UiO : University of Oslo

institutt for informatikk

IN3050/IN4050 - Introduction to Artificial Intelligence and Machine Learning

Lecture 9: **More on supervised learning**

Pooya Zakeri, Fall 2025

EVALUATION

Topics in supervised machine learning

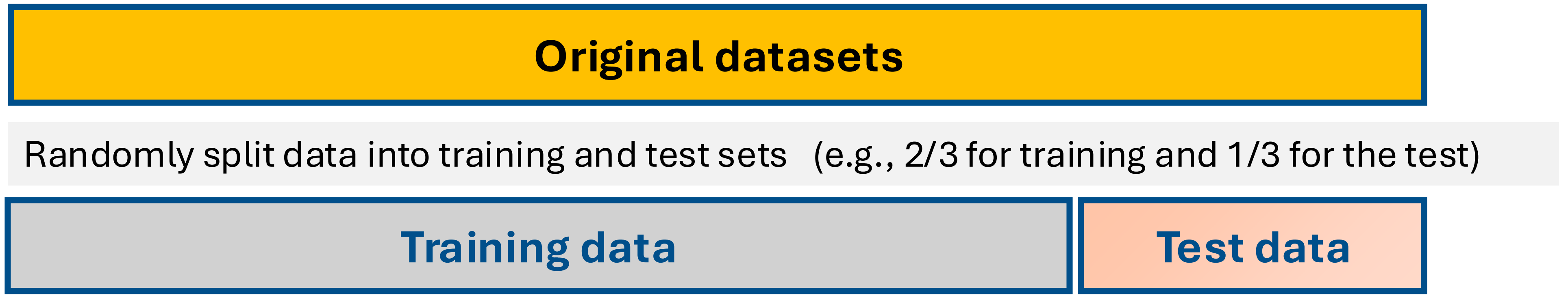
- Model Selection
- Evaluation
  - Evaluation strategies
  - Evaluation metrics
- Bias-variance tradeoff
- Overfitting
- Regularization

Why Evaluation?

- There are many models and algorithms, which one is the best?
- Performance on the training data (data used to build models) is *not* a good indicator of performance on future data
    - ***Q: Why?***
    - A: Because new data will probably not be **exactly** the same as the training data!
- How predictive is the model we learned?
    - For regression, usually MSE or MAE, RMSE
    - For classification, many options (discuss later today)
        - Accuracy can be used, with caution

Train and test subsets

- Model does not generalize to unseen data
  - Fail to predict samples that are not in the training samples
  - Pick a model that has a **lower generalization error**
- How to evaluate generalization error?
  - Split your data randomly into ***train***, *validation*, and ***test sets*** (*holdout* method)
  - Use the train subset to train models.
  - Use **test subset** error as an *estimator* of generalization error



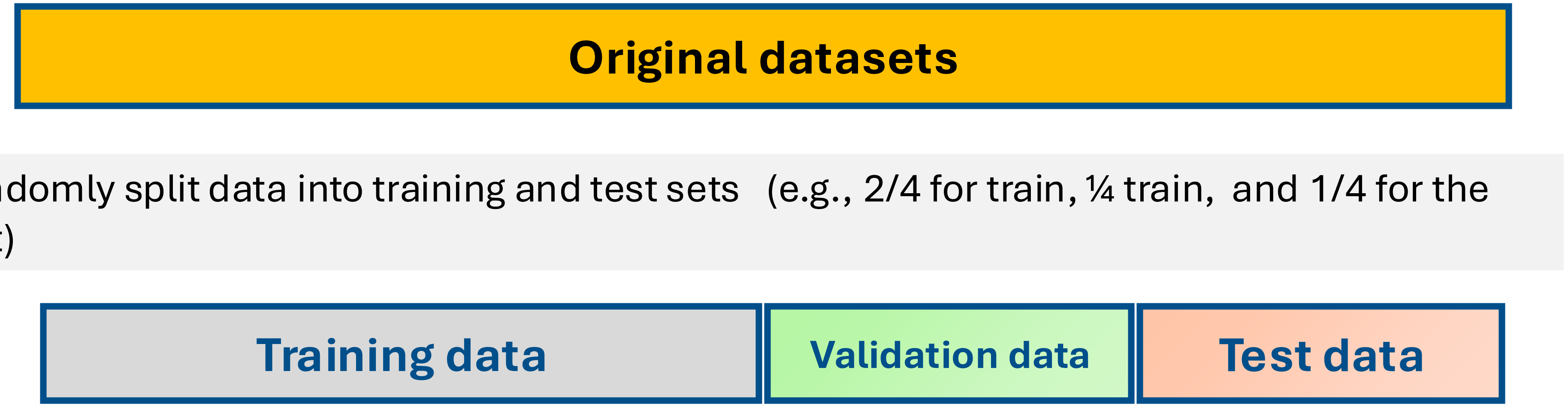
Randomly split data into training and test sets (e.g., 2/3 for training and 1/3 for the test)

Validation subset

- How to tune Hyperparameters?

  It is important that the test data is not used *in any way* to build the model

  A common procedure uses three sets: **training data, validation data, and test data, then validation data is used to optimize the model’s parameters**



Randomly split data into training and test sets (e.g., 2/4 for train, ¼ train, and 1/4 for the test)

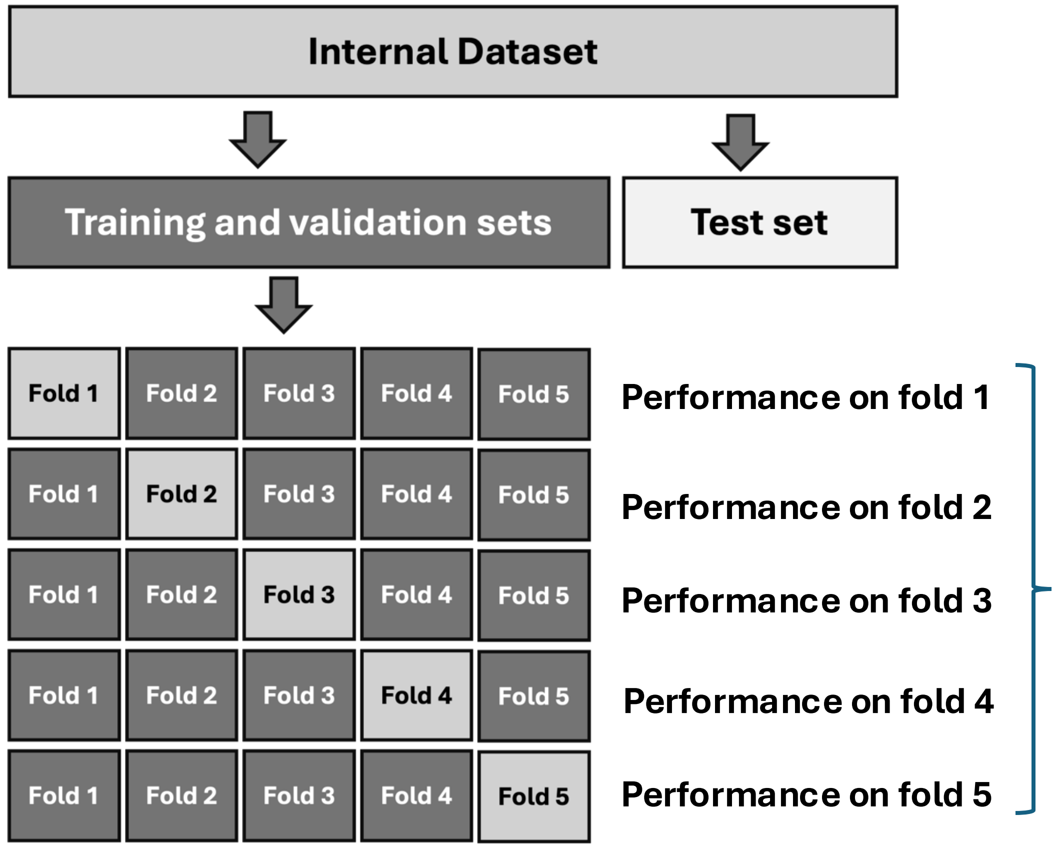
Notes and Considerations:

- The *holdout* method reserves a certain amount for testing and uses the remainder for training
    - Usually, one third for testing, the rest for training
- For "unbalanced" datasets, samples might not be representative
    - Few or no instances of some classes
- *Stratified sample: an advanced version of balancing the data*
    - Make sure that each class is represented with approximately equal proportions in both subsets

Notes and Considerations:

- Unbiased estimation using the internal dataset requires:
  - A test subset set aside before starting to experiment.
  - Only a single analysis performed using the test samples.
    - ➢ Only a single (ensemble) model.
    - ➢ Ideally, also pre-specify what performance evaluation metric you would use.
    - ➢ It is fine to perform additional analyses but note that these might be biased by the multiple comparisons and learning from previous analyses.

Cross-validation



Average the scores

Tuning parameters using k-fold cross-validation

1. Randomly partition the non-test subset into k folds of equal size.
2. Repeat k times:

   Use the kth fold for validation and the other k-1 folds for training.
3. Use the average of the performance estimates obtained for the validation folds
   - ➢ Calculate the standard deviation or other measures of uncertainty.

Evaluation on “small” data

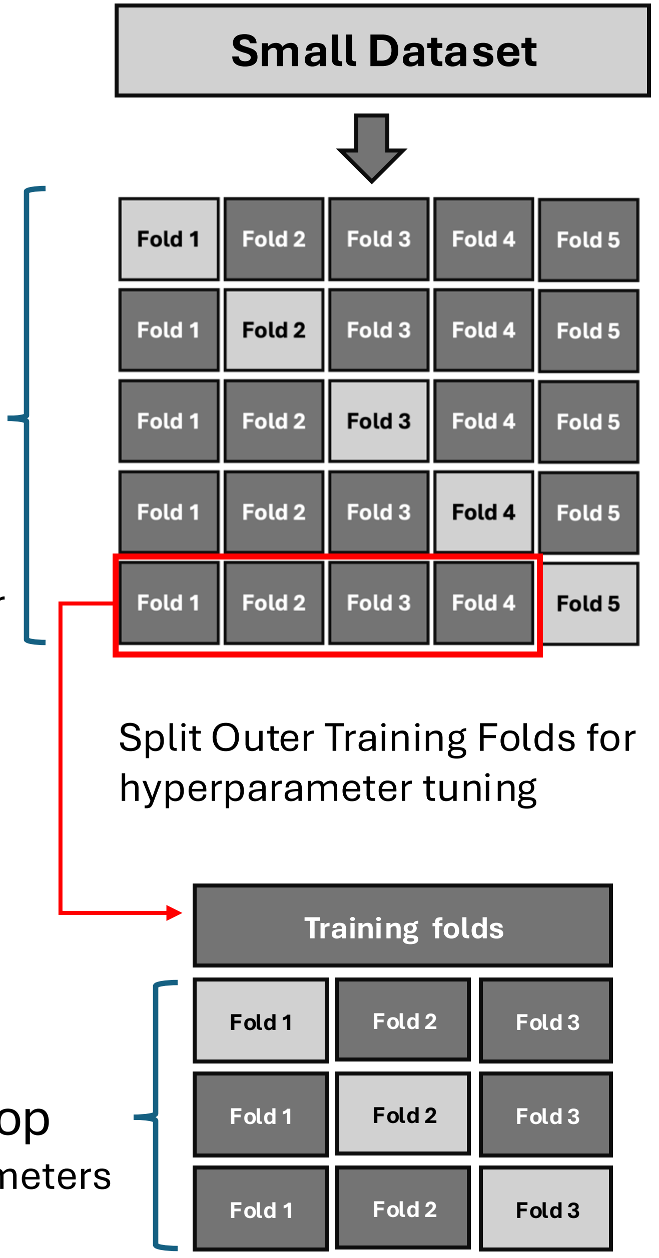
- What if we have a small data set?
  - The chosen 2/3 for training may not be representative.
  - The chosen 1/3 for testing may not be representative.
- *Cross-validation is more useful in small datasets*
  - First step: data is split into *k* subsets of equal size
  - Second step: each subset in turn is used for testing and the remainder for training
- For classification, often the subsets are stratified before the cross-validation is performed (*Stratified k-Fold cross-validation*)
- The error estimates are averaged to yield an overall error estimate

Notes and consideration in Cross-validation

- **Leave-One-Out Cross-Validation (LOOCV)**
  - A special case of K-Fold where K = number of data points
  - More computationally expensive
  - Useful for small datasets
- **Repeated k-Fold cross-validation (Repeated random sub-sampling CV**)
  - K is the number of times we will train the model and not the number of folds
  - Robust
  - the proportion of train/test split is not dependent on the number of iterations
  - Some samples may never be selected to be in the test set at all
- **Nested resampling**
  - It estimates the generalization error of the underlying model and its (hyper)parameter search.
  - For tuning hyperparameters (in small datasets).
  - Computationally very expensive

Further readings

Nested cross-validation



Outer loop

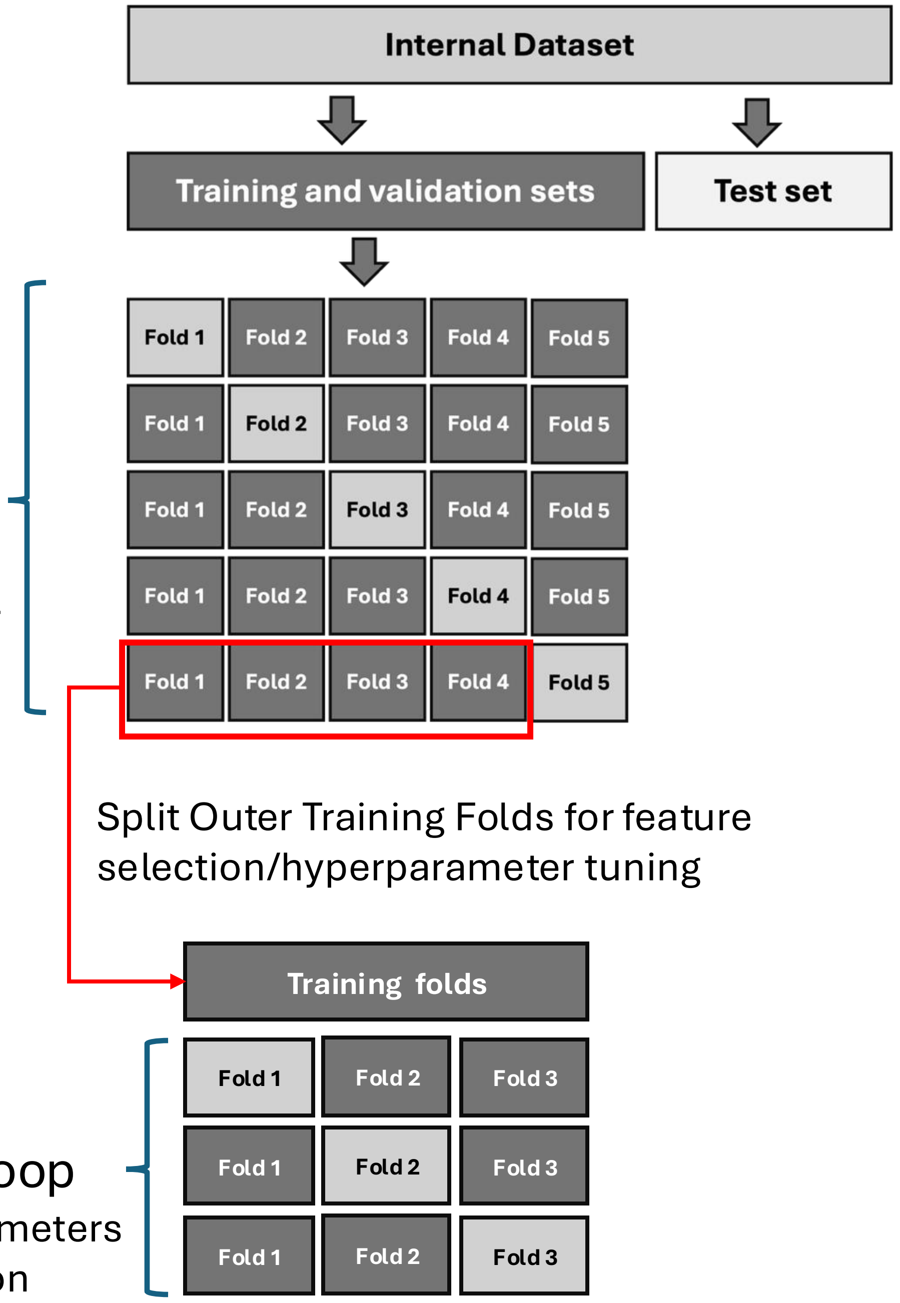
1. Train each split with optimal hyperparameter/features
2. Average each split's test error

Split Outer Training Folds for hyperparameter tuning

Inner loop

Tune hyperparameters

Nested cross-validation



Outer loop

1. Train each split with optimal hyperparameter/features
2. Average each split's test error

Split Outer Training Folds for feature selection/hyperparameter tuning

Inner loop

Tune hyperparameters
Feature selection

Cross-validation: Pros and Cons

- **Pros**
  - Estimates are less influenced by one particular training.
  - Estimates are less influenced by one particular validation subset.
  - Ends up with k models which can be used in an ensemble model.
  - Critical for small datasets: Maximizes data utility by recycling limited samples for both training and validation, avoiding unreliable single splits.
- **Cons:**
  - Training time (and validation time) multiplied by k
    - Could be acceptable for reasonably small n, e.g., 5.

