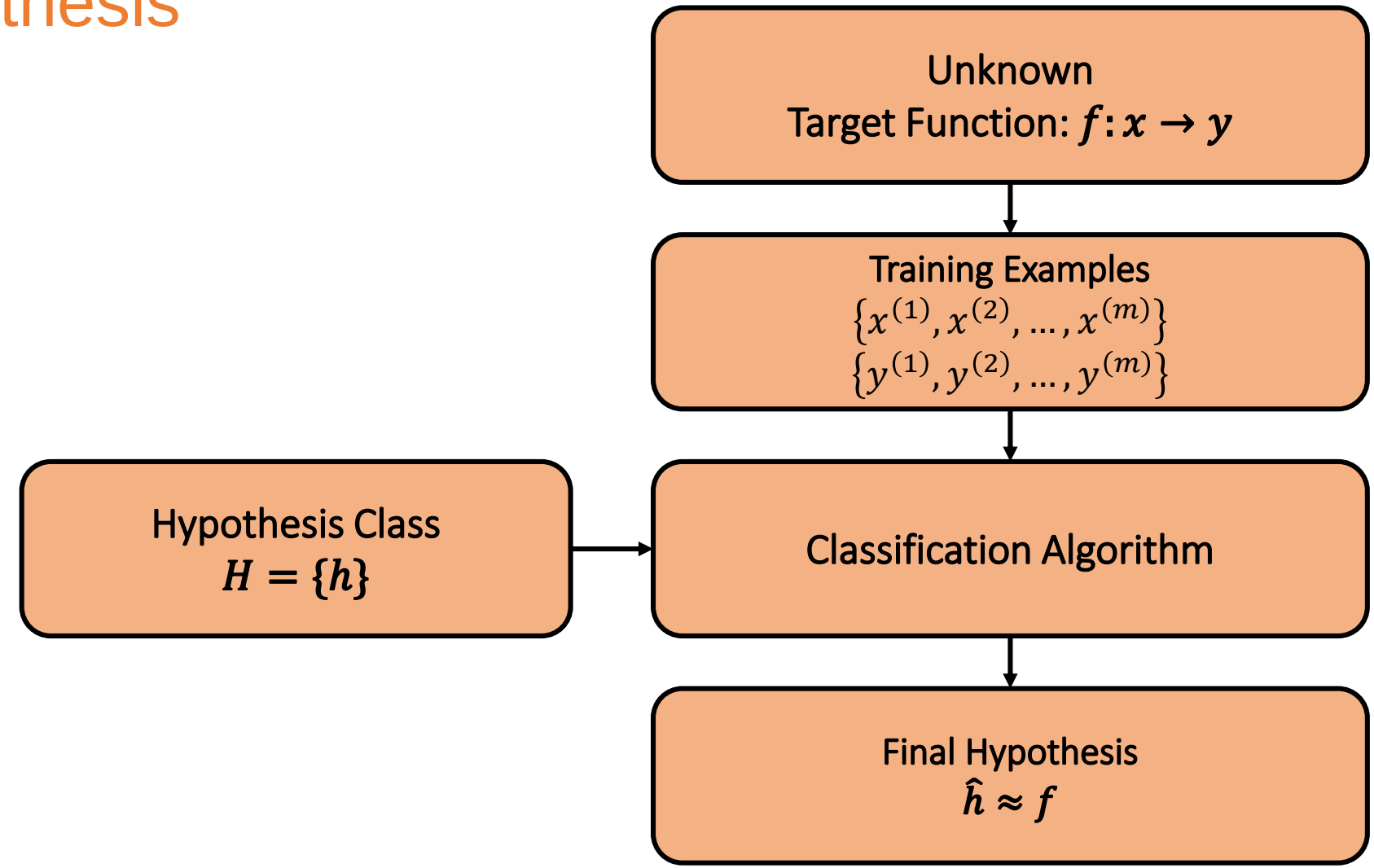
Machine Learning in Healthcare

#C07 Practical Considerations on Training a ML Model

Technion-IIT, Haifa, Israel

Assist. Prof. Joachim Behar
Biomedical Engineering Faculty
Technion-IIT

Hypothesis


Unknown
Target Function: $f: x \rightarrow y$
Training Examples
$\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$
$\{y^{(1)}, y^{(2)}, \ldots, y^{(m)}\}$
Hypothesis Class
$H = \{h\}$
Classification Algorithm
Final Hypothesis
$\hat{h} \approx f$

