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!
- In this presentation, 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 using R
- I will demonstrate how to evaluate model performance and
- how to optimize models to address different disease-related questions using a case study
- all slides and the complete code behind the figures produced in this presentation will be available to you after this talk

About me

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 evolutionary genetics and immune memory in Drosophila

- I am also an R-blogger and dance teacher in my spare time

Table of contents

Table of contents

Building meaningful machine learning models for disease prediction

1. Machine Learning (ML) in disease modeling
2. What makes a model meaningful?
3. A quick recap of ML basics
4. How to build ML models in R
5. Evaluating model performance

- Before I begin, let me give you a short overview over the structure of this talk:
- I titled my talk: "Building meaningful machine learning models for disease prediction"
- with the emphasis on **meaningful**
- So, after an introduction to how machine learning is currently applied to disease modeling and clinical data science
- I want to introduce to you what I mean exactly when I talk about **meaningful models**
- Before I delve into the nitty-gritty of hands-on modeling, I will quickly recap the most important concepts of ML
- And finally, I will show how to build ML models in R
- and - most importantly - how to evaluate the performance of such models

ML in disease modeling

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

Examples:

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

Image source: Wikimedia Commons

[1] **Kunio**.
[2] **doi:10.1146/annurev.bioeng.8.061505.095802**.
[3] **Levenson**.

- Machine learning is a powerful approach for developing sophisticated, automatic, and objective models for the analysis of data
- Machine learning specifically promises to help physicians make better or faster 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.

- 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

- there is not really a place in medicine where AI doesn't have potential applications:
- more and more data is being collected that needs to be interpreted,
- meaning that diagnosis is increasingly data driven, e.g. in radiology.
- Moreover, ML allows the incorporation of data from heterogenous inputs:
- clinical, genomics, drugs, electronic health records, etc.
- to predict e.g. drug responses, relapses, etc. for individual patients
- This way, ML will be a step towards personalized or precision medicine

ML in disease modeling

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

Examples:

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

Image source: Wikimedia Commons

[1] **Kunio**.
[2] **doi:10.1146/annurev.bioeng.8.061505.095802**.
[3] **Levenson**.

- Examples for how ML is already being used in the medical field today come mainly from
- image recognition
- this is similar to training ML models to recognize images of cats (classification in images)
- This way, ML models can be trained to identify breast cancer, lung cancer, osteoporosis, brain tumors, etc. from medical images
- With Computer-Aided Diagnosis (CAD), radiologists use the computer output as a 'second opinion' and make the final decisions

- Another example is that facial analysis technology can help diagnose genetic disorders that often involve dysmorphic facial features, like trisomies
- Researchers at the University of Oxford in the United Kingdom have the developed
- Face2Gene, an app that uses machine learning.
- Doctors can upload a photo of patient's face to get a prediction for the type of genetic disorder he or she might have

- What I want to mention at the end is that ML is not meant to replace medical professionals!
- Doctors are still important!
- But the computer will do the tasks that we humans are not good at, like interpreting complex images.
- Today, we already collect more data than humans can manage to make sense of (e.g. genomics data),
- so the clinicians will be freed up to think about best treatment options and talk more with the patients
- Moreover, good models are built on strong knowledge of the question and the biology behind it
- and features need to be evaluated in context

Can we trust ML models?

- most ML algorithms model high-degree interactions between variables
- ML models are hard (or impossible) to interpret!
- we often don't know WHY they make decisions
- therefore, it is crucial that our models are meaningful

