Building meaningful machine learning models for disease prediction

Dr Shirin Glander

Dep. of Genetic Epidemiology
Institute of Human Genetics
University of Münster

shirin.glander@wwu.de

https://shiring.github.io
https://github.com/ShirinG

Friday, 31st March 2017

Building meaningful machine learning models for disease prediction

Dr Shirin Glander

Dep. of Genetic Epidemiology
Institute of Human Genetics
University of Münster

shirin.glander@wwu.de

https://shiring.github.io
https://github.com/ShirinG

Friday, 31st March 2017

- Welcome everybody!
- Thank you very much for the invitation and the opportunity to present this talk
- I will go over my work on using machine learning models in disease prediction
- I want to specifically give a hands-on demonstration of how you can build meaningful models yourselves using R
- I will demonstrate how to evaluate model performance and
- how to optimize models to address different disease-related questions

About me

since 2015 Bioinformatics Postdoc
Next Generation Sequencing
autoinflammatory diseases &
innate immunity

2011 - 2015 PhD in Biology
Is the immune system of plants required to adapt to flowering time change?

2005 - 2011 BSc and MSc of Science in Biology
evolutionary genetics,
immune memory in Drosophila

About me

since 2015 Bioinformatics Postdoc
Next Generation Sequencing
autoinflammatory diseases &
innate immunity

2011 - 2015 PhD in Biology
Is the immune system of plants required to adapt to flowering time change?

2005 - 2011 BSc and MSc of Science in Biology
evolutionary genetics,
immune memory in Drosophila

- Before I start, I want to quickly introduce myself:
- I am a bioinformatics postdoc
- working with next generation sequencing data,
- like RNA-seq for transcriptomics,
- whole genome sequencing for variant analysis,
- ATAC- or Chip-seq for chromatin and epigenetic information.
- My own research focuses on questions relating to autoinflammatory diseases
- and innate immune mechanisms

- I earned my PhD in biology from the University of Münster in 2015
- working with RNA-seq data to determine how plant defense has co-evolved with and potentially shaped different life-history strategies

- Before that, during my BSc and MSc I worked on questions about evolutionary genetics and immune memory in Drosophila

Table of contents

Building meaningful machine learning models for disease prediction

- Now, let's start with the topic about which you're here:
- Machine learning is a powerful approach for developing sophisticated, automatic, and objective models for the analysis of complex biomedical data
- Machine learning promises to help physicians make near-perfect diagnoses, ...
- ... choose the best medications for their patients, ...
- ... predict readmissions, ...
- ... identify patients at high-risk for poor outcomes, ...
- ... and in general improve patients' health while minimizing costs.

- I titled my talk: "Building meaningful machine learning models for disease prediction"
- with the emphasis on **meaningful**
- So, first, I want to introduce to you what it is I mean exactly with **meaningful models**
- Then, I will give you a few examples of how machine learning is currently being used in disease modeling and clinical data science
- Before I delve into the nitty-gritty of modeling, I will quickly recap the most important concepts of ML
- And finally, I will show how to build ML models in R
- and how to evaluate the performance of such models

What makes a model meaningful?

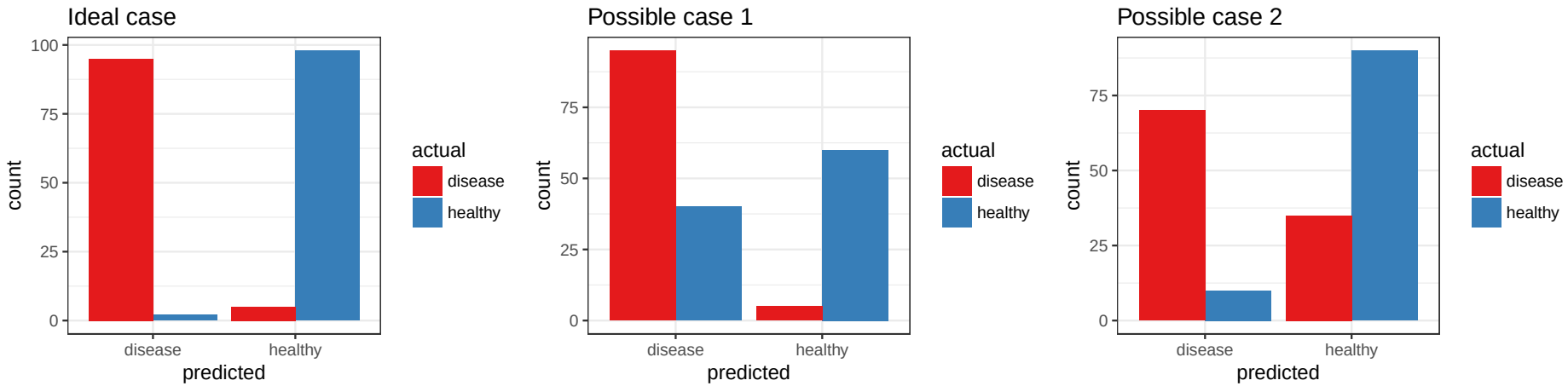
What makes a model meaningful?

- creating ML models is relatively easy
- creating good or meaningful models is hard

Meaningful models

- are generalizable
- answer the question(s) posed...
- ... with sufficient accuracy to be trustworthy

Accuracy depends on the problem!



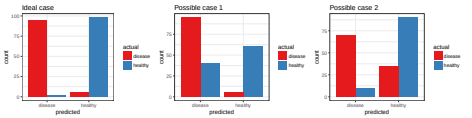
What makes a model meaningful?

- creating ML models is relatively easy
- creating good or meaningful models is hard

Meaningful models

- are generalizable
- answer the question(s) posed...
- ... with sufficient accuracy to be trustworthy

Accuracy depends on the problem!



- With a little bit of practice, anybody can build machine learning models
- what is hard though, is to build 'good' or 'meaningful' models
- I want to begin with an introduction to the main question of my talk:
- what makes a model *good* or *meaningful*?

1. it needs to be generalizable, i.e. it needs to perform well on unseen data
2. It answers a **specific** question or addresses a specific problem,
   e.g. does a mammogram image show a healthy breast or is there a tumor?
   Or does an ECG show a normal heart rhythm or arrhythmia?
3. And it produces a correct outcome (e.g. a diagnosis) often enough that we trust it!

- If we build models, we therefore need to evaluate its accuracy
- before we can decide whether it is trustworthy enough to implement in real-life,
- like in a hospital where it could e.g. assist doctors in making decisions on treatment

- But what exactly is *high accuracy* can not be defined with a one-size-fits-all approach:
- it depends on the problem we want to model.
- a model needs to perform well in the real world, not in the lab (i.e. on training data)
- just because a model performs well in the lab, doesn't mean it will also perform well in deployment
- the context might change: distribution, missing data, noise, class size, etc.

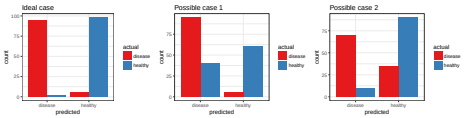
What makes a model meaningful?

- creating ML models is relatively easy
- creating good or meaningful models is hard

Meaningful models

- are generalizable
- answer the question(s) posed...
- ... with sufficient accuracy to be trustworthy

Accuracy depends on the problem!



Let me illustrate what I mean with the following examples:

- Ideal case: Of course, we all want to achieve ideal modeling results where overall prediction accuracy is very high. With a model like that, we can be very confident that a healthy person is indeed healthy and a sick person is not.
- But in reality, we often achieve prediction accuracies that are much less nice.
- Scenarios 1 and 2: Le's consider two possible scenarios:
- in scenario 1, we can be very confident that a person who got classified as "healthy" is indeed healthy, while a person who has been diagnosed as diseased might as well be healthy based on these prediction accuracies
- in case 2, it is the other way around.
- We now need to make a decision which scenario is better and in which direction we want to optimize our model: do we rather want to refer a few healthy people for further checking because the model predicted them as diseased? Or do we rather want to be as certain as possible that a predicted disease is actually true and accept that we might miss a few disease cases?

Machine Learning (ML) in disease modeling

ML in disease modeling

- tools that can interpret "big medical data"
- and provide fast, accurate and actionable information
- for precision or personalized medicine

Examples:

- computer-aided diagnosis of breast cancer from mammograms[1]
- identifying gene defects with facial recognition software[2]
- identifying signatures of Brain Cancer from MRSI[3]
- ... and many more ...

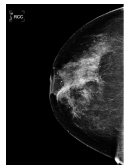
[1] Doi 2007.
[2] Levenson 2014.
[3] Sadja 2006.

ML in disease modeling

- tools that can interpret "big medical data"
- and provide fast, accurate and actionable information
- for precision or personalized medicine

Examples:

- computer-aided diagnosis of breast cancer from mammograms[1]
- identifying gene defects with facial recognition software[2]
- identifying signatures of Brain Cancer from MRSI[3]
- ... and many more ...



[1] Doi 2007.
[2] Levenson 2014.
[3] Sadja 2006.

- there is not a place in medicine where AI doesn't have potential applications:
- more data is being collected that needs to be interpretated
- increasingly data driven diagnosis, e.g. in radiology
- image recognition of e.g. tumors or pneumonia in medical images
- similar to training ML models to recognize images of cats (classification in images)
- ML allows incorporation of data from heterogenous inputs:
- clinical, genomics, drugs, electronic health records, etc.
- = personalised medicine

- A key aspect of precision medicine is the development of informatics tools
- that can analyze and interpret 'big data' sets
- in an automated and adaptive fashion
- while providing accurate and actionable clinical information
- ML based models can improve detection, diagnosis, and therapeutic monitoring of disease

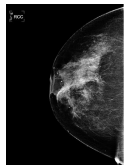
- you can find things with ML that you wouldn't be able to find otherwise
- many models today perform better than humans!
- a patient with a rare difficult condition can be matched to similar cases from the past
- faster diagnosis, better treatment

ML in disease modeling

- tools that can interpret "big medical data"
- and provide fast, accurate and actionable information
- for precision or personalized medicine

Examples:

- computer-aided diagnosis of breast cancer from mammograms[1]
- identifying gene defects with facial recognition software[2]
- identifying signatures of Brain Cancer from MRSI[3]
- ... and many more ...



[1] Doi 2007.
[2] Levenson 2014.
[3] Sadja 2006.

- identifying breast cancer, lung cancer, osteoporosis, brain tumors, etc. from medical images
- predicting response of different cancer types to different treatments
- Computer-Aided Diagnosis (CAD) has become one of the major research subjects in medical imaging and diagnostic radiology
- With CAD, radiologists use the computer output as a 'second opinion' and make the final decisions
- In vivo magnetic resonance spectroscopy imaging (MRSI) allows noninvasive characterization
- and quantification of molecular markers of potentially high clinical utility
- for improving detection, identification, and treatment for a variety of diseases, most notably brain cancers

- facial analysis technology aids diagnoses of genetic disorders
- researchers at the University of Oxford in the United Kingdom
- Face2Gene uses machine learning
- helps geneticists narrow down possible disorders that often involve dysmorphic facial features

- doctors are still important!
- computer does the tasks that we humans are not good at, like interpreting complex images
- we have more data than humans can manage to make sense of (e.g. genomics data)
- the clinicians will be freed up to think about best treatment options and talk more with the patients
- good models are built on strong knowledge of the question and the biology behind it
- features need to be evaluated in context

the elephant in the room

Trust!

