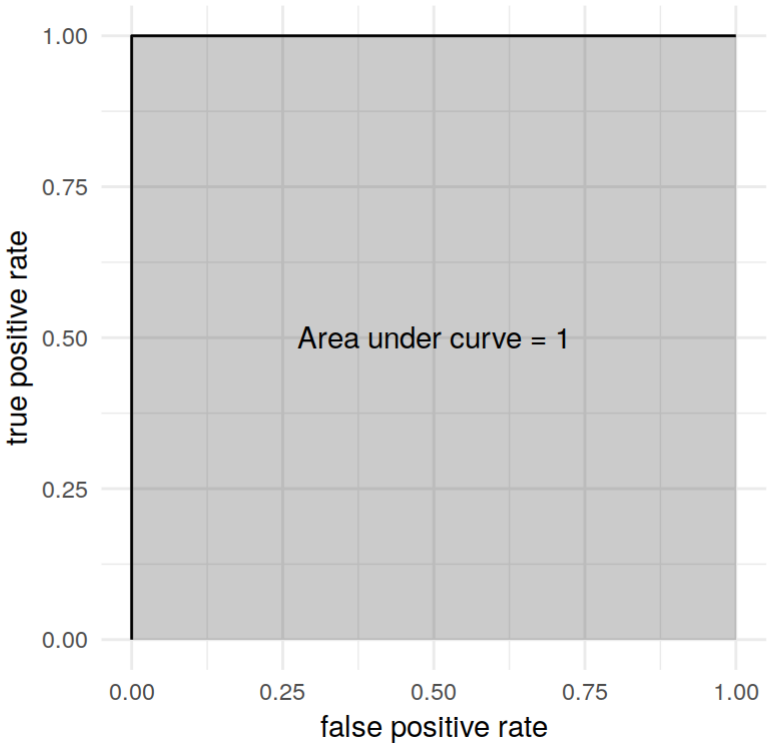
In the given example, a model with a threshold of 0.3 would achieve a higher sensitivity (0.6 vs. 0.4) than a model with a threshold of 0.5 for the cost of sacrificing specificity (0.786 vs. 0.929).

WZB Wissenschaftszentrum Berlin für Sozialforschung

Area under ROC Curve (AUC)

ROC curves allow quantitive assessment of overall model performance.

A perfect model would have a true positive rate of 100% and false positive rate of 0%. The area under the ROC curve would be 1:

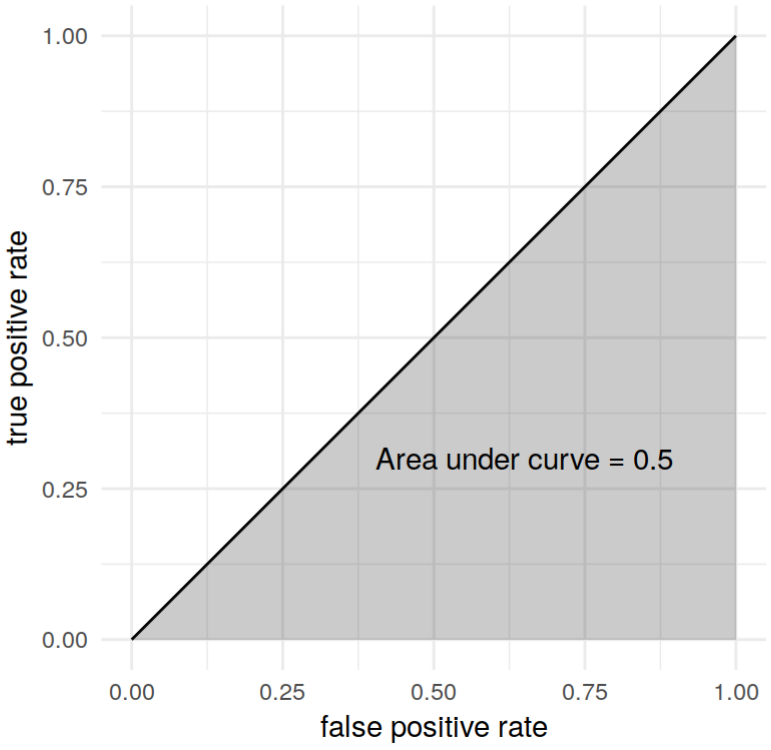


WZB Wissenschaftszentrum Berlin für Sozialforschung

Area under ROC Curve (AUC)

ROC curves allow quantitive assessment of overall model performance.

