Model evaluation and cross-validation

German Demidov

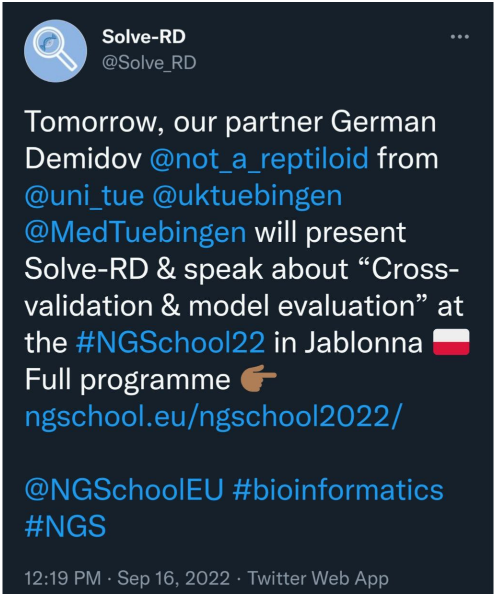
Institut für Medizinische Genetik und Angewandte Genomik, Tübingen

german.demidov at med.uni-tuebingen.de

September 16, 2022

Overview

Twitter Mixup



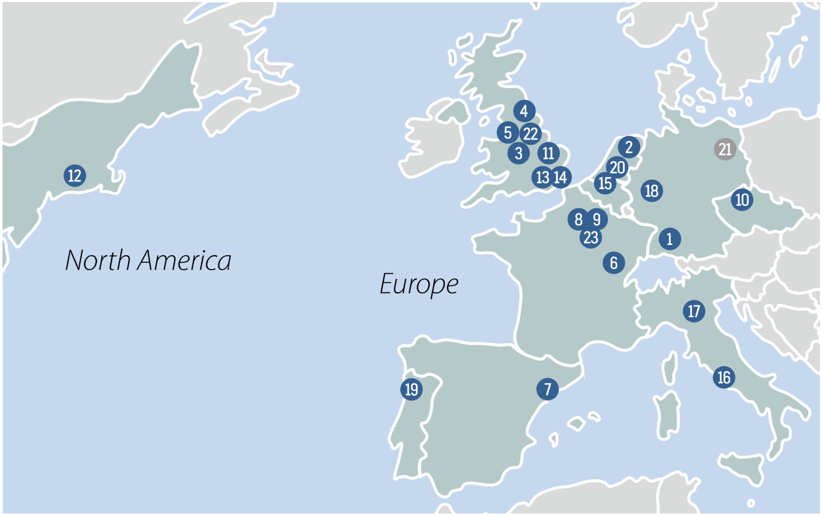
Solve-RD
@Solve_RD

Tomorrow, our partner German Demidov @not_a_reptiloid from @uni_tue @uktuebingen @MedTuebingen will present Solve-RD & speak about “Cross-validation & model evaluation” at the #NGSchool22 in Jablonna 🇵🇱
Full programme 👉 ngschool.eu/ngschool2022/

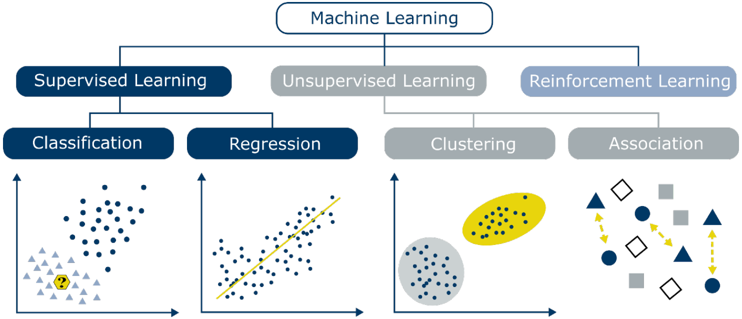
@NGSchoolEU #bioinformatics #NGS

12:19 PM · Sep 16, 2022 · Twitter Web App

SolveRD


North America
Europe
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

Machine learning problems



Machine learning model

Model:

$$y \sim M(X)$$

Real data: $\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$

Evaluation metric

A way to quantify the performance of a machine learning model

We need it to 1) compare with other models, 2) select the best hyperparameters for our model

(Disclaimer: everything which is in this presentation I copied from somewhere: Introduction to Stat Learning, youtube lectures of M. Khalusova, MarinStatsLectures and others, personal blog of Fabio Sigrist and others I found on Google)