Further readings

More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
  - Why ten? Extensive experiments have shown that this is the best choice to get an accurate estimate
  - Stratification reduces the estimate’s variance
- Even better: repeated stratified cross-validation
  - E.g., ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Sampling with replacement (*bootstrap validation*)

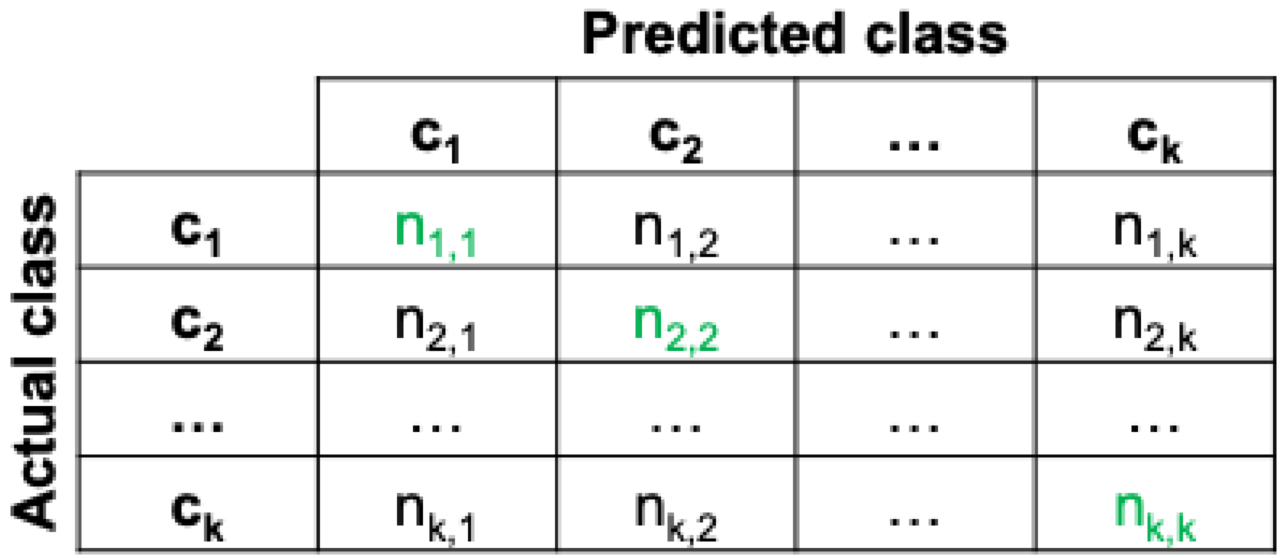
- k-fold cross-validation applies a fixed partitioning into folds.
- We can instead repeatedly partition the non-test subset into a train subset and a validation subset. For e.g., using bootstrapping.
- Bootstrapping example: repeat 10 times:
  1. Sample with replacement from the non-test data to obtain a train subset with the same size as the non-test subset.
     - ➢ Each train subset will, on average, contain 63.2% of the non-test data but will be the same size as the non-test subset due to duplicates.
  2. Train using the train subset (with duplicates) and use the other non-test data for validation

Homework Reflection

- Recap: If a random test subset is used for only a single analysis, then the resulting estimate is expected to be unbiased for the internal data.
- Does that imply unbiased estimation for the intended application?

Confusion Matrix

- A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data
- Each element specifies the number of samples with a particular class c_A predicted to be a particular class c_P

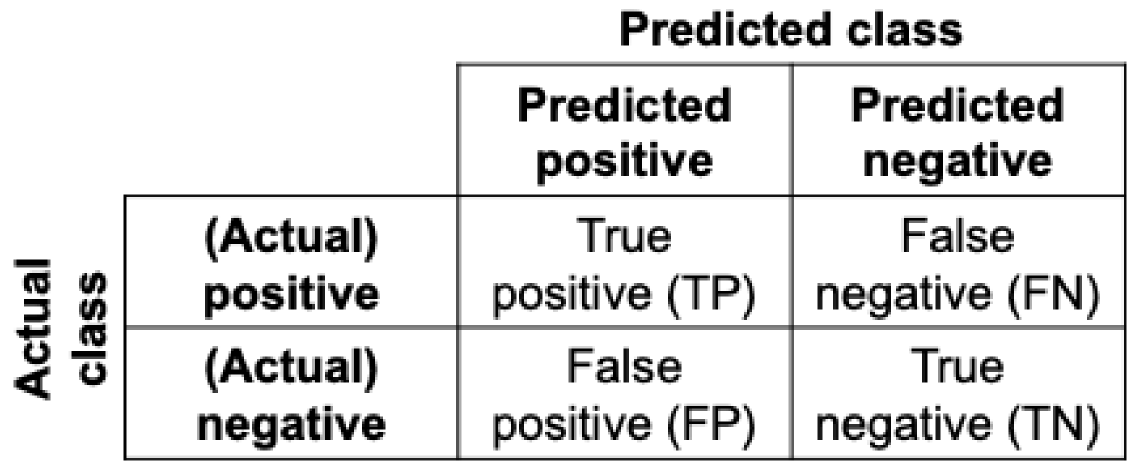


		Predicted class			
		c_1	c_2	...	c_k
Actual class	c_1	$n_{1,1}$	$n_{1,2}$	...	$n_{1,k}$
	c_2	$n_{2,1}$	$n_{2,2}$	...	$n_{2,k}$
	...	...	...	...	...
	c_k	$n_{k,1}$	$n_{k,2}$	...	$n_{k,k}$

- c_i specifies a class and $n_{a,b}$ specifies a count
- Diagonal elements specify correctly classified samples.

Confusion Matrix

- In the special case of two classes, the confusion matrix contains only four values, often named as:



		Predicted class	
		Predicted positive	**Predicted negative**
Actual class	**(Actual) positive**	True positive (TP)	False negative (FN)
	(Actual) negative	False positive (FP)	True negative (TN)

- ➢ True Positives (TP): These are cases in which we predicted positive, and they are actually positive(e.g. they have the disease, and they do have the disease.)
- ➢ True Negatives (TN): We predicted negative, and they are negative (e.g., we predict they are healthy, and they don't have the disease.)
- ➢ False Positives (FP): We predicted positive, but they are actually negative. (e.g., we predicted they do have the disease, but they don't actually have the disease. (Also known as a "**Type I error**.")
- ➢ False Negatives (FN): We predicted negative, but they are actually positive, (e.g., we predicted them as healthy, but they actually do have the disease. (Also known as a "**Type II error**.")

Evaluation Metrics: Accuracy

- Overall correct predictions (all classes)

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

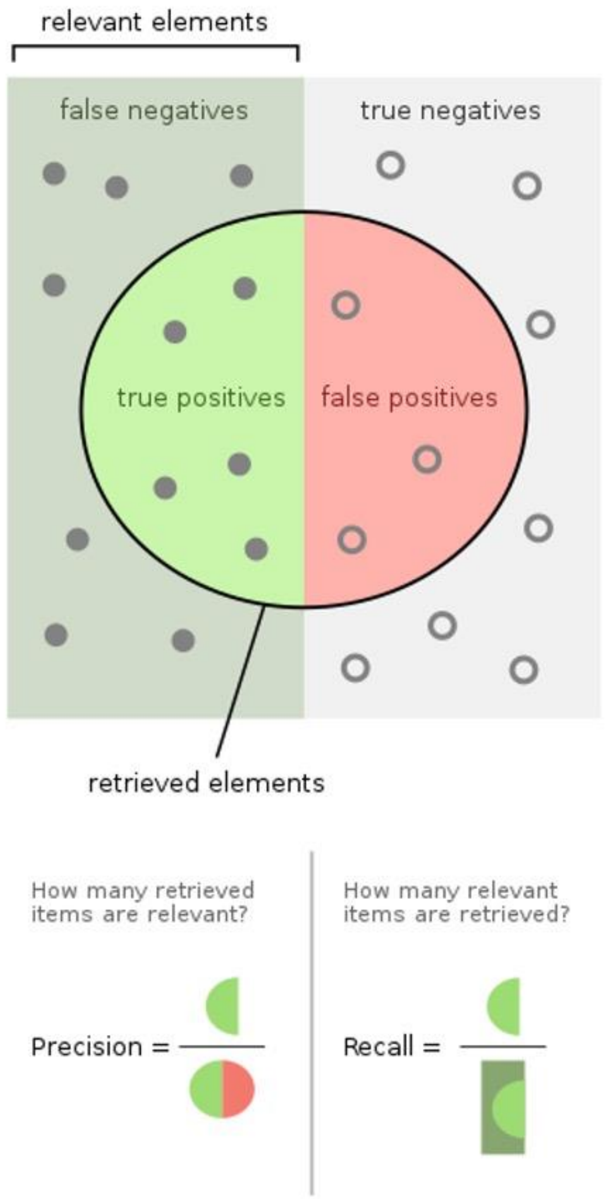
- When Accuracy is Misleading?
  - Imbalanced Datasets (e.g., Medical Diagnosis, Fraud Detection)
    - Example: 1000 patients, only 10 have a disease (1% positive cases).
    - A naive classifier that always predicts ”No Disease” achieves:
    - Accuracy = 99% (because it correctly predicts 990 negatives). But it completely fails at identifying the 10 actual cases!
  - Overconfident Incorrect Predictions in Softmax-based Classification
    - Accuracy does not distinguish between confident and uncertain predictions.
    - A model predicting a 0.51 probability for the correct class is treated as 100% correct. 0.49 probability is treated as 100% wrong.
  - And more?

Sensitivity and specificity

- From the confusion matrix for two classes

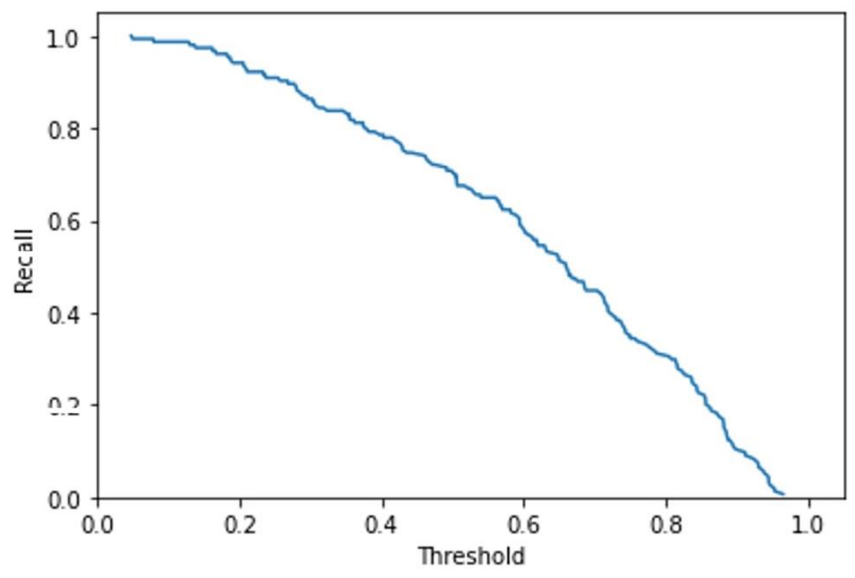
$$Sensitivity = \frac{TP}{TP+FN}$$

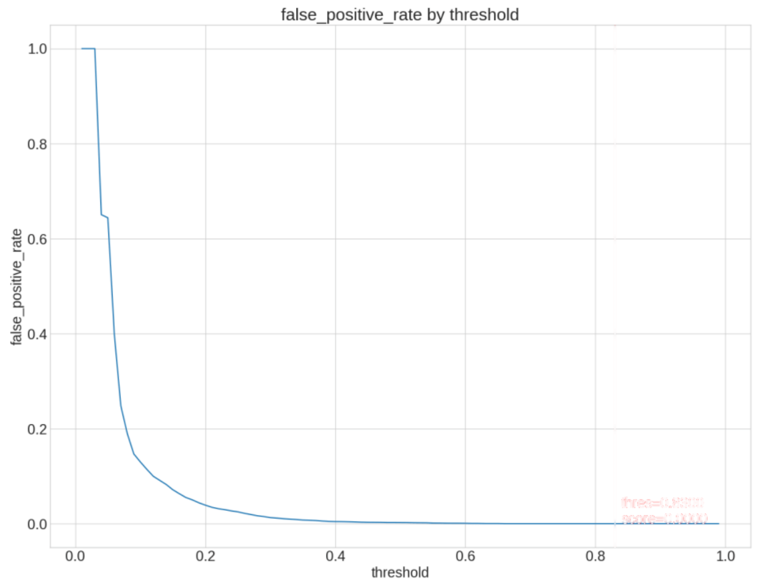
- Proportion of actual positives that are positive predictions.
- Same as recall (a.k.a. true positive rate).
- Proportion of actual negatives that are negative predictions (a.k.a. true negative rate.)
- False positive rate = 1 - Specificity = type I error rate
- False negative rate = 1 - Sensitivity = type II error rate



TPR and FPR

- For a trained binary classifier:
  - If we increase the threshold, only the higher scored observations will be classified as positive. As you increase the threshold for classifying the input as positive, both the FPR and recall will typically go down.



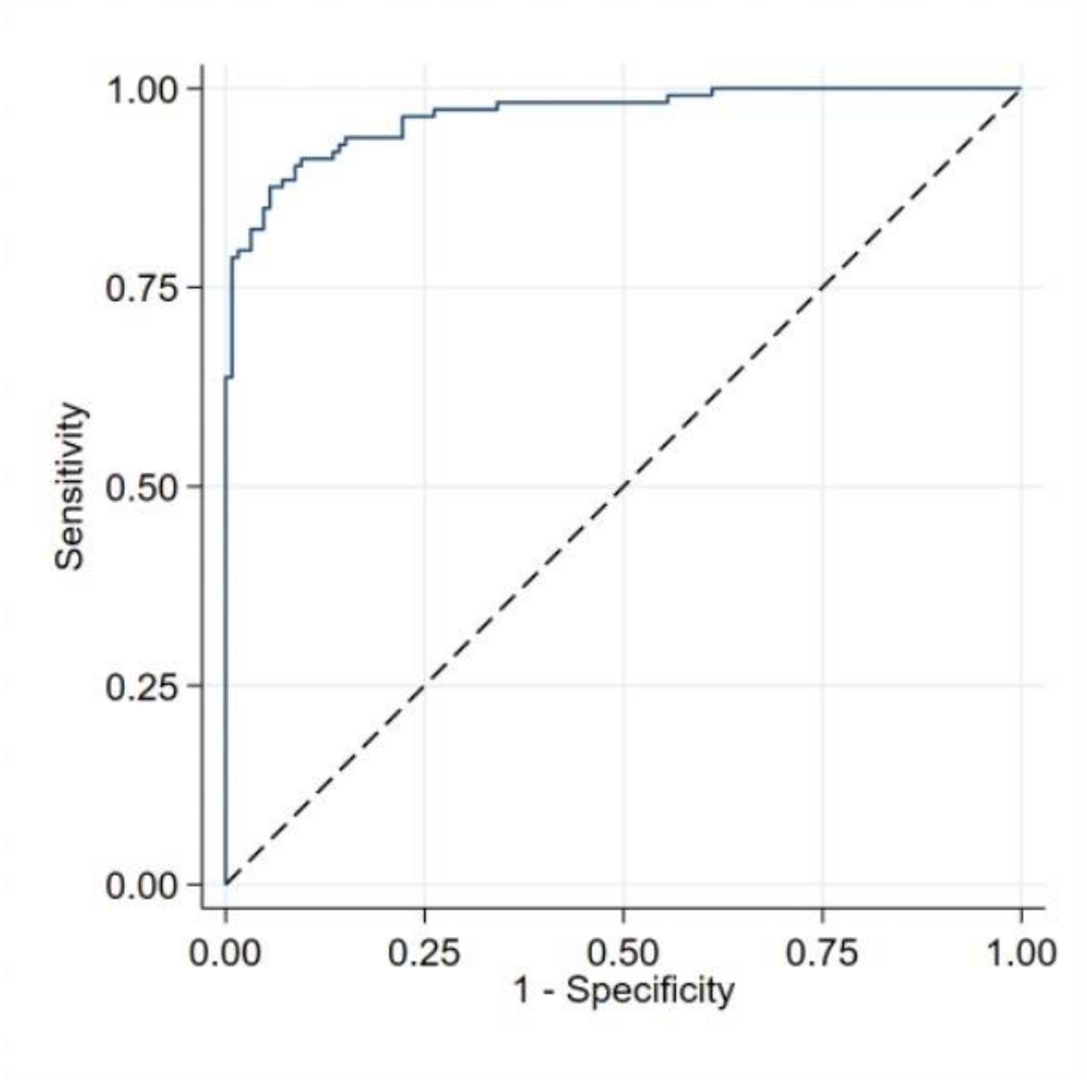


Receiver Operator Characteristic (ROC) curves

A curve depicting the pairs of sensitivity (recall, a.k.a. TPR) vs. 1-specificity (a.k.a. FPR) for different thresholds

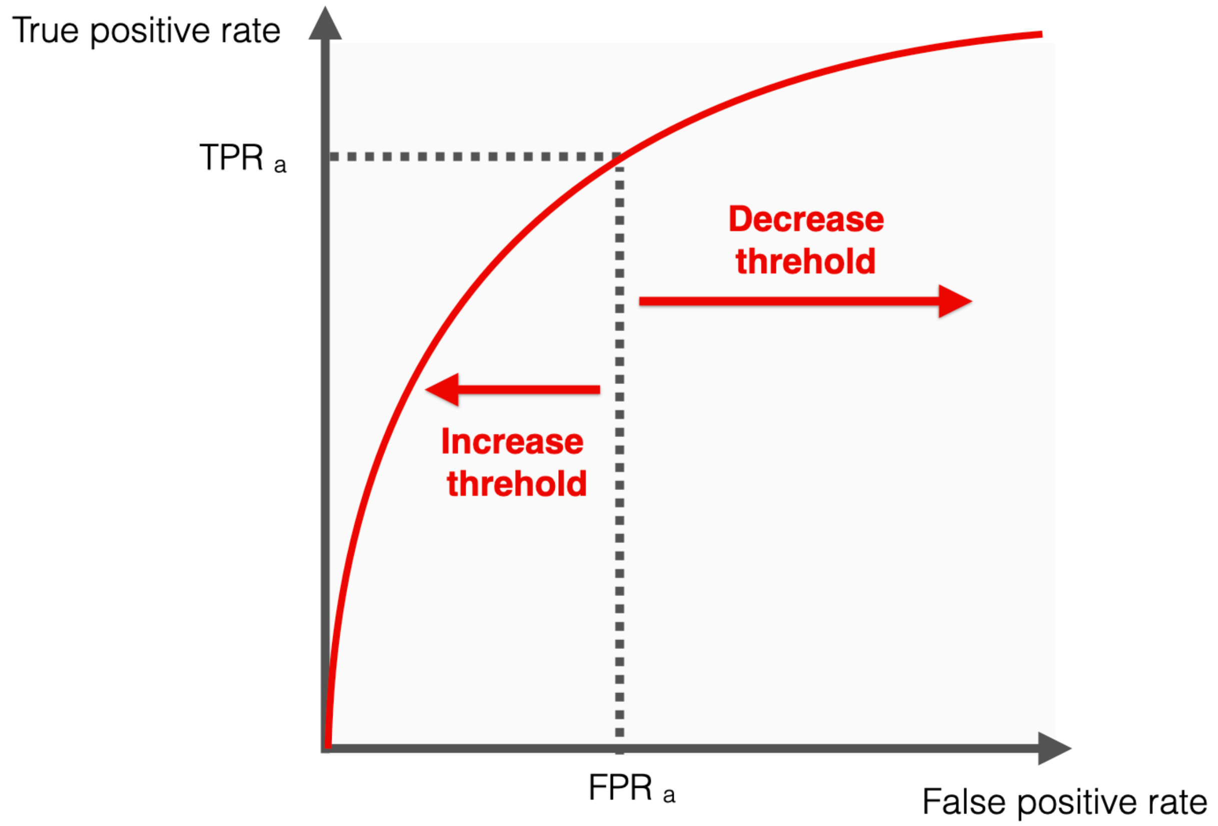
  - Each point on the curve corresponds to a combination of FPR and TPR values at a specific decision threshold.
  - Linear interpolation is used between the new and the previous (1-Specificity, Sensitivity) pair.