An ineffective model's ROC curve would be equal to the diagnoal. The area under the ROC curve would be 0.5:



WZB Wissenschaftszentrum Berlin für Sozialforschung

Classification models

The number of choices is large for classification models:

- linear models (e.g. Logistic Regression)
- penalized linear models (e.g. Ridge, Lasso)
- non-linear models (e.g. Support Vector Machines, K-Nearest Neighbors, Naive Bayes)
- partitioning models (e.g. Decision Trees, Random Forest)

We will have a look at Decision Trees and Random Forest for classification and compare them with Logistic Regression.

An example data set

I decided to use the PRODAT data set (former WZB project – Dieter Rucht / Simon Teune) as an example data set.

- data set of protests in Germany (including former GDR) from 1950 to 2002 with their coded characteristics
- original data set contains 15,973 observations and 223 variables
- most variables are categoricals
- many categoricals are coded as "primary" and "secondary" indicators, e.g. `polfeld1` and `polfeld2` for primary and secondary policy fields
- codebook available in German and English

Question of interest for the following examples: **How accurately can we predict if a protest (also) exhibits physical violence from the given characteristics?**

WZB Wissenschaftszentrum Berlin für Sozialforschung

PRODAT data preparation

- remove variables not relevant for analysis (e.g. coder's name, year, unique event ID, comments, derived variables)
- remove observations for protest forms that do not potentially exhibit violence (e.g., petitions, leaflets, litigation)
- remove variables with many missings (more than 60% NAs)
- create a binary outcome variable `y` that captures if a protest also included violent actions (derived from variable `dform4_4`)
- remove observations were outcome is missing
- create dummy variables from categoricals and unify primary and secondary indicator dummies (e.g. when `polfeld1` is "social security" and `polfeld2` is "environment" then both dummies `polfeld.social_security` and `polfeld.environment` would be set to `1`)
- perform sampling (stratified by outcome `y`) to create a training set with 75% of the data and a test set with the rest

After that, we have 8,887 observations with 240 variables (most of which are dummy variables) in the training set and 2,961 observations in the test set.

PRODAT predictors and characteristics

Important predictors seen later:

- `formleg`: captures whether a protest was legal, illegal or of unknown status[1]
- `zahl`: number of participants
- `tallzahl`: number of groups
- `polfeld`: policy field
- `dauer`: duration of protest
- `tragsoz`: social supporters[2] (e.g. workers, unemployed, women, etc.)
- `nminder`: captures whether a protest was pro or against ethnic minorities
- `reaktion`: captures whether a protest caused "immediate reactions"

[1] the codebook gives no further information in what "legal" or "illegal" means in the context of protests

[2] I'm not sure if this is the right term

PRODAT predictors and characteristics

Other characteristics of the data set:

- outcome y has class imbalance: ~16% of protests are coded as including violence
- many missings in the data set:
  - categorical predictors: NA dummy created[1]
  - numeric predictors: some methods can handle missings (decision trees), for others these predictors were removed

[1] in order to uncover "informative missingness"; however in practice this should probably not be done for all predictors

WZB Wissenschaftszentrum Berlin für Sozialforschung

A Logistic Regression model

We compute a logistic regression model as baseline model. This model uses a reduced set of predictors:

- only primary indicators (e.g. `polfeld1` and not `polfeld2`)
- no predictors with missings (`zahl`, `tallzahl`)
- no predictors with very low variance (i.e. almost only zeros or only ones for all observations for dummies)

We create the model as simple additive function of all remaining 89 predictors:

```
# family = binomial for logistic regression
logreg_model <- glm(y ~ ., data = train_data, family = binomia
```