Image source: Pixabay

- I want to begin with the arguably central question concerning ML:
- Can we trust them?
- I consider this kind of the elephant in the room when we hear all the amazing things that ML models can do, like how th beat doctors in diagnosing cancers, etc.
- And when we test models on known data, it is easy to be wowed by high prediction accuracy.
- But what we really want, is to be able to deploy our models to truly unknown data and still be confident that we can trus them
- Trust is not an easy thing to achieve, because the inherent problem with ML models is that they are hard (or impossible to interpret
- This is because ML algorithms model high-degree interactions between variables
- meaning the effect of combining many variables together
- This way, they tend to create nonlinear,
- non-monotonic,
- non-polynomial,
- and even non-continuous functions
- that approximate the relationship between independent and dependent variables in a data set
- This complexity, which bestows the extraordinary predictive abilities on machine learning algorithms also makes the answers these algorithms produce difficult (or even impossible) to understand.
- and we humans, especially non-experts in modeling, will have a hard time trusting a model that they don't understand
- difficulties in interpretation still present maybe the biggest barrier for the widespread, practical use of ML in clinical settings

Can we trust ML models?

- most ML algorithms model high-degree interactions between variables
- ML models are hard (or impossible) to interpret!
- we often don't know WHY they make decisions
- therefore, it is crucial that our models are meaningful

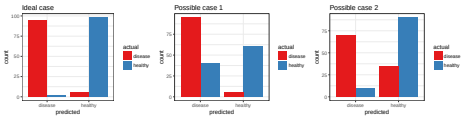
Image source: Pixabay

- So, when working with data we should be asking ourselves some very hard questions:
- do I understand my data?
- Do I understand the model and the answers my machine learning algorithm is giving me?
- And do I trust these answers?
- trusting models and results is a general requirement for good (data) science
- so we need to make sure, that we are modeling is actually 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!



- So, what makes a model 'meaningful'?
- With a little bit of practice, most anybody can build machine learning models.
- What is hard though, is to build 'good' or 'meaningful' models

1. This means that the model needs to be generalizable, i.e. it needs to perform well on unseen data
2. It needs to answer a **specific** question or address 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 needs to produce a correct prediction (such as a diagnosis) with high enough accuracy that we trust it!

- We therefore need to evaluate model accuracy when we want to establish ML for a specific task
- 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 about treatment

- But what exactly denotes a *high enough accuracy* can not be defined with a one-size-fits-all approach:
- accuracy depends on the problem we want to model because
- a model needs to perform well in the real world, not in the lab on training data

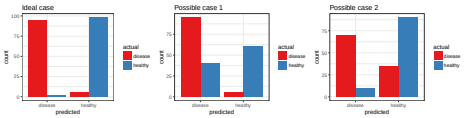
What makes a model meaningful?

- 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!

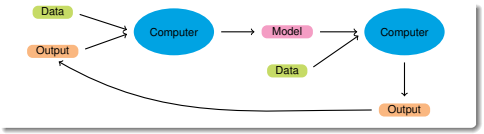


Let me illustrate what I mean with the following examples:

- 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

- artificial intelligence (AI)
- data-driven
- algorithms learn by being trained on observed data...
- ... and predict unknown data
- the increase in computational capacity has made ML more accessible



- ML is a subfield of computer science
- that falls in the general category of AI
- In ML, algorithms learn models from known data
- and are then able to use this inferred knowledge for predicting unknown data
- ML concepts are not new, but the increase in computational capacity and the bigger amounts of data that we are able t collect in shorter periods of time, has made them more accessible and applicable

Supervised vs Unsupervised learning

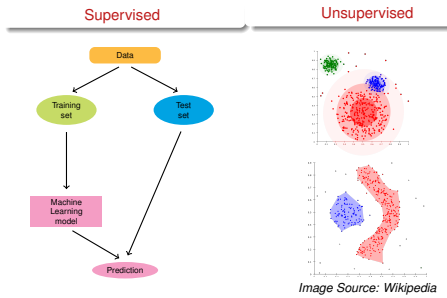


Image Source: Wikipedia

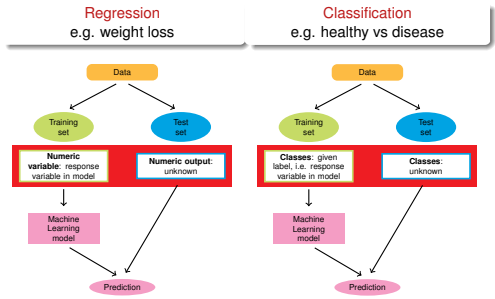
- ML is generally divided into two categories: Supervised and unsupervised
- even though there are also hybrid methods, called semi-supervised