- we don't know WHY ML models make decisions
- inherent problem with ML models:
  they are hard (or impossible) to interpret

- therefore, it is crucial that our models are accurate
  and meaningful

the elephant in the room

Trust!

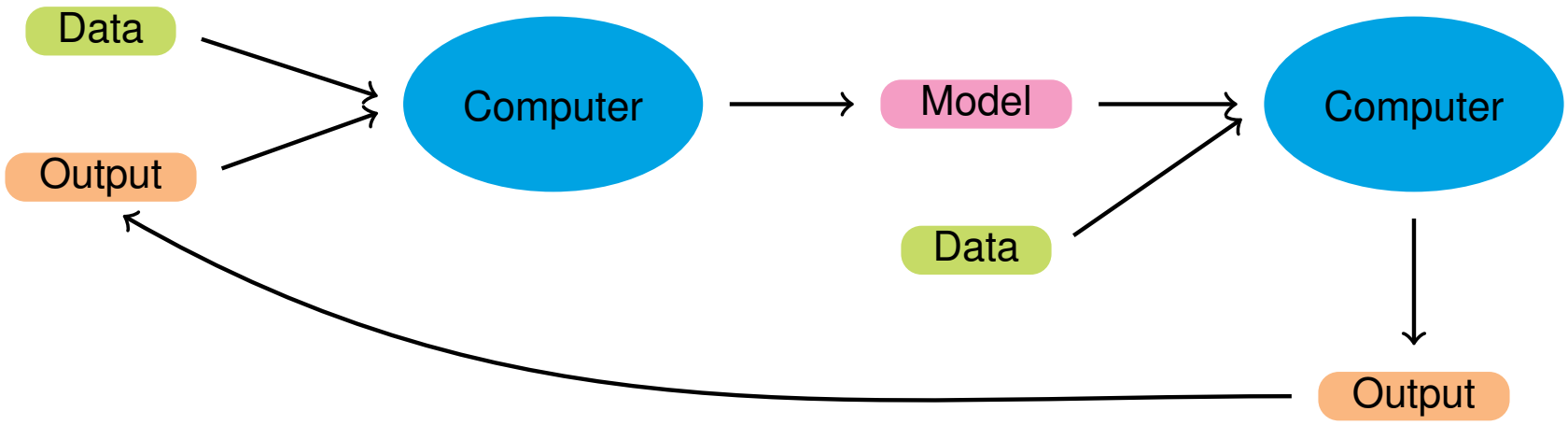
- we don't know WHY ML models make decisions
- inherent problem with ML models:
  they are hard (or impossible) to interpret
- therefore, it is crucial that our models are accurate and meaningful

- almost by definition one cannot really understand a good model.
-

A quick recap of ML basics

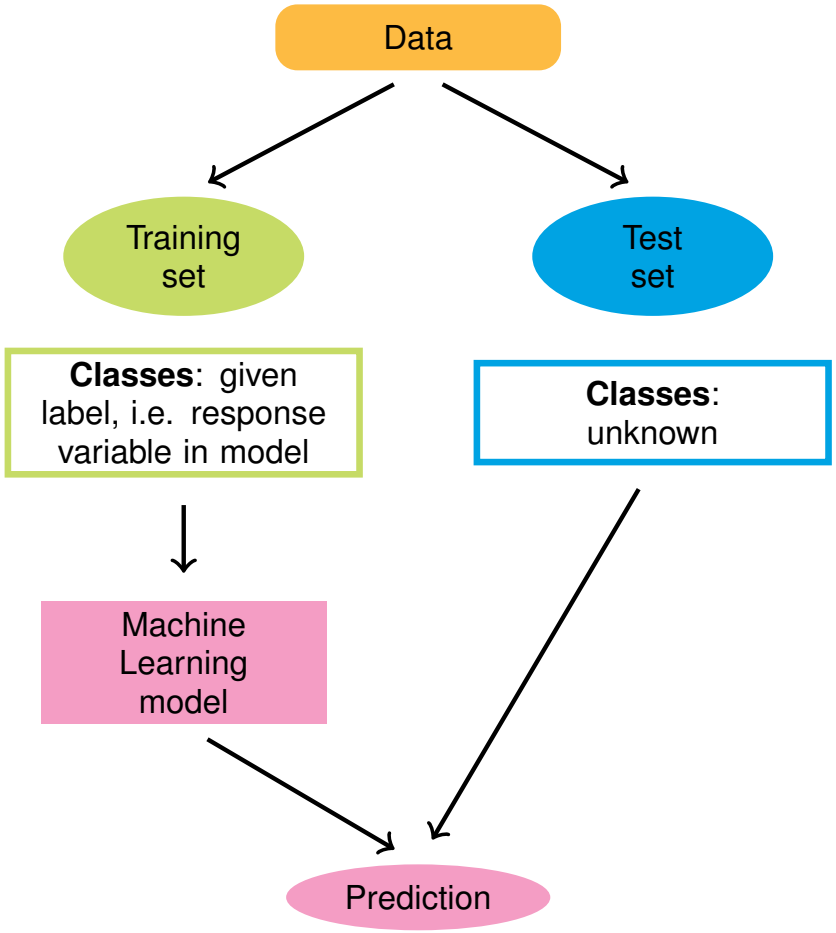
Machine learning

- artificial intelligence (AI)
- data-driven
- algorithms learn by being trained on observed data...
- ... and predict unknown data
- ML concepts are not new, but the increase in computational capacity has made them more accessible

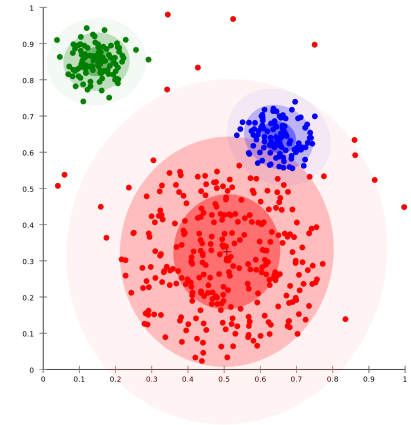


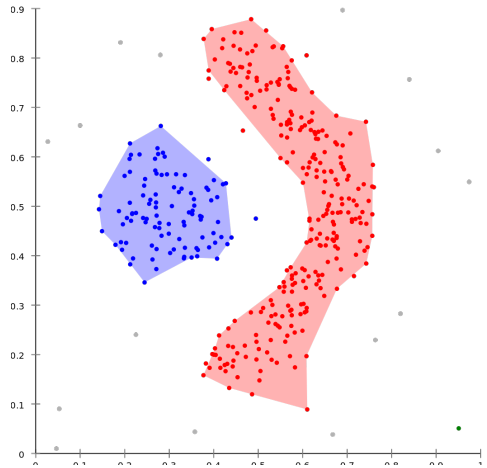
Supervised vs Unsupervised algorithms

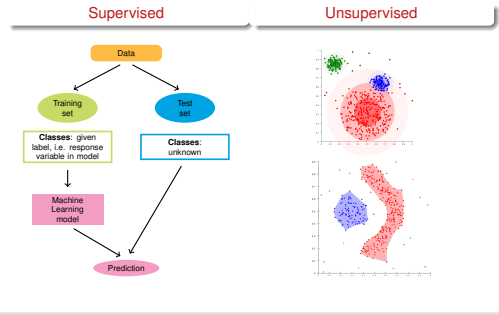
Supervised



Unsupervised







- Unsupervised:
- no input!
- matrix decomposition methods, e.g. nonnegative matrix factorization
- pattern recognition
- clusters data into different groups
- able to recover biomarkers of disease and toxicity

- Supervised
- classification labels are given
- e.g. support vector machine

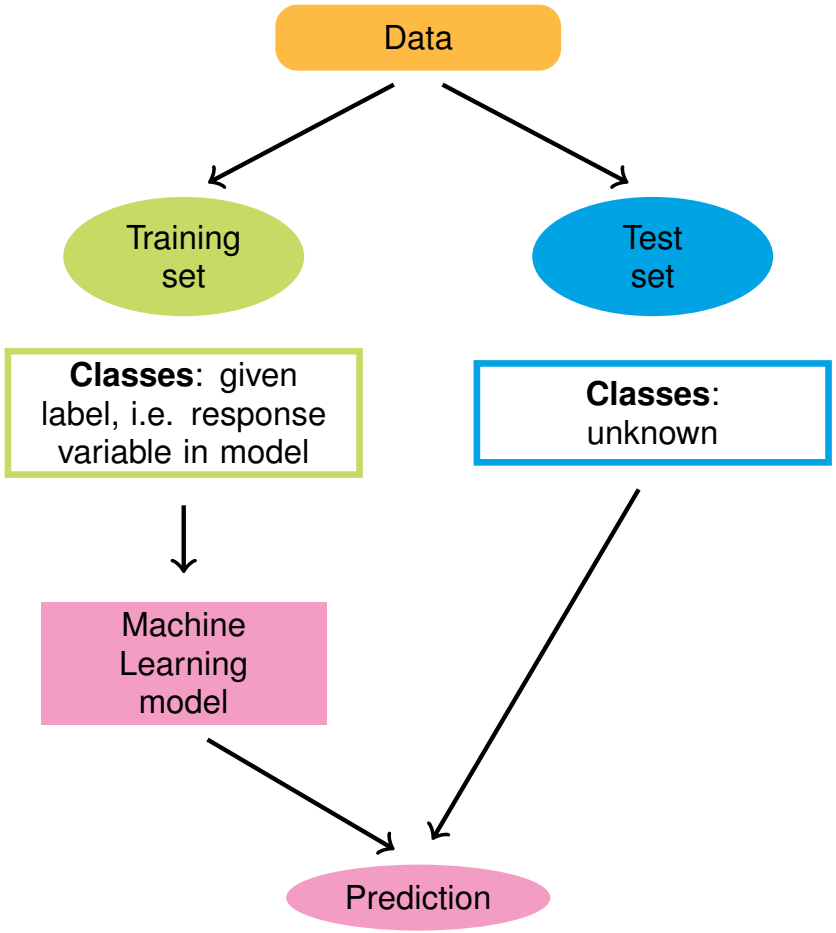
- Semi-supervised:
- a little bit of input, like how many classes to find

Here, I will focus on supervised learning!