Linear regression example

$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$,
where

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

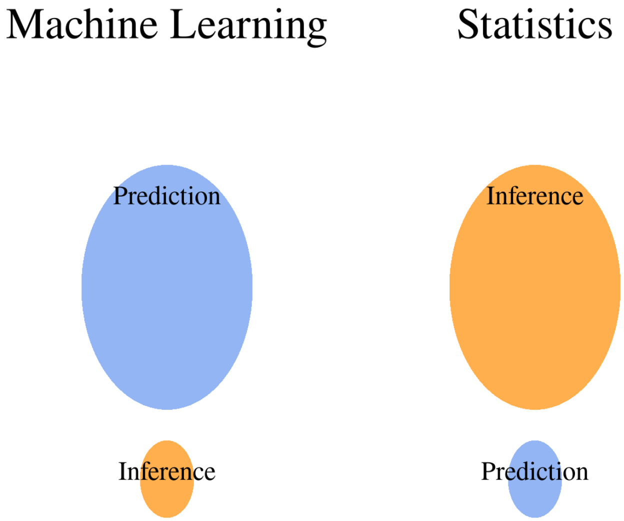
$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Types of problems and Metrics

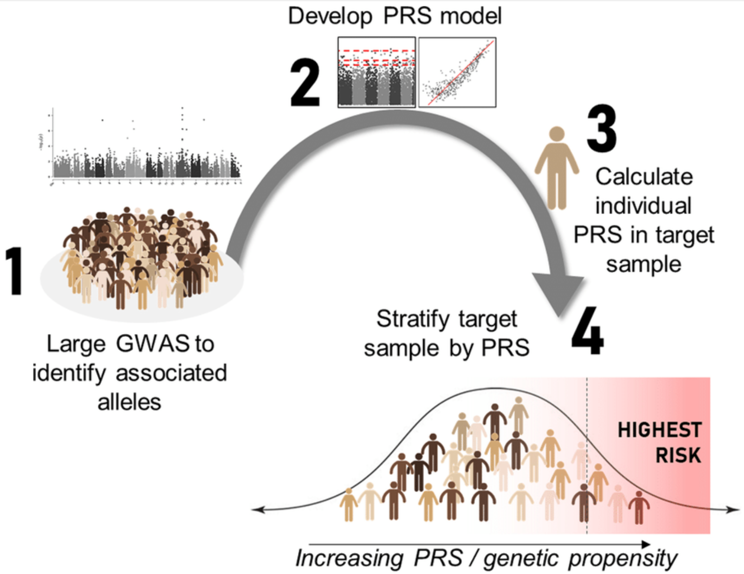
$y \sim M(X)$

- Regression (continuous outcome): R^2, MAE, RMSE
- Ordinal outcome (RMSE)
- Classification (factor outcome): accuracy, Precision, Recall, F1 Score, ROC/AUC, Precision/Recall AUC, Matthews correlation coefficient

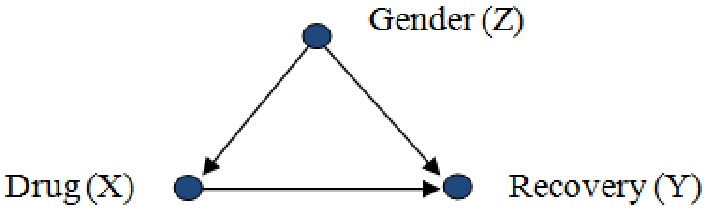
Machine learning vs statistics



Inference vs Prediction


Develop PRS model
2
1
Large GWAS to identify associated alleles
3
Calculate individual PRS in target sample
4
Stratify target sample by PRS
HIGHEST RISK
Increasing PRS / genetic propensity

Difference between inference vs Prediction: Confounding

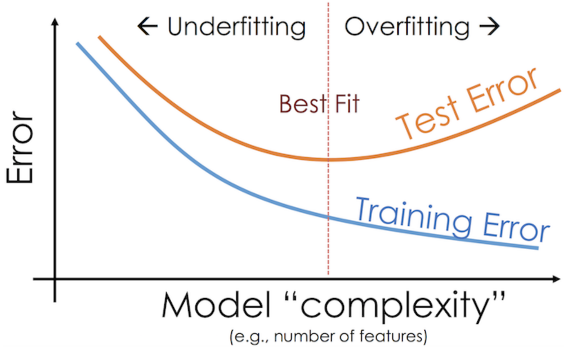


Q: When it is important?

- Estimation of the effect size
- Prediction

Types of error

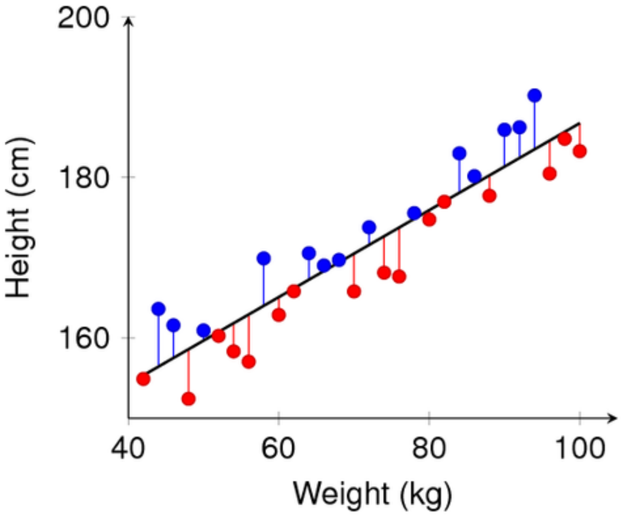
- Train error
- Test error



Evaluation of a regression model

- Coefficient of Determination (R^2)
- RMSE (root mean squared error)
- MAE (mean absolute error)