Predictive performance

Output of significant (at 5% sign. level) coefficients ordered by absolute magnitude:

```
   term                                  log_odds_ratio odds_ratio  p.value
 1 formleg.illegal                                3.66       39.0  9.40e- 4
 2 tragsoz1.arbeitnehmer                         -3.15        0.0430 6.20e- 8
 3 reapoli.ja                                     1.74        5.71  1.06e- 3
 4 tragsoz1.ausl_nder_asyl_ethn_grupp            -1.67        0.187 3.63e-16
 5 adress.30                                     -1.44        0.236 2.55e-16
 6 tragsoz1.sonstige                             -1.38        0.251 5.87e-11
 7 nminder1.andere_proteste                      -1.20        0.300 2.18e-11
 8 tragsoz1.studenten                            -0.978       0.376 1.50e- 4
 9 objekt.privatpersonen                          0.931       2.54  3.48e- 7
10 tragsoz1.ka                                   -0.811       0.444 3.39e-12
   ...
```

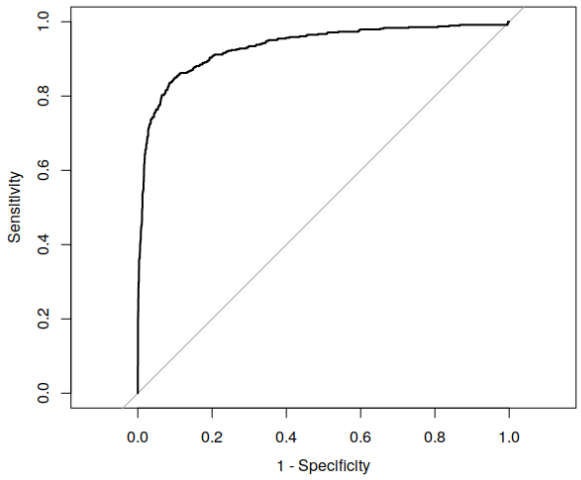
The model has some extremely large coefficients, so it would be advisable to use a penalized model instead (like ridge, lasso), but for the moment we only care about the predictive performance:

- 10-fold cross validation on training data yields AUC of 0.93, sensitivity of 0.70 and specificy 0.96 → model is better at prediciting true negatives ("no violence") than true positives
- similar values for the test data set

Predictive performance

The model has some extremely large coefficients, so it would be advisable to use a penalized model instead (like ridge, lasso), but for the moment we only care about the predictive performance:

- 10-fold cross validation on training data yields AUC of 0.93, sensitivity of 0.70 and specificy 0.96 → model is better at prediciting true negatives ("no violence") than true positives
- similar values for the test data set



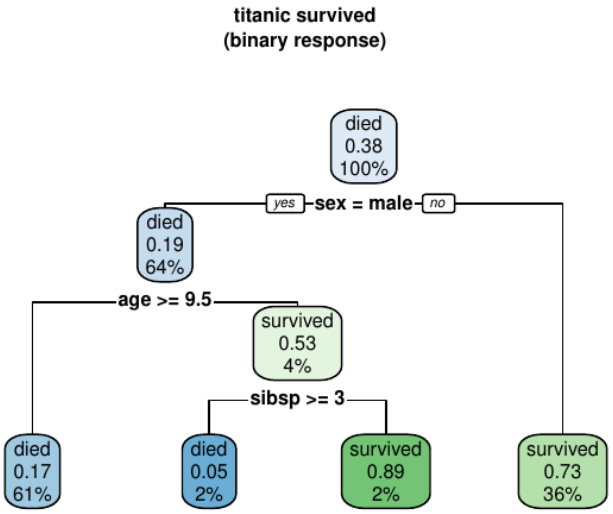
ROC curve

Decision trees

Decision trees

Decision trees or classification trees (CART algorithm – Breiman et al. 1984) recursively split the data into "purer" groups regarding the response. This requires a **measure of "group purity"** in order to find out the **predictor for splitting and its split value**.

The resulting decision tree is a binary tree: At each node there is a split into two sub-groups. The terminal nodes are also called leaves.



Example decision tree (source: Milborrow 2018)

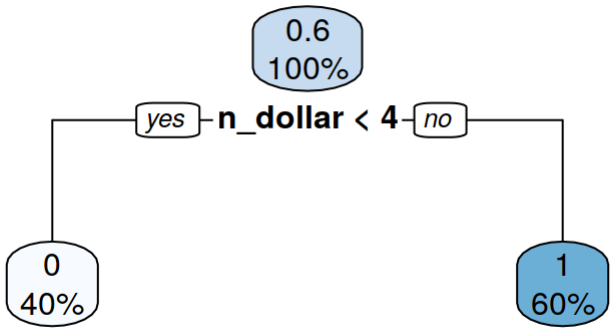
Decision trees

→ example data with outcome `is_spam` and predictors that reflect how often a word "dollar" or "hello" occurs in an email:

```
##   is_spam n_dollar n_hello
## 1    TRUE        5       1
## 2    TRUE        8       2
## 3    TRUE        6       0
## 4   FALSE        0       1
## 5   FALSE        2       1
```

How would you split the data?

