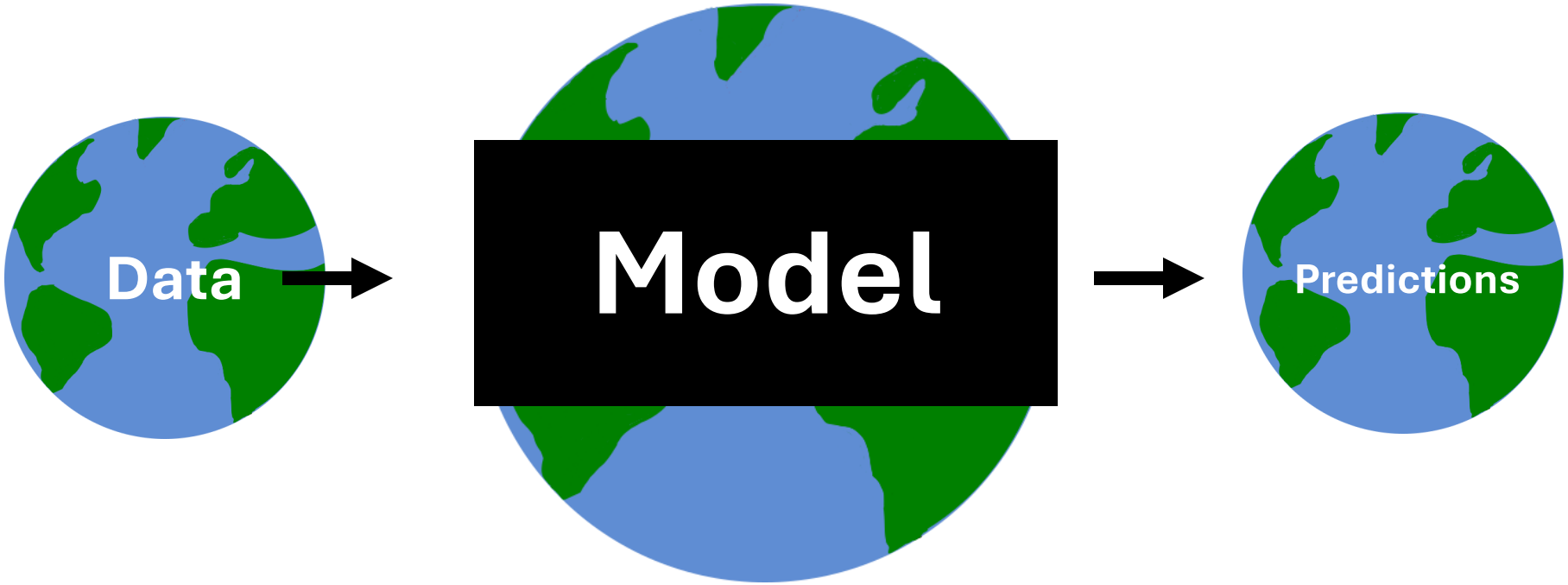
Validation

27 January 2026

Alex Lyman

Train, Validation, Test

This Slide Again



Training/Testing

- How do we know how good a model is?
  - Train long enough?
  - Feels good enough?

- Four methods that we commonly use:
  - Training set method
  - Static split test set
  - Random split test set CV
  - *N*-fold cross-validation
  - The last two are the more accurate approaches

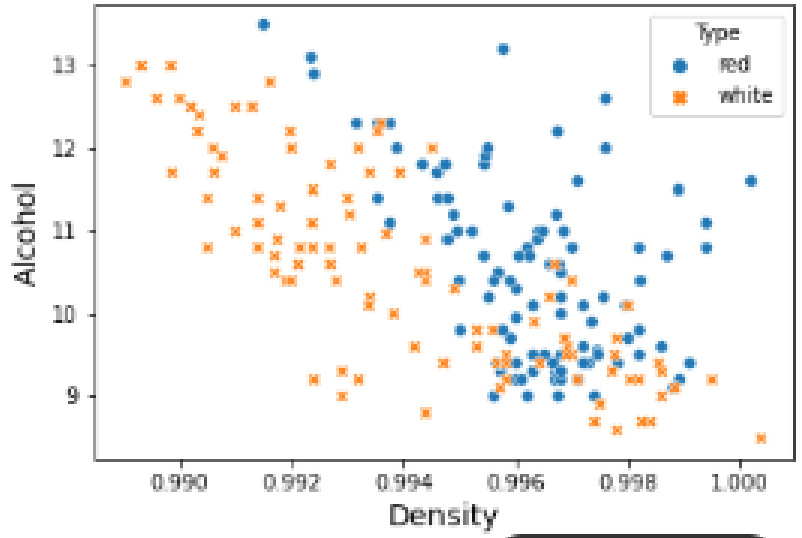
Training Set Method

- Procedure
  - Build model from the training set
  - Compute accuracy on the same training set
- Simple but least reliable estimate of future performance on unseen data (a rote learner could score 100%!)
- Not used as a performance metric but it is often important information in understanding how a machine learning model learns
- This is information which you will often report in your labs and then compare it with how the learner does on a better method

Static Training/Test Set

- Static Split Approach
  - The data owner makes available to the machine learner two distinct datasets:
    - One is used for learning/training (i.e., inducing a model), and
    - One is used exclusively for testing
- Note that this gives you a way to do repeatable tests
- Can be used for challenges (e.g. to see how everyone does on one particular unseen set, method we use for helping grade your labs.)
- Be careful not to overfit the Test Set ("Gold Standard")