Residuals



Coefficient of Determination

We calculate the mean of our outcome:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

The sum of squares of residuals, also called the residual sum of squares:

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

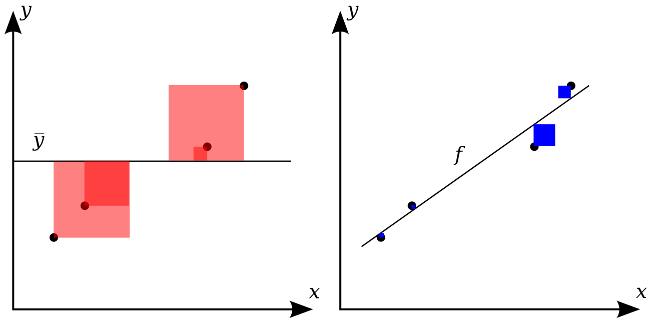
The total sum of squares (proportional to the variance of the data):

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$$

The most general definition of the coefficient of determination is

$$R^2 = 1 - \frac{SS_{\rm res}}{SS_{\rm tot}}$$

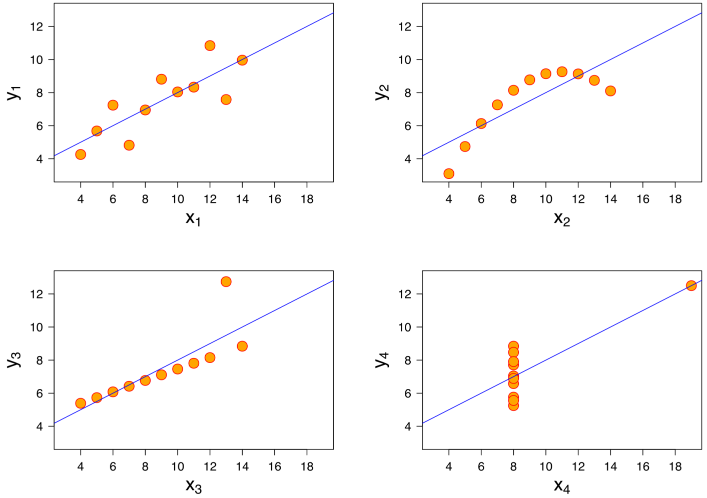
Coefficient of Determination: Graphical Explanation



$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$

In linear least squares multiple regression with an intercept term, R^2 = the square of the Pearson correlation coefficient between the observed y and modeled (predicted) f data values of the dependent variable.

Drawbacks



Multiple linear regression

```
set.seed(1)
outcome <- rnorm(100)
predictor <- rnorm(100)
newlm <- lm(outcome ~ predictor)
summary(newlm)
```

Multiple linear regression

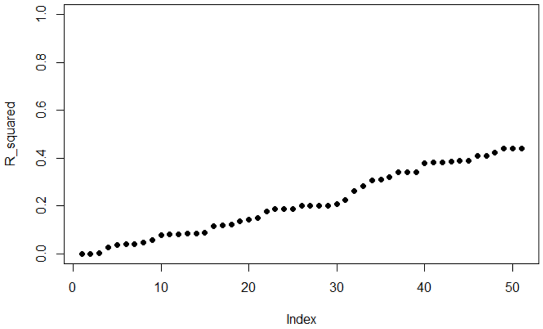
```
Call:
lm(formula = outcome ~ predictor)
Residuals:
     Min       1Q   Median       3Q      Max
-2.32416 -0.60361  0.00536  0.58305  2.29316
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.1088521  0.0903480   1.205    0.231
predictor   -0.0009324  0.0947216  -0.010    0.992
Residual standard error: 0.9028 on 98 degrees of freedom
Multiple R-squared:  9.887e-07,Adjusted R-squared:  -0.010
F-statistic: 9.689e-05 on 1 and 98 DF,  p-value: 0.9922
```

Multiple linear regression

```
R_squared <- c(summary(newlm)$r.squared)
for (i in 1:50) {
  new_predictor <- rnorm(100)
  predictor = cbind(predictor, new_predictor)
  newlm <- lm(outcome ~ predictor)
  R_squared <- c(R_squared, summary(newlm)$r.squared)
}
plot(R_squared)
```

R^2 with more random predictors

R^2 alone cannot be used as a meaningful comparison of models with very different numbers of independent variables. F-test can be performed on the residual sum of squares.



Adjusted R^2

Replace r.squared with adj.r.squared:

```
R_squared <- c(R_squared, summary(newlm)$adj.r.squared)
```

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p}$$

where p is the total number of explanatory variables in the model, and n is the sample size.