→ choose `n_dollar` to be a good predictor for splitting with a value that splits the data into two groups, each purely consisting of "spam" or "not spam" samples:



Decision trees

A decision tree algorithm evaluates each predictor and many possible split points (i.e. thresholds for splitting) by calculating a measure of purity for each prospective pair of sub-groups. One such measure is the Gini-index. The predictor-value combination that maximizes the Gini-index is used for splitting.

The splitting process is done recursively until a stopping criterion is met, e.g.:

- number of samples in a group is below a certain threshold
- maximum tree depth is reached
- no increase in node purity

Pruning

Large decision trees tend to overfit the data: They split the training data into very small and very specific sub-groups and hence do not generalize well.

Usually a complexity parameter cp is used to penalize the purity criterion by a factor of the total number of terminal nodes in the tree (→ "pruning"). The larger a tree grows, the more penalty is put on the purity criterion.

An optimal value for cp can be found with hyperparameter tuning on the training data.

A decision tree for PRODAT

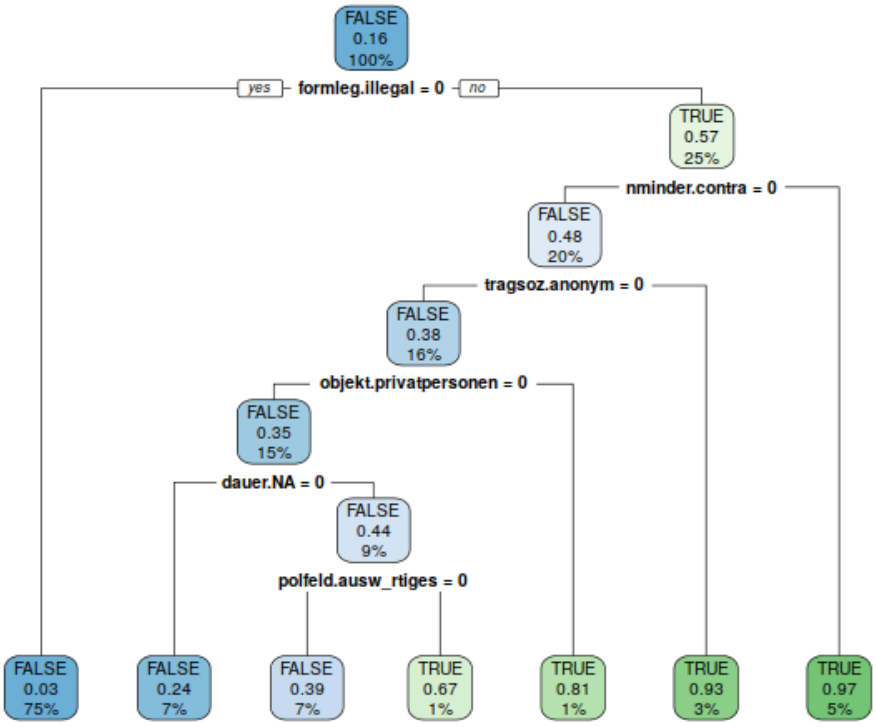
We build a small decision tree (high complexity penalty `cp = 0.01`) using the package `rpart`:

```
library(rpart)
treemodel <- rpart(y ~ ., data = train_data,
                   control = rpart.control(cp = 0.01))
```

The resulting model can be visualized with `rpart.plot`, which allows for easy interpretation:

```
library(rpart.plot)
rpart.plot(treemodel)
```

A decision tree for PRODAT



WZB Wissenschaftszentrum Berlin für Sozialforschung

A decision tree for PRODAT

Model interpretation can also be assisted by variable importance. We can measure how each predictor overall contributes to increasing the purity criterion (i.e. increasing the Gini-index). → predictors that occur higher / multiple times in tree contribute more to purity optimization

```
head(treemodel$variable.importance, 10)
```

```
formleg.illegal                965.40788
formleg.legal                  525.13945
tragsoz.anonym                 332.99901
nminder.contra                 329.73977
adress.27                       85.19441
objekt.privatpersonen           67.40235
aufpo.ja                        60.29222
dauer.NA                        25.28559
polfeld.ausw_rtiges             16.65478
dauer.1_bis_12_h                14.65957
```

Note: `formleg` has more than two levels.