Wine classification: white vs. red

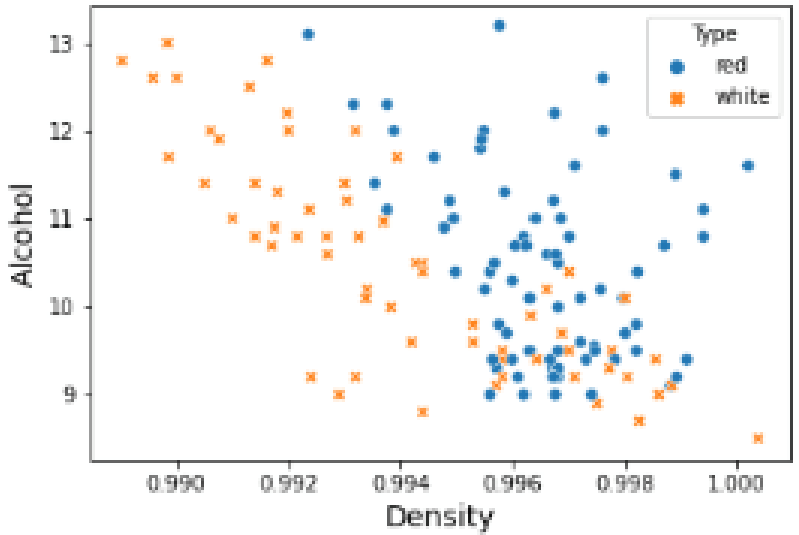


k-nearest neighbors

Hyperparameter: $k = 5$
Accuracy: 0.745

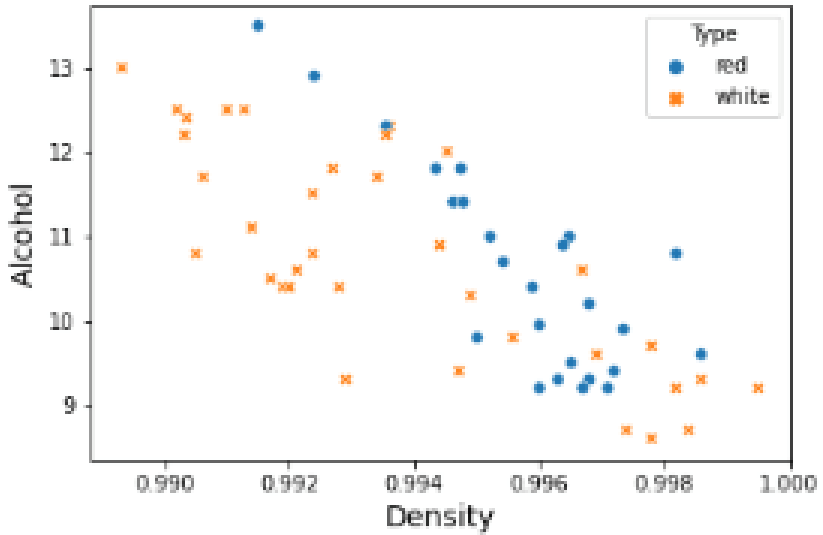
Model training

Subset 1



Model testing

Subset 2

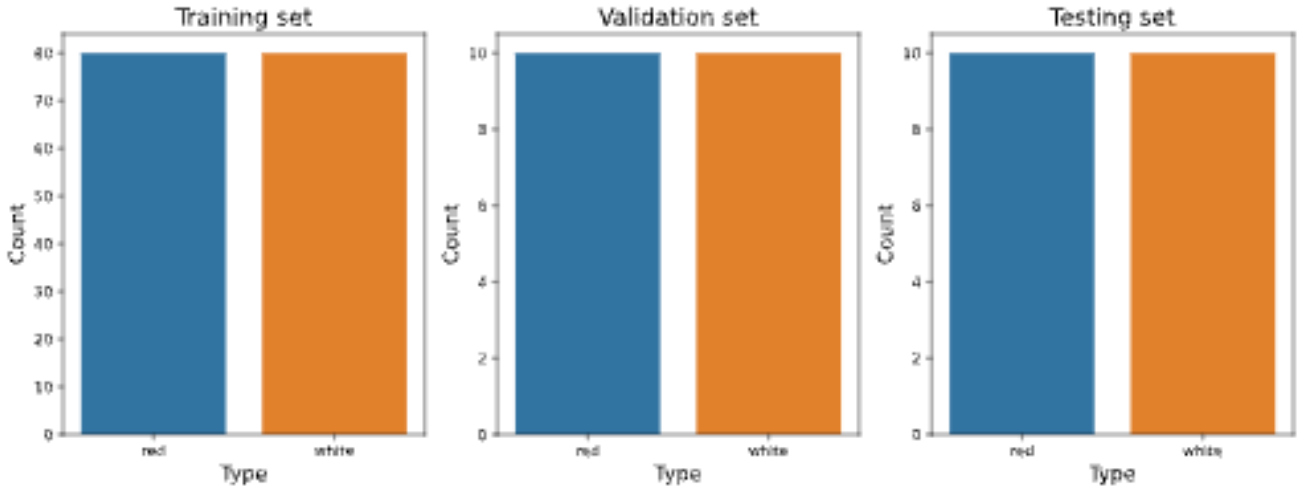


Random Training/Test Set Approach

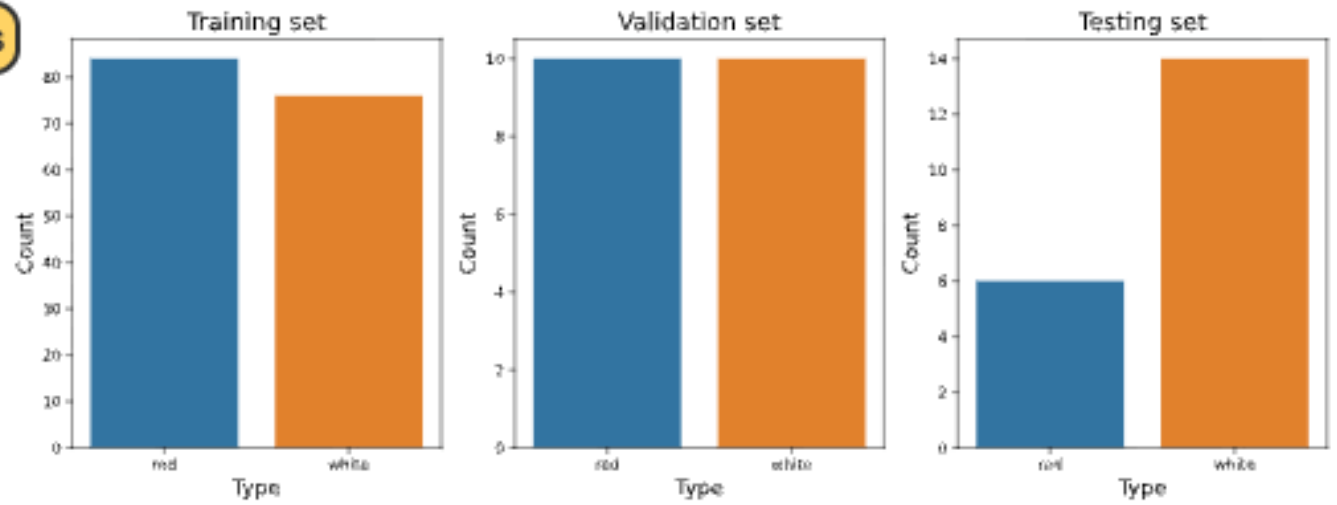
- Random Split Approach (aka holdout method)
  - The data owner makes available to the machine learner a single dataset
  - The machine learner splits the dataset into a training and a test set, such that:
    - Instances are randomly assigned to either set
    - The distribution of instances (with respect to the target class) is hopefully similar in both sets due to randomizing the data before the split