Performance Statistics

Confusion Matrix

- Confusion matrix:

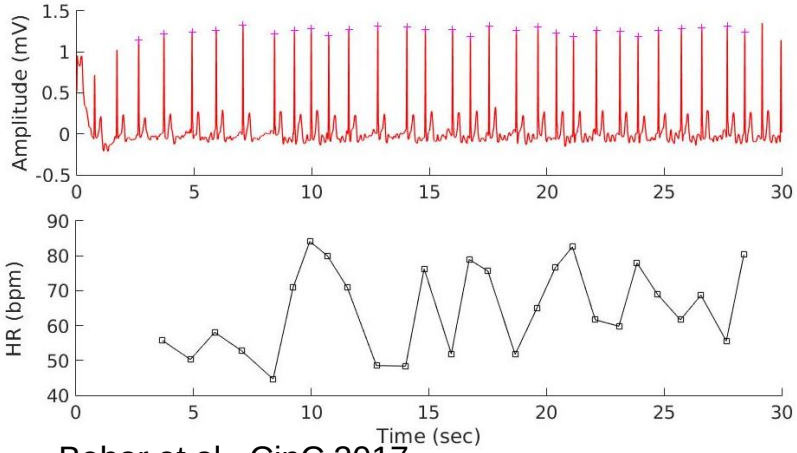
	True label Yes	True label No	
Predicted Yes	TP	FP	PPV = TP/(TP+FP)
Predicted No	FN	TN	NPV = TN/(FN+TN)
	Se = TP/(TP/FN)	Sp = TN/(TN+FP)	

Limitation of Accuracy

- Let's assume a population of 900 patients.
- 100 of them have AF and 800 are non-AF.
- This is the confusion matrix we get from our classifier:

	True label Yes	True label No
Predicted Yes	50	20
Predicted No	50	780

- If we compute the accuracy:
  - $Ac = \frac{TP+TN}{TP+TN+FP+FN} = \frac{50+780}{50+780+20+50} = 0.95$
- Should we conclude our classifier is doing a great job?



Behar et al., CinC 2017

Performance Statistics

My test identified 10 patients with the condition correctly. What is the proportion of patients with the condition that were correctly identified out of all with the conditions?

My test identified 20 patients without the condition correctly. What is the proportion of patients without the condition that were correctly identified out of all without the condition?

	True label Yes	True label No
Predicted Yes	TP	FP
Predicted No	FN	TN

- **Sensitivity:** proportion of people with a condition who are correctly identified by a test as indeed having that condition.
  - $Se = \frac{TP}{TP+FN}$
- **Specificity:** proportion of people without a condition who are correctly identified by a test as indeed not having the condition.
  - $Sp = \frac{TN}{TN+FP}$

Performance Statistics

Got a patient with a positive test. What is the probability that this patient has indeed the condition?

Got a patient with a negative test. What is the probability that this patient does not indeed have the condition?

	True label Yes	True label No
Predicted Yes	TP	FP
Predicted No	FN	TN

- **Positive Predictive Value**: is the probability that people with a positive test result indeed do have the condition of interest.
  - $PPV = \frac{TP}{TP+FP}$
- **Negative Predictive Value**: probability that people with a negative test result indeed do not have the condition of interest.
  - $NPV = \frac{TN}{TN+FP}$.