Why are there predictors listed in the top 10 that did not appear in the tree plot?

Because each predictor's contribution is also measured for alternative trees (surrogate splits).

A decision tree for PRODAT

Evaluating the model using held-out `test_data`:

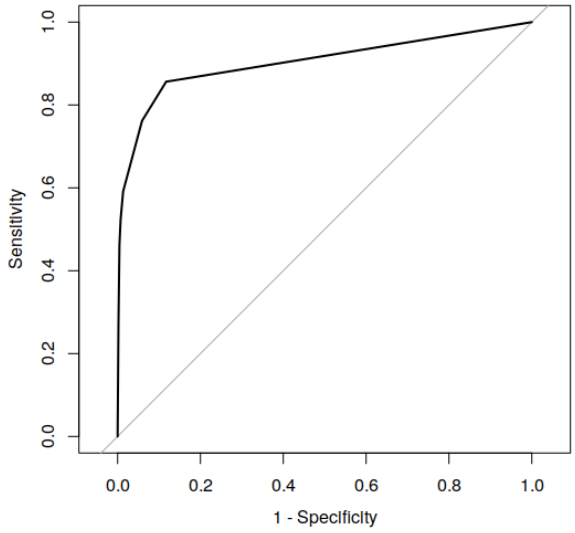
```
# get class predictions (TRUE / FALSE)
pred_classes <- predict(treemodel, newdata = test_data_X, type = 'class')
confusionMatrix(pred_classes, test_data$y, positive = 'TRUE')
```

```
          Reference
Prediction FALSE TRUE
     FALSE  2441  199
     TRUE     33  288
               Accuracy : 0.9216
                  Kappa : 0.6697
                  [...]
            Sensitivity : 0.59138
            Specificity : 0.98666
                  [...]
     Balanced Accuracy : 0.78902
```

- like log. regression, this model has low sensitivity (true positive rate)
- using alternative threshold would yield in better sensitivity (~ 0.85) for the cost of reduced specificity ((~ 0.89))

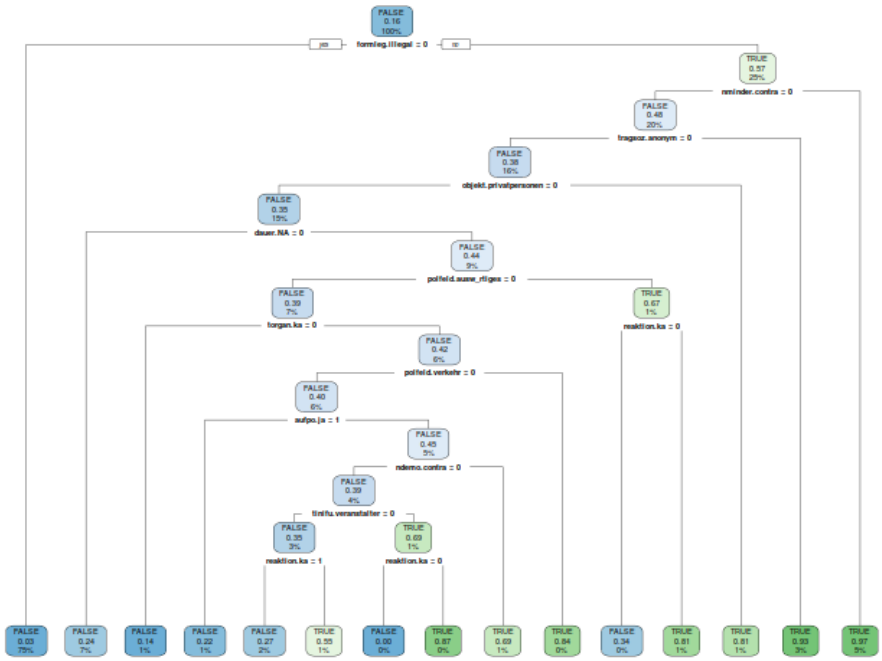
A decision tree for PRODAT

```
# get probability predictions
# (matrix with prob. for FALSE and TRUE response respectively)
pred_prob <- predict(treemodel, newdata = test_data_X, type = 'prob')
library(pROC)
roc_curve <- roc(response = test_data$y, predictor = pred_prob[,1])
auc(roc_curve)  # AUC ~ 0.90
plot.roc(roc_curve, legacy.axes = TRUE)
```