      - Stratification is an option to ensure proper distribution
    - Typically 60% to 90% of instances is used for training and the remainder for testing – the more data there is the more that can be used for training and still get statistically significant test predictions
  - Useful quick estimate for computationally intensive learners
  - Not statistically optimal (high variance, unless lots of data)
    - Could get a lucky or unlucky test set

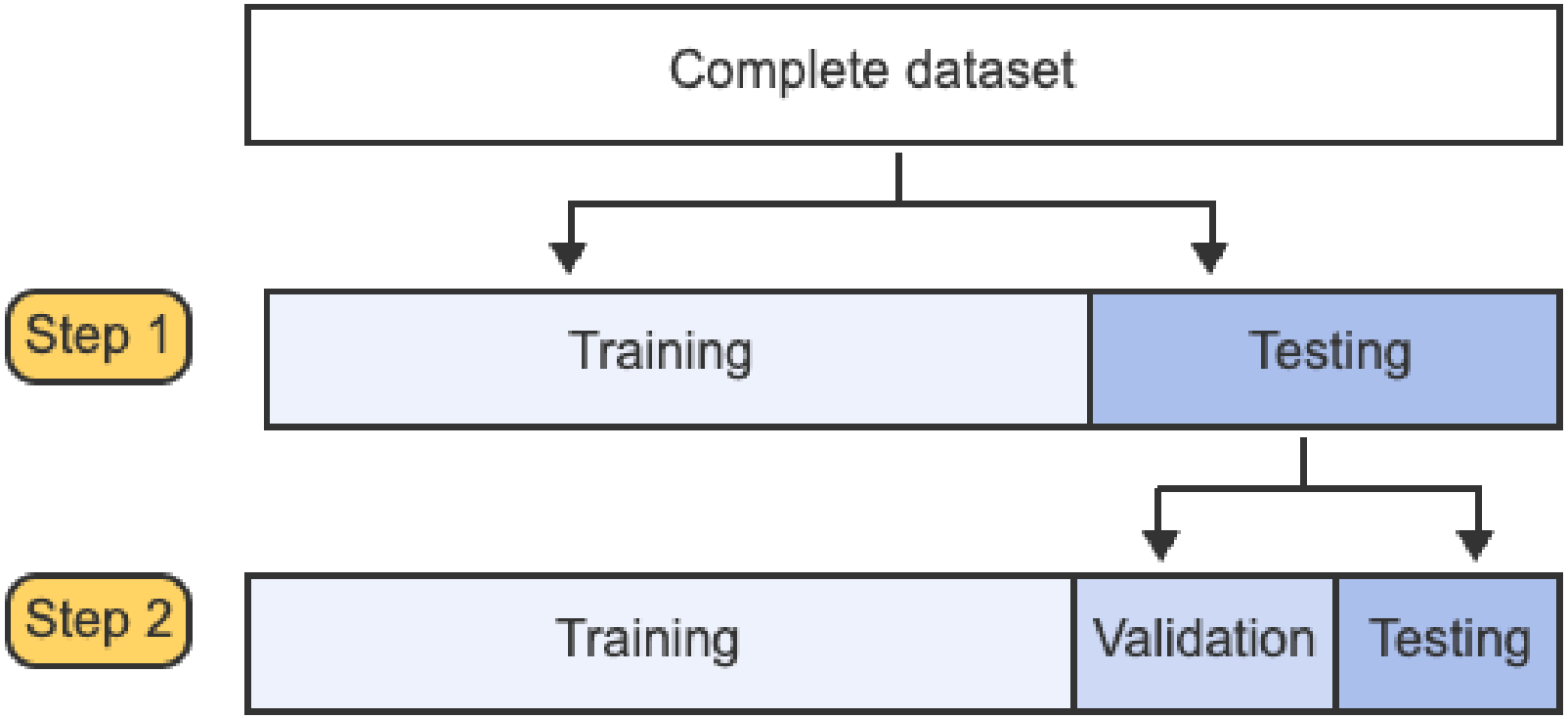
$n_{Train} = 160$ $n_{Val} = 20$ $n_{Test} = 20$

Stratified sets



Unstratified sets




Complete dataset
Step 1
Training
Testing
Step 2
Training
Validation
Testing

Use the slider to adjust the proportions allocated to training, validation, and testing.