Performance Statistics

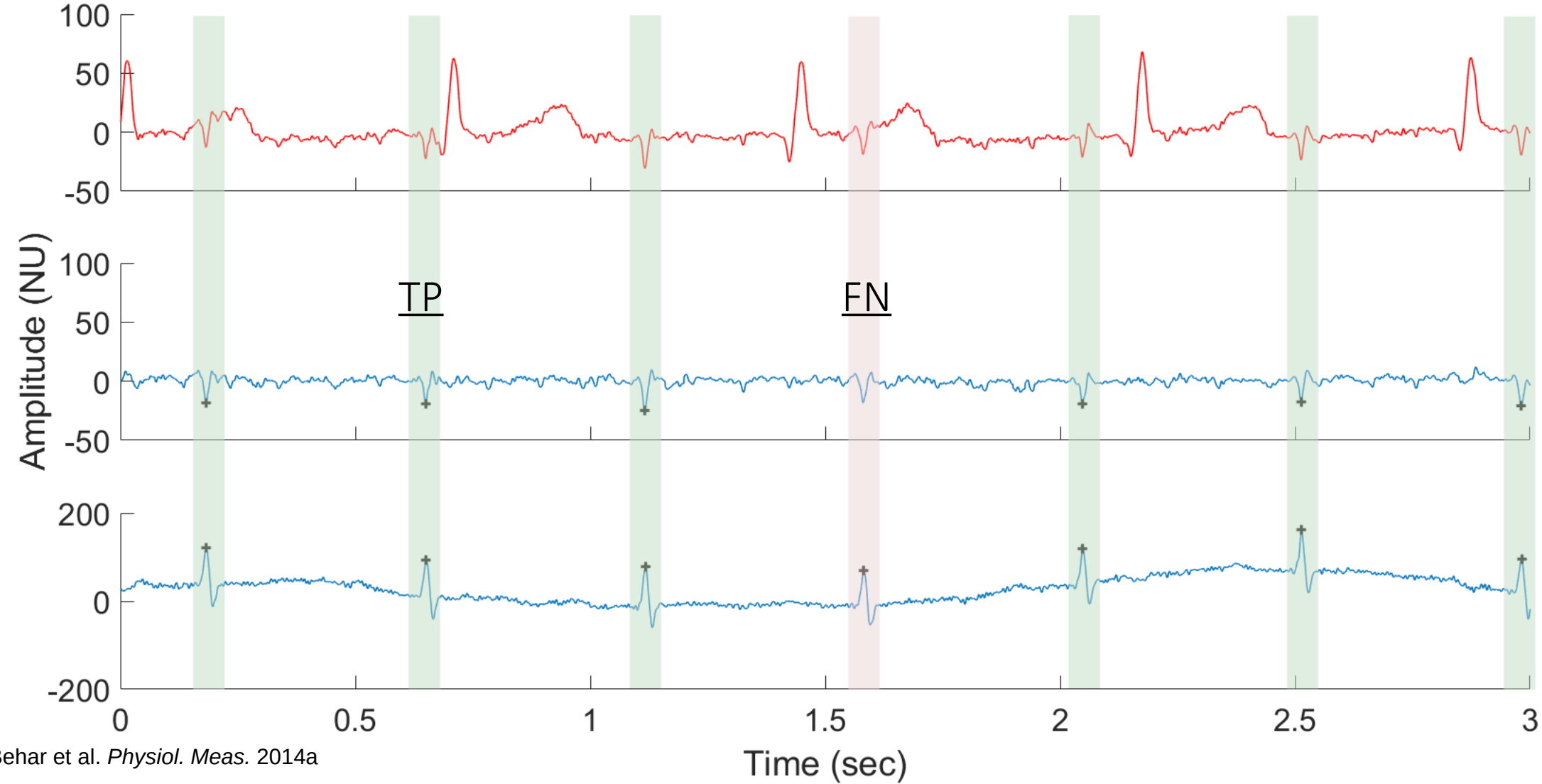
I want to optimize my classifier. How can I quantify its "accuracy" with one single statistical measure?