Classification vs Regression

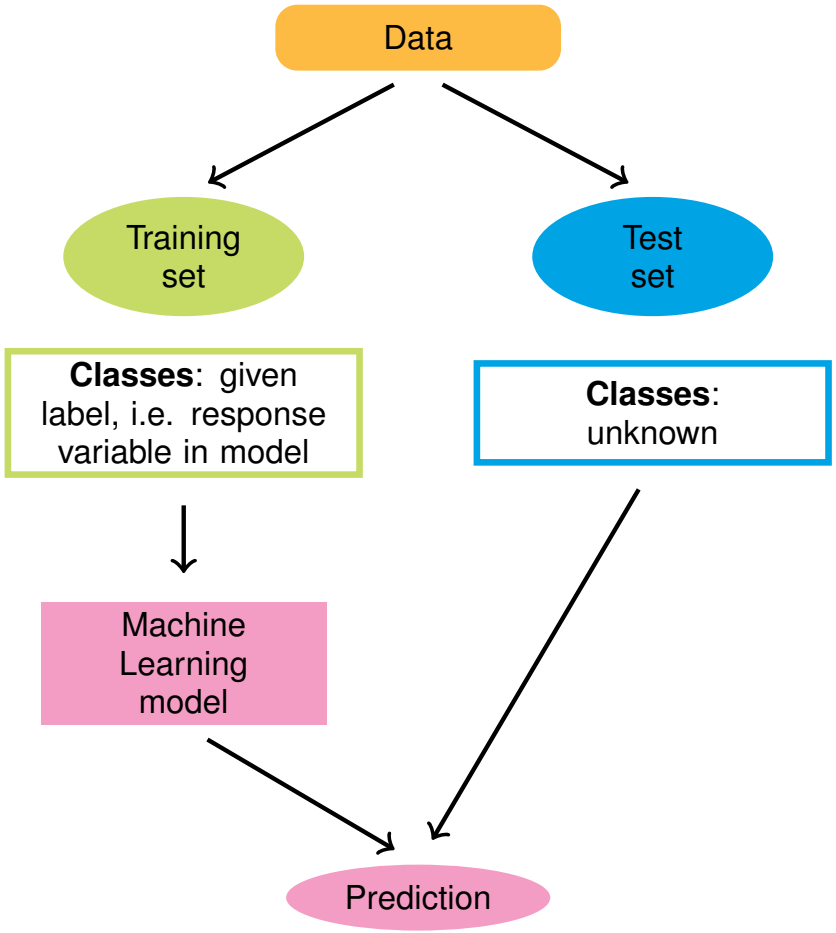
Classification

e.g. healthy vs disease

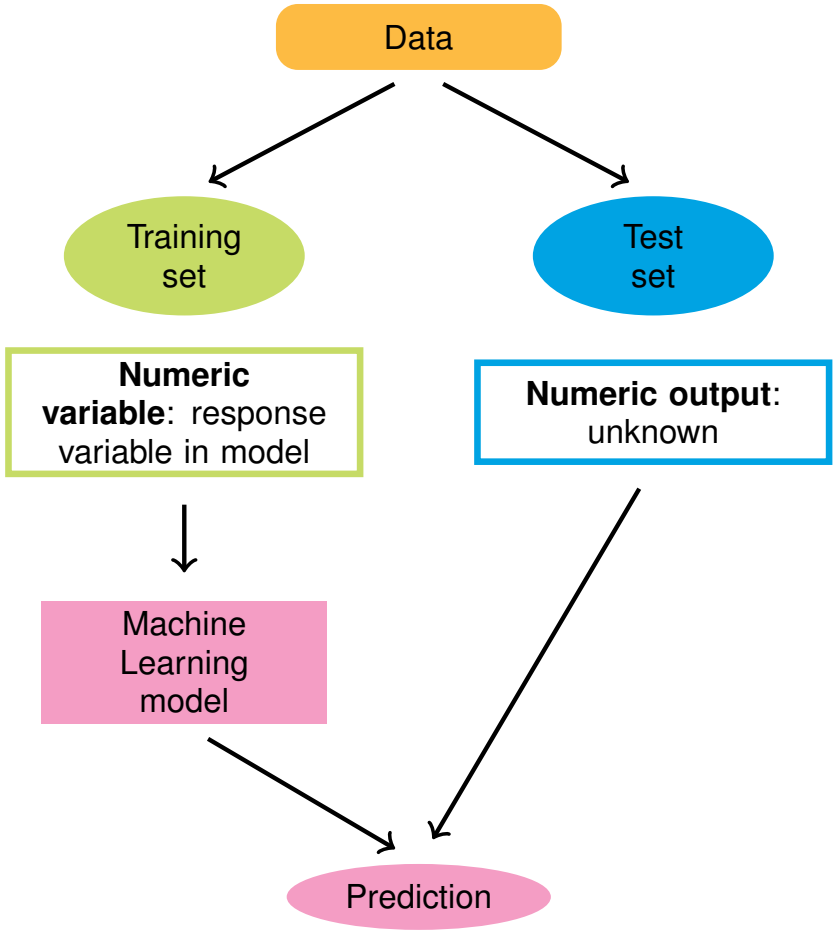


Classification vs Regression

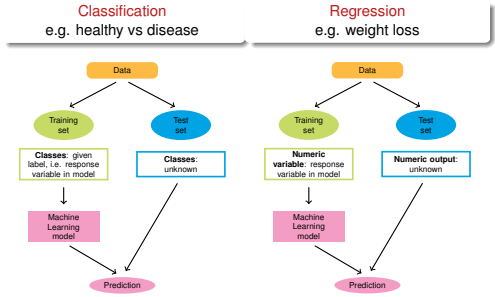
Classification
e.g. healthy vs disease



Regression
e.g. weight loss



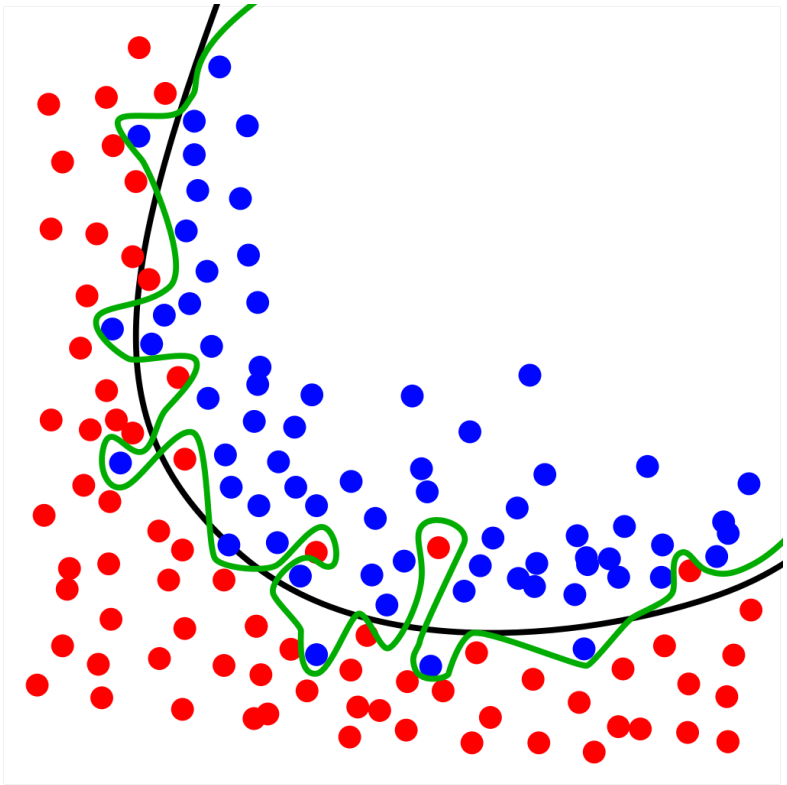
Classification vs Regression



- Supervised learning!
- Your learning algorithm seeks a function from inputs to the respective targets.
- If the targets are expressed in some classes, it is called classification problem.
- Alternatively, if the target space is continuous, it is called regression problem.

Features

- Features are the variables used for model training.
- Using the right features is crucial.

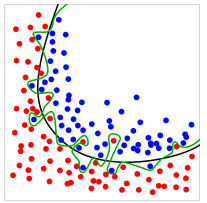


- More is not necessarily better (overfitting)!

- feature selection
- feature extraction/ engineering

Features

- Features are the variables used for model training.
- Using the right features is crucial.
- More is not necessarily better (overfitting)!
- feature selection
- feature extraction/ engineering



Machine learning uses so called features (i.e. variables or attributes) to generate predictive models. Using a suitable combination of features is essential for obtaining high precision and accuracy. Because too many (unspecific) features pose the problem of overfitting the model, we generally want to restrict the features in our models to those, that are most relevant for the response variable we want to predict. Using as few features as possible will also reduce the complexity of our models, which means it needs less time and computer power to run and is easier to understand.

extraction of salient structure in the data that is more informative than the raw data itself (the feature extraction

problem)

Training, (cross-) validation and test data

Training, (cross-) validation and test data

Decide cross validation strategy - To avoid overfitting, make sure you've set up a cross validation strategy in early stages. A nice CV strategy will help you get reliable score on leaderboard.
preventing overfitting, we want our models to be generalizable
Cross validation means that from my main set, I create RANDOMLY 2 sets. I built (train) my algorithm with the first one (let?s call it training set) and score the other (let?s call it validation set). I repeat this process multiple times and always check how my model performs on the test set in respect to the metric I want to optimize.
The process may look like:

For 10 (you choose how many X) times Split the set in training (50And validation (50Then fit the algorithm on the

training set Score the validation set. Save the result of that scoring in respect to the chosen metric. Calculate the

average of these 10 (X) times. That how much you expect this score in real life and is generally a good estimate.

Remember to use a SEED to be able to replicate these X splits Other things to consider is Kfold and stratified KFold .

Read here. For time sensitive data, make certain you always the rule of having past predicting future when testings.

Hyper Parameter Tuning

Grid Search

Start hyper parameter tuning - Once CV is at place, try improving model's accuracy using hyper parameter tuning. It further includes the following steps: Data transformations: It involve steps like scaling, removing outliers, treating null values, transform categorical variables, do feature selections, create interactions etc. Choosing algorithms and tuning their hyper parameters: Try multiple algorithms to understand how model performance changes. Saving results: From all the models trained above, make sure you save their predictions. They will be useful for ensembling. Combining models: At last, ensemble the models, possibly on multiple levels. Make sure the models are correlated for best results.

Take home messages:

- ...

How to build ML models in R

Session setup

- RStudio
- Breast Cancer Wisconsin Dataset
- caret[4]
- h2o[5]

Code will be available on my website and on Github

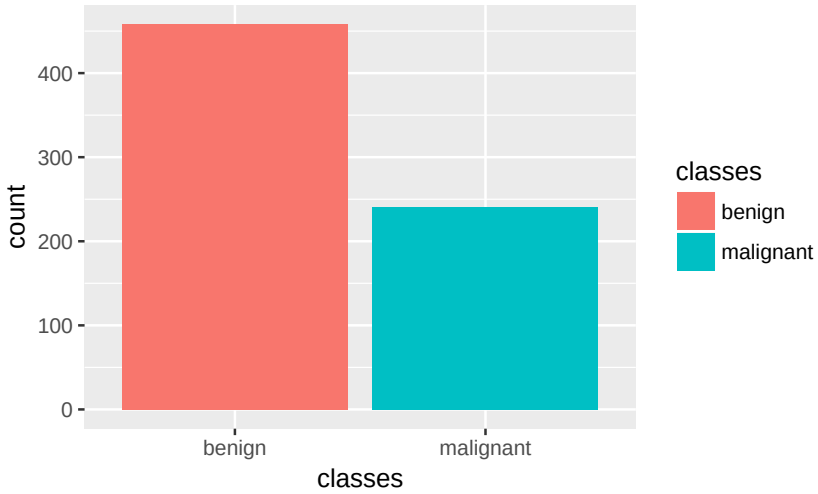
[4] M. Kuhn et al. (2016). *caret: Classification and Regression Training*. R package version 6.0-71.