- In supervised learning,
- classification labels are assigned to each training sample
- these labels are then used as output variables for the model
- Examples are SVM or decision trees

- Unsupervised learning on the other hand does not require an input, like class lables
- Methods, like matrix decomposition, e.g. nonnegative matrix factorization
- basically perform pattern recognition and
- cluster similar data points into groups

- Semi-supervised methods require a little bit of input, like how many clusters or classes to find in the data

Here, I will focus on supervised learning!



- Within supervised learning, the two main task are classification and regression.
- Let's say your learning algorithm seeks a function that predicts a response variable from input variables.
- If the response variable is expressed as classes, it is a classification problem.
- Alternatively, if the response variable space is continuous, it is a regression problem.

- Functions created by linear regression algorithms are probably the most interpretable class of models.
- For this reason, linear models were the go-to applied predictive modeling tool for decades,
- even though this usually meant obtaining a somewhat lower accuracy could have been achieved with more complex models.
- In linear models, for every change in any given independent variable, the response function changes at a defined rate,
- in only one direction, and at a magnitude represented by a readily available coefficient

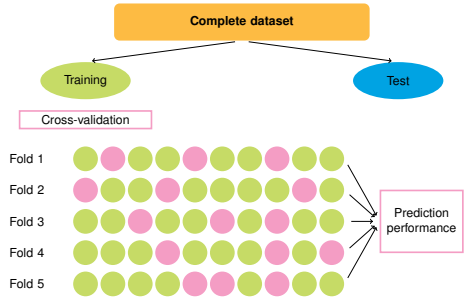
- Most machine learning algorithms however, create nonlinear response functions.
- These more complex models are the most difficult to interpret, because they can change in a positive and negative direction and at a varying rate for any change in an independent variable.
- Typically, the only standard interpretability measure these functions provide are relative variable importance measures

Features

- 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 (sometimes also called variables or attributes) to generate predictive models
- Using a suitable combination of features is essential for obtaining high 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.
- Sometimes extraction of salient structure in the data that is more informative than the raw data itself
- this describes the feature extraction or feature engineering problem

Training, cross-validation and test data



- A central aspect of modeling is to make them generalizable and avoid overfitting
- to achieve this, we can want to train our model on a dataset that separate from a test set
- in addition, we can add cross-validation strategies to the modeling process itself
- CV means that the main training set is randomly subdivided into 2 sets:
- one for actual training and one for validation
- the model is then trained on the training subset and predictions on the validation-hold-out-set are used to calculate and accuracy score
- this process is repeated several times (e.g. 5-fold)
- the prediction performance of the cross-validation can be used to optimize modeling

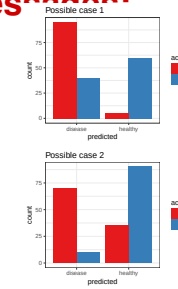
Take home messages:

- ML models learn on observed data
- and predict unknown data

- creating ML models is easy
- creating good models is hard

Meaningful models

- answer specific questions
- are able to generalize to unseen data
- can be trusted



- Here, I want to briefly summarize the most important aspects from my talk so far
- We will now look at a hands-on example of ML modeling in R
- When doing the actual work, I know from experience that it is easy to get caught up in the nitty gritty of writing code and trying out different things
- but I want to encourage you to try and not loose sight of the bigger picture

Session setup

- Breast Cancer Wisconsin Dataset[4]

Image Source: Wikipedia

- caret[5]
- h2o[6]

Code will be available on my website and on Github

[4] **Wolberg01121990**.
[5] **caret**.
[6] **h2o**.