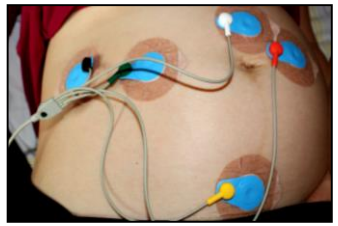
- We do not want to miss the AF patients, right?
- $h_w(x) \geq 0.5$ predicts AF.
- Perhaps $h_w(x) \geq 0.3$ predicts AF is better?
- Say TN are not what we care about (as much).
- We look for a measure that "average" Se and PPV.
- F_1: harmonic average between Se and PPV.
- Measure of the tradeoff between Se and PPV.

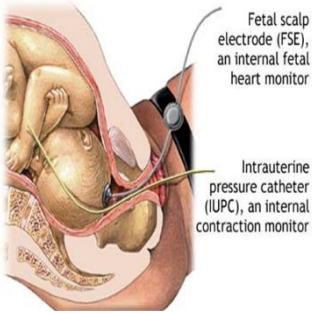
	True label Yes	True label No
Predicted Yes	TP	FP
Predicted No	FN	TN

$$F_1 = 2 \cdot \frac{PPV \cdot Se}{PPV + Se} = \frac{2 \cdot TP \cdot Se}{2 \cdot TP + FP + FN}$$

Algorithm Evaluation







Behar et al. *Physiol. Meas.* 2014a

Going back to our example

	True label Yes	True label No
Predicted Yes	50	20
Predicted No	50	780

- Let's assume a population of 900 patients.
- 100 of them have AF and 800 are non-AF.
- This is the confusion matrix we get from our classifier:
- If we compute the performance statics:
  - $Ac = \frac{TP+TN}{TP+TN+FP+FN} = \frac{50+780}{50+780+20+50} = 0.95$
  - $Se = \frac{TP}{TP+FN} = \frac{50}{50+50} = 0.5$
  - $Sp = \frac{TN}{TN+FP} = \frac{780}{780+20} = 0.98$
  - $PPV = \frac{TP}{TP+FP} = \frac{50}{50+20} = 0.71$
  - $NPV = \frac{TN}{FN+TN} = \frac{780}{780+50} = 0.94$
  - $F_1 = 2 \cdot \frac{PPV \cdot Se}{PPV+Se} = 0.59$
- So other stats are needed and provide better insights when classes are **skewed**.