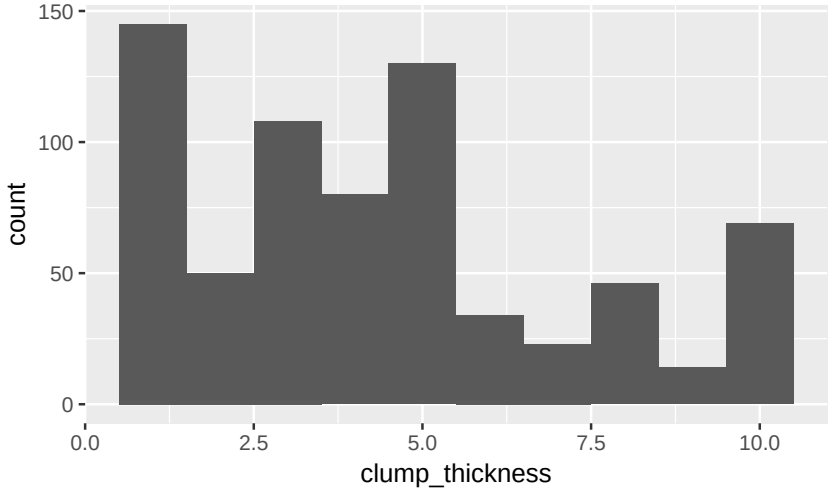
[5] H2O.ai (2017). *h2o: R Interface for H2O*. . R package version 3.10.3.6.

Get to know your data

Response variable

- Is it balanced?





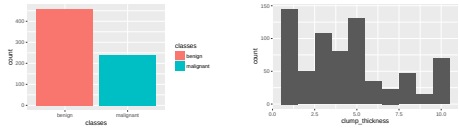
Missing data

- Is there missing data?
- Can we afford to loose data points with missing values?
- Or do we use imputation (and introduce additional uncertainty)?

Get to know your data

Response variable

- Is it balanced?



Missing data

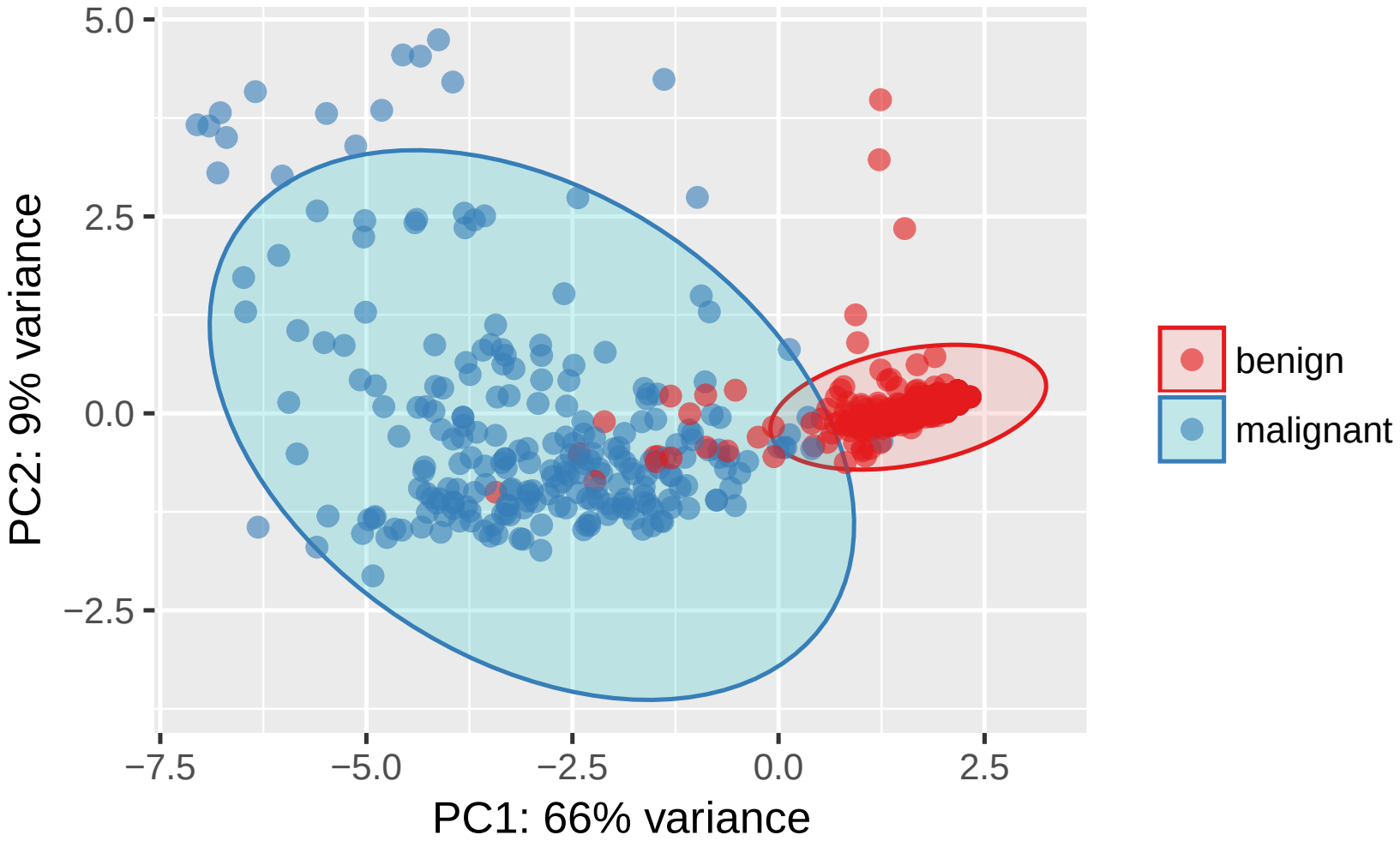
- Is there missing data?
- Can we afford to loose data points with missing values?
- Or do we use imputation (and introduce additional uncertainty)?

- Understand the data
- Look at data types.
- Check variable classes.
- Create some univariate - bivariate plots to understand the nature of variables.
- Most important to know is the distribution: are the classes balanced?
- If we have unbalanced data, this will effect our model later on
  We would have to consider over- or undersampling.

- Missing data
- If we have lots of data and few missing values, we can afford to loose these data points.
- If we don't have that much data though, our model will probably loose significant power if we remove the samples.
- In that case, we would rather introduce a certain uncertainty by imputing missing values.

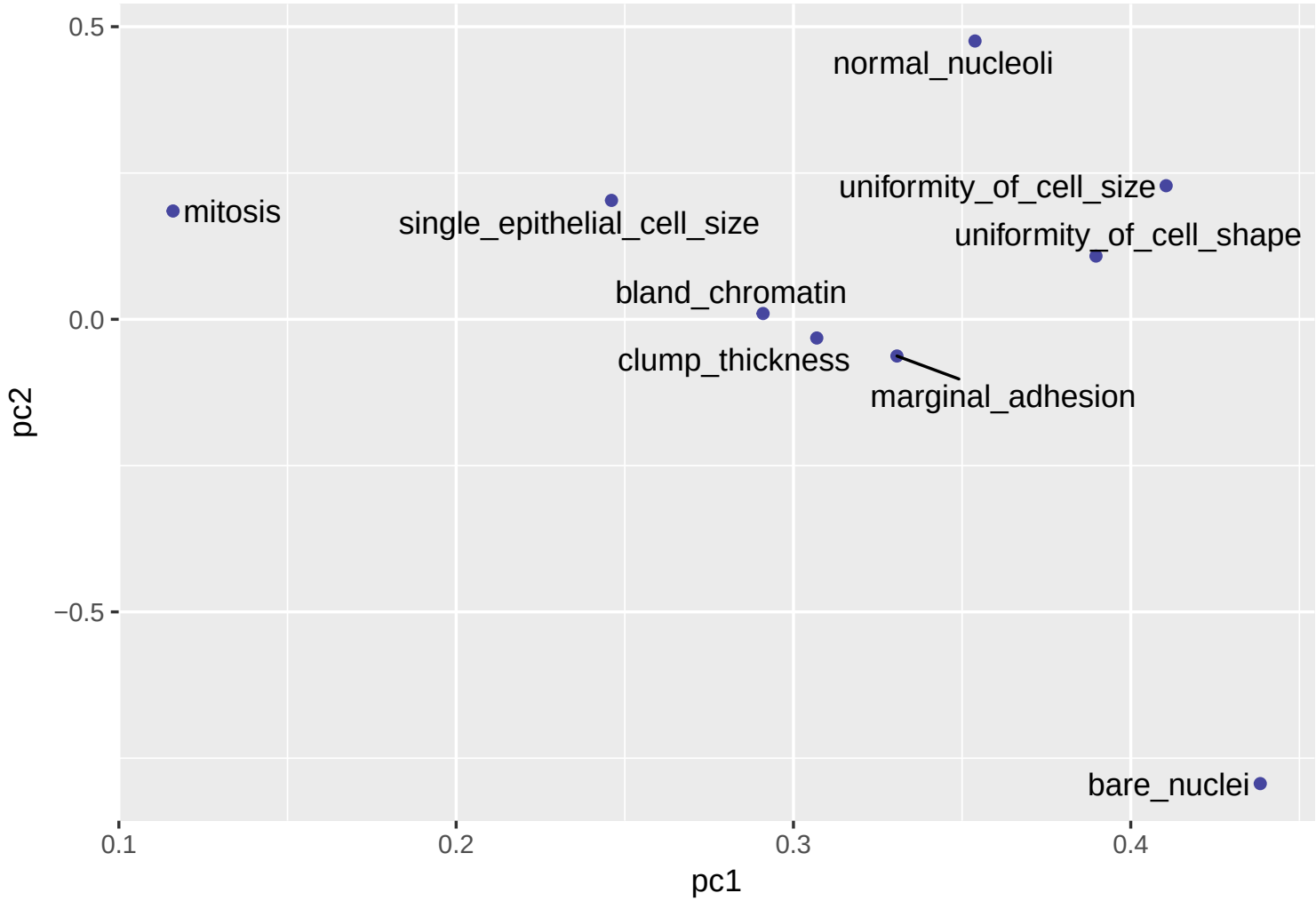
Get to know your data

Principal Component Analysis (PCA)

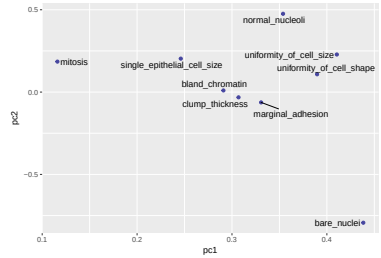


Get to know your data

Principal Component Analysis (PCA)



Principal Component Analysis (PCA)