- First, a few words about the logistics:
- All analyses have been done in R 3.3.3 using RStudio 1.0.136
- As case study example, I am using the breast cancer dataset from Dr. William H. Wolberg of the University of Wisconsi Hospitals, Madison

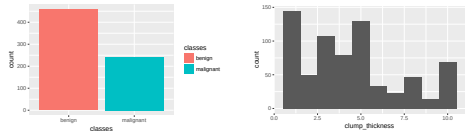
- It looks at the predictor classes:
- malignant or
- benign breast mass

- The features characterise cell nucleus properties
- and were generated from image analysis of fine needle aspirates (FNA) of breast masses:
- Sample ID (code number)
- Clump thickness
- Uniformity of cell size
- Uniformity of cell shape
- Marginal adhesion
- Single epithelial cell size
- Number of bare nuclei
- Bland chromatin
- Number of normal nuclei
- Mitosis
- Classes, i.e. diagnosis

Get to know your data

Response variable

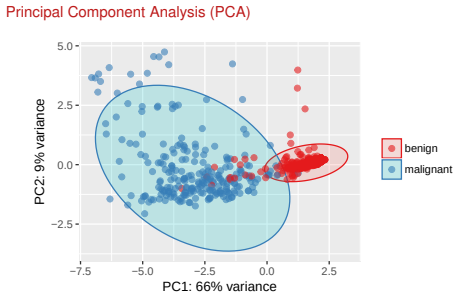
- Is it balanced?



Missing data

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

- Before begin thinking about machine learning, we should get to know our data
- This is crucial to understanding the data and later on the model and it's interpretations
- A few basic things you can do are: Look at data types (categorical vs continuous, etc.)
- Check variable classes (what is good outcome to train on).
- Two specific things that are important to know for ML are:
- 1) 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.
- the two plots show on the left: the data used to demonstrate classification, where we want to predict the classes 'benig or 'malignant'
- the right plot show the example variable used to demonstrate regression: clump thickness
- 2) Missing data is another important aspect for modeling
- 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 some uncertainty by imputing missing values.



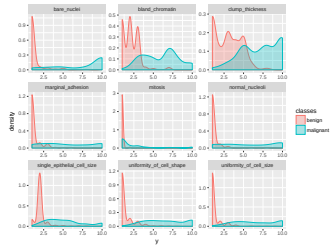
- Most real data sets are hard to see because they have many variables and many rows
- visualization can help when we reduce the complexity
- There are many techniques for projecting the rows of a data set from a high-dimensional original space to a more visua understandable lower-dimensional space:
- Principal Component Analysis (PCA)
- Multidimensional Scaling (MDS)
- t-distributed Stochastic Neighbor Embedding (t-SNE)
- Each of these techniques has strengths and weaknesses, but the key idea they all share is to represent the rows of a data set in a meaningful low-dimensional space.

- To get an idea about the dimensionality and variance of the datasets, I am first looking at PCA plots
- Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation ...
- ... to convert a set of observations into a set of values of linearly uncorrelated variables called principal components
- The first two principal components (PCs) show the two major trends in the data that explain the majority of variation in t data
- PCA can be done on any data matrix. Here I color the points by class label.
- From this plot, you can already see that there is a general trend in the total dataset that samples from the two classes a distinguishable: the majority of variation comes from whether a sample came from a benign or malignant mass

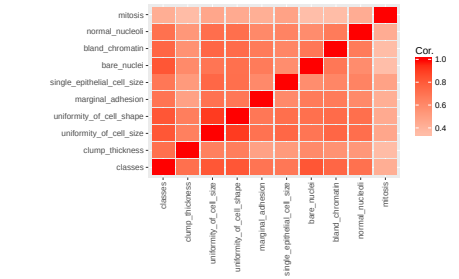
Get to know your data

Features

- factors or numeric
- pre-processing



- Next, we want to get an idea about the features.
- By simply plotting the distribution, we can already get a feeling of how good they are for telling apart our two class labels.
- Here, we only have a small dataset with 9 features
- If you have bigger dataset, this might of course not be practical to look at individually any more.
- But here, we can nicely see that malignant samples seem to have a much broader range of values in all features than the benign samples
- Based on this, we could already assume that a sample with a generally higher value in most features is likely malignant