- ROC curves start at (0,0) and end at (1,1).



ROC Curve and AUC

- Area under the curve measures overall performance
- AUC and AUROC is the area under the ROC curve. AUC falls within 0 and 1.
- Useful for binary classification problems
- ROC Curve is usually used in binary classification, but it can also be used in multi-class classification using averaging methods.
- Thresholds are not explicitly shown in the roc curves



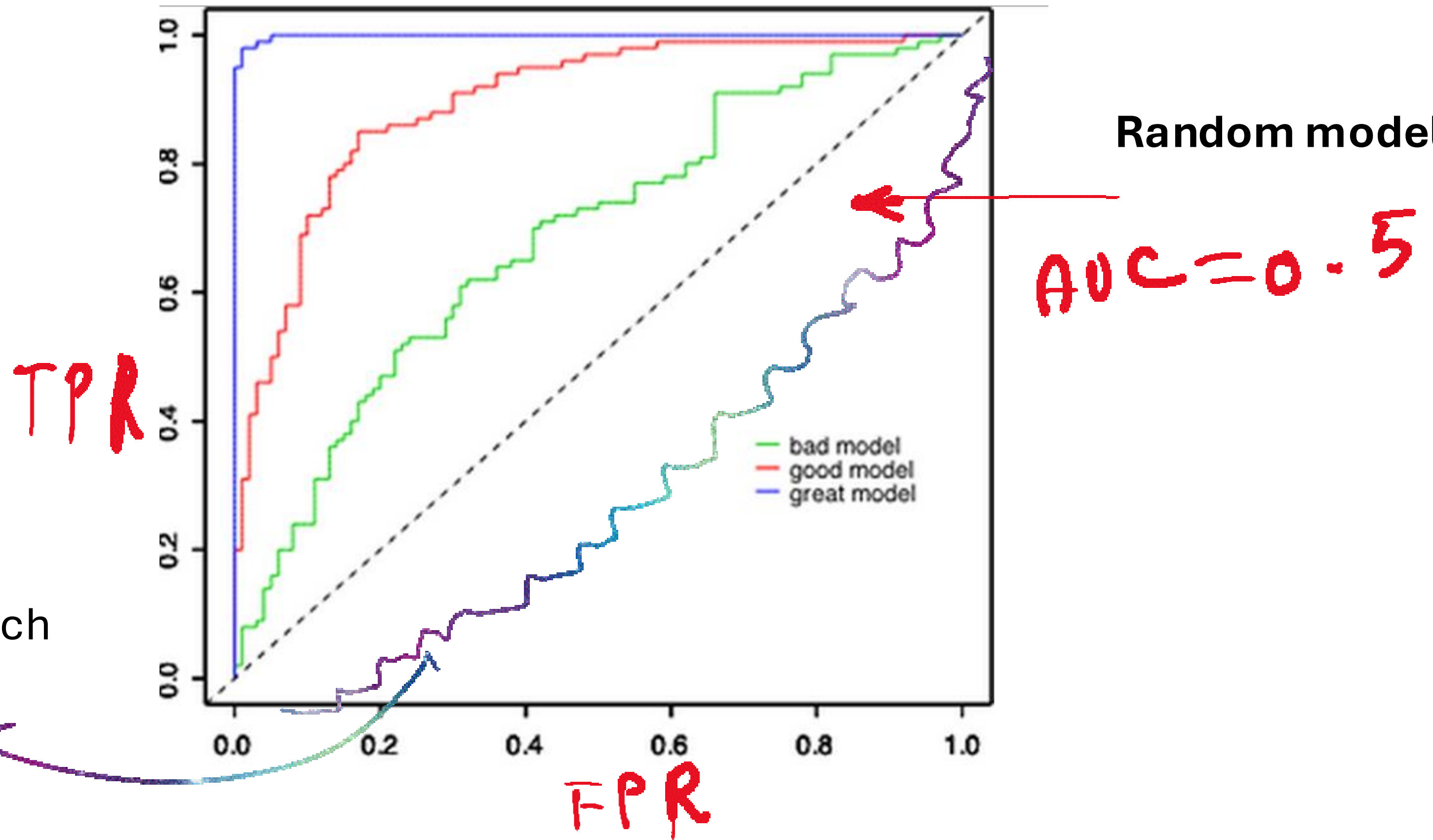
Plot the ROC Curve

- **Step 1:** Make predictions using the trained model (compute the scores or probabilities) for each test sample.
- **Step 2:** Order the scores (descending).
- **Step 3:** Select thresholds, equally spaced across the range of scores (e.g., ten thresholds).
- **Step 4:** For each threshold, repeat steps 5 to 7.
- **Step 5:** For the current threshold, generate the predicted classes for the test data:
  - Assign the positive class to samples with scores > the threshold.
  - Assign the negative class to samples with scores < the threshold.
- **Step 6:** Compare the predicted classes and compute the **TPR and FPR**.
- **Step 7:** Plot each pair of (TPR, FPR) points for each threshold.

Source

Multiple ROC Curves

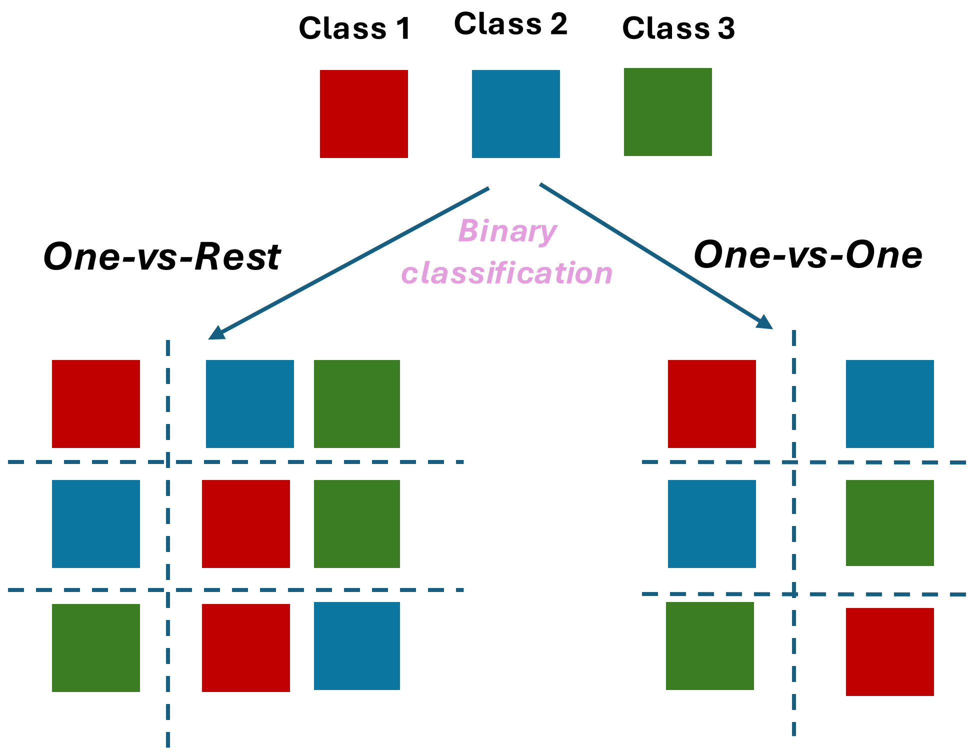
- Comparison of multiple classifiers is usually straight-forward especially when no curves cross each other. Curves close to the perfect ROC curve have a better performance level than the ones closes to the baseline.



This model has an AUC lower than 0.5, which means it performs worse than chance.

Multiclass ROC

- ROC curves are typically used in binary classification
- In the case of multiclass classification, a notion of TPR or FPR is obtained only after binarizing the output. How?
    - The One-vs-Rest scheme compares each class against all the others (N classifiers)
    - The One-vs-One scheme compares every unique pairwise combination of classes ($\frac{N(N-1)}{2}$ classifiers)



Multiclass ROC

- Macro-average (for OVR): To obtain the macro-average, the metric is calculated separately for each class and then averaged across all classes, thereby treating all classes equally beforehand.

$$Recall_{macro} = \frac{1}{C}\sum_{c}\frac{TP_C}{TP_C + FN_C}$$

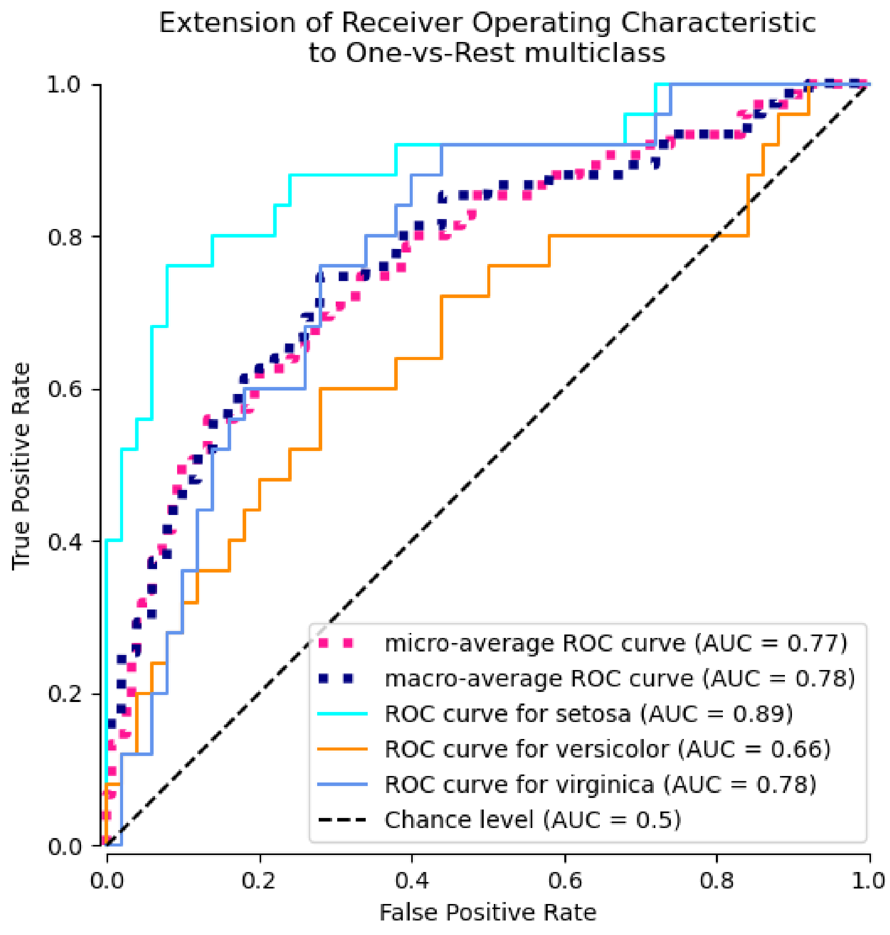
$$FPR_{macro} = \frac{1}{C}\sum_{c}\frac{FP_C}{FP_C + TN_C}$$

- Micro-averaging (for OVR): It combines the contributions from all classes and calculates the average metric as follows

$$Recall_{micro} = \frac{\sum_C TP_C 1}{\sum_C TP_C + FN_C}$$

$$FPR_{micro} = \frac{\sum_C FP_C 1}{\sum_C FP_C + TN_C}$$

- What about macro and micro averaging for OVO?



Iris setosa

Iris versicolor

Iris virginica

Iris flower data set

More on ROC curves

- ROC Curve shows the trade-off between **true positive rate (TPR)** and **false positive rate (FPR)**.
- In **unbalanced datasets**, the FPR can remain low even when the classifier performs poorly on the minority class.
- ROC Curve may appear misleadingly optimistic, as the model can show good performance by predicting the majority class correctly.

Precision and Recall

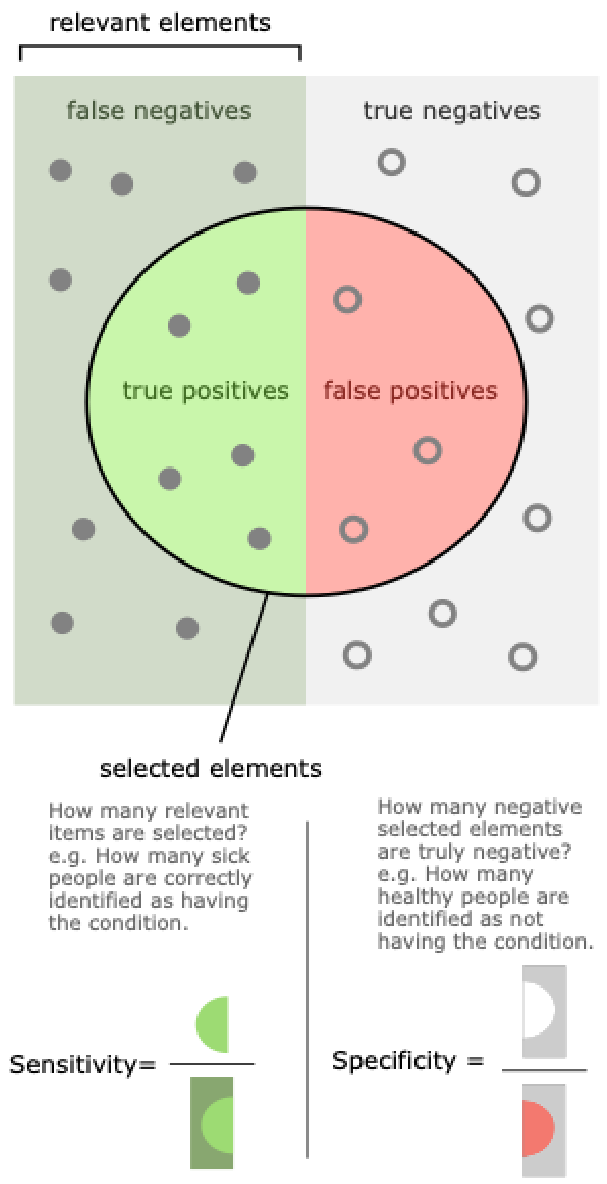
- Can be deduced from the confusion matrix for two classes.

$$Precision = \frac{TP}{TP+FP}$$

- Precision = Accuracy of positive predictions

$$Recall = \frac{TP}{TP+FN}$$

- Recall = Percentage of positive elements found (a.k.a. TPR)
- **Precision:** Focuses false positives
- **Recall:** Focuses false negatives


relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many relevant items are selected? e.g. How many sick people are correctly identified as having the condition.
How many negative selected elements are truly negative? e.g. How many healthy people are identified as not having the condition.
Sensitivity=
Specificity =

F-score

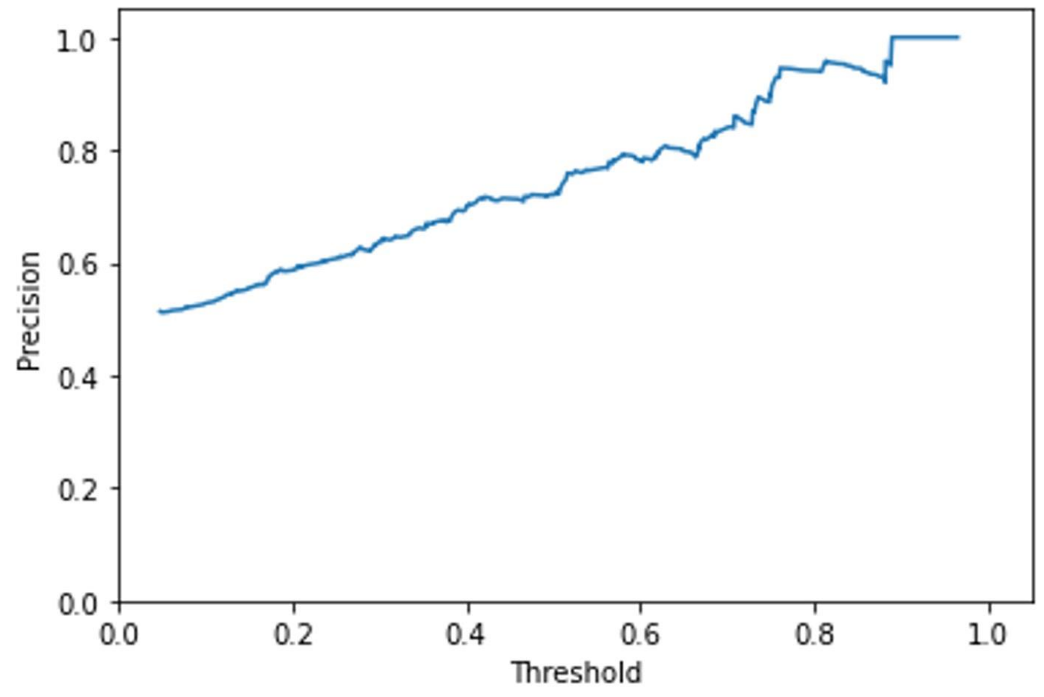
- For a fixed threshold, we can compute a single precision and recall value and combine these values into a single metric called F-score.
- The traditional or balanced F-score (F1 score) is the harmonic of precision and recall.