Model selection via BIC and AIC

Not model evaluation, but selection

$$\mathrm{AIC} \; = \; 2k - 2\ln(\hat{L})$$

$$\mathrm{BIC} = k\ln(n) - 2\ln(\widehat{L})$$

where k is the number of estimated parameters, n is the number of observations.

Explained variance in Random Forest regression

```
library(randomForest)
library(ggplot2)
dat <- data.frame(ggplot2::diamonds[1:1000,1:7])
rf <- randomForest(formula = carat ~ .,
data = dat, ntree = 500)
rf
# Call:
#   randomForest(formula = carat ~ ., data = dat, ntree = 5
#                Type of random forest: regression
#                      Number of trees: 500
# No. of variables tried at each split: 2
# Mean of squared residuals: 0.001820046
# % Var explained: 95.22
```

Manual R^2 for random forest regression models

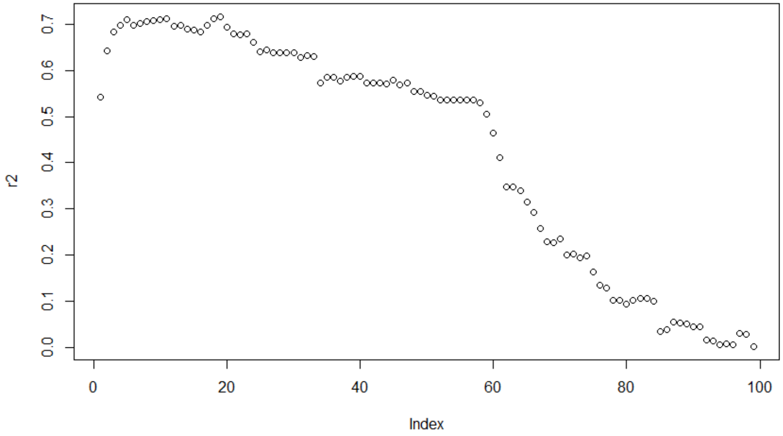
```
actual = dat$carat
predicted <- unname(predict(rf, dat))

R2 <- 1 - (sum((actual-predicted)^2)/sum(
(actual-mean(actual))^2))
```

R^2 using test data

```
library(faux)
num_of_predictors <- 100
bvn <- rnorm_multi(200, num_of_predictors, 0, 1, .75)
colnames(bvn)[1] = "a"
r2 <- c()
for (i in 2:num_of_predictors) {
train <- bvn[1:100,1:i]
test <- bvn[101:200,1:i]
newlm <- lm(a ~ ., data=train)
predictions <- predict(newlm, test)
r2 <- c(r2, cor(predictions, test$a)**2)
}
plot(r2)
```

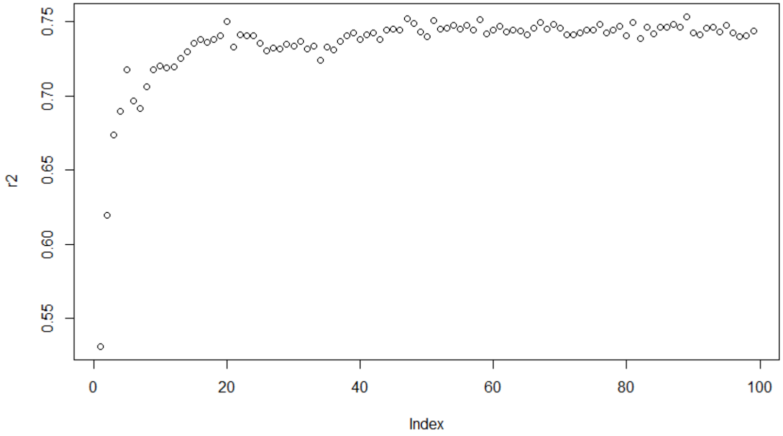
R^2 using test data



R^2 using test data

```
library(randomForest)
r2 <- c()
for (i in 2:num_of_predictors) {
  train <- bvn[1:100,1:i]
  test <- bvn[101:200,1:i]
  randf <- randomForest(a ~ ., data=train)
  predictions <- predict(randf, test)
  r2 <- c(r2, cor(predictions, test$a)**2)
}
plot(r2)
```

R^2 using test data



The use of coefficient of determination

R squared has an intuitive scale and does not depend of y unit
R squared gives you no information about the prediction error

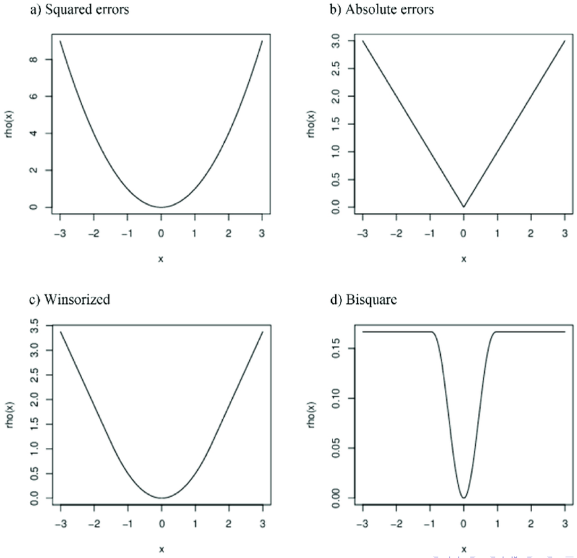
RMSE and MAE

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- most commonly used metric
Should Mean Absolute Error be used instead?