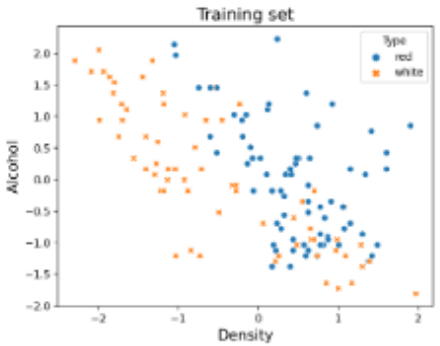
0.70 0.85

0.00 1.00

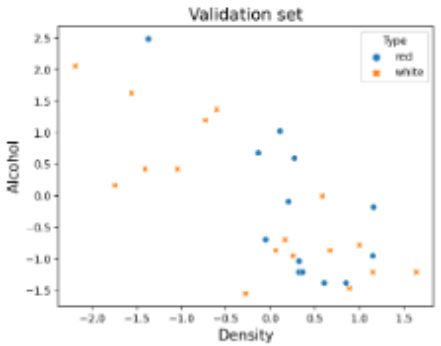
Set a random state value between 0 and 1000.

124 − +

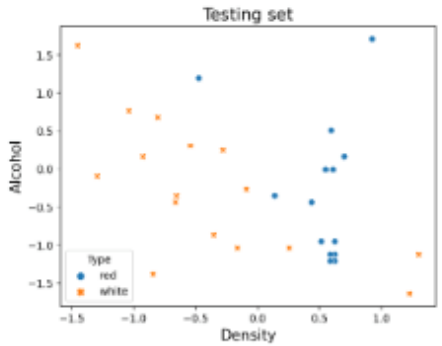
Training proportion: `0.7`



Validation proportion: `0.15`



Testing proportion: `0.15`



Fit k-nearest neighbors with k={3, 5, 7, 9} to the training set.

	k	Score
0	3	0.8741
1	5	0.8328
2	7	0.8327
3	9	0.8202

Choose the best performing k on the validation set.

	k	Score
0	3	0.7487
1	5	0.7772
2	7	0.7704
3	9	0.7704

Evaluate the "best" k from validation using the testing set.

Best k's score: `0.7742`

Best k = `5`

Cross-Validation

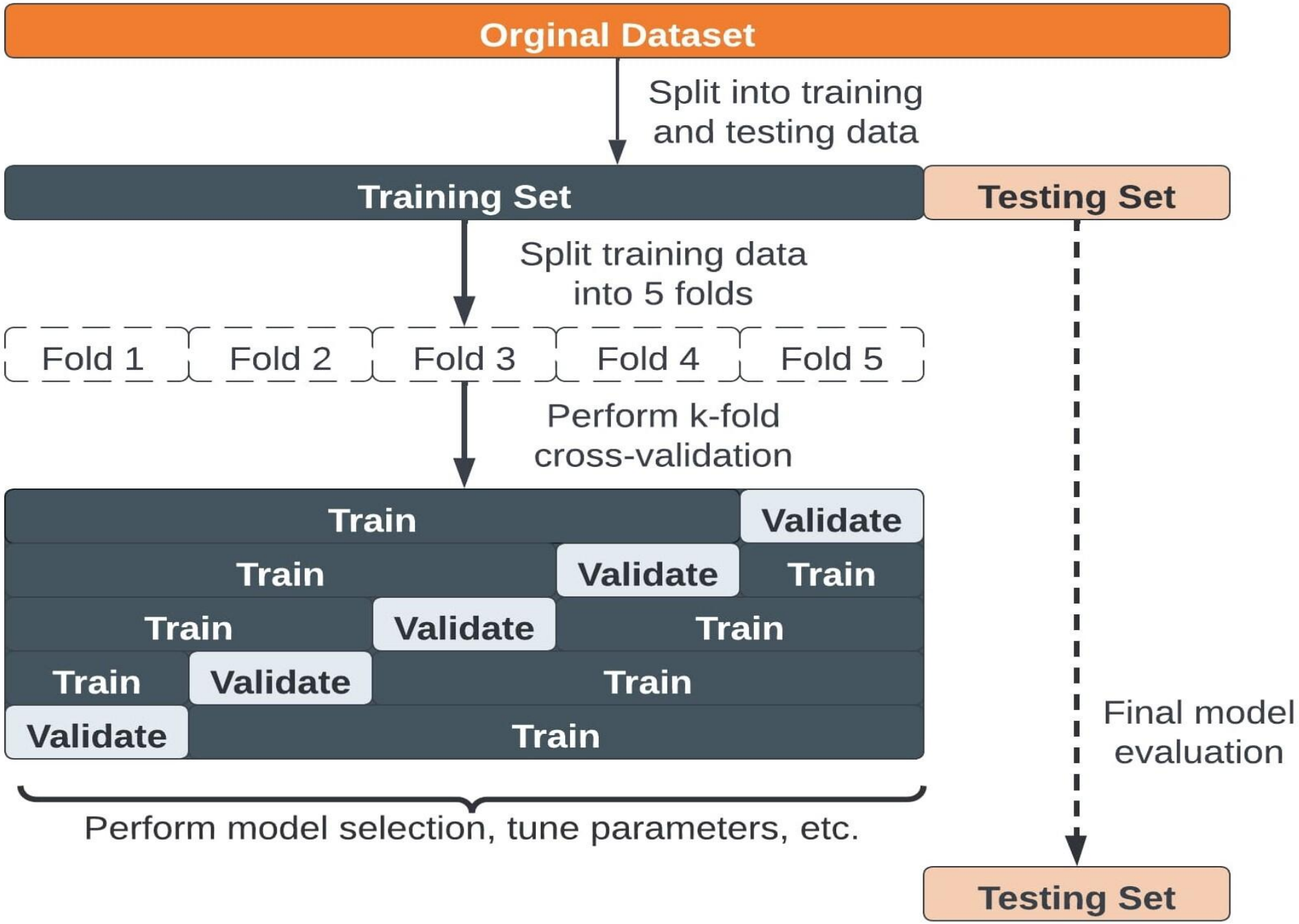
Cross-Validation (CV)

- Cross-Validation (CV) – Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations
- We then average the results of these iterations
- With CV we avoid having data just used for either training or validation, and give all training data a chance to be part of each, thus getting more accurate results