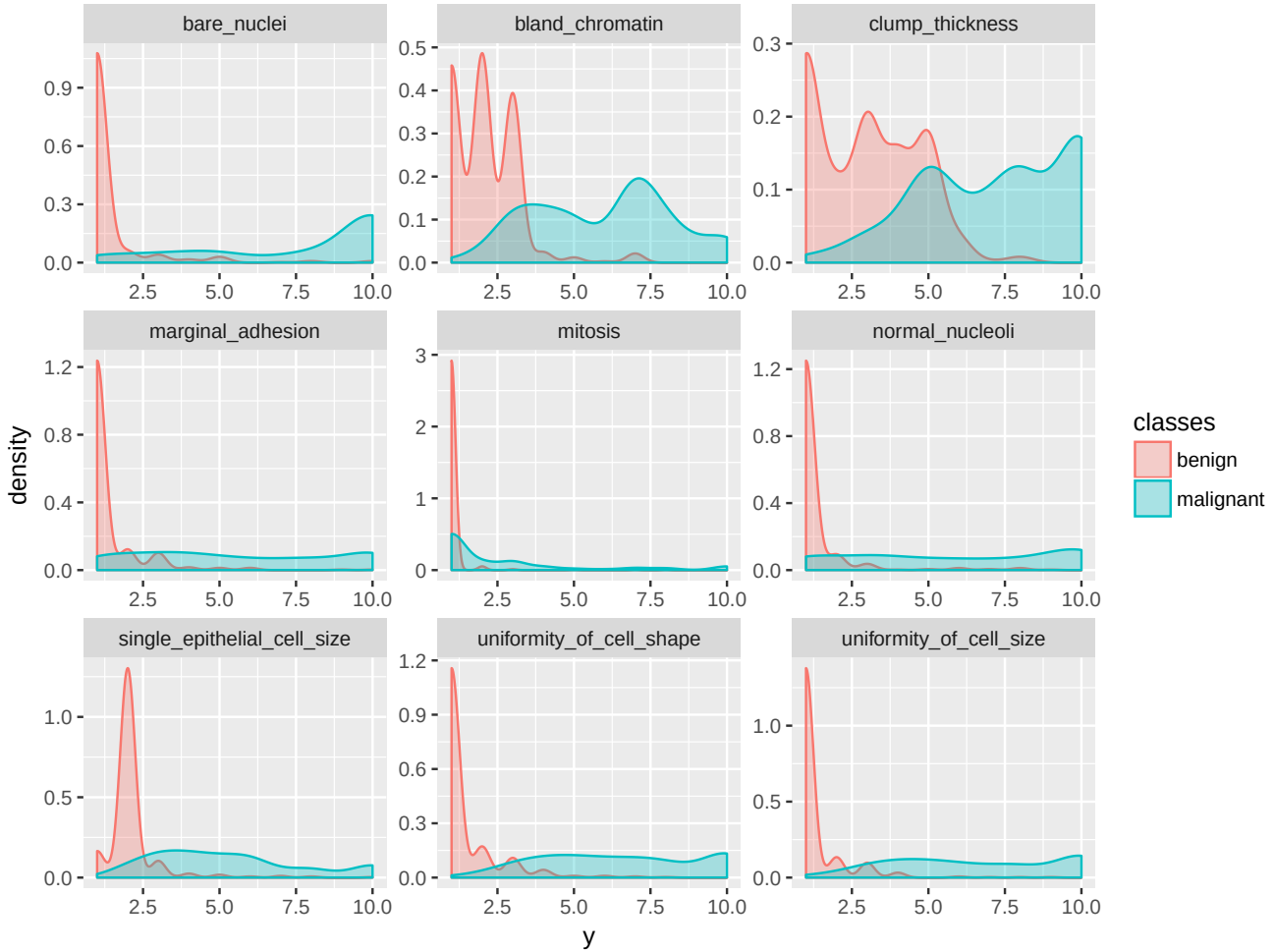


- To get an idea about the dimensionality and variance of the datasets, I am first looking at PCA plots
- for samples and features
- The first two principal components (PCs) show the two components that explain the majority of variation in the data

Get to know your data

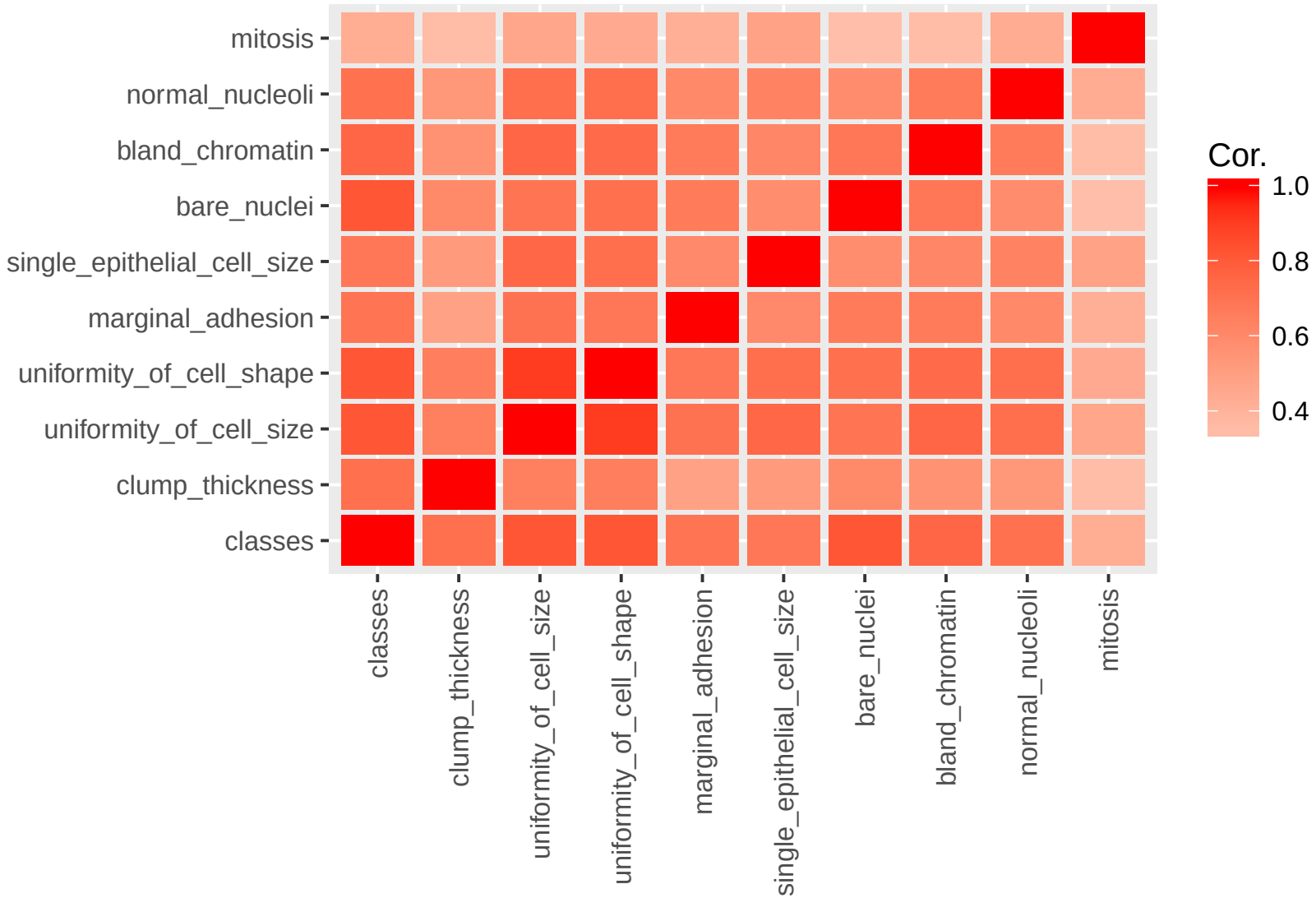
Features

- factors or numeric
- pre-processing



Get to know your data

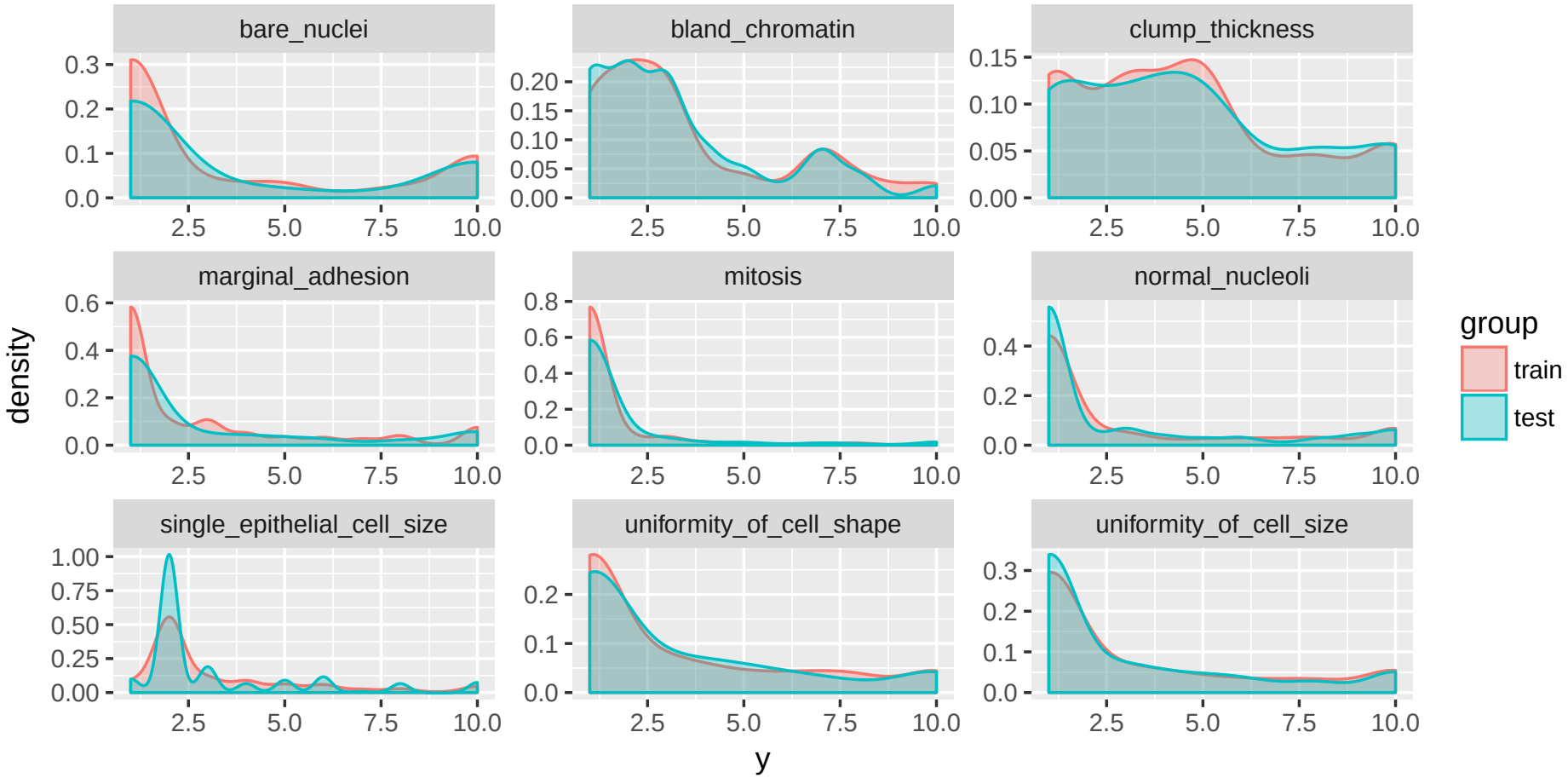
Correlation



Training, validation and test data

We need to split the data into training and test sets - ideally stratified by response class.

Density distribution

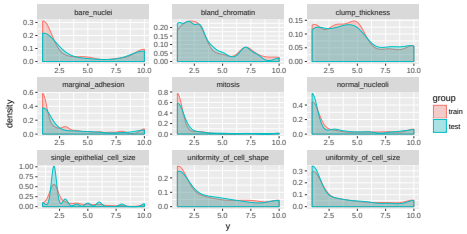


Training, validation and test data

Training, validation and test data

We need to split the data into training and test sets - ideally stratified by response class.