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

RMSE and MAE and other losses



MAE and RMSE

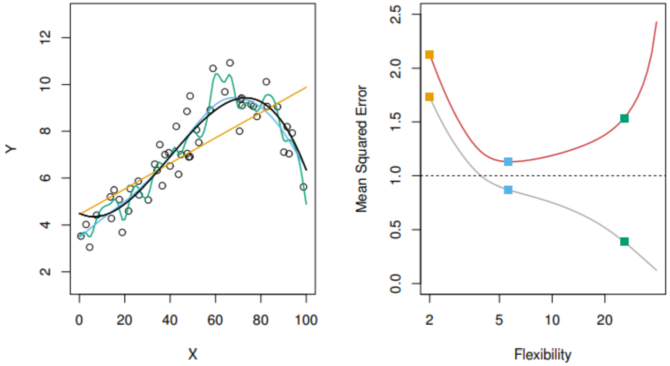
In common:

- Same units as the y response values
- Don't care about the direction of errors

Different:

- Higher weights to large errors in RMSE
- RMSE is differentiable

RMSE application



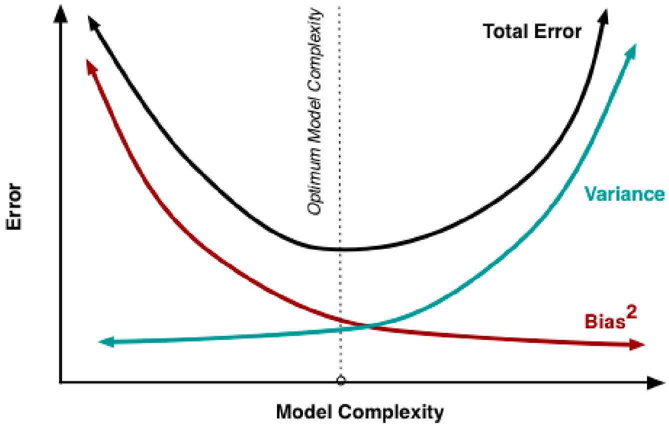
Bias and variance

D is the training dataset

$$\mathrm{E}_{D,\varepsilon}\left[\left(y - \hat{f}(x;D)\right)^2\right] = \left(\mathrm{Bias}_D\left[\hat{f}(x;D)\right]\right)^2 + \mathrm{Var}_D\left[\hat{f}(x;D)\right] + \sigma^2$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.

Bias and variance



Bias and variance

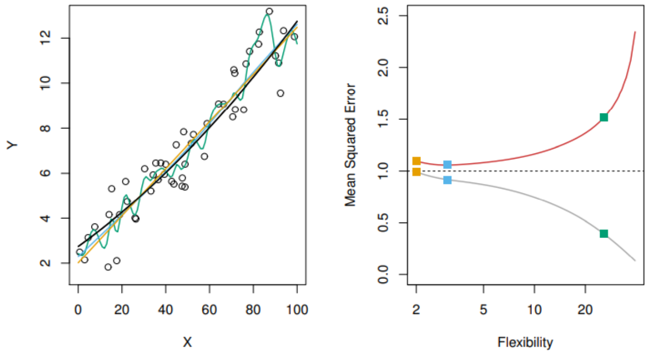
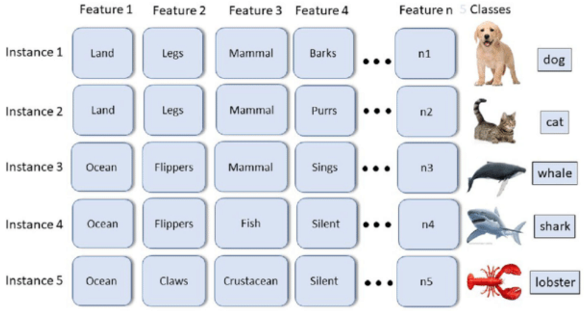


FIGURE 2.10. *Details are as in Figure 2.9, using a different true f that is much closer to linear. In this setting, linear regression provides a very good fit to the data.*

Classification



	Feature 1	Feature 2	Feature 3	Feature 4		Feature n	Classes
Instance 1	Land	Legs	Mammal	Barks	• • •	n1	dog
Instance 2	Land	Legs	Mammal	Purrs	• • •	n2	cat
Instance 3	Ocean	Flippers	Mammal	Sings	• • •	n3	whale
Instance 4	Ocean	Flippers	Fish	Silent	• • •	n4	shark
Instance 5	Ocean	Claws	Crustacean	Silent	• • •	n5	lobster

Classification

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of predictions}}$$

Is accuracy of 99% good?