Performance Statistics

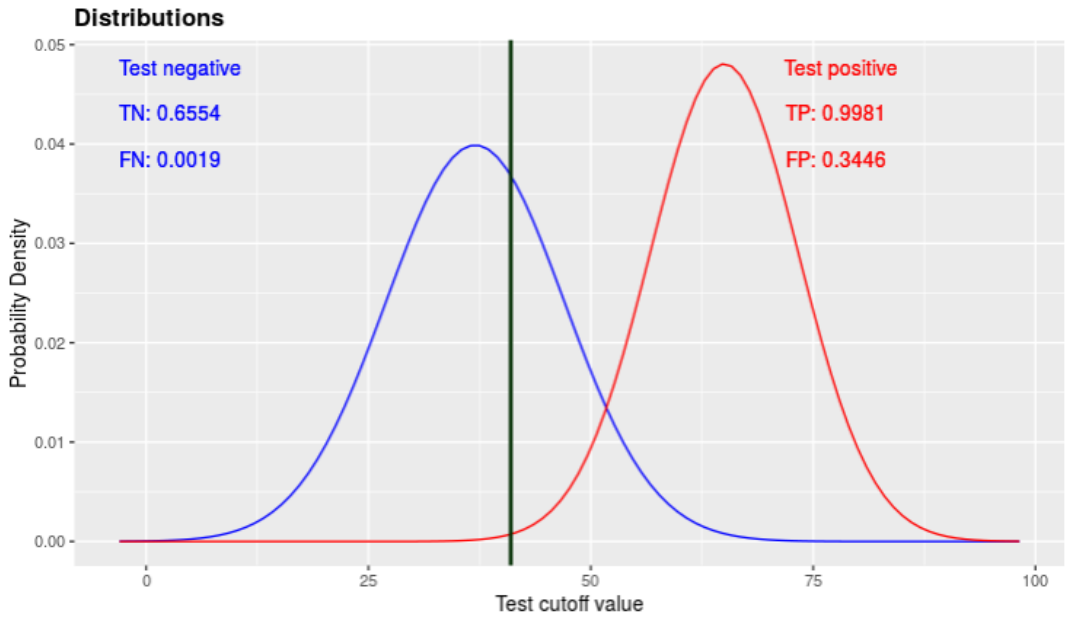
- To summarize:
  - Se: proportion of people with a condition who are correctly identified by a test as indeed having that condition.
  - Sp: proportion of people without a condition who are correctly identified by a test as indeed not having the condition
  - PPV: is the probability that people with a positive test result indeed do have the condition of interest.
  - NPV: probability that people with a negative test result indeed do not have the condition of interest.
  - F_1: harmonic average between Se and PPV. Useful as a single measures for classifier optimization. Measure of the tradeoff between Se and PPV.

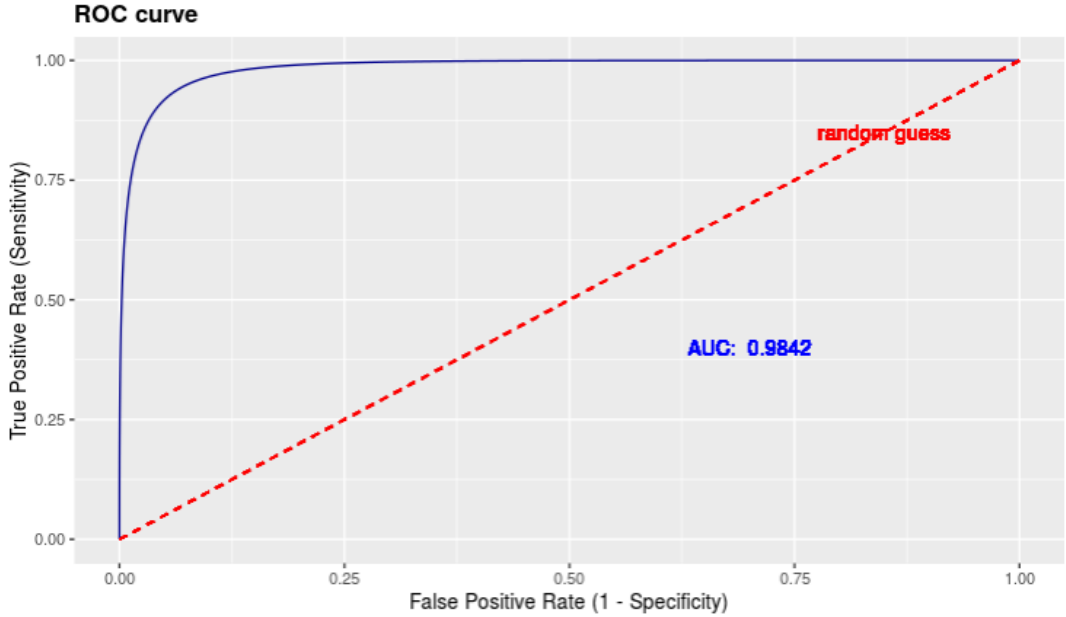
Performance Statistics

- Difference between Se, Sp and PPV, NPV:
  - Se and Sp quantify how accurate the classifier performs with respect to a reference ("ground truth").
  - PPV and NPV encapsulate the information about prevalence of the condition. In other words the imbalance of the classes is taken into account which might be a good thing if our population sample is characteristic of our population of interest.
  - Thus PPV and NPV are particularly appropriate when considering the performance of a medical screening test for example.
  - In practice, report Se, Sp, PPV and NPV and interpret carefully with respect to the research question.