Density distribution

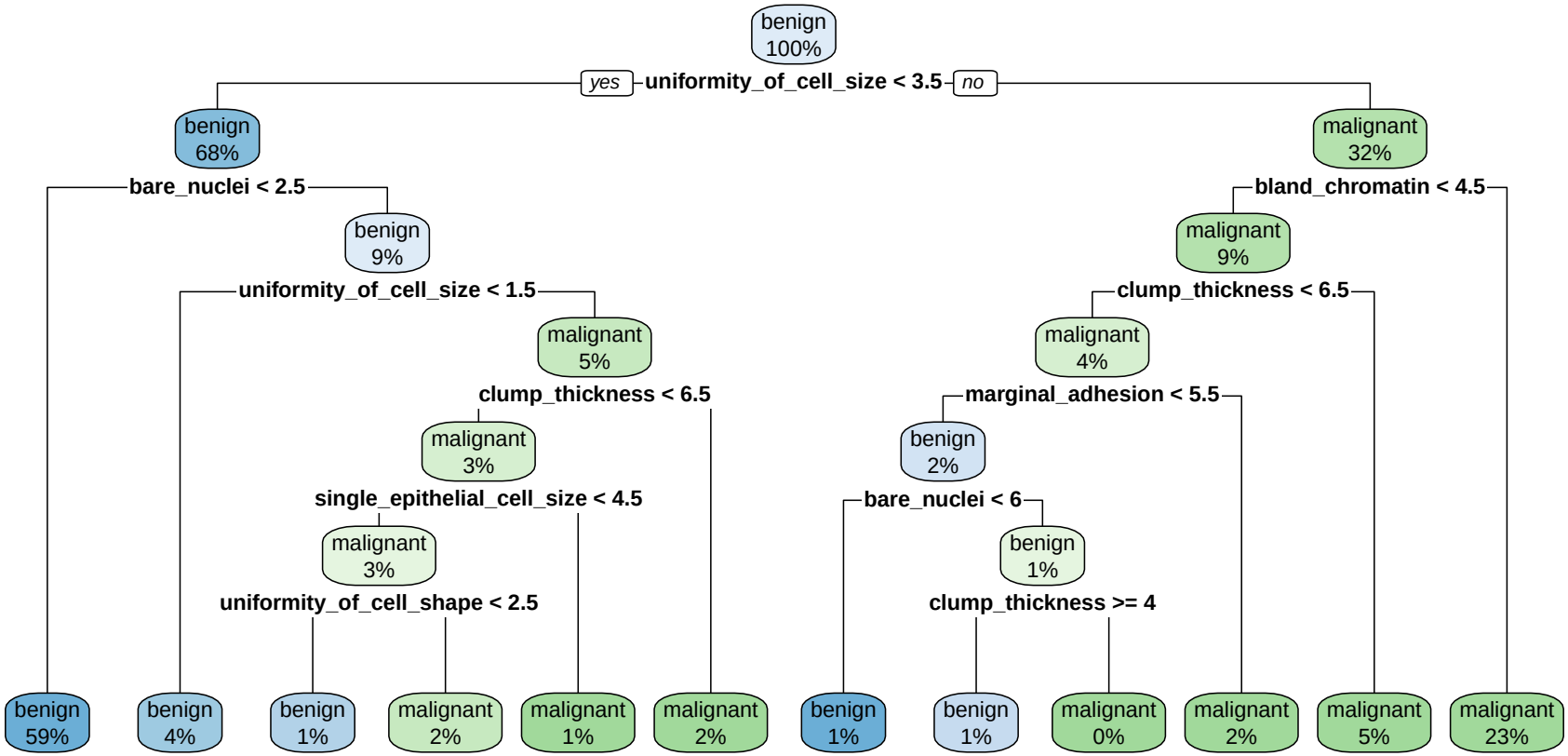


For accurate predictions, density distribution should be similar in training and test data!

Classification with tree-based models

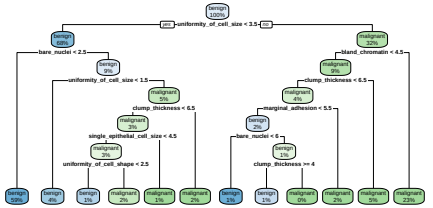
Decision trees

e.g. Random Forest and gradient boosting trees



Decision trees

e.g. Random Forest and gradient boosting trees



- We start with a group of samples
- for each sample, we assign a class: e.g. it comes from a benign or malignant tumor
- for each sample, we also have a number of features
- a decision trees separates the data at several nodes to end up with classifications at the final leaves
- the model learns a conditional structure of discriminative features

- Random Forests produces multiple decision trees
- with some level of randomness
- each tree is evaluated regarding how well it classified the samples (cross-validation)
- if two trees are equally accurate, the tree with fewer nodes is preferred
- 'information gain' is the measure that tells us how good a tree model is
- better generalization than individual trees

Tree-based classification

- Random Forest or Gradient boosting
- with *caret*[6]

```
caret::train(classes ~ .,
             data = train_data,
             method = "rf",  # or "xgb"
             preProcess = c("scale", "center"),
             trControl = trainControl(method = "repeatedcv",
                                      number = 10,
                                      repeats = 10,
                                      savePredictions = TRUE,
                                      verboseIter = FALSE))
```

```
##           benign malignant class.error
## benign       313         8  0.02492212
## malignant      4       165  0.02366864
```

[6] M. Kuhn et al. (2016). *caret: Classification and Regression Training*. R package version 6.0-71.

Tree-based classification

- Random Forest or Gradient boosting
- with *caret*[6]

```
caret::train(classes ~ .,
             data = train_data,
             method = "rf",  # or "xgb"
             preProcess = c("scale", "center"),
             trControl = trainControl(method = "repeatedcv",
                                      number = 10,
                                      repeats = 10,
                                      savePredictions = TRUE,
                                      verboseIter = FALSE))
```

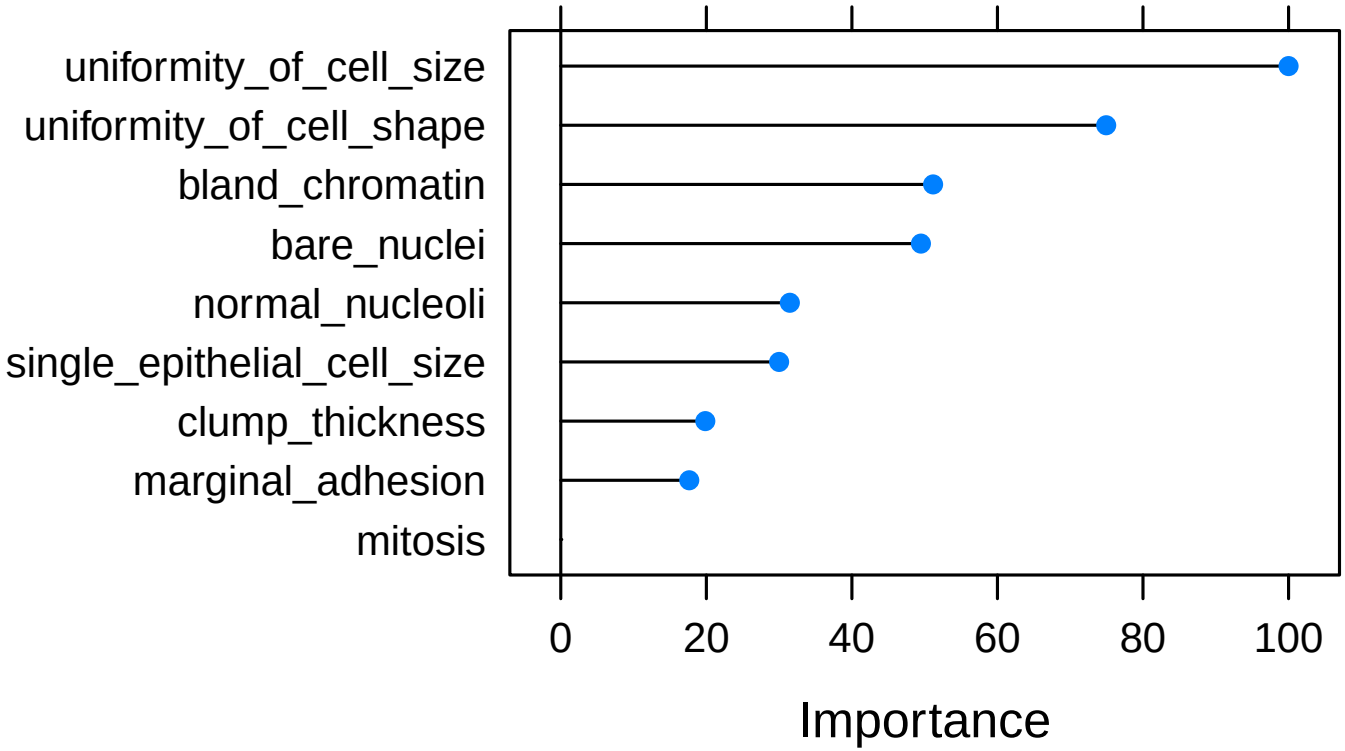
```
##            benign malignant class.error
## benign        313         8  0.02492212
## malignant       4       165  0.02366864
```

[6] M. Kuhn et al. (2016). *caret: Classification and Regression Training.* R package version 6.0-71.

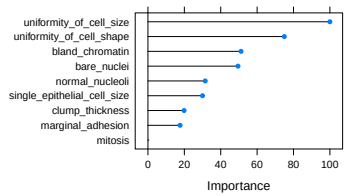
This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure.
train can be used to tune models by picking the complexity parameters that are associated with the optimal resampling statistics. For particular model, a grid of parameters (if any) is created and the model is trained on slightly different data for each candidate combination of tuning parameters. Across each data set, the performance of held-out samples is calculated and the mean and standard deviation is summarized for each combination. The combination with the optimal resampling statistic is chosen as the final model and the entire training set is used to fit a final model.

The predictors in x can be most any object as long as the underlying model fit function can deal with the object class. The function was designed to work with simple matrices and data frame inputs, so some functionality may not work (e.g. pre-processing). When using string kernels, the vector of character strings should be converted to a matrix with a single column.

Feature importance



Feature importance



Not all of the features I created will be equally important to the model. The decision tree already gave me an idea of which features might be most important but I also want to estimate feature importance using a Random Forest approach with repeated cross validation.

Regression with Linear Models

- Generalized Linear Models
- with *caret*[7]

```
model_glm <- caret::train(clump_thickness ~ .,
                          data = train_data,
                          method = "glm",
                          preProcess = c("scale", "center"),
                          trControl = trainControl(
                                        method = "repeatedcv",
                                        number = 10,
                                        repeats = 10,
                                        savePredictions = TRUE,
                                        verboseIter = FALSE))
```

[7] M. Kuhn et al. (2016). *caret: Classification and Regression Training*. R package version 6.0-71.

Hyper-parameter tuning

Grid Search

- with *h2o*[8]

```
hyper_params <- list(
                     ntrees = c(25, 50, 75, 100),
                     max_depth = c(10, 20, 30),
                     min_rows = c(1, 3, 5)
                     )

search_criteria <- list(
                        strategy = "RandomDiscrete",
                        max_models = 50,
                        max_runtime_secs = 360,
                        stopping_rounds = 5,
                        stopping_metric = "AUC",
                        stopping_tolerance = 0.0005,
                        seed = 42
                        )
```

[8] H2O.ai (2017). *h2o: R Interface for H2O*. . R package version 3.10.3.6.