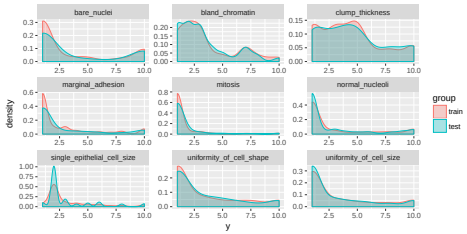
- Correlation between features is also of interest.
- features with a high correlation might be regulated by common factors
- on the other hand, highly correlated features will also give redundant information and we might want to consider either leaving only one of these features or combining them into a new engineered feature.
- features that are highly correlated with the classes are also probably features of high importance for the prediction model later on
- here, I show a correlation heatmap of Pearson correlation coefficients
- we can directly see that uniformity of cell shape and cell size are very highly correlated
- and mitosis is the one feature with comparably low correlation - also with the classes, so it likely will not contribute much in terms of predictive power

Training, validation and test data

Training, validation and test data

Splitting the data into training and test sets - ideally stratified by response class.

Density distribution

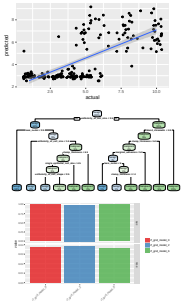


- We now want to split the data randomly into training and test sets
- the density distribution should be similar in training and test data!

Model examples
Regression with Linear Models
- e.g. Generalized Linear Models
- with *caret*

Tree-based classification
- Random Forest or Gradient boosting trees
- with *caret*

Hyper-parameter tuning
- Grid Search
- with *h2o*

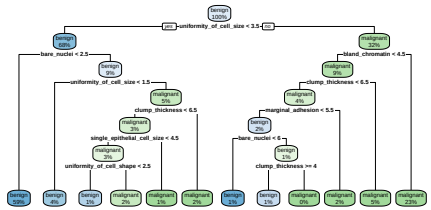


- Now we can go ahead with actual modeling:
- There are over 250 ML algorithms implemented in caret alone.
- But I am going to focus on two of the most widely used: GLM and RF

- traditional linear models tend to create linear, monotonic, and continuous functions
- Even though they're not always the most accurate predictors, the simplicity of linear models makes the results they generate easy to interpret.
- Random Forests are decision tree based algorithms and I will explain them in more detail in a minute

- Of course, you need quite a bit of experience and intuition to hit on a good combination of parameters for modeling.
- That's why it usually makes sense to do a grid search for hyper-parameter tuning.
- It allows us to test different combinations of hyper-parameters and find one with improved accuracy.

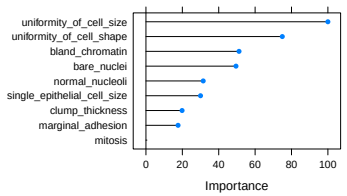
Decision trees



- Decision trees are the basis for Random forest and boosting trees
- One decision tree is built the following way:
- We start with a group of samples
- for each sample, we assign a class: e.g. it being from benign or malignant
- for each sample, we also have a number of features
- a decision trees then separates the data at several nodes to end up with classifications at the final leaves
- this way, the model learns a conditional structure of discriminative features

- Random Forests produces multiple decision trees
- with some level of randomness
- Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set
- each tree is evaluated regarding how well it classified the samples (using e.g. cross-validation)
- an ensemble of all trees is then used for prediction
- this affords RF better generalization abilities than individual trees
- Individual trees are interpretable
- but with more trees and more features the final models become vastly more complex
- So, while theoretically we would be able to understand models based on decision trees, in reality, we usually can not

Feature importance



- Not all of the features will be equally important to the model.
- A benefit of using ensembles of decision tree methods is that they can automatically provide estimates of feature importance from a trained predictive model.
- Generally, importance provides a score that indicates how useful or valuable each feature was in the construction of the decision trees within the model.
- The more an attribute is used to make key decisions with decision trees, the higher its relative importance.
- Importance is calculated for a single decision tree by how much each attribute split improves the performance measure, weighted by the number of observations the node is responsible for.
- The feature importances are then averaged across all of the the decision trees within the model.
- 'information gain' is the measure that tells us how good a tree model is
- the measure based on which the (locally) optimal condition is chosen is called impurity. For classification, it is typically either Gini impurity or information gain/entropy and for regression trees it is variance
- If you think back to the correlation heatmap we generated earlier, we can see that the features with the highest correlation with the output classes also have the highest importance weights!