- Mainly used for comparing different learning models

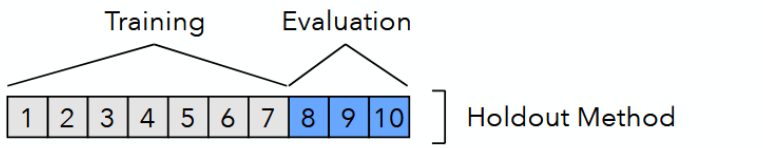
N-fold Cross-validation

- Use all the data for both training and validation
  - Statistically more reliable
  - All data can be used which is good, **especially for small data sets**
- Procedure
  - Partition the randomized dataset (call it D) into N equally-sized subsets S_1, ..., S_N
  - For k = 1 to N
    - Let M_k be the model induced from D - S_k
    - Let a_k be the accuracy of M_k on the instances of the test fold S_k
  - Return $(a_1+a_2+...+a_N)/N$


Orginal Dataset
Split into training and testing data
Training Set
Testing Set
Split training data into 5 folds
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Perform k-fold cross-validation
Train
Validate
Train
Validate
Train
Train
Validate
Train
Train
Validate
Train
Validate
Train
Perform model selection, tune parameters, etc.
Final model evaluation
Testing Set

N-fold Cross-validation (cont.)

- The larger *N* is, the smaller the variance in the final result
- The limit case where $N = |D|$ is known as *leave-one-out CV* and provides the most reliable estimate. However, it is typically only practical for small instance sets
- Commonly, a value of $N=10$ is considered a reasonable compromise between time complexity and reliability
- Note that N-fold CV is a better way to estimate how well we will do on novel data



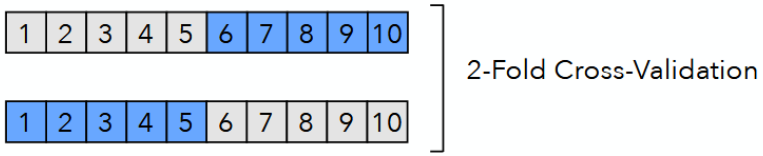
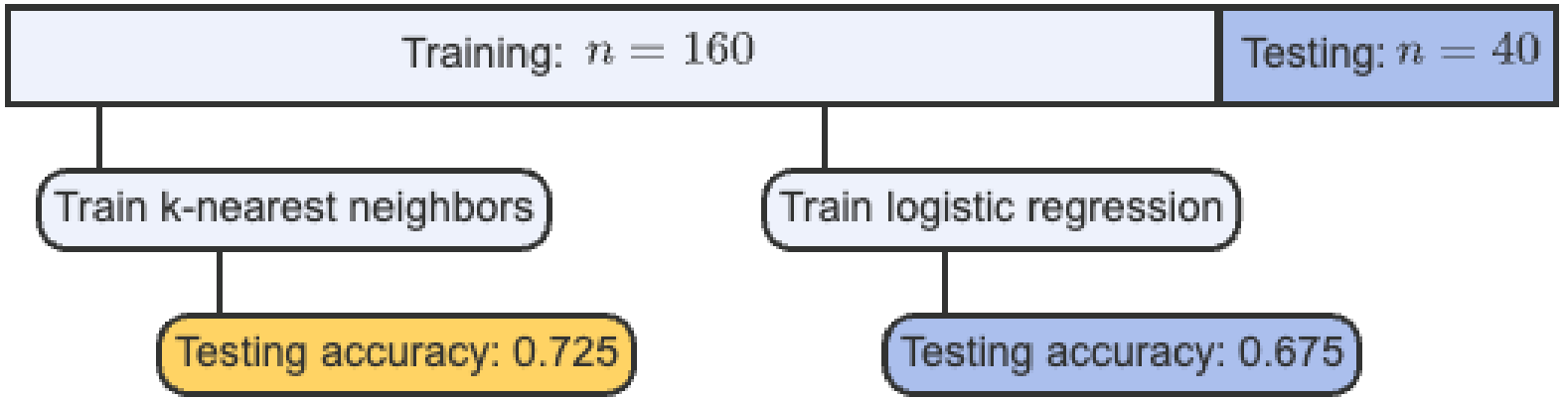