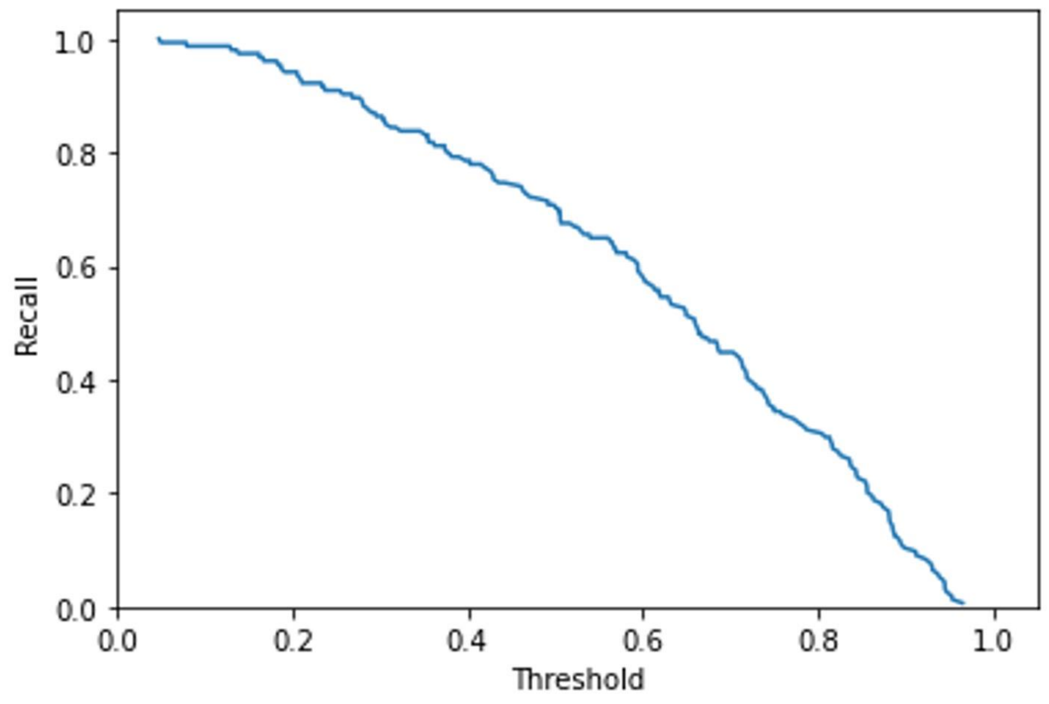
$$F_1 = \frac{2}{recall^{-1}+precision^{-1}} = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2TP}{2TP+FN+FP}$$

- It gives equal importance to precision and recall. In practice, different types of misclassifications incur different costs (Fβscore)

Precision and recall

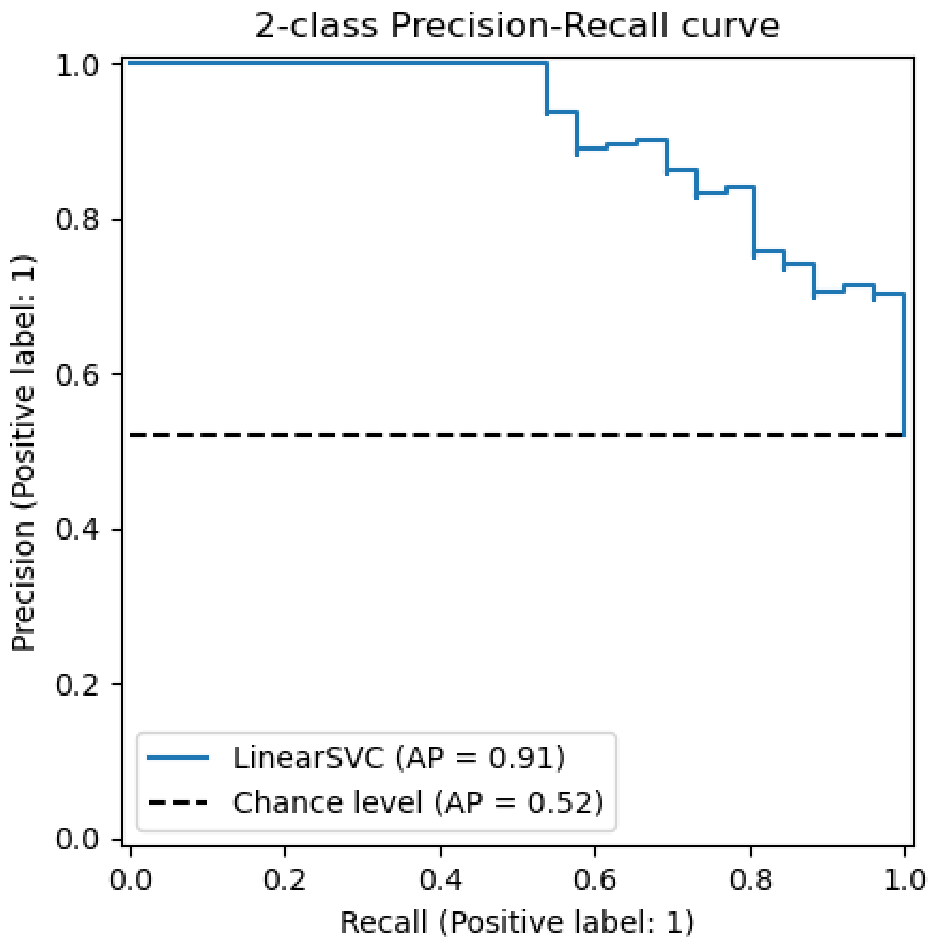
- For a trained binary classifier:
  - As you increase the threshold for classifying the input as positive, precision will typically go up, and recall will typically go down.





Introduction to Precision-Recall Curves (PR Curves)

- Precision-recall curve can be created similarly as ROC curve, except that:
  - Using different thresholds of the prediction score of the positive class, the precision and recall are calculated and plotted instead of the recall and false positive rate.
- PR Curves are sometimes more useful when dealing with **imbalanced datasets**, where the positive class is rare.
- Like ROC curves, we can summarize the information in a precision-recall curve with a single value. This summary metric is the **AUC-PR**. AUC-PR stands for **area under the (precision-recall) curve.**
- Generally, the higher the AUC-PR score, the better a classifier performs for the given task.
- One way to calculate AUC-PR is to find the **AP**, or *average precision*
- In a perfect classifier, AUC-PR =1.



Interpreting PR Curves

- **Shape of the Curve:**A higher area under the PR curve indicates better performance.
- A curve that stays near the top-right corner (high precision and recall) is ideal.
- **Key Insights:** A steep decline indicates the model struggles with trade-offs between precision and recall.

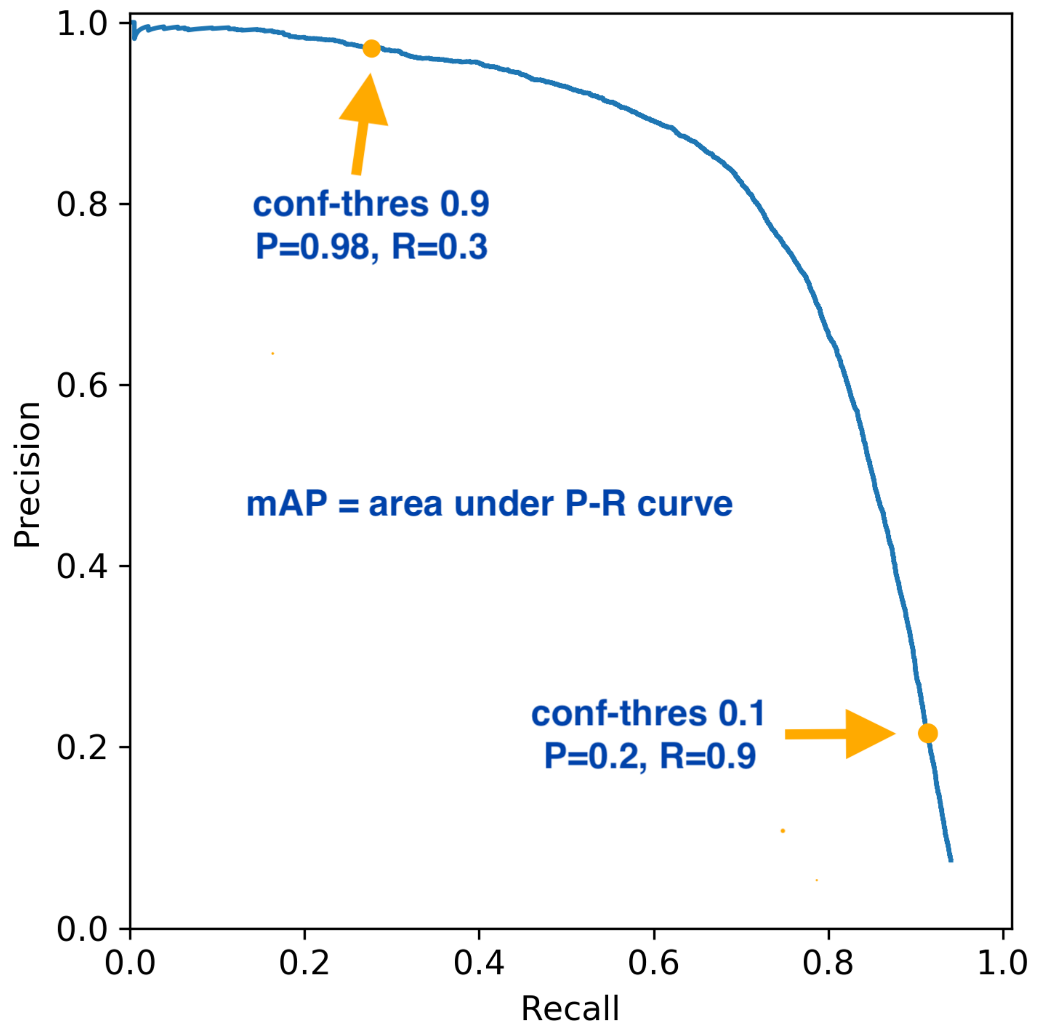


Figure source

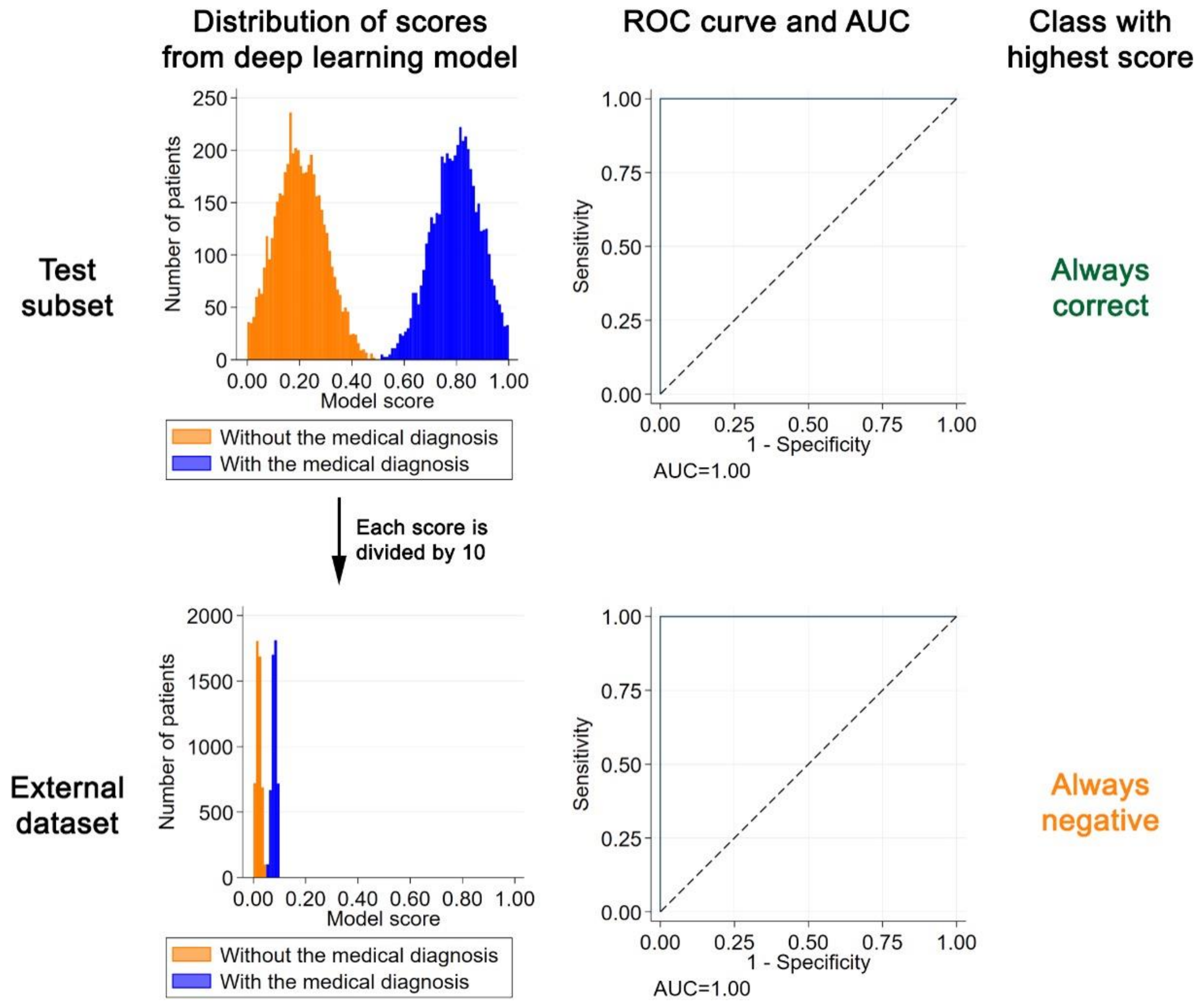
AP and AUROC

- Summarize the discriminatory value of the prediction scores.
- Only use the ranking of the prediction scores, not their absolute values.
- A concern when combined with external testing.
  - A distribution shift between the internal dataset and the external dataset can result in changes in the score distribution that resemble those caused by some strictly increasing transform.
  - *AP and AUROC may thus hide a type of model bias that is not uncommon.*

- ***Questions:***
  - Are AP and AUROC sensitive to class imbalance?
  - How is AUROC misleading in the case of class imbalance?

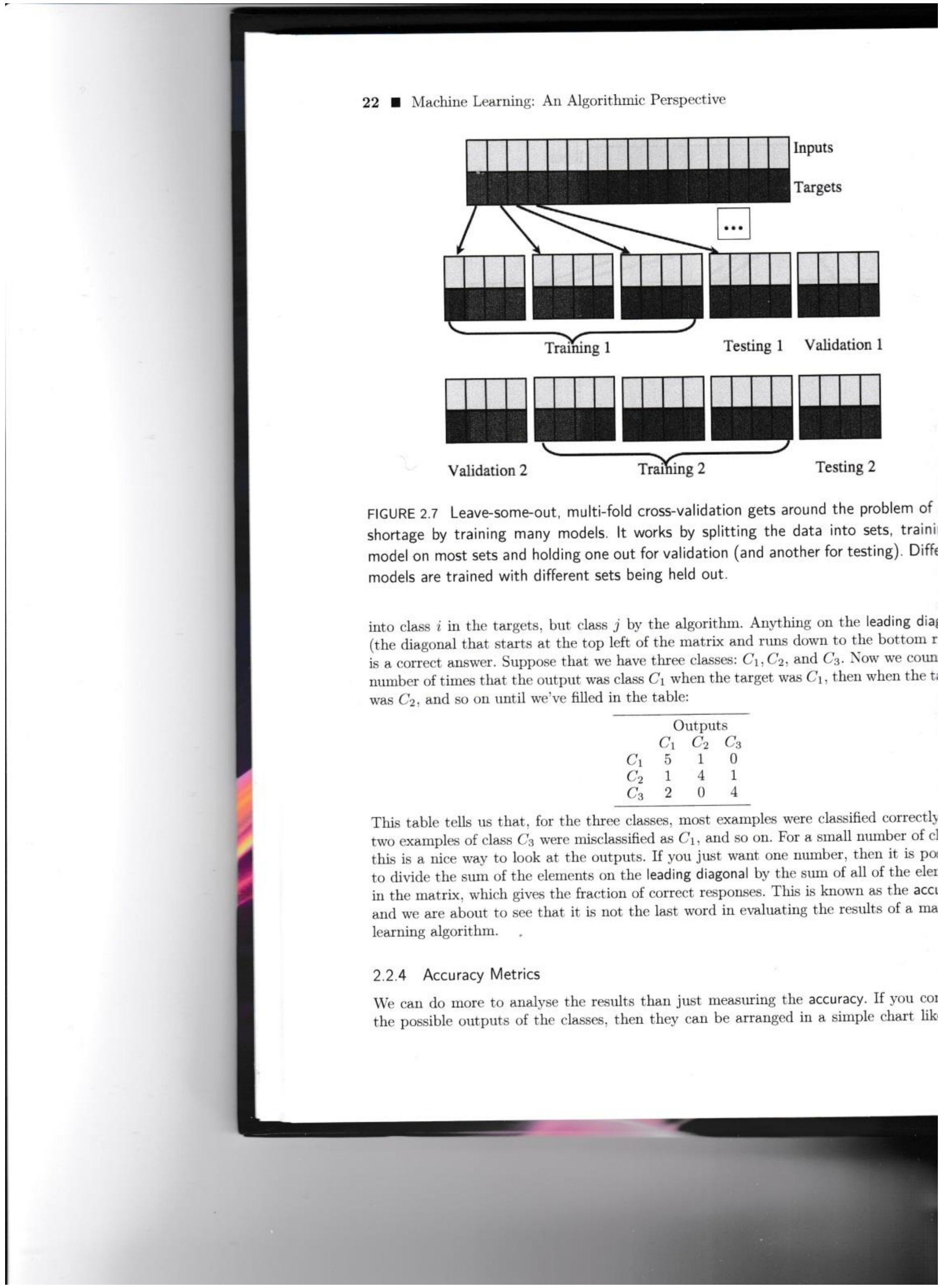
- What if the scores of a second external dataset were transformed by s1/10 compared to the test subset?
  - AUC will still be 1.00, but the class with the highest score will nearly always be positive (specifically if s > 0.510 ≈ 0.00098).