A decision tree for PRODAT

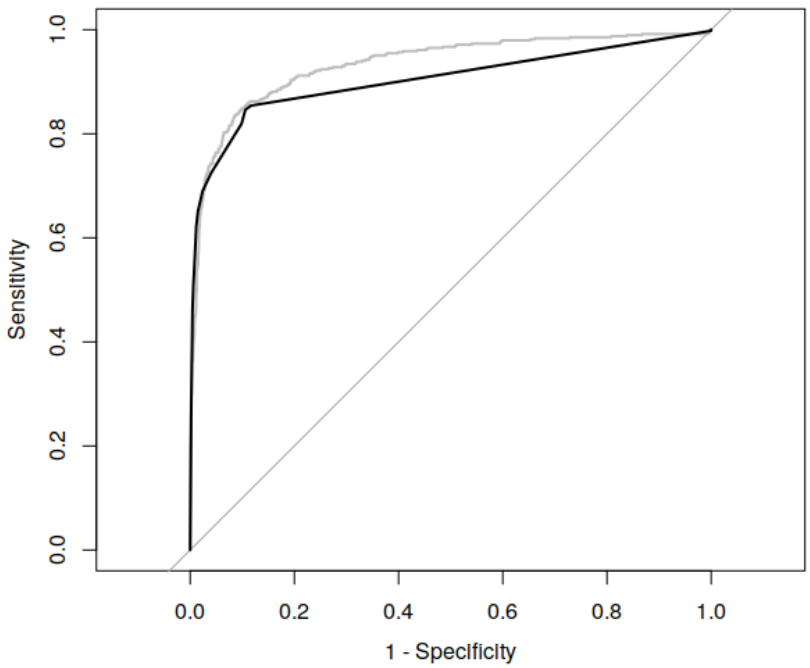
Cross-validation yielded an optimal `cp = 0.005` which results in slightly better performance but a much more complex tree:



WZB Wissenschaftszentrum Berlin für Sozialforschung

A decision tree for PRODAT

The AUC is at about 0.90 which is worse than for the logistic regression:

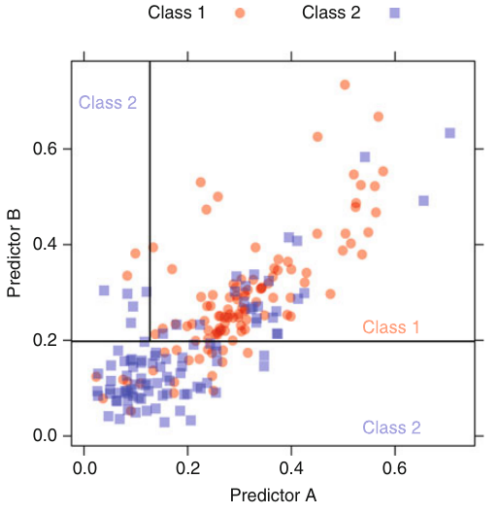


ROC curves: Gray line is log. regr., black line is dec. tree model

Advantages and disadvantages

Advantages:

- fast model computation
- fast prediction
- automatic feature selection
- can handle missings via surrogate splits
- models are interpretable



Decision boundaries defined by a tree based model (source: Kuhn & Johnson 2016)

Disadvantages:

- partition data into rectangular regions → "sub-optimal predictive performance compared to other modeling approaches" (Kuhn & Johnson 2016)
- high model variance → unstable (small changes in data might lead to very different tree)

Random forests

Motivation for Random Forests

Decision trees exhibit several properties that make them a **weak learner** (high variance, low predictive performance). **Ensemble methods** try to combine several (hundreds, thousands) weak learners in order to form a "strong learner". Each weak learner's training process involves a **random component** to lower the variance and increase predictive performance.