Hyper-parameter tuning

Grid Search

- with *h2o*[8]

```
hyper_params <- list(
                     ntrees = c(25, 50, 75, 100),
                     max_depth = c(10, 20, 30),
                     min_rows = c(1, 3, 5)
                     )

search_criteria <- list(
                       strategy = "RandomDiscrete",
                       max_models = 50,
                       max_runtime_secs = 360,
                       stopping_rounds = 5,
                       stopping_metric = "AUC",
                       stopping_tolerance = 0.0005,
                       seed = 42
                       )
```

[8] H2O.ai (2017). *h2o: R Interface for H2O.* . R package version 3.10.3.6.

Of course, you need quite a bit of experience and intuition to hit on a good combination of parameters. That?s why it usually makes sense to do a grid search for hyper-parameter tuning. Hyper-parameter tuning with grid search allows us to test different combinations of hyper-parameters and find one with improved accuracy.
Keep in mind though, that hyper-parameter tuning can only improve the model so much without overfitting. If you can?t achieve sufficient accuracy, the input features might simply not be adequate for the predictions you are trying to model. It might be necessary to go back to the original features and try e.g. feature engineering methods.
We can use the h2o.grid() function to perform a Random Grid Search (RGS). We could also test all possible combinations of parameters with Cartesian Grid or exhaustive search, but RGS is much faster when we have a large number of possible combinations and usually finds sufficiently accurate models.
For RGS, we first define a set of hyper-parameters and search criteria to fine-tune our models. Because there are many hyper-parameters, each with a range of possible values, we want to find an (ideally) optimal combination to maximize our model?s accuracy. We can also specify how long we want to run the grid search for. Based on the results of each model tested in the grid, we can choose the one with the highest accuracy or best performance for the question on hand.

We now want to extract the best model from the grid model list. What makes a model the best depends on the question you want to address with it: in some cases, the model with highest AUC is the most suitable, or the one with the lowest mean squared error, etc. We first use the h2o.getGrid() function to sort all models by the quality metric we choose (depending on the metric, you want it ordered by descending or ascending values). We can then get the model that?s the first in the list to work with further. This model?s hyper-parameters can be found with . You can now work with your best model as with any regular model in h2o.

Evaluating model performance

Never use the same data for evaluation that you used for training!

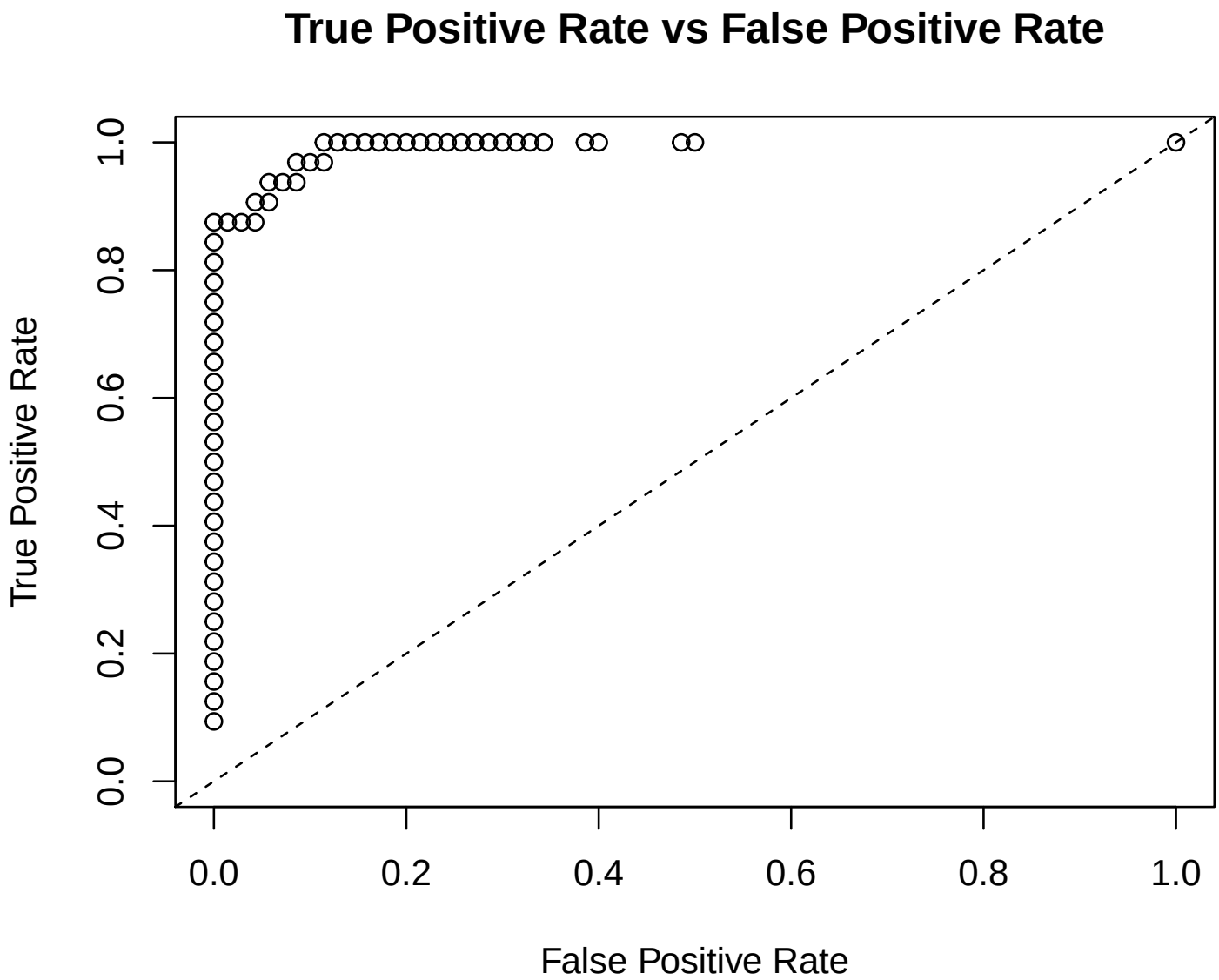
Never use the same data for evaluation that you used for training!

- Golden Rule of ML: always test model performance on indepenent data
- otherwise you will get overly optimistic performance measures

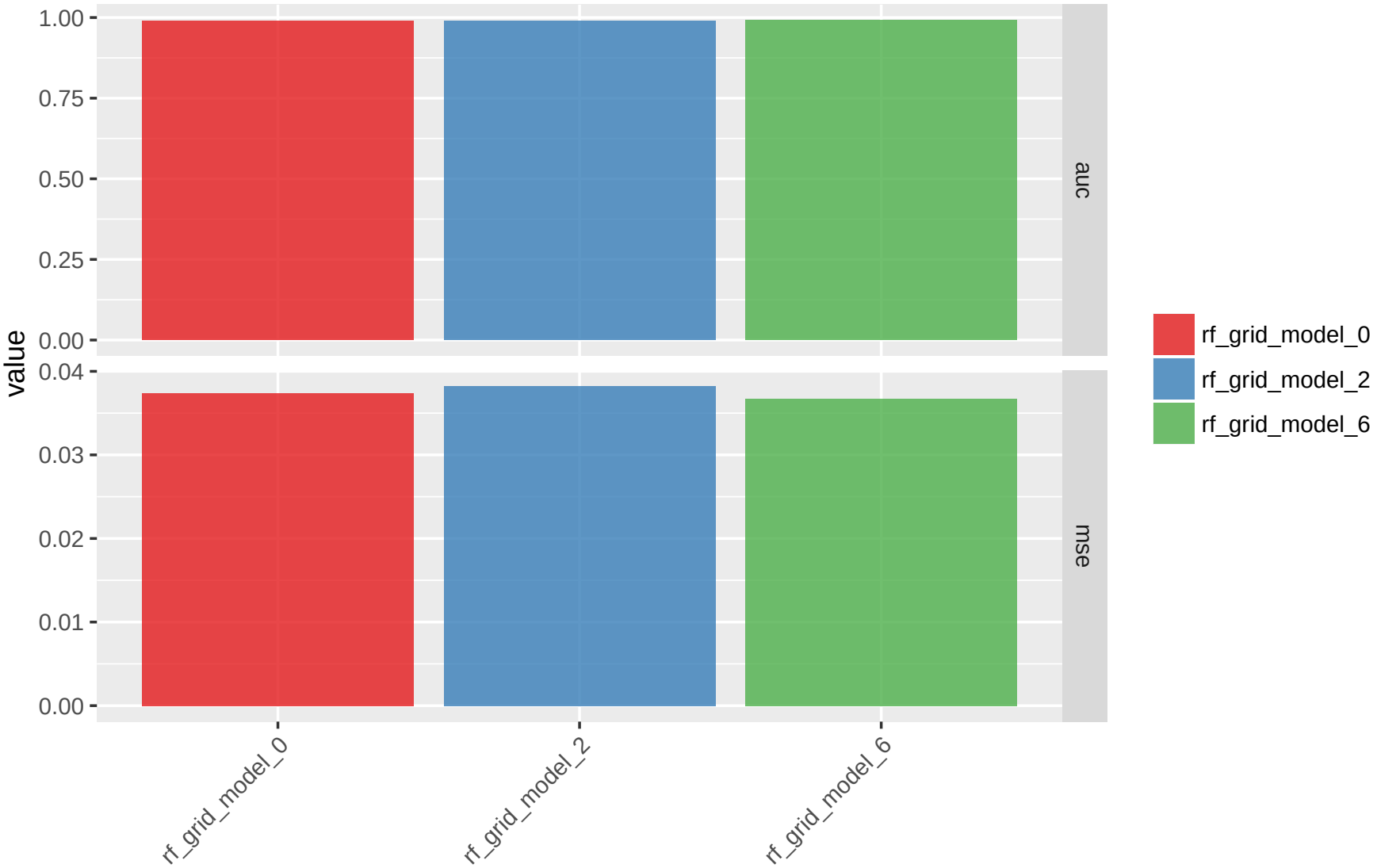
The ultimate performance test for our model will be it?s prediction accuracy on the test set it hasn?t seen before.

Here, I will compare the AUC and mean squared error for each best model from before. You could of course look at any other quality metric that is most appropriate for your model.