One could use any other metric (like accuracy) along with AP or AUROC.



Marsland Ch. 2

- First edition 2009
- Second edition 2015:
    - Removed mistakes
    - Added chapter 2
        - With mistakes
- Please read sections 2.2.4 and 2.2.5 and **check for statements that look incorrect or misleading**. For each one you find, explain why it's wrong and propose a corrected version (with a short justification or example).



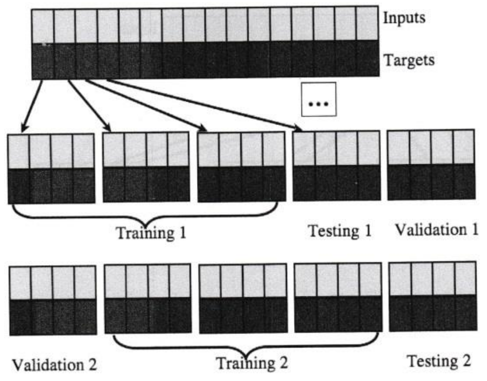


FIGURE 2.7 Leave-some-out, multi-fold cross-validation gets around the problem of shortage by training many models. It works by splitting the data into sets, traini model on most sets and holding one out for validation (and another for testing). Diffe models are trained with different sets being held out.

into class i in the targets, but class j by the algorithm. Anything on the leading dia (the diagonal that starts at the top left of the matrix and runs down to the bottom r is a correct answer. Suppose that we have three classes: C_1, C_2, and C_3. Now we coun number of times that the output was class C_1 when the target was C_1, then when the t was C_2, and so on until we've filled in the table:

	Outputs		
	C_1	C_2	C_3
C_1	5	1	0
C_2	1	4	1
C_3	2	0	4

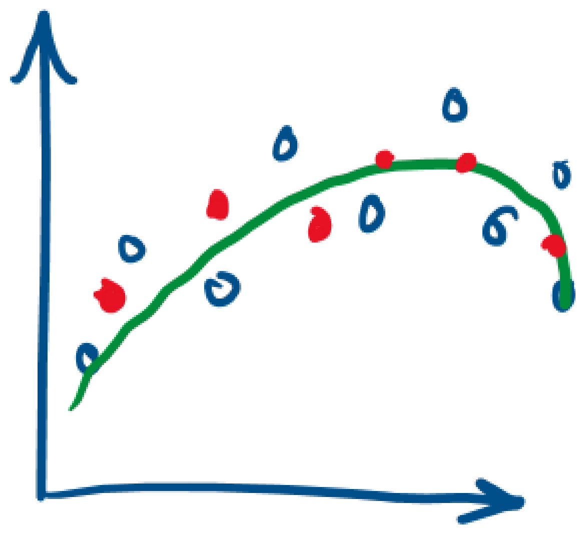
This table tells us that, for the three classes, most examples were classified correctly two examples of class C_3 were misclassified as C_1, and so on. For a small number of cl this is a nice way to look at the outputs. If you just want one number, then it is po to divide the sum of the elements on the leading diagonal by the sum of all of the ele in the matrix, which gives the fraction of correct responses. This is known as the acc and we are about to see that it is not the last word in evaluating the results of a ma learning algorithm.

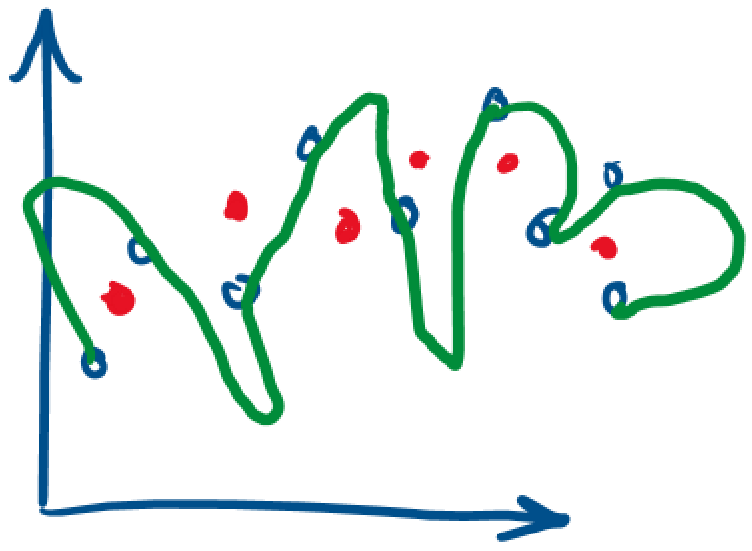
2.2.4 Accuracy Metrics

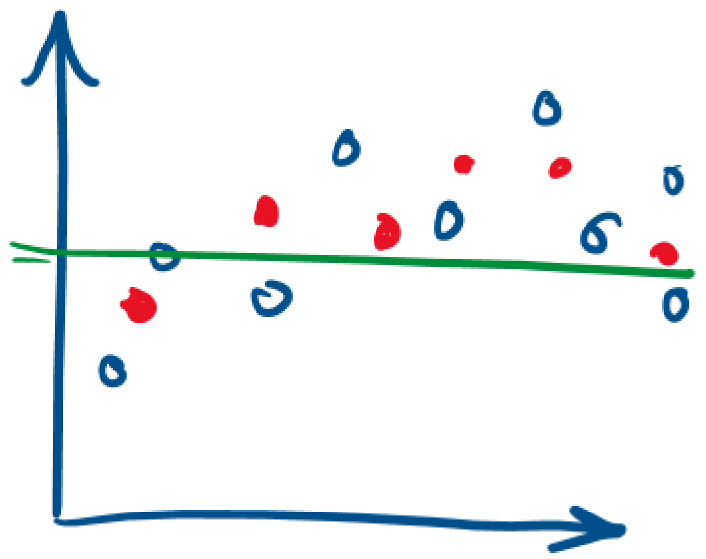
We can do more to analyse the results than just measuring the accuracy. If you co the possible outputs of the classes, then they can be arranged in a simple chart lik

Overfitting vs underfitting (intuition)

- **Overfitting** – fitting the training data too precisely - usually leads to poor results on new data
- **Underfitting** – the model does not fit training data well


Good fit

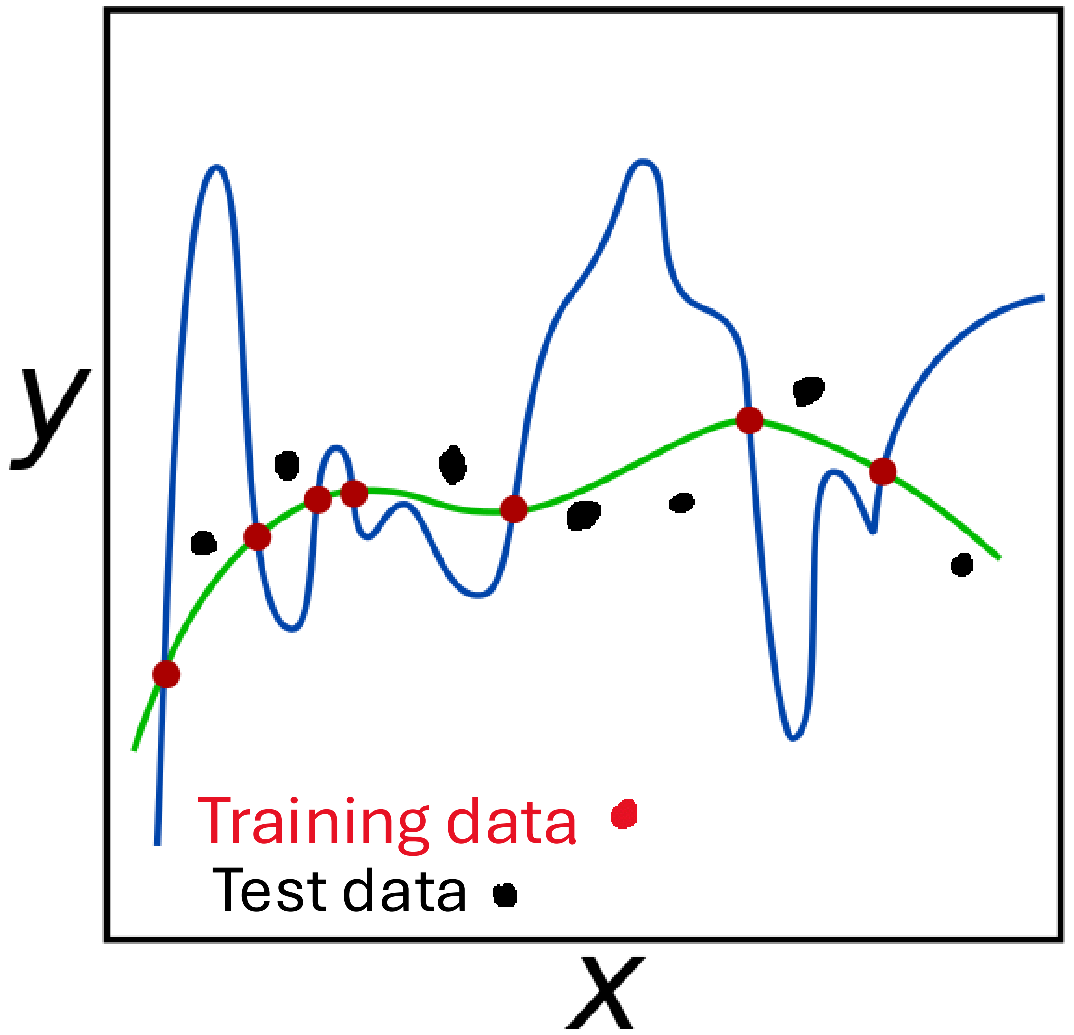

overfitting


underfitting

Overfitting

- Goal in supervised ML:
  - Construct models that fit the training set
- Sometimes, we succeed too well:
  - Model fits training data very well
  - Does not generalize to other data

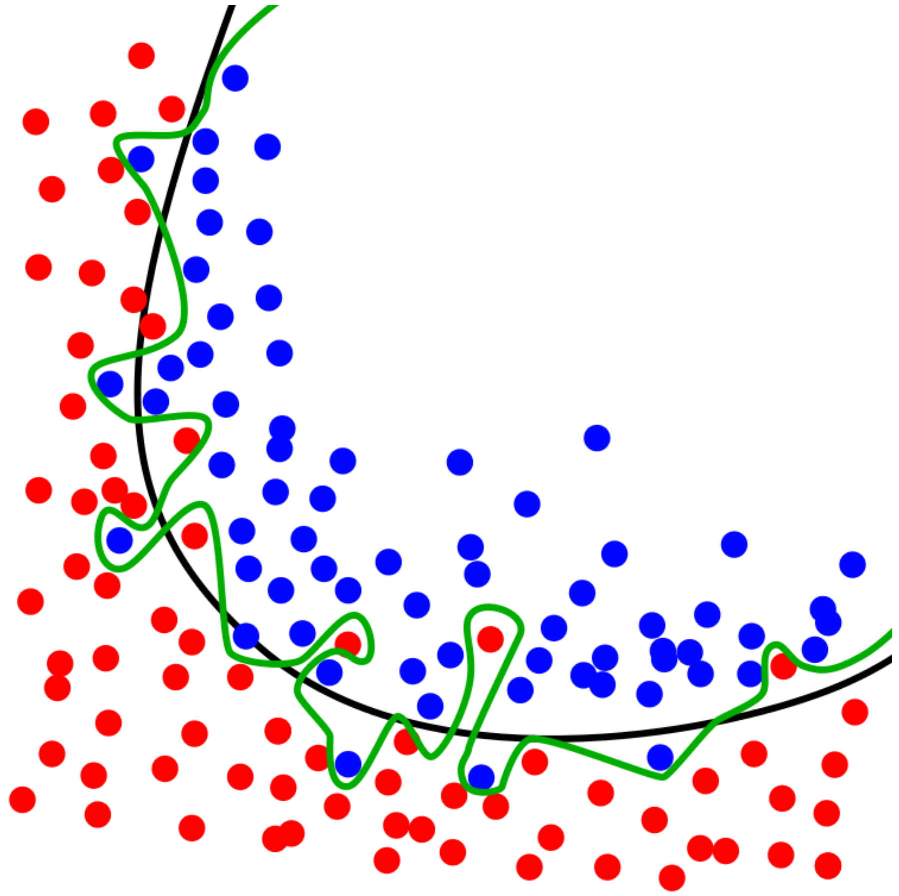
Overfitting happens when a model learns noise and details in the training data, performing well on it but poorly on unseen data.



Source: Wikipedia

Overfitting

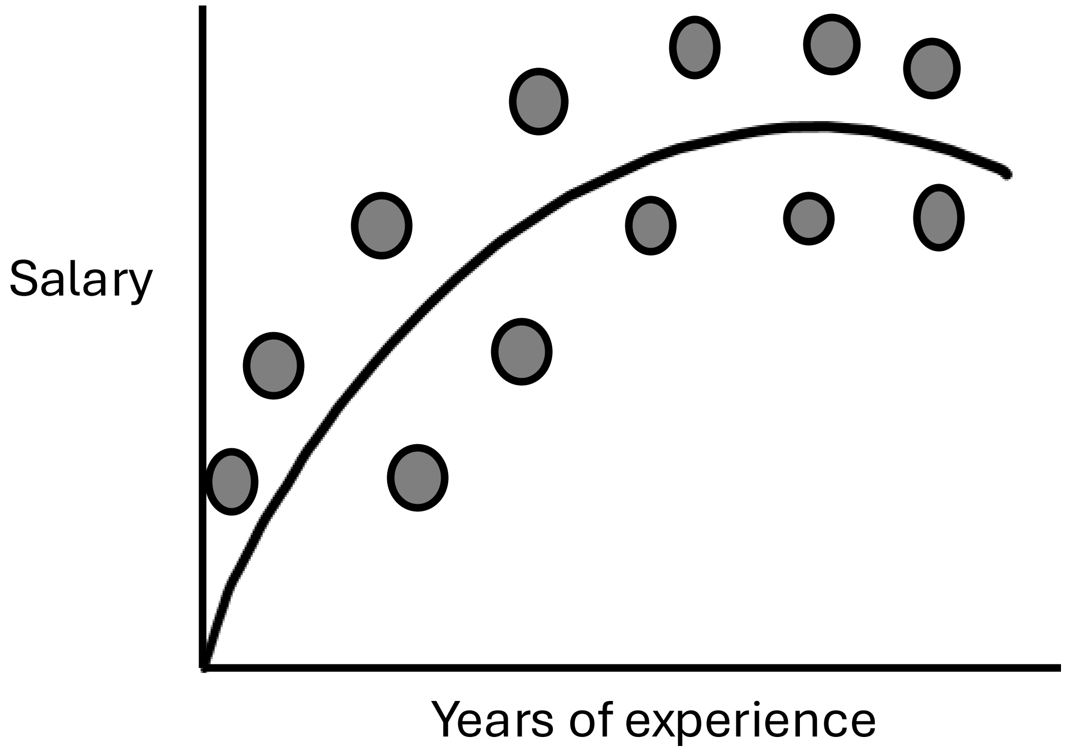
- Classification



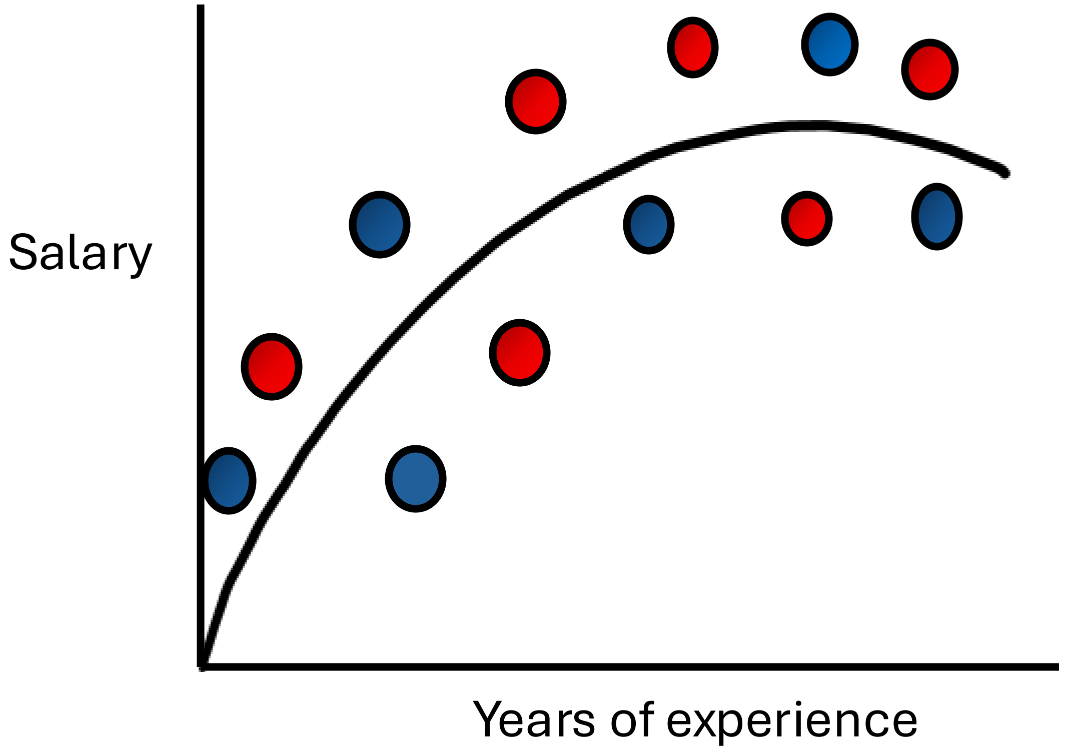
Source: Wikipedia

Bias-variance tradeoff (intuition)