Some ensemble methods do random resampling of the observations in the training data for each weak learner (e.g. bootstrap aggregation – bagging). Random Forest is an ensamble technique that uses both **random resampling and random predictor selection** for each decision tree.

Algorithm

Hyperparameters:

- number of trees to grow n_{tree}
- number of randomly selected predictors m_{try}

```
For each tree from 1 to n_tree:

  Resample training data with replacement
(bootstrap sample)

  Train a tree model until stopping criterion is
reached:

    Randomly select m_try predictors out of original
predictors

    Select the best predictor-value combination for
splitting
    (i.e. that maximizes Gini-index)
```

In the end, we have a Random Forest ensemble which consists of n_{tree} decision trees.

Prediction

For predicting the outcome of a new sample:

- each tree model predicts the outcome independently and thereby casts a "vote"
- votes are counted for each class and divided by n_{tree} → probability vector for each class
- classification according to class with highest probability

Example:

- classes "positive", "neutral", "negative"
- ensemble with n_{tree} = 1000
- new sample gets votes from each tree, we divide it by n_{tree}:

```
      negative  neutral positive
votes      687      201      112
prob.     0.69     0.20     0.11
```

→ final class for new sample would be "negative" with probability 0.69

WZB Wissenschaftszentrum Berlin für Sozialforschung

A Random Forest model for PRODAT

In contrast to decision trees, Random Forest (RF) can't handle missings in the predictors.[1] Hence we remove `zahl` and `tallzahl` from the predictors and then build the model using the `randomForest` package:

```
# hyperparameters ntree = 500 and mtry = 50 were chosen from
# prior experience here; I will later show a proper tuning
rfmodel <- randomForest(x = train_data_X, y = train_data$y,
                        ntree = 500, mtry = 50)
```

The model can now be used for prediction:

```
# to predict classes:
predict(rfmodel, test_data_X, type = 'response')

# to predict class probabilities:
predict(rfmodel, test_data_X, type = 'prob')
```

[1] One reason for that is that reasonable surrogate splits in RF context may not always exist, since predictors are chosen randomly for each split (there may not be a good, correlated surrogate predictor to split on).

Interpretation

Can we see how the outcome for a new sample is actually predicted?

The model is an ensemble of 500 decision trees, each consisting of hundreds or thousands of nodes. For a prediction, each tree generates one class vote. **To retrace the decision for a single sample, we would have to investigate how each of the 500 votes were generated.**

As an example, we can have a look at the 10th tree that was created (this tree has more than 1000 nodes):

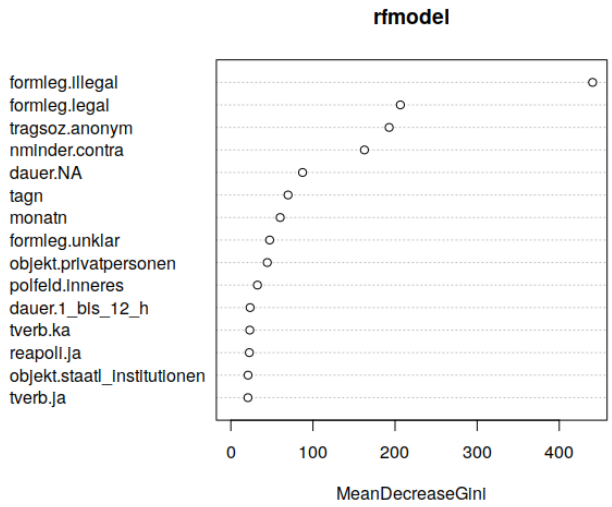
```
getTree(rfmodel, k = 10, labelVar = TRUE)
```

```
  left daughter right daughter              split var split point status pred
1             2              3               tverb.ka         0.5      1
2             4              5             reapoli.ja         0.5      1
3             6              7 objekt.privatpersonen         0.5      1
4             8              9        nfrauen.contra         0.5      1
5            10             11   nandere.mit_familie         0.5      1
...
```

(couldn't find a way to quickly visualize this)

Interpretation

So, a Random Forest model is essentially a **"black box model"**; its decisions are almost impossible to retrace. However, it is possible to derive the variable importance using `varImp()` or `varImpPlot()`:



There is currently a lot of research on "explainable ML models" (e.g. Ribeiro et al. 2016).