Performance Statistics

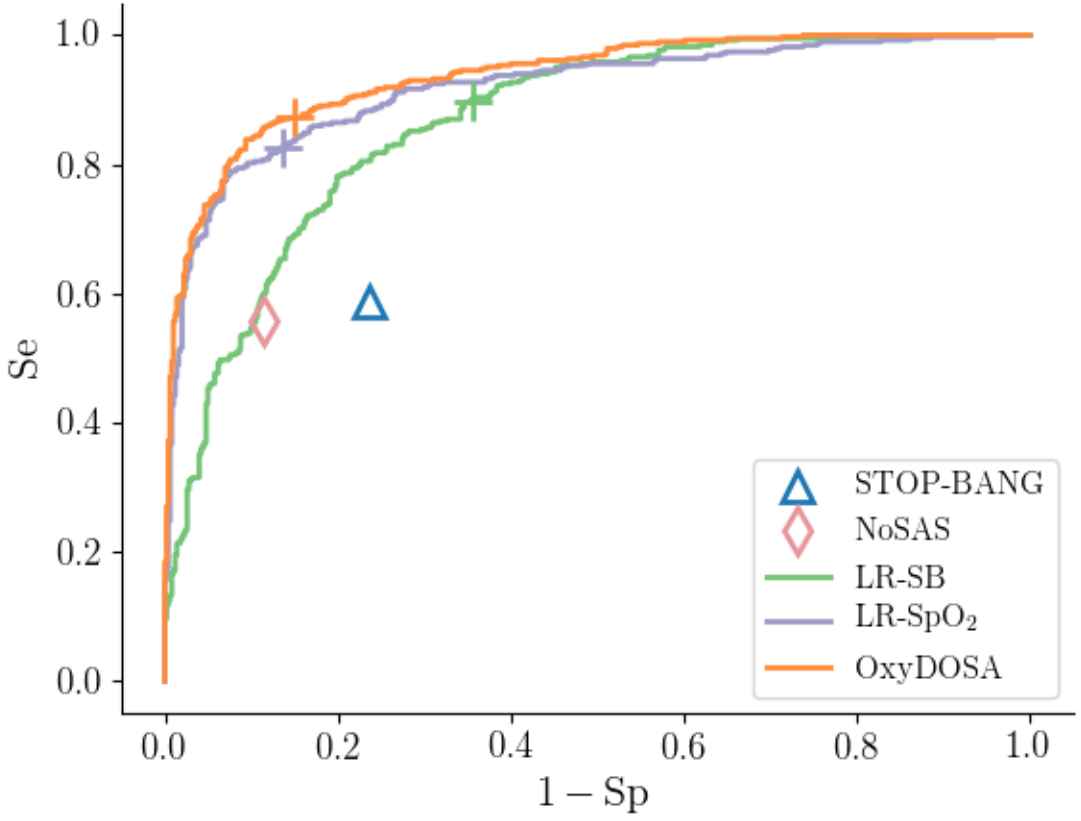
- Receiver Operating Characteristic (ROC)
  - Diagnostic ability of a binary classifier.
  - Se is plotted against $1 - Sp$.
  - Performance measurement at different threshold values.
- The Area Under the ROC (AUROC)
  - Quantify the separability of the classes.
  - The closer from 1 the better.
- Extension to multiple classes:
  - Hand, David J., and Robert J. Till. "A simple generalisation of the area under the ROC curve for multiple class classification problems." Machine learning 45.2 (2001): 171-186.