Confusion matrix

```
library(caret)
library(InformationValue)
library(ISLR)

data <- Default # Credit Card Default Data
set.seed(1)
sample <- sample(c(TRUE, FALSE), nrow(data),
replace=TRUE, prob=c(0.7,0.3))
train <- data[sample, ]
test <- data[!sample, ]

model <- glm(default~student+balance+income,
family="binomial", data=train)
```

Confusion matrix

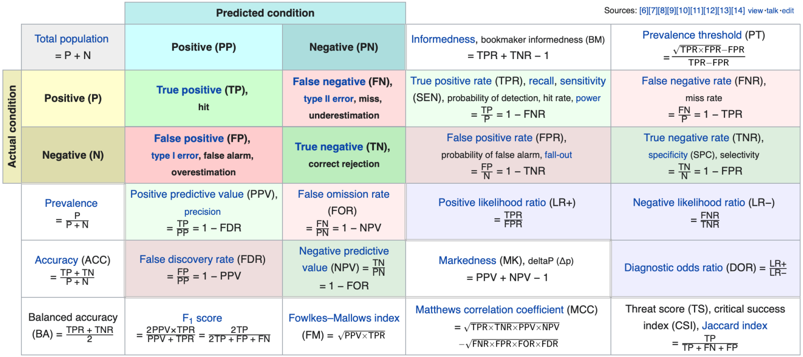
```
predicted <- predict(model, test, type="response")
#convert defaults from "Yes" and "No" to 1's and 0's
test$default <- ifelse(test$default=="Yes", 1, 0)
#find optimal cutoff probability to use to maximize accurac
optimal <- optimalCutoff(test$default, predicted)[1]
#create confusion matrix
confusionMatrix(test$default, predicted //def thresh 0.5
     0  1
0 2912 64
1   21 39
confusionMatrix(test$default, predicted, threshold=optimal
     0  1
0 2919 68
1   14 35
```

Confusion matrix

```
optimal <- optimalCutoff(test$default, predicted,
optimiseFor="Ones")[1]

confusionMatrix(test$default, predicted,
threshold=optimal)
     0   1
0 2003   3
1  930 100
```

Precision, Recall



		Predicted condition		Sources: [6][7][8][9][10][11][12][13][14] view·talk·edit	
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP), hit**	**False negative (FN), type II error, miss, underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP), type I error, false alarm, overestimation**	**True negative (TN), correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR–) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2 PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

Precision, Recall

Precision $= \frac{TP}{TP+FP}$ - concentrate on True Positives
Recall $= \frac{TP}{TP+FN}$ - concentrate on False Negatives

What's more important to minimize? You decide

F1

Harmonic mean between precision and recall

$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$

Matthews Correlation Coefficient

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Takes into account all four components

MCC vs F1

```
     0      1
0   TN=0   FP=5
1   FN=0   TP=95
```

Accuracy: 95%
F1 score = 0.974
MCC = Undefined (dividing by 0)

MCC vs F1

```
      0      1
0   TN=1   FP=4
1   FN=5   TP=90
```

Accuracy: 95%
F1 score $= 0.952$
MCC $= 0.135$

MCC vs F1

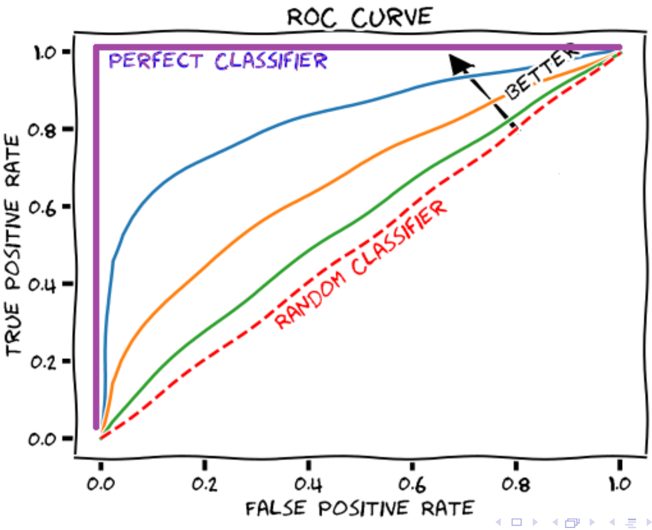
	0	1
0	TN=90	FP=5
1	FN=4	TP=1

Accuracy: 95%
F1 score = 0.182
MCC = 0.135

MCC vs F1

MCC can be better to summarize the confusion matrix (only for binary problems)