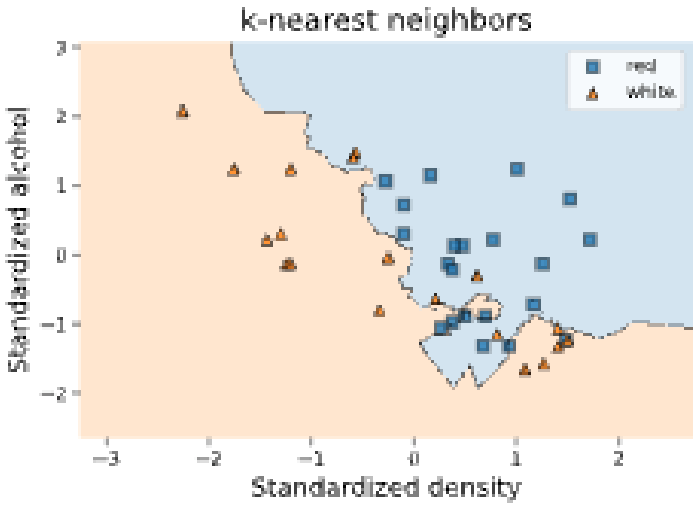
Image: Sebastian Raschka

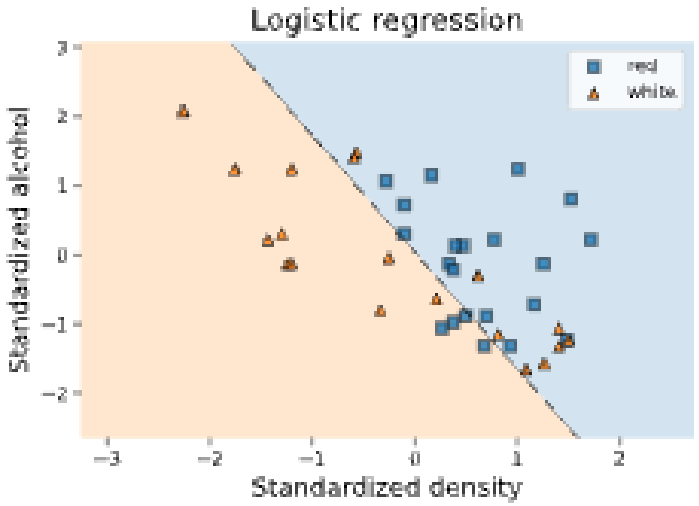
LOOCV

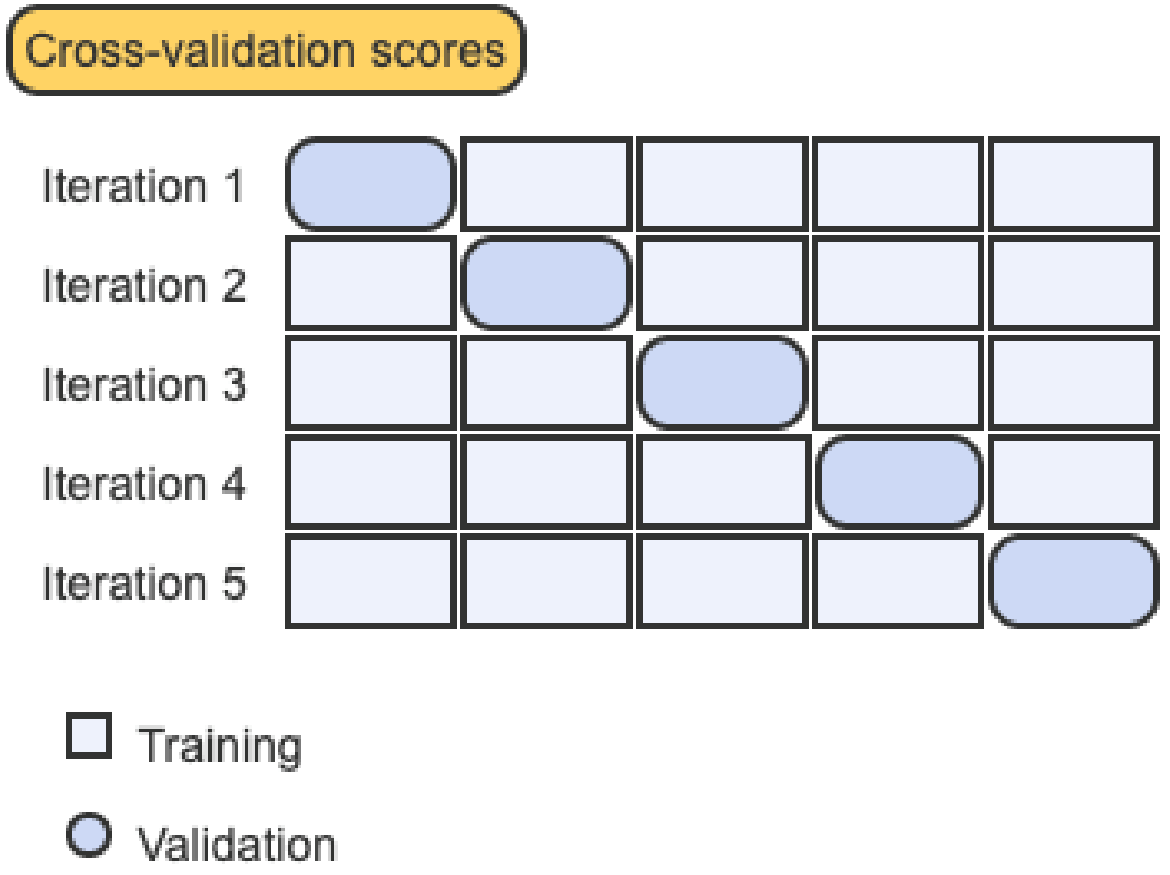
Model Selection With CV

- Still must chose an actual model to use during execution - how?
  - Could select the one model that was best on its fold?
  - We should use a method that takes advantage of as much data as we can.