Salary Prediction (Years of Experience) example



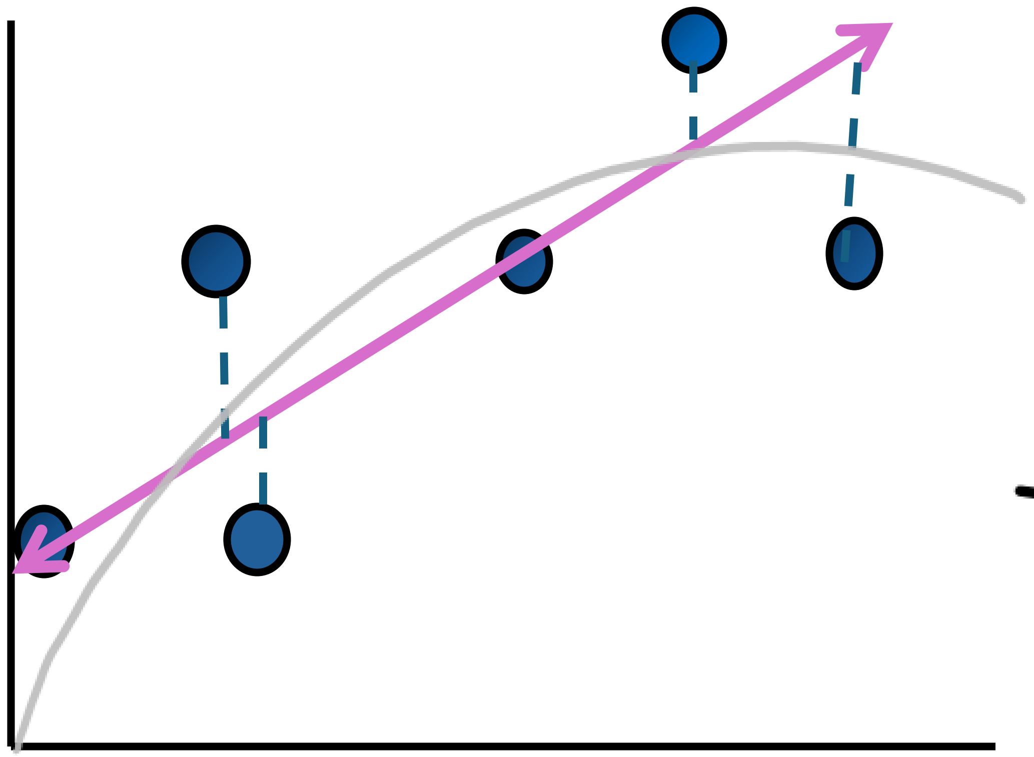
Training data and test data



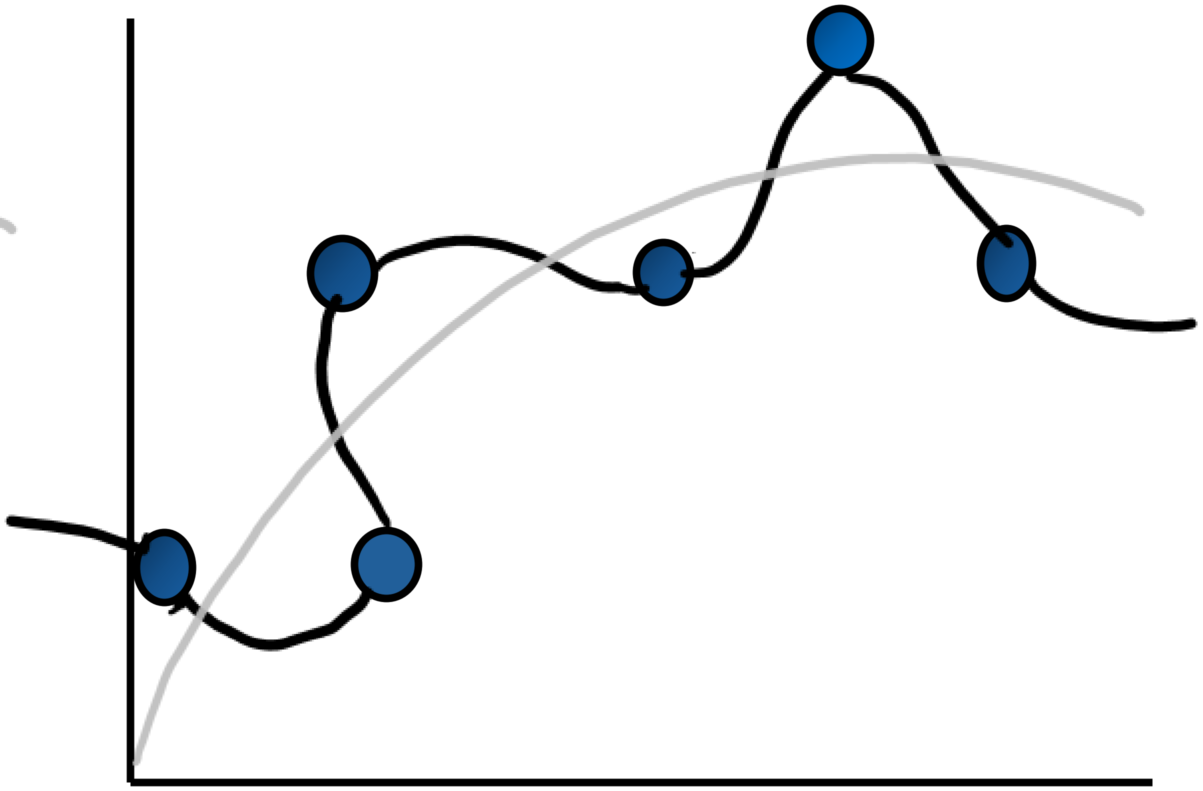
Simple models Vs complicated models:

Straight line (linear regression) vs. Wavy line (on training data)

Not flexible to capture the true relationship (high bias)



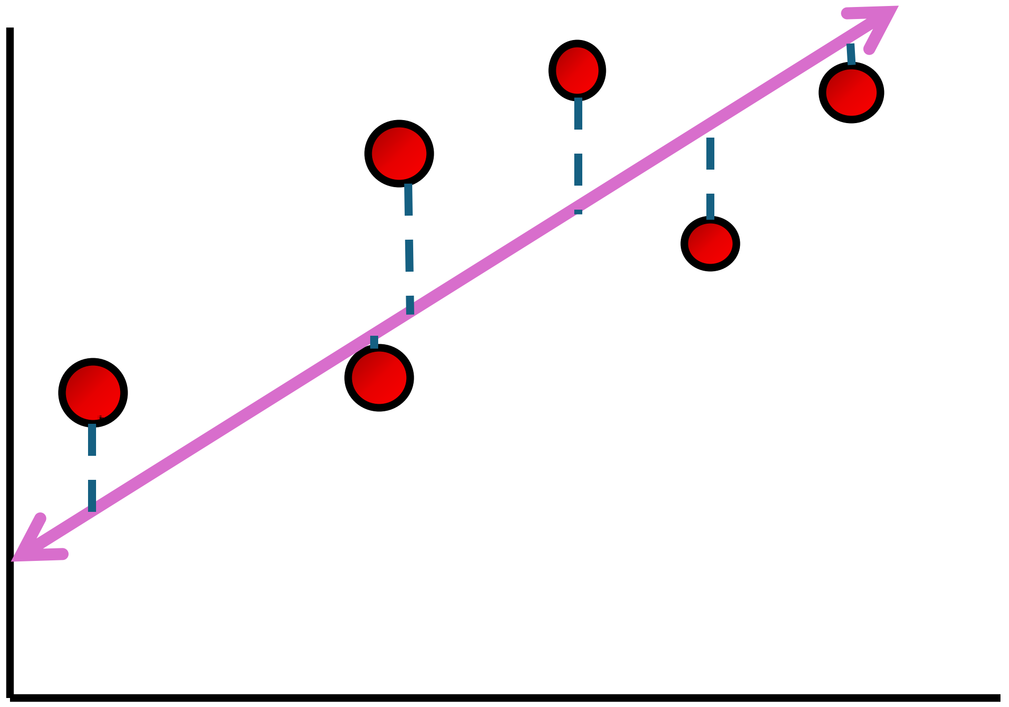
Very flexible to capture the true relationship in training data (low bias)



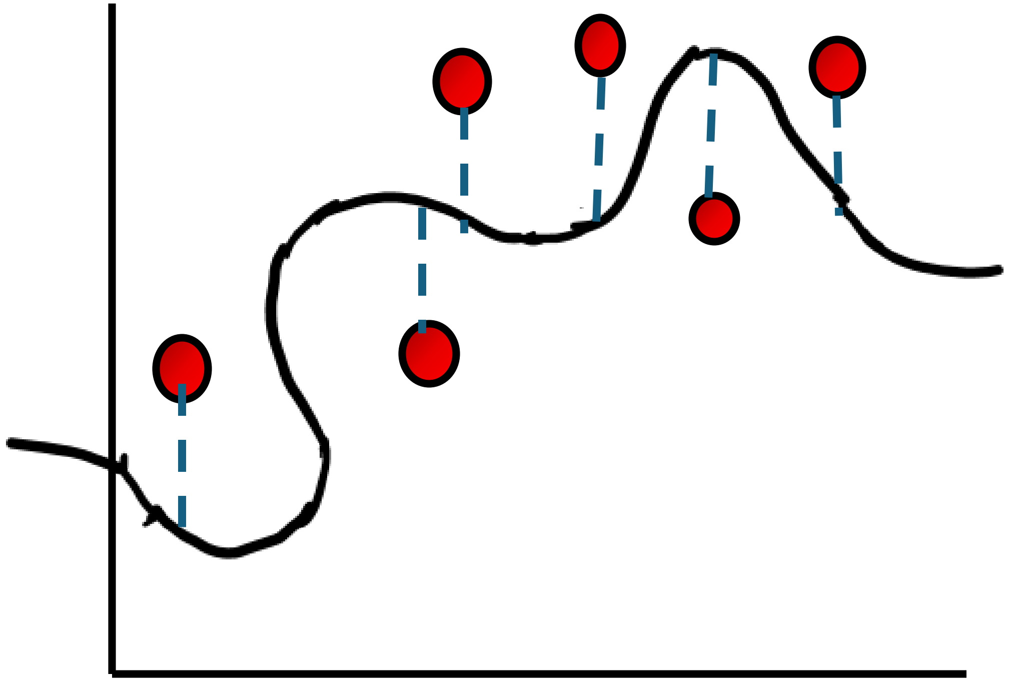
Simple models Vs complicated models:

Straight line (linear regression) vs. Wavy line (on test data)

The sums of squares are very similar for different data sets. (a relatively low variance)

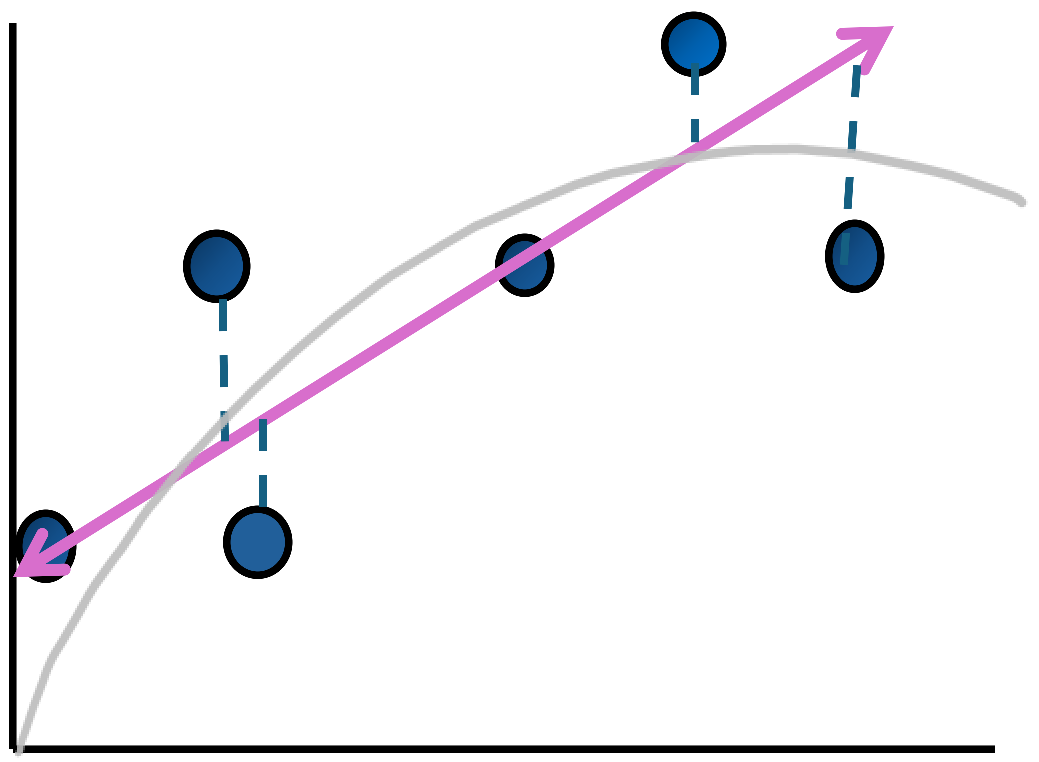


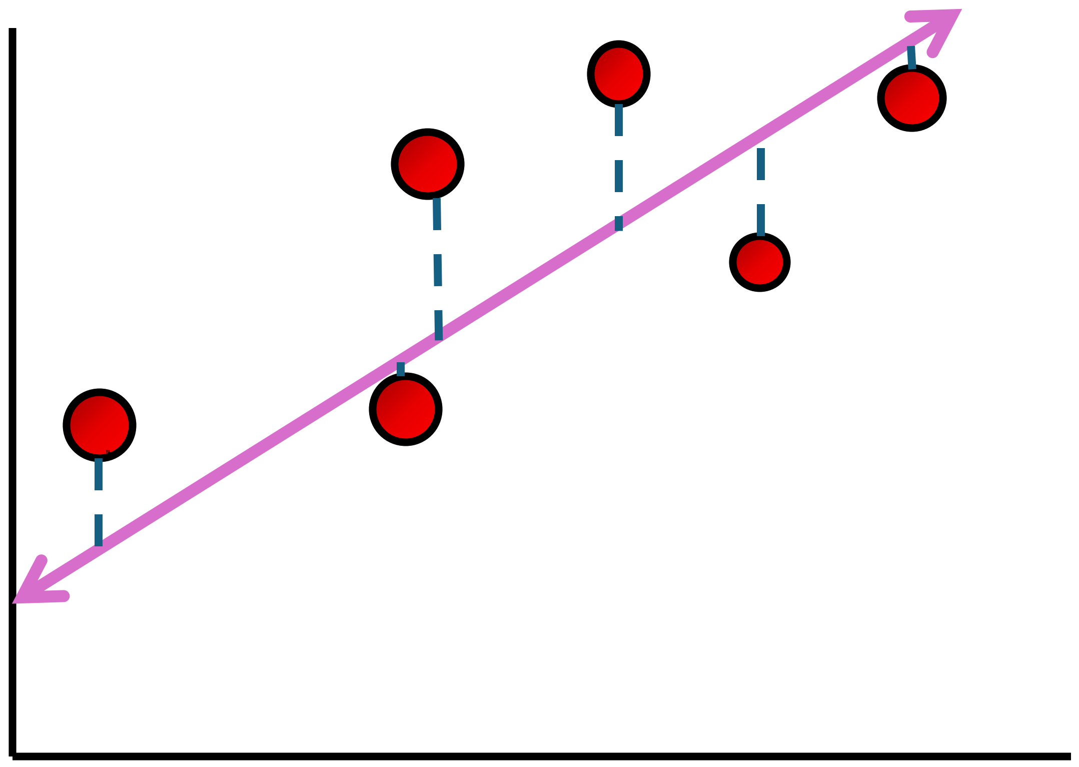
It results in vastly different sums of squares for different data sets (High variance)