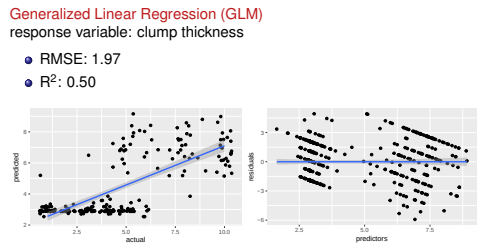
Evaluating model performance

- Because we usually build models that are too complex to really understand, we need other ways of assessing the trustworthiness of our models

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

- This is the Golden Rule of ML: always test model performance on independent data
- otherwise you will get overly optimistic performance measures
- The ultimate performance test for our model will always be it's prediction accuracy on a test set it hasn't seen before.

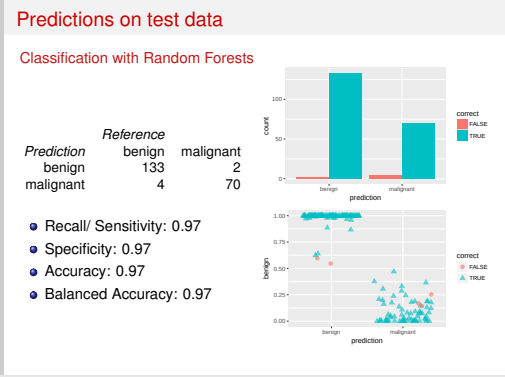
Predictions on test data



- Here, you can see the results from predicting clump thickness with a GLM model
- To assess the accuracy, the model gives us the following output values: RMSE = root of the mean squared error and R-squared
- In regression analysis, the term mean squared error refers to the unbiased estimate of error variance:
- the residual sum of squares divided by the number of degrees of freedom
- in regression the predictor variable is a real number, therefore to measure the quality of the predicted value from some algorithm you need to find some sort of difference between them. You do this by calculating the square of the error, tak the mean across all test objects and take square root
- A small RMSE means good prediction accuracy.
- RMSE is strongly influenced by outliers!

- R-squared is the ratio of explained variance to the total variance.
- it is a measure of how much of the variance in y is explained by the model. If R2 is close to one, then the model's predictions are close to the true outcome

- The plot on the left shows actual clump thickness (x-axis) vs predicted clump thickness from our model (y-axis)
- the differences between actual and predicted values are the residulas
- the plot on the right shows predicted values (x-axis) vs residuals (y-axis)
- Generally, the residuals of a well-fit model should be randomly distributed because good models will account for most phenomena in a data set, except for random error.
- We can conclude here, that our model was not very good at predicting clump thickness!

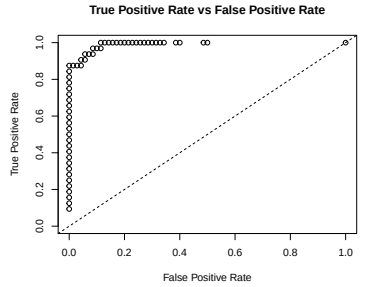


- Now we come to the classification models, here a example for predicting the class labels benign and malignant with Random Forests
- When comparing true vs predicted class labels, we can generate a confusion matrix
- We can also ask for a number of different measures about model performance:
- recall == sensitivity == True Positive Rate: proportion of positives that are correctly identified (e.g., the percentage of si people who are correctly identified as having the condition).
- Specificity == True Negative Rate: proportion of negatives that are correctly identified (e.g., the percentage of healthy people who are correctly identified as not having the condition)
- Accuracy: is the weighted arithmetic mean of Precision and Recall