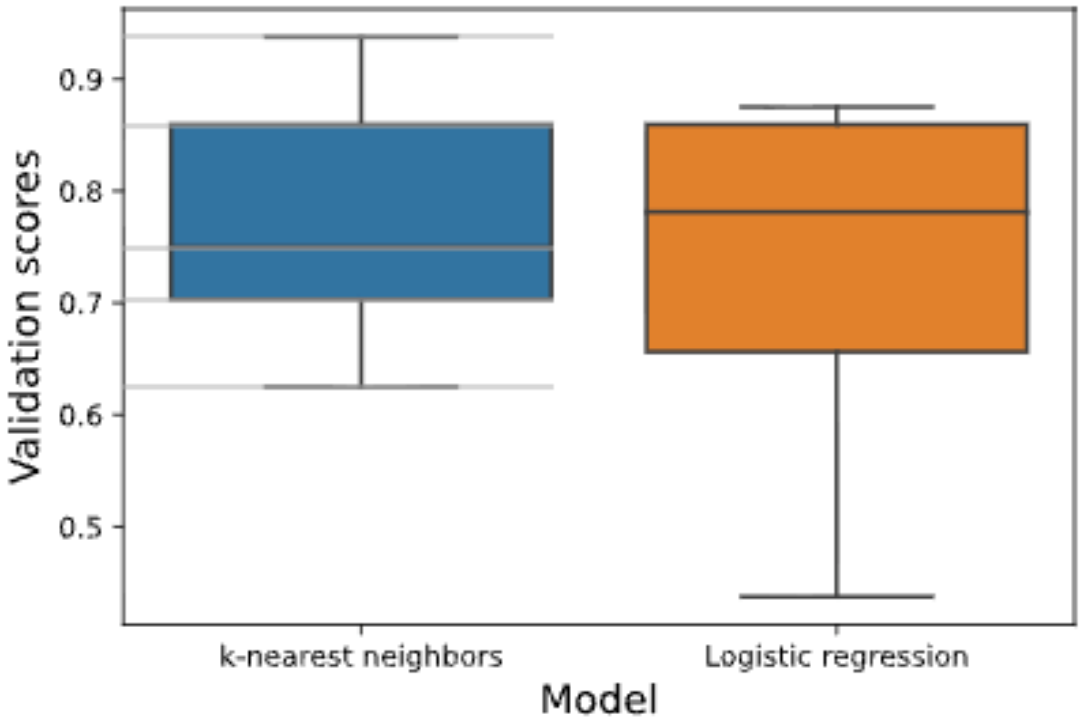

Training: $n = 160$
Testing: $n = 40$
Train k-nearest neighbors
Train logistic regression
Testing accuracy: 0.725
Testing accuracy: 0.675


k-nearest neighbors
red
white
Standardized alcohol
Standardized density

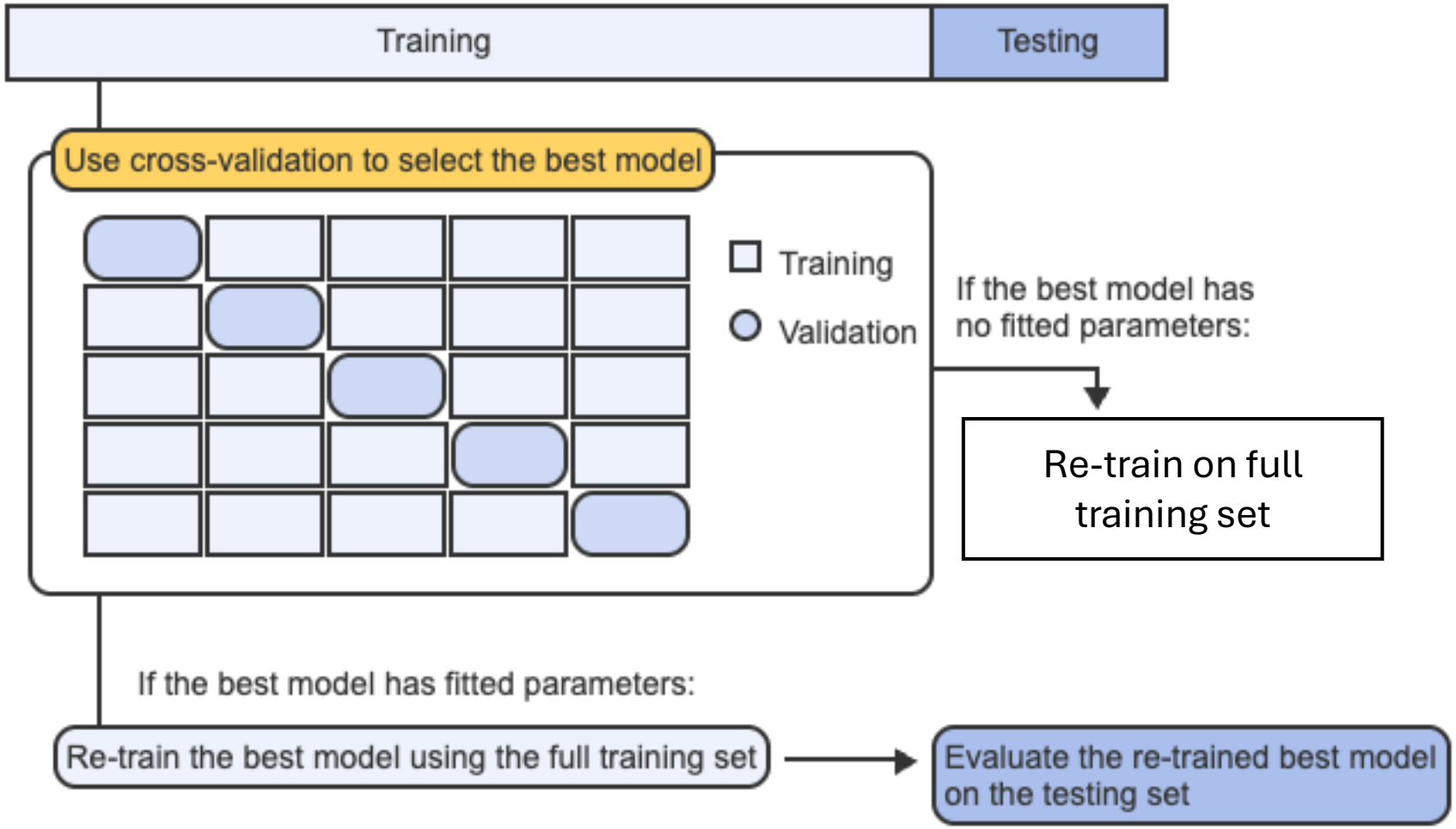

Logistic regression
red
white
Standardized alcohol
Standardized density



	k-nearest neighbors	Logistic regression
Iteration 1	0.750	0.750
Iteration 2	0.813	0.750
Iteration 3	0.813	0.750
Iteration 4	0.750	0.719
Iteration 5	0.750	0.844



	k-nearest neighbors	Logistic regression
IQR	0.85 - 0.70 = 0.15	0.85 - 0.65 = 0.20
Median	0.75	0.78
Minimum	0.63	0.44
Maximum	0.93	0.88


Training
Testing
Use cross-validation to select the best model
Training
Validation
If the best model has no fitted parameters:
Re-train on full training set
If the best model has fitted parameters:
Re-train the best model using the full training set
Evaluate the re-trained best model on the testing set

Model Selection

- What is the goal of our model evaluation?
- Why do we have so many metrics?
- How do we choose a metric?
- What is the goal of evaluating models?