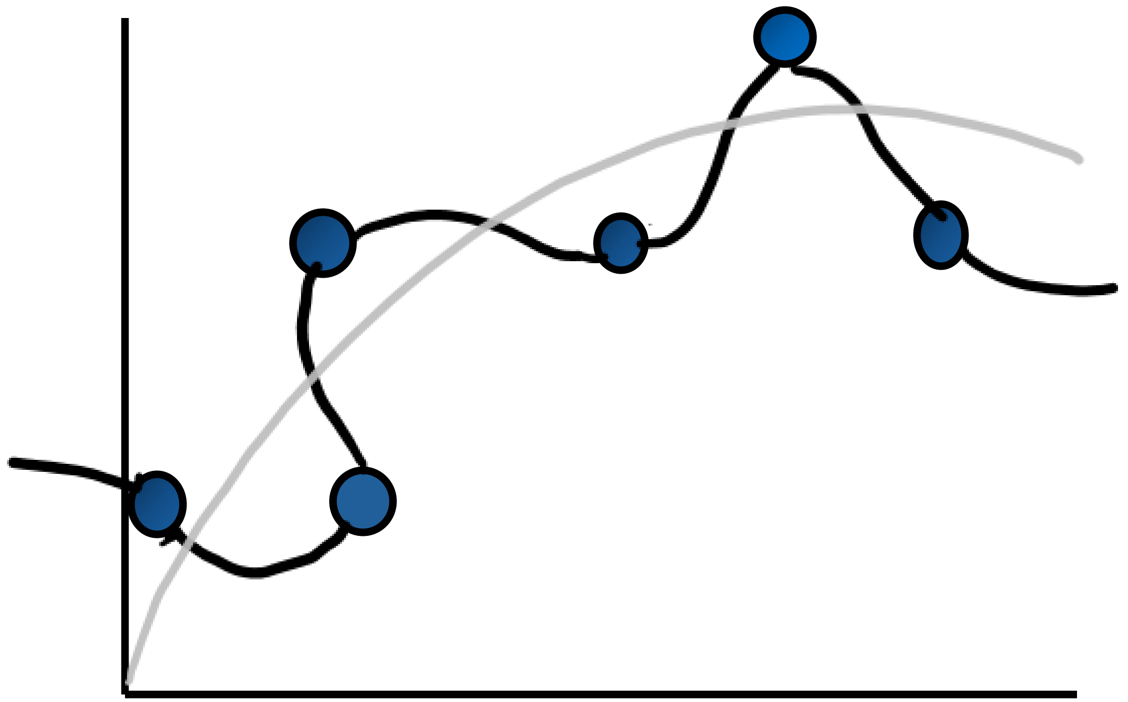
The difference in fits between data sets is called variance

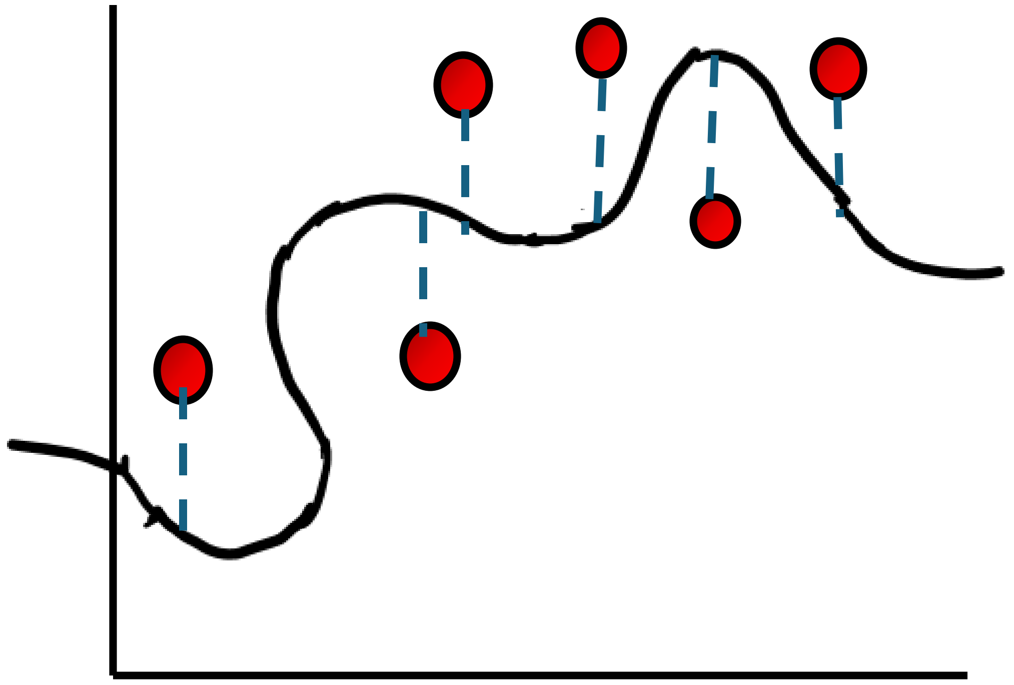
High bias



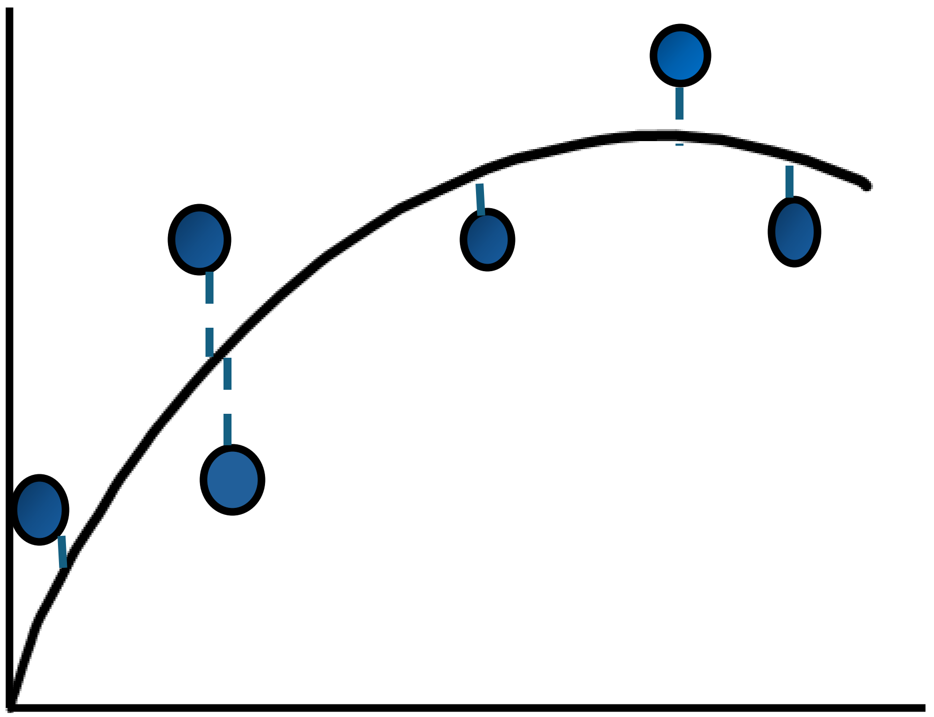


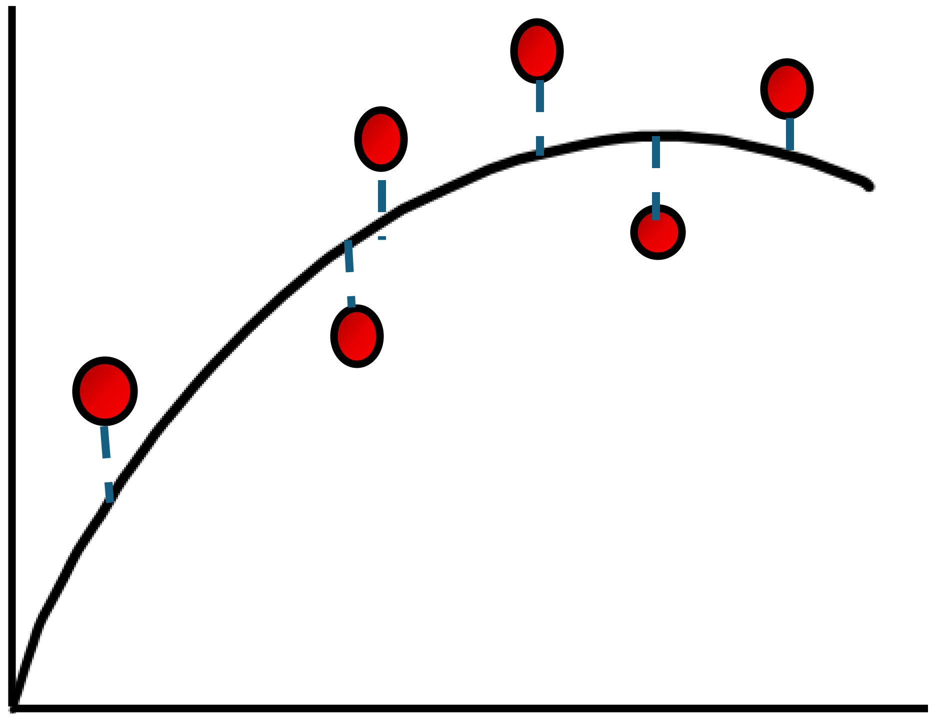
High variance



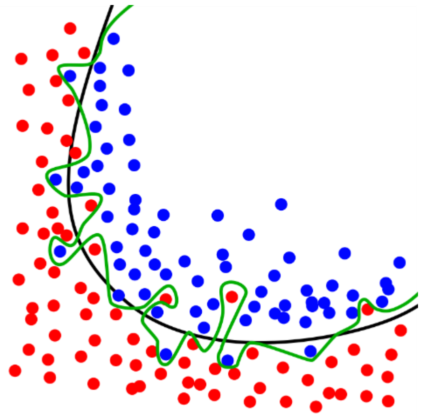


Balanced model





Bias-variance tradeoff

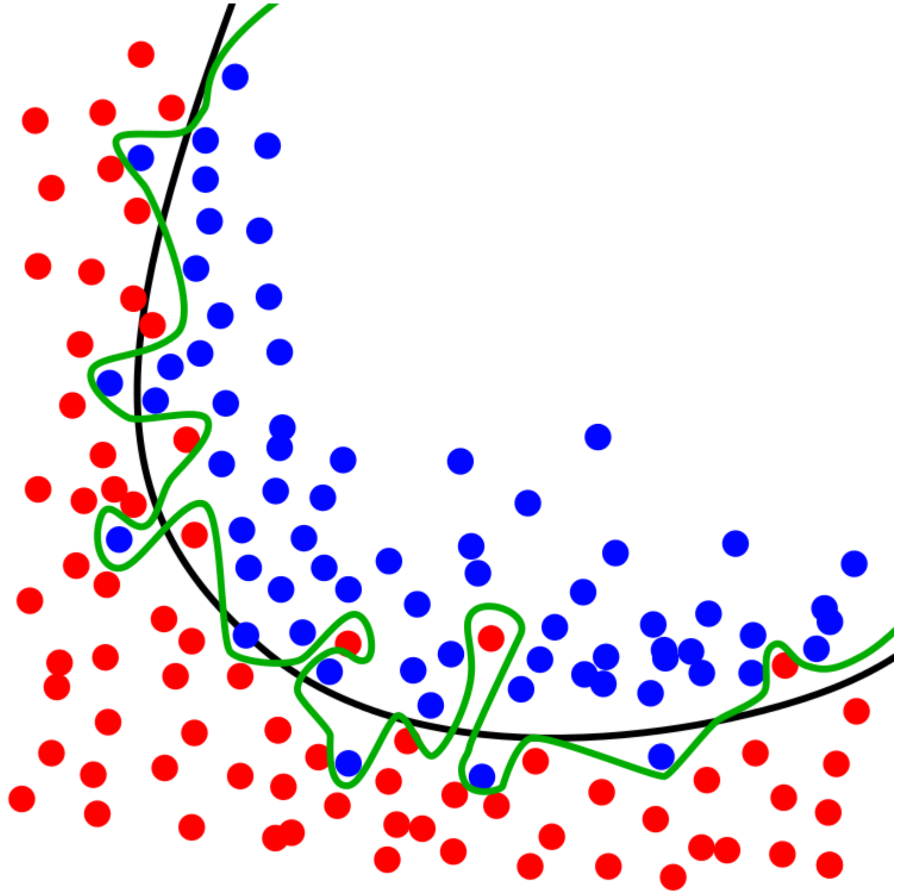


- In general, we will not succeed in finding g identical to f
- The expectation of the error can be split into
  - bias
  - variance

- Bias
  - the classifier systematically misses the target,
  - e.g., searching for linear classifiers for a non-linear problem
  - Under-fitting
- Variance
  - Picking up some of the noise, being too sensible to small variations in the input data
  - cf. the green decision boundary
  - Over-fitting

Bias-variance tradeoff

- Underlying idea
  - (from statistics, can be formalized)
- The training data $\mathrm{D} = (\boldsymbol{X}, \boldsymbol{t})$ can be described by an expression
- $t = f(\boldsymbol{x}) + \varepsilon$, where
  - ε is irreducible (noise), has zero mean and variance σ^2
- The goal of the ML training is to find (an approximation) g to f, where $(f(x) - g(x))^2$ is minimal for x both from *D*, and from outside of D.

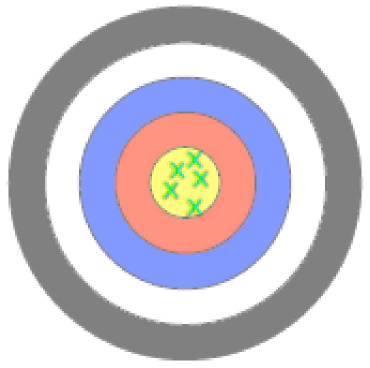


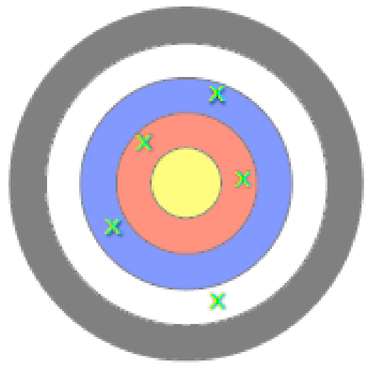
$E[(t - g(x))^2] = \text{Bias}^2 + \text{Variance} + \sigma^2$

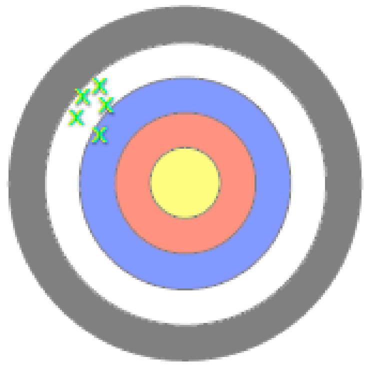
Homework: Check Marsland Ch. 1, page number 36, equation 2.29

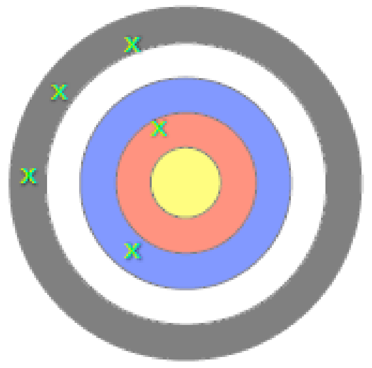
Source: Wikipedia

Metaphor of bias vs variance


Low bias, low variance


Low bias, high variance

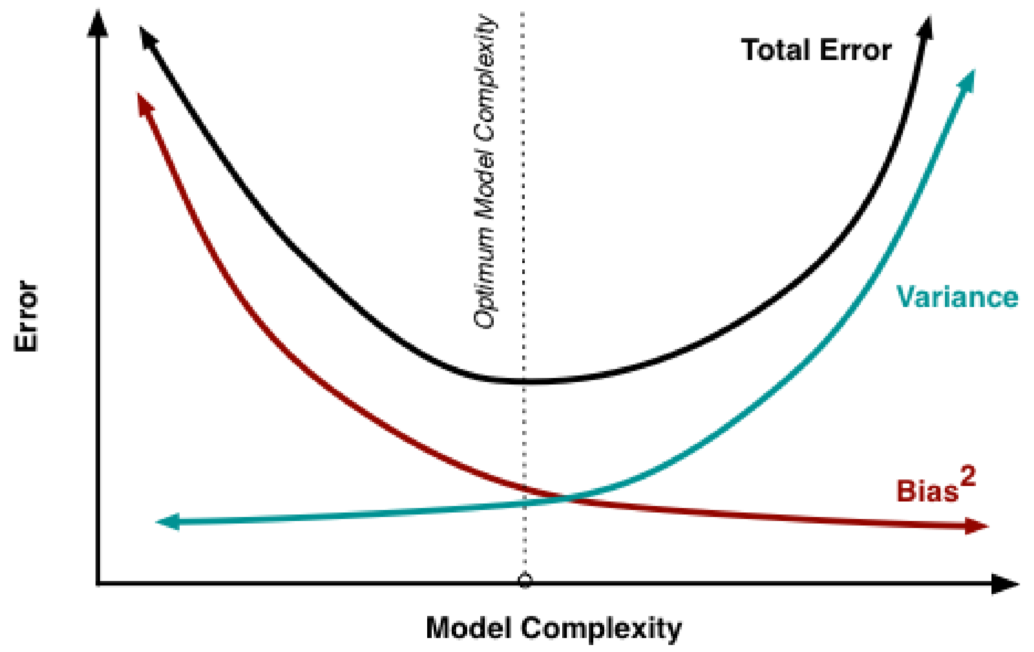

High bias, low variance


High bias, high variance

source

The tradeoff

- Traditional wisdom:
  - Lower bias gives higher variance,
  - and the other way around
- There is a sweet spot in the middle