ROC / AUC


ROC CURVE
PERFECT CLASSIFIER
BETTER
RANDOM CLASSIFIER
TRUE POSITIVE RATE
FALSE POSITIVE RATE
0.0
0.2
0.4
0.6
0.8
1.0

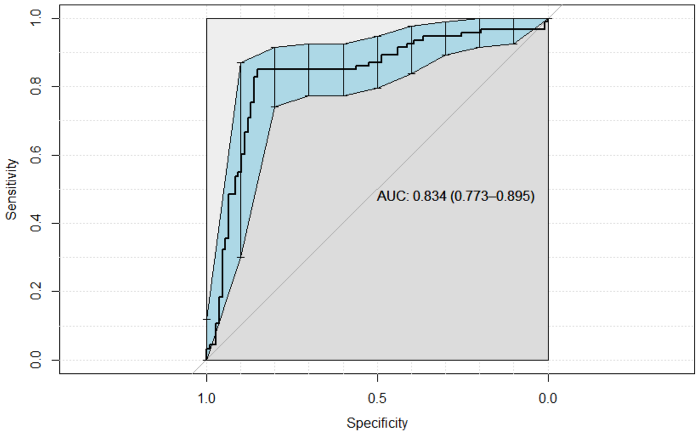
How to calculate ROC / AUC in R

Packages: pROC, ROCR
https://www.r-bloggers.com/2016/11/calculating-auc-the-area-under-a-roc-curve/

How to calculate ROC / AUC in R

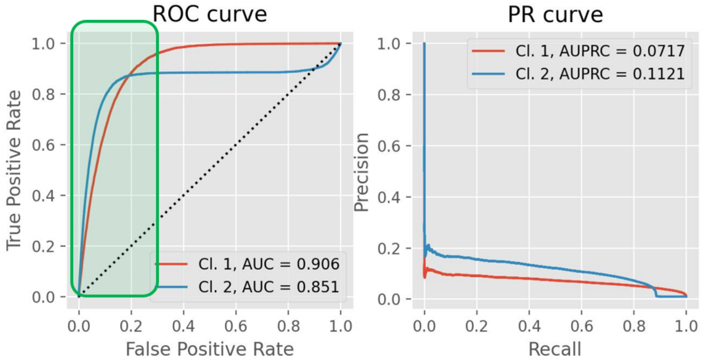
```
library(ROCR)
data(ROCR.simple)
df <- data.frame(ROCR.simple)
pred <- prediction(df$predictions, df$labels)
perf <- performance(pred,"tpr","fpr")
library(pROC)
pROC_obj <- roc(df$labels,df$predictions, smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.95, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=T
            print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(pROC_obj)
plot(sens.ci, type="shape", col="lightblue")
plot(sens.ci, type="bars")
```

How to calculate ROC / AUC in R

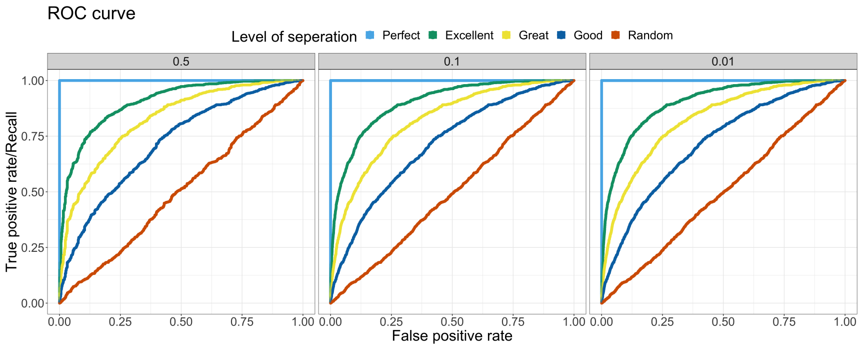


Precision / Recall curve

2 classifiers and imbalanced data

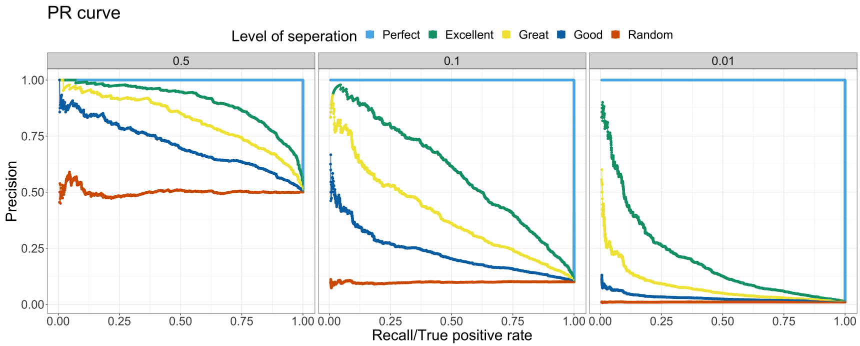


AUC ROC in imbalanced data



(Sin-Yi Chou github)