- accuracy can be a misleading performance measure. Specifically, it may falsely suggest above-chance generalizability
- in binary classification, an unbalanced dataset may result in a classifier that is biased towards the more frequent class
- Balanced accuracy: average accuracy obtained on either class. Based on a confusion matrix

- On the right, you can see the corresponding plots
- above: how many predictions were correct
- below: what prediction probabilities did our samples have if they were correctly or wrongly classified
- Sometimes we can already see from this whether the default threshold of 0.5 makes sense
- but more on prediction thresholds later

Area Under the Curve (AUC)



- AUC usually refers to area under ROC curve (mathematically known as definite integral): Receiver Operating Characteristic
- metric for binary classification
- Accuracy deals with ones and zeros, meaning you either got the class label right or you didn?t. But many classifiers are able to quantify their uncertainty about the answer by outputting a probability value.
- From a random classifier you can expect as many true positives as false positives. That?s the dashed line on the plot.
- A score for a perfect classifier would be 1

Hyper-parameter tuning with grid search

- h2o.grid()
- Random Grid Search (RGS) or Cartesian Grid

Define a set of hyper-parameters:

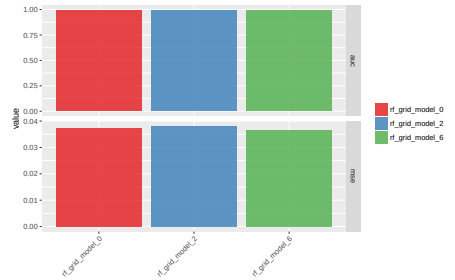
- number of trees
- maximum tree depth
- fewest allowed (weighted) observations in a leaf
- etc.

Choose best model from grid:

- h2o.getGrid()
- AUC, error, accuracy, etc.

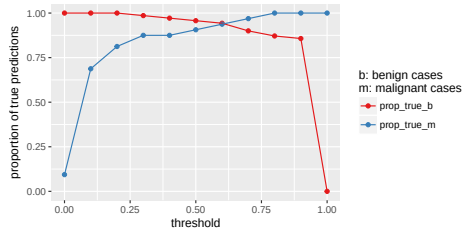
- When we build a models, we can define a number of parameters
- to get the best possible model, we should test different parameter combinations
- this, we can do with a grid search
- We could test all possible combinations of parameters with Cartesian Grid or exhaustive search, but Random GS is much faster when we have a large number of possible combinations and usually finds sufficiently accurate models.
- We can use the h2o.grid() function to perform a Random Grid Search (RGS).
- 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.
- 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.

AUC and mean squared error (MSE)



- The MSE as described for regression is a measure of the quality of an estimator. it is always non-negative, and values closer to zero are better.

Choosing a prediction threshold



- The default prediction threshold is 0.5
- meaning that every sample where the model give a higher prediction probability than 0.5 for benign, will get predicted as benign and vice versa for malignant
- this threshold isn't always the best, though and we should look at the proportions of true predictions for varying thresholds
- here, we can also choose what we want to optimize for: a balanced accuracy for the two classes or do we want to maximize accuracy for one class?

- And lastly, we can compare different models' prediction accuracies
- and choose which one we want
- here you see above the default predictions with a threshold of 0.5 for three models chosen from grid search
- below I set a more stringent threshold of 0.8 for both classes, meaning that I loose samples between 0.2 and 0.8 and consider them uncertain
- now we have only correct classifications left for model 6 and could choose this for further validation we entirely new samples

Take home messages:

- there is no 'one-size-fits-all' approach to ML
- We want to create meaningful models that we can trust to answer our specific questions!
- know your data well before modeling
- take time to think about pre-processing & features
- test different models & hyper-parameters
- evaluate model performance on independent data
- choose performance measure based on your specific problem
- choose prediction threshold based on your specific problem

- 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.