Area Under the Curve (AUC



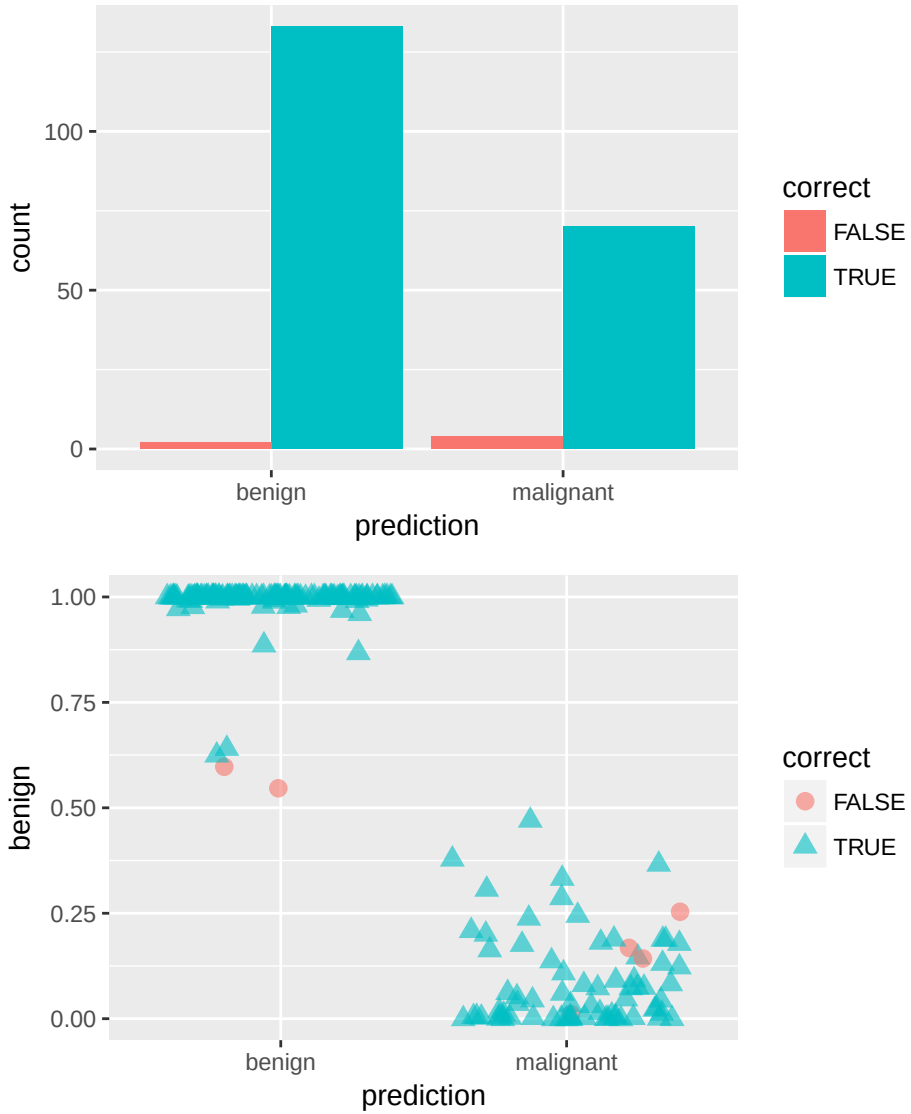
AUC and mean squared error (MSE)



Predictions on test data

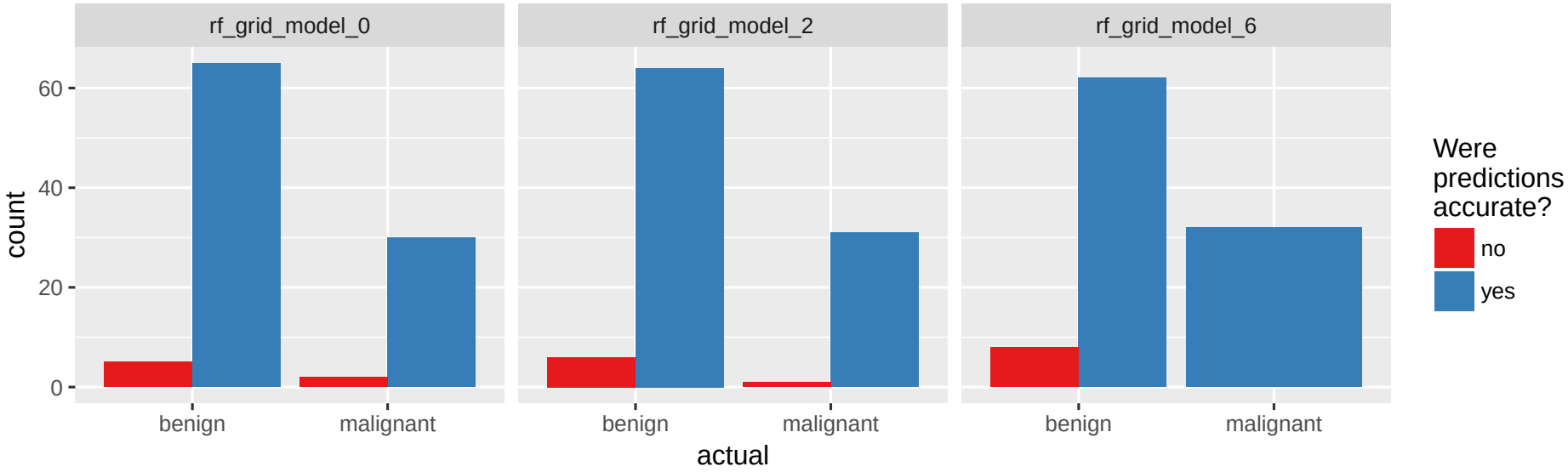
Classification

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       133         2
##   malignant      4        70
##
##                Accuracy : 0.9713
##                  95% CI : (0.9386, 0.9894)
##     No Information Rate : 0.6555
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9369
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9708
##             Specificity : 0.9722
##          Pos Pred Value : 0.9852
##          Neg Pred Value : 0.9459
##              Prevalence : 0.6555
##          Detection Rate : 0.6364
##    Detection Prevalence : 0.6459
##       Balanced Accuracy : 0.9715
##
##        'Positive' Class : benign
##
```

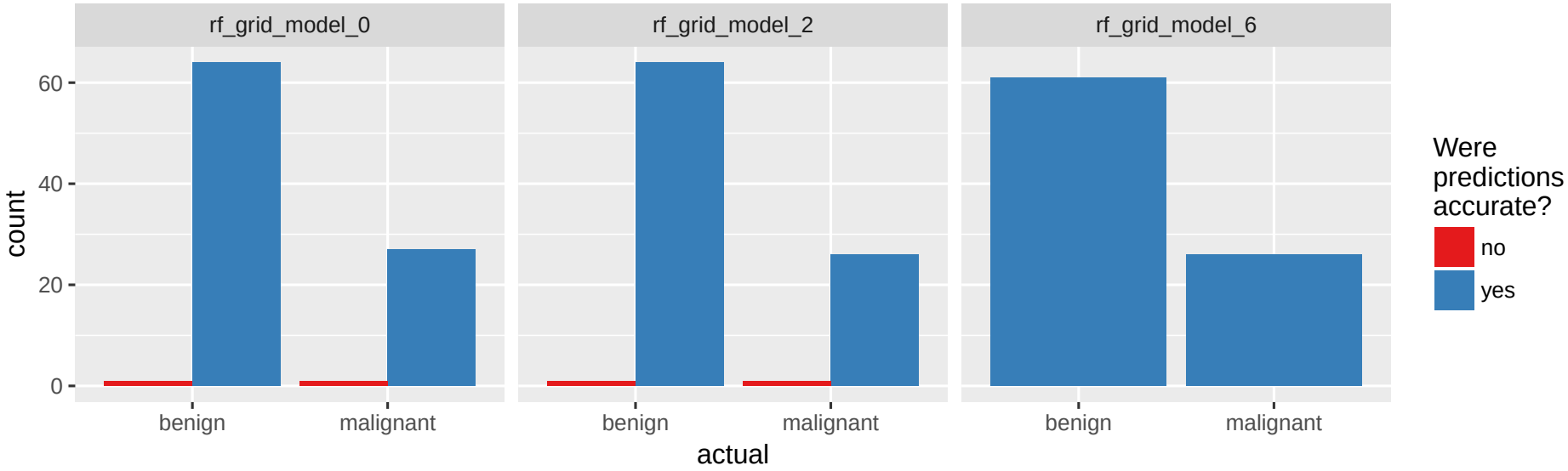


Predictions on test data

Default predictions



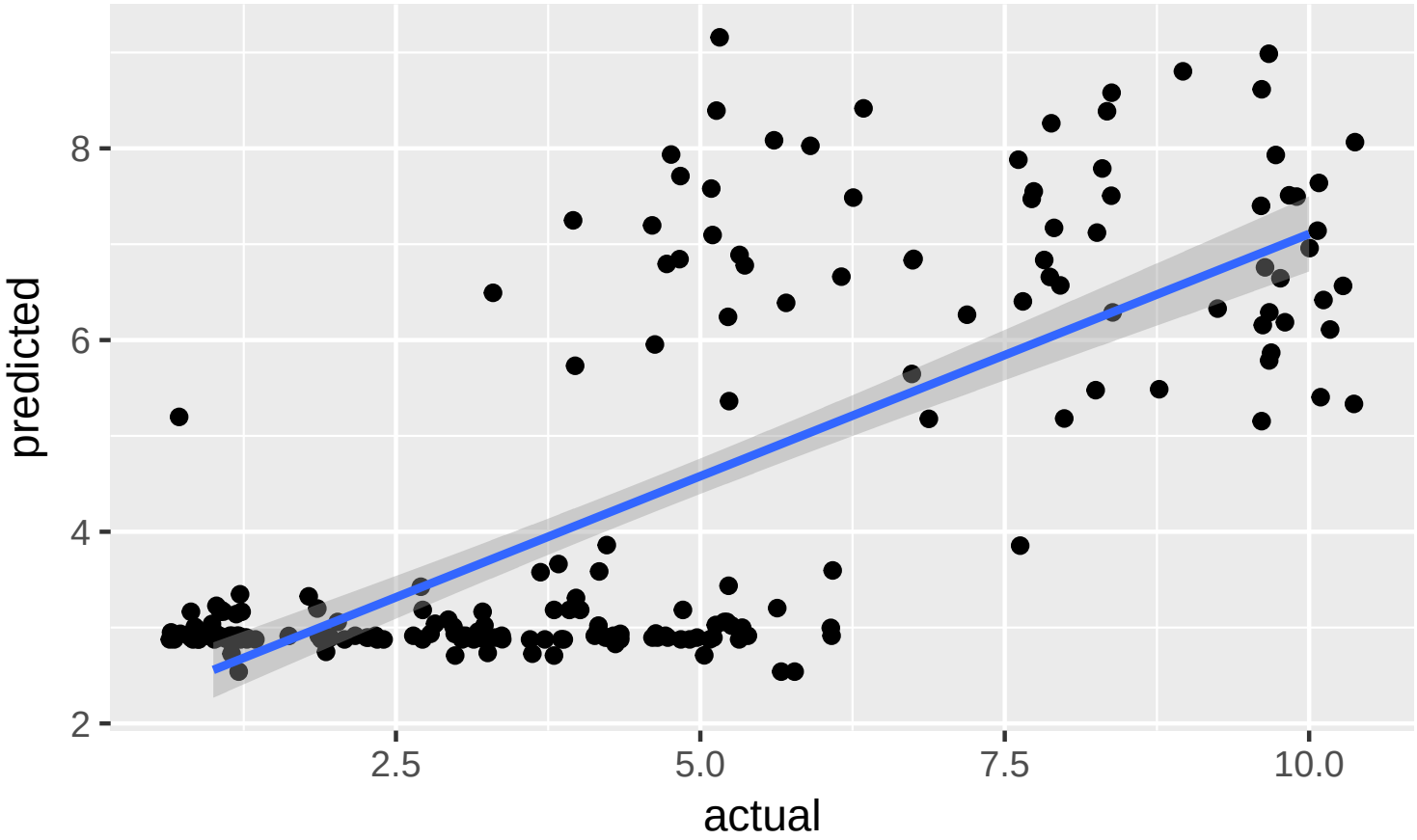
Stringent predictions



Predictions on test data

Regression

- RMSE: 1.974296
- Rsquared: 0.5016141



Take home messages:

- ...

Outlook

- ‘big data’ needs to be big!
- for really meaningful models, data needs to be shared
- the more data, the more accurate and generalizable the models will be
- issues: privacy, platform, quality standards
- ML could make health care more cost-effective by reducing the energy required for interpretation

Thank you for your attention!

Questions?

Slides and code will be available on Github:
https://github.com/ShirinG/Webinar_ML_for_disease

Code will also be on my website: https://shiring.github.io

shirin.glander@wwu.de