AUC PR in imbalanced data



AUC PR vs AUC ROC

- ROC in case the model is biased towards one class can't be large - unlike Accuracy. If the goal of the model is to perform equally well on both classes, ROC is the choice
- In case you care about the small positive class, PR AUC can be better
- If the model doesn't work after the metric is changed, there are still other remedies to deal with imbalanced data, such as downsampling/upsampling

Multiple class: Precision

	Bird	Cat	Dog
Bird	1	0	1
Cat	0	4	0
Dog	0	1	2

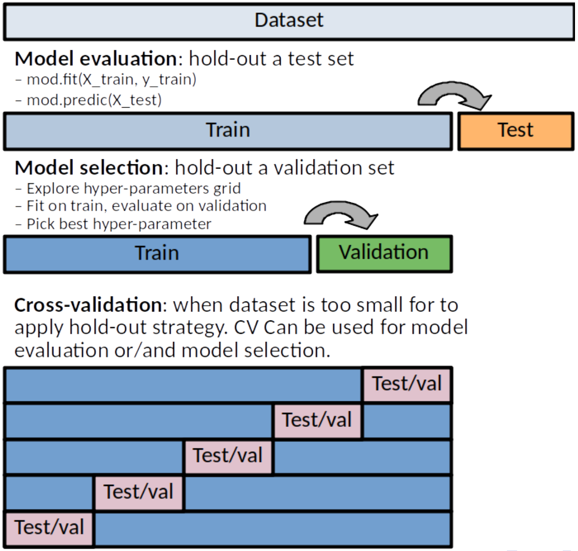
	TP	FP	Pr	N samples
Bird	1	0	1	2
Cat	4	1	0.8	4
Dog	2	1	0.66	3
Total	7	2		

Microprecision$= \frac{7}{7+2} = 0.7777$
Macroprecision$= \frac{1}{3}(1 + 0.8 + 0.66) = 0.822$
Weighted precision$= \frac{1 \cdot 2 + 0.8 \cdot 4 + 0.66 \cdot 3}{2+4+3} = 0.8$

Which to choose

- Microprecision: every sample contributed
- Macroprecision: all the classes equally contributed (good for underrepresented classes)
- Weighted: each classes' contribute to the precision according to the size of the class

Train-test approach



Model evaluation: hold-out a test set
- mod.fit(X_train, y_train)
- mod.predic(X_test)

Model selection: hold-out a validation set
- Explore hyper-parameters grid
- Fit on train, evaluate on validation
- Pick best hyper-parameter

Cross-validation: when dataset is too small for to apply hold-out strategy. CV Can be used for model evaluation or/and model selection.

Resampling methods

- Estimation: bootstrap, permutation-based tests
- Model evaluation: cross-validation

Cross-validation types

- Leave one out CV
- k-fold CV
- repeated k-fold CV

Summary of the error

$$CV_k = \frac{1}{k}\Sigma_{i=1}^{k} MSE_i$$

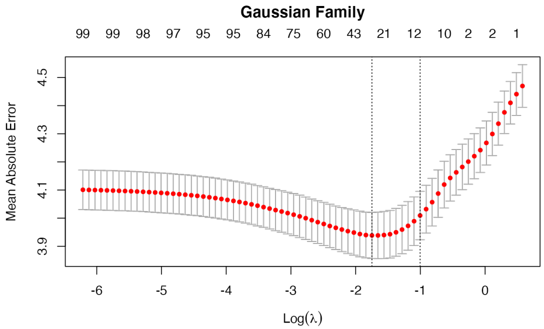
Cross-validation types

Which cross-validation is better?

- LOOCV can take incredibly long, 10-fold can be faster
- LOOCV has almost the lowest bias (almost all the samples used for the training)
- LOOCV has a very high variance

Typically one performs k-fold cross-validation using k = 5 or k = 10 since empiricially it shows good balance between variance and bias

Cross-validation as parts of methods



lambda.min - value of lambda that gives minimum cvm.
lambda.1*se* - largest value of lambda such that error is within 1 standard error of the minimum.

Cross-validation in caret (R)

```
set.seed(1)
data("swiss")
library(caret)
```

Cross-validation in caret (LOOCV)

```
set.seed(1)
train.control <- trainControl(method = "LOOCV")
# Train the model
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
print(model)
Linear Regression
47 samples
 5 predictor
No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 46, 46, 46, 46, 46, 46, ...
Resampling results:

  RMSE      Rsquared   MAE
  7.738618  0.6128307  6.116021
```

Cross-validation in caret (K-fold)

```
set.seed(1)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
print(model)
Linear Regression
47 samples
 5 predictor
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 42, 42, 43, 41, 42, 44, ...
Resampling results:

  RMSE      Rsquared   MAE
  7.464863  0.7227897  5.953467
```

The End