Tuning a Random Forest model

RF is computationally extensive: a single RF model took ~10 minutes to compute on my laptop

→ parameter tuning should be done with parallel processing on a cluster machine

```
library(doMC)
registerDoMC(32)  # use 32 out of 64 CPU cores on cluster machine

rf_tuning <- train(x = train_data_X, y = train_data$y, method = 'rf',
                   tuneGrid = data.frame(.mtry = c(2,25,50,75,100,125,150,20
                   trControl = trainControl(method = 'repeatedcv', repeats =
```

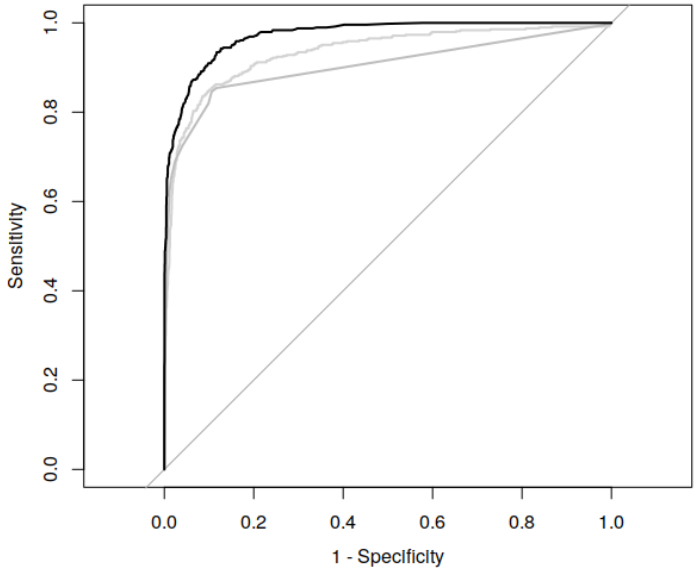
- note that n_{tree} is left at default value (500 trees)
- best predictive preformance with $m_{try} = 50$

Tuning a Random Forest model

Achieved metrics on test data:

```
            Accuracy : 0.9395
               [...]
               Kappa : 0.7635
               [...]
         Sensitivity : 0.7290
         Specificity : 0.9810
               [...]
Balanced Accuracy : 0.8550
```

Tuning a Random Forest model



ROC curves: Light gray line is log. regr., dark gray line is dec. tree, black line is RF model

AUC: 0.972 → better than logistic regression, which had AUC of 0.93, sensitivity 0.70 and specificy 0.96

WZB Wissenschaftszentrum Berlin für Sozialforschung

Advantages and disadvantages

Advantages:

- excellent predictive performance in many situations
- minimum data preprocessing required (handles skew, near zero variance, collinearities[1])
- automatic feature selection

Disadvantages:

- decisions almost impossible to comprehend → creates a "black box model"
- computationally extensive
- does not work with missing data (there are extensions that do that)

[1] Having several strongly collinear predictors has impact on reliability of variable importance measure.

Reflection

Reflect on what we just did: We build a classifier for predicting violence in protests.

- What do you think of the practical implications of such a model?
- Are ML models more fair / objective than "mental models"?
- What are possible sources of bias inside the data?

Literature

- Kuhn & Johnson 2016: Applied Predictive Modeling
- Ribeiro et al. 2016: "Why Should I Trust You?" – Explaining the Predictions of Any Classifier
- Breiman et al. 1984: Classification and Regression Trees
- S. Milborrow 2018: Plotting rpart trees with the rpart.plot package
- O'Neil 2016: Weapons of Math Destruction
- Bellovin 2019: Yes, “algorithms” can be biased. Here’s why

End of the tutorial

Where to go from here?

Exercise:

- use R in your daily work!
- SWIRL
- R Exercises
- Coding Club tutorials

Stay up-to-date:

- R-Bloggers
- RWeekly

Network:

- Berlin R Users Group
- R-Ladies Berlin

Read:

- Grolemund & Wickham 2017: R for Data Science
- Grolemund 2014: Hands-On Programming with R
- Wickham 2014: Advanced R
- Peng 2016: Exploratory Data Analysis with R
- Healy 2018: Data Visualization – A practical introduction
- Navarro 2019: Learning Statistics with R

WZB Wissenschaftszentrum Berlin für Sozialforschung