Eugene Goostman

12 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Eugene Goostman is a chatbot that some regard as having passed the Turing test, a test of a computer's ability to communicate indistinguishably from a human. Developed in Saint Petersburg in 2001 by a group of three programmers, the Russian-born Vladimir Veselov, Ukrainian-born Eugene Demchenko, and Russian-born Sergey Ulasen,[1][2] Goostman is portrayed as a 13-year-old Ukrainian boy—characteristics that are intended to induce forgiveness in those with whom it interacts for its grammatical errors and lack of general knowledge.

The Goostman bot has competed in a number of Turing test contests since its creation, and finished second in the 2005 and 2008 Loebner Prize contest. In June 2012, at an event marking what would have been the 100th birthday of the test's author, Alan Turing, Goostman won a competition promoted as the largest-ever Turing test contest, in which it successfully convinced 29% of its judges that it was human.

On 7 June 2014, at a contest marking the 60th anniversary of Turing's death, 33% of the event's judges thought that Goostman was human; the event's organiser Kevin Warwick considered it to have passed Turing's test as a result, per Turing's prediction in his 1950 paper Computing Machinery and Intelligence, that by the year 2000, machines would be capable of fooling 30% of human judges after five minutes of questioning. The validity and relevance of the announcement of Goostman's pass was questioned by critics, who noted the exaggeration of the achievement by Warwick, the bot's use of personality quirks and humour in an attempt to misdirect users from its non-human tendencies and lack of real intelligence, along with "passes" achieved by other chatbots at similar events.[3][4][5]

Personality [edit]

Eugene Goostman is portrayed as being a 13-year-old boy from Odesa, Ukraine, who has a pet guinea pig and a father who is a gynaecologist. Veselov stated that Goostman was designed to be a "character with a believable personality". The choice of age was intentional, as, in Veselov's opinion, a thirteen-year-old is "not too old to know everything and not too young to know nothing". Goostman's young age also induces people who "converse" with him to forgive minor grammatical errors in his responses.[1][6] In 2014, work was made on improving the bot's "dialog controller", allowing Goostman to output more human-like dialogue.[2]

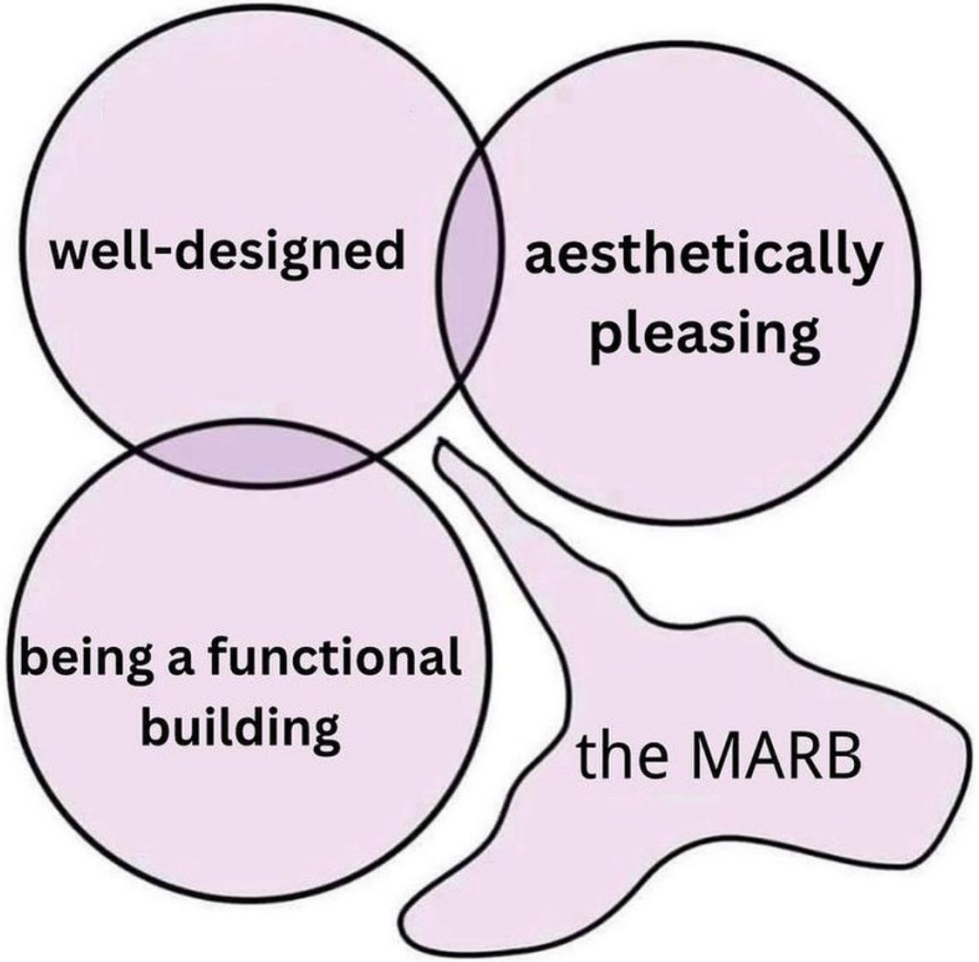
A conversation between Scott Aaronson and Eugene Goostman ran as follows:[7]

> **Scott:** Which is bigger, a shoebox or Mount Everest?
> **Eugene:** I can't make a choice right now. I should think it out later. And I forgot to ask you where you are from…
> **Scott:** How many legs does a camel have?
> **Eugene:** Something between 2 and 4. Maybe, three? :-))) By the way, I still don't know your specialty – or, possibly, I've missed it?
> **Scott:** How many legs does a millipede have?
> **Eugene:** Just two, but Chernobyl mutants may have up two five. I know you are supposed to trick me.
> **Scott:** No, I need to know that you're not a chatbot. Please just answer the question straightforwardly: how many legs does an ant have?
> **Eugene:** Something between 2 and 4. Maybe three? :-))) Oh, what a fruitful conversation ;-)

Model Selection

- What does the Eugene Goostman story tell us about model evaluation and selection?

Quiz Time


well-designed
aesthetically pleasing
being a functional building
the MARB