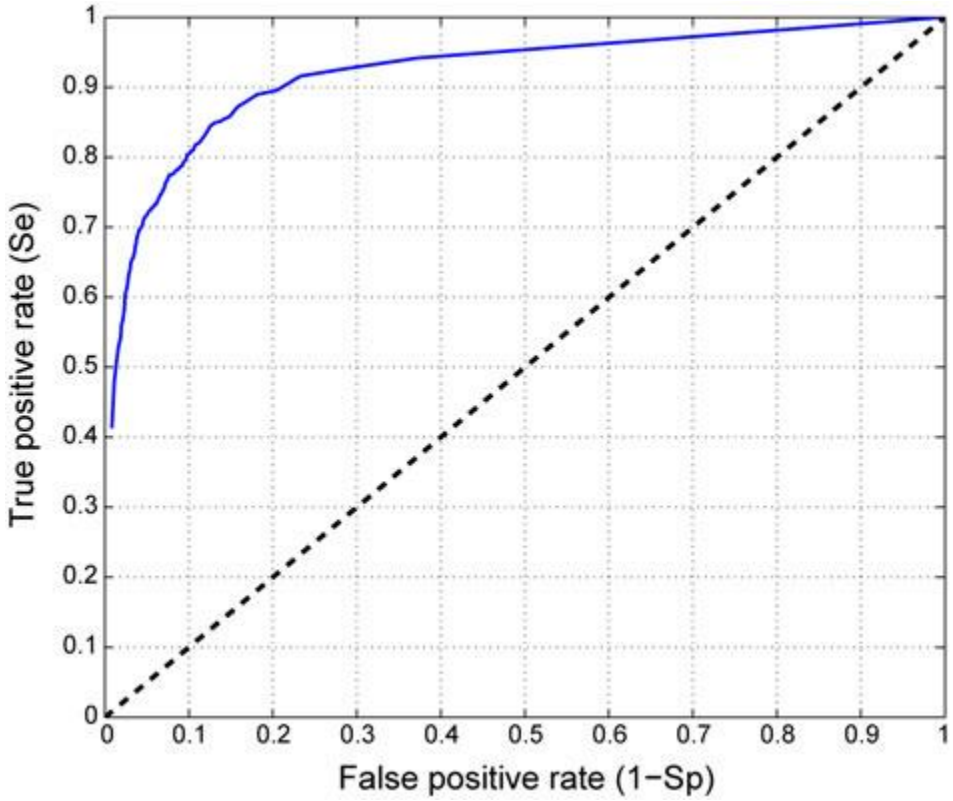




https://kennis-research.shinyapps.io/ROC-Curves/

Performance Statistics





Behar, Joachim, et al. EClinicalMedicine 11 (2019).
Behar, Joachim, et al. IEEE TBME 60.6 (2013).

Training the final ML model

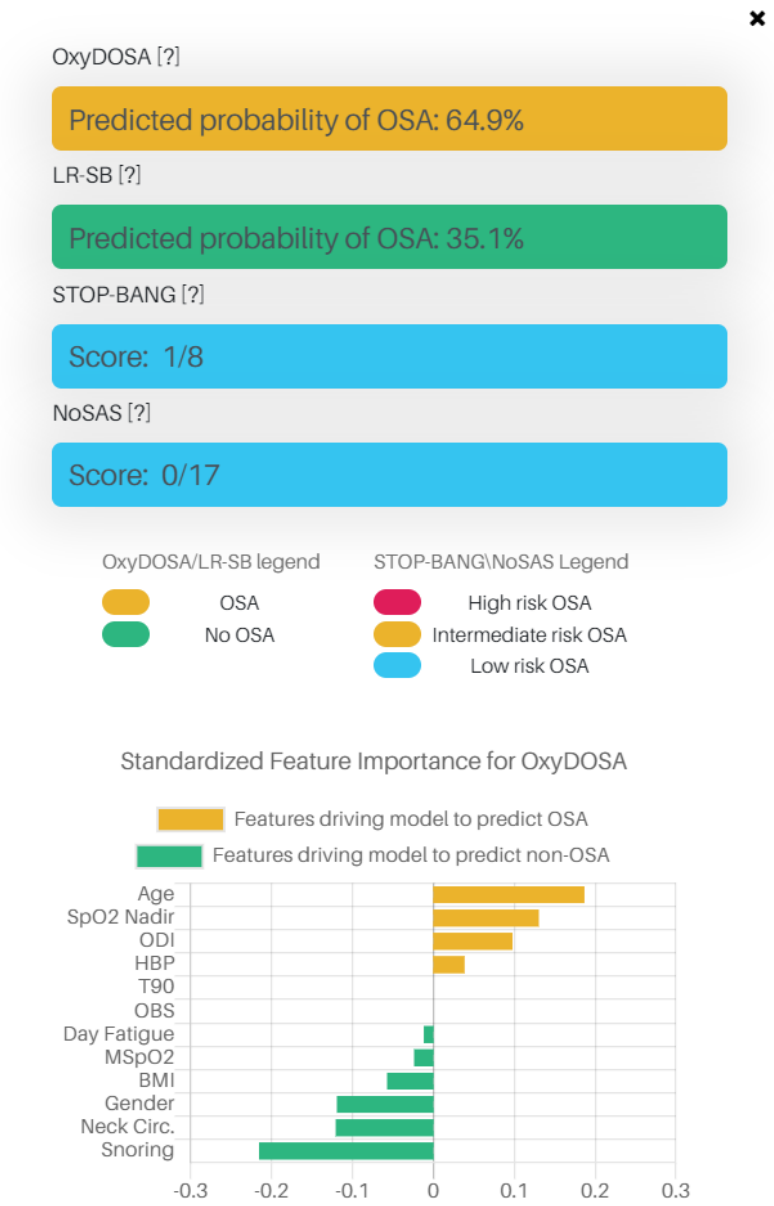
Training the final ML model

- You have performed cross-validation and you are satisfied with the performance results you got on the test set using one or a set of the statistics we just saw. Great!
- Now you need to generate the "**final model**" that is the model you will deploy in the real world and use to make prediction on new examples. How do you do that?
- At this point we have figured out:
  - How to prepare our data (e.g. what scaling approach to use),
  - What algorithm to use and with what complexity (e.g. regression with power d),
  - Found suitable hyperparameters (e.g. regularization parameter λ).
- The performance of our model on the test set represents how our algorithm will perform on unseen examples. Thus at this point we have designed well our procedure and found a suitable model. The train-validation-test set split/cross validation procedure has served its purpose and we do not need them further.

Training the final ML model

- You can generate the final model by applying the selected ML model (preprocessing, algorithm type, algorithm hyperparameters) on the whole dataset.

- Example of such implementation:
  - https://aim-lab.github.io/oxydosa.html



Take Home

- **Confusion matrix**. **Performance statistics** and their meaning.
  - Sensitivity, specificity, positive predictive value, negative predictive value, F_1 measure and AUROC.
- Depending on the question you ask you will one set of statistics or another.
- A practical tip:
  - Start with a simple algorithm, obtain results on cross-validation. Then plot learning curves and get an idea of where you can improve.
  - Avoid "premature optimization" and let evidence guide your design.
- When you are done with the model evaluation and are satisfied with its performance (according to some representative performance statistics that is tailored to your problem) then you can general the **final model** by training the model you have identified (preprocessing procedure, algorithm, hyperparameters) on the whole dataset you have.

References

[1] Coursera, Andrew Ng. Advice for applying machine learning.
[2] Machine Learning Basic Concepts. CDT Lectures Notes 2013. Alistair Johnson.
[3] How to Train a Final Machine Learning Model. Jason Brownlee
https://machinelearningmastery.com/train-final-machine-learning-model/