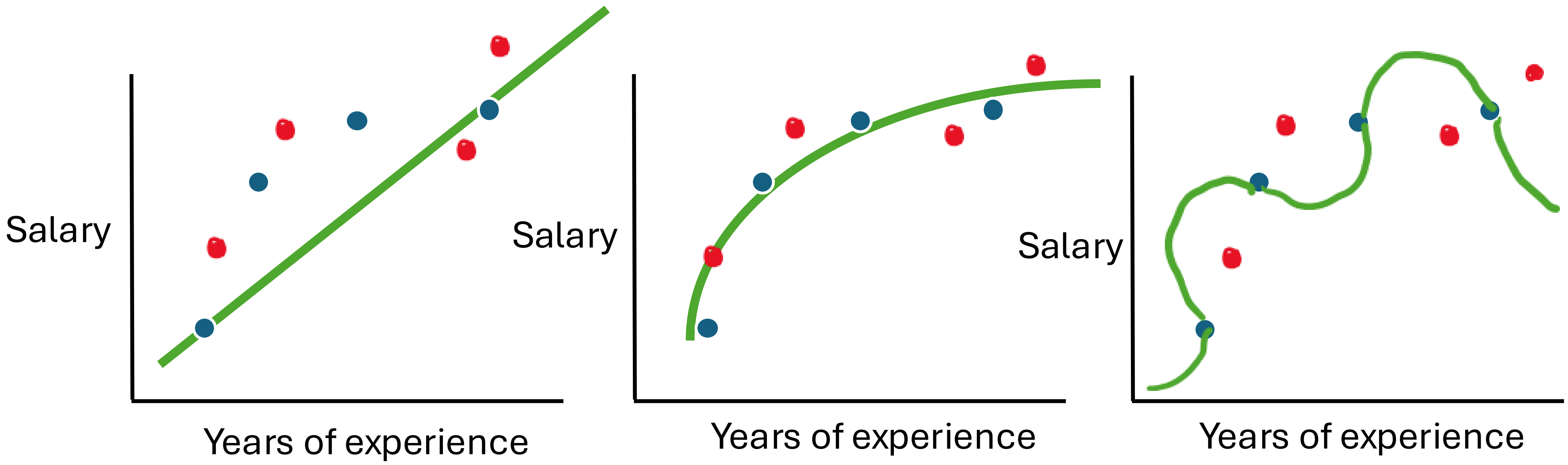
http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-variance tradeoff

- High bias
  - Leads to underfitting
  - Remedies:
    - Train more
    - Increase model complexity
    - Try different model techniques

- High variance
  - Leads to overfitting
  - Remedies:
    - Introduce more data
    - **Use regularization**
    - Try different model techniques

Model Selection (Overfitting vs Underfitting)



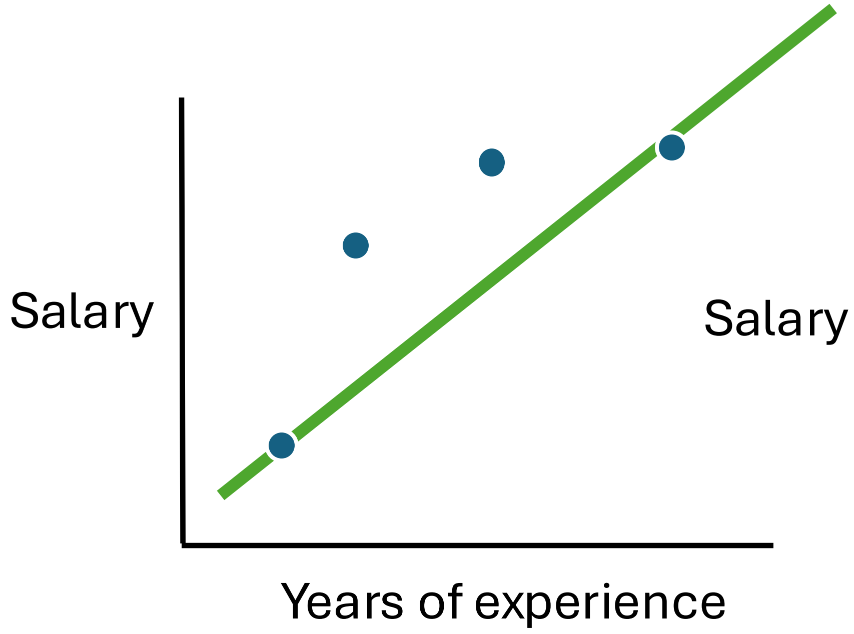
Underfitting
High bias
Too simple

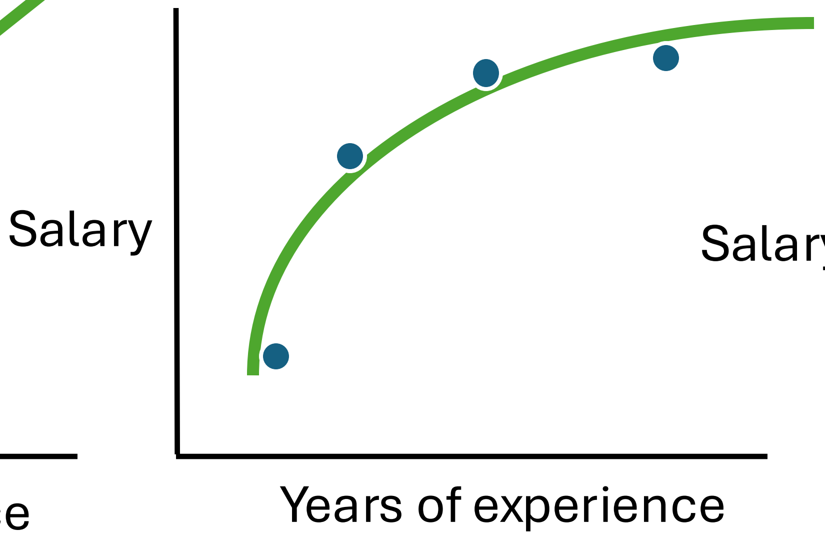
Good fit
Balance
Right model

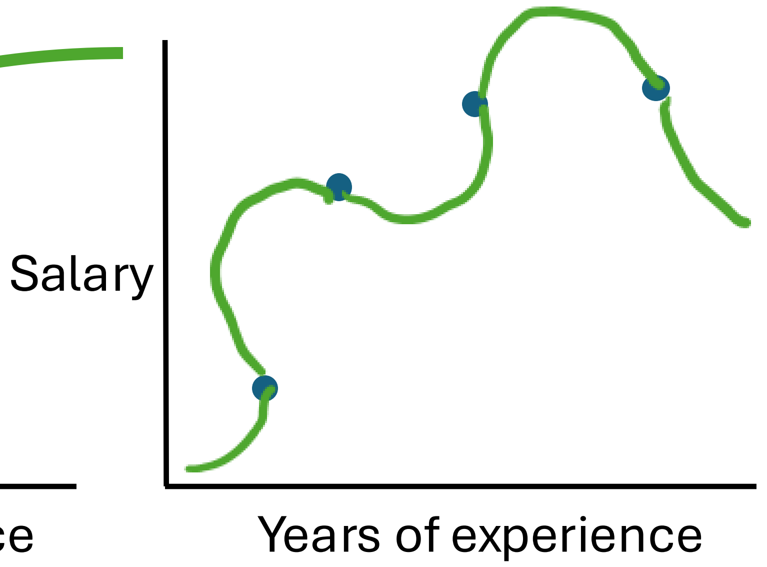
Overfitting
High variance
Too complex

Source: Andrew Ng

Regression (Salary predictions)







$w_1 x + b$

$w_1 x + w_2 x^2 + b$

$w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

Underfit

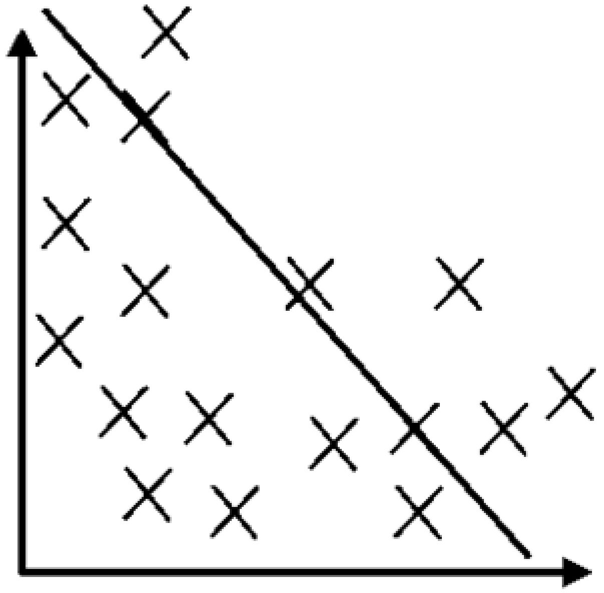
Just right

Overfit

- What if the feature dimension is too high?

Source: Andrew Ng

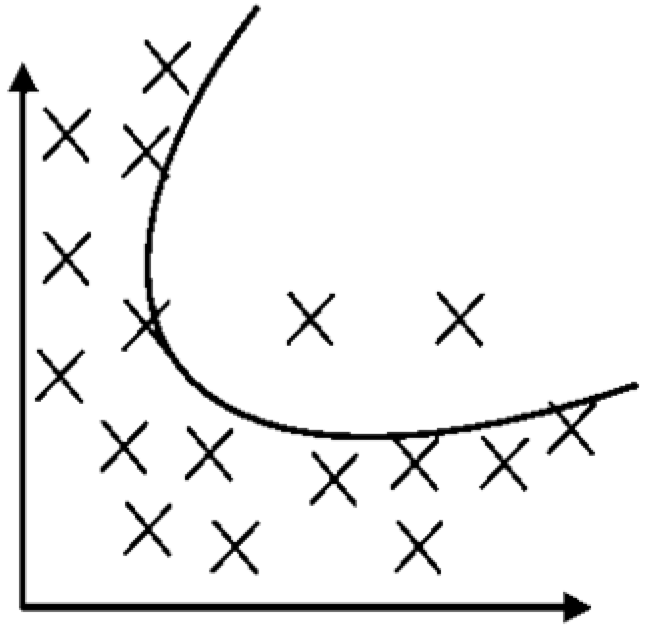
Classification



$$z = w_1 x_1 + w_2 x_2 + b$$

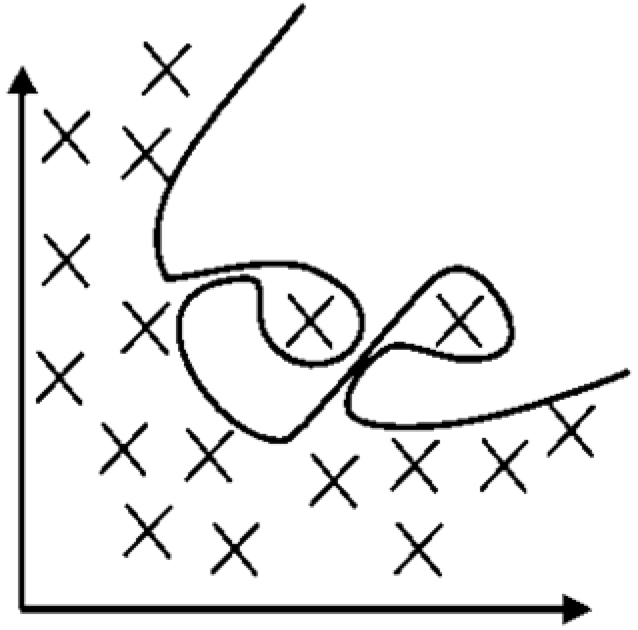
$$f_{\vec{w},b}(\vec{x}) = g(z)$$

Underfit



$$Z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + b$$

Just right



$$Z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^1 + w_5 x_1^2 x_2^3 + w_6 x_1^2 x_4^2 + \cdots + b$$

Overfit

Further reading: Andrew Ng

Remedy: Regularization

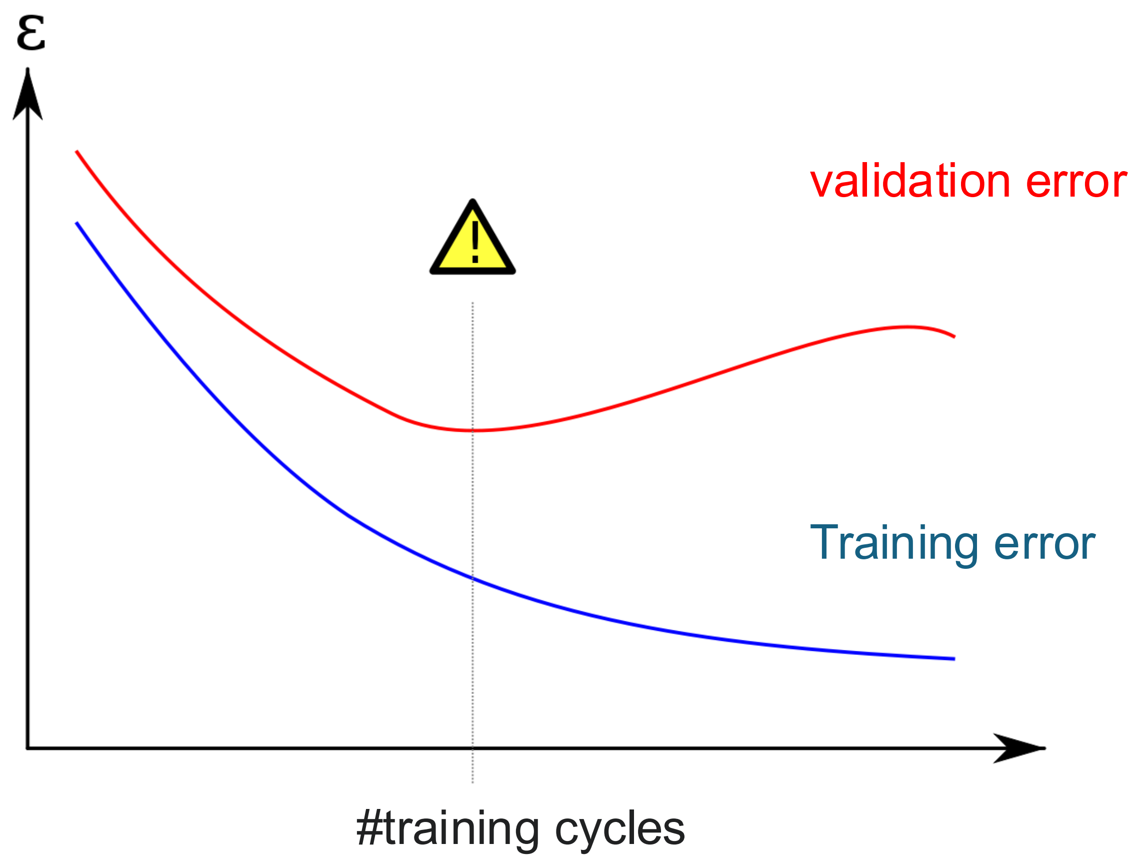
Explicit regularization

- Explicitly adding a term to the optimization problem, reducing the size of parameters
  - Ridge regularization
  - Lasso regularization
  - Lasso regularization

- **Implicit regularization**
  - Early stopping
  - Add more data
  - Feature selection
  - Dropout Regularization
  - using a robust loss function
  - discarding outliers.
  - Ensemble methods
    - Bagging (random forests)
    - Boosting (AdaBoost, gradient boosted trees).
    - Voting

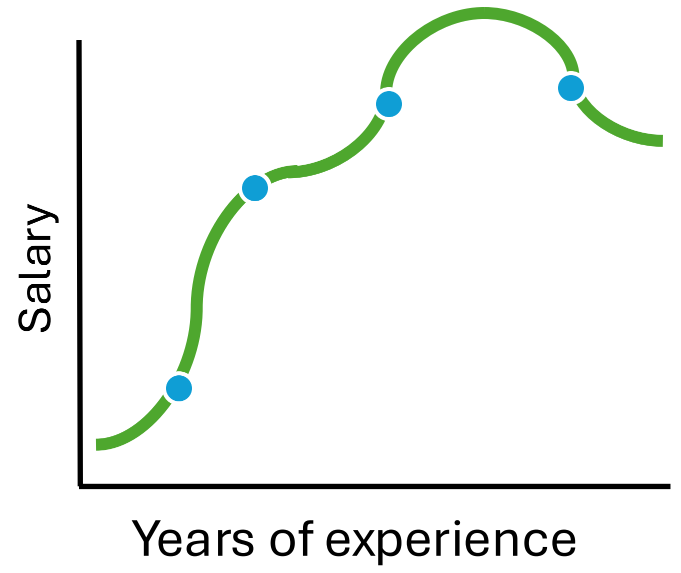
Remedies (Early Stopping)

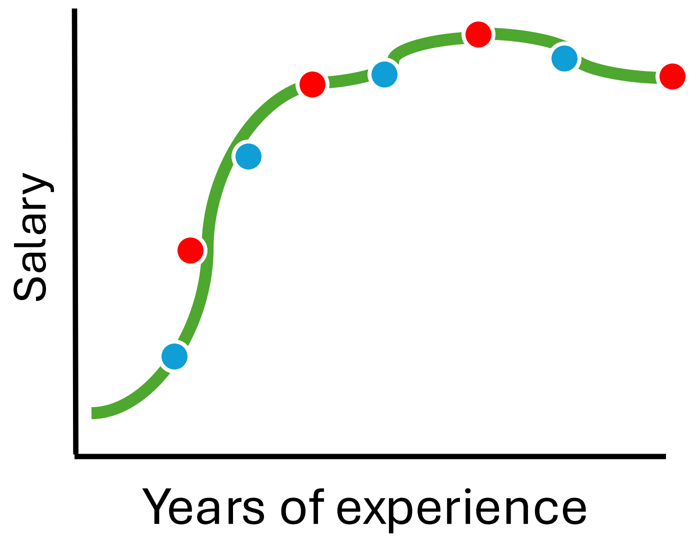
Implicit regularization



Source: Wikipedia

Does really adding more training data help?





More data is likely to help when your model has high variance
What about when your model has high bias?

Feature selctions

n features

Years of experience	Age	Educaltional level	Position	Skill level	Company size	. . .	Diversity factor	Geographical location	Salary
x_1	x_2	x_3	x_4	x_5	x_6		x_7	x_{50}	y

all features + insufficient data $\longrightarrow$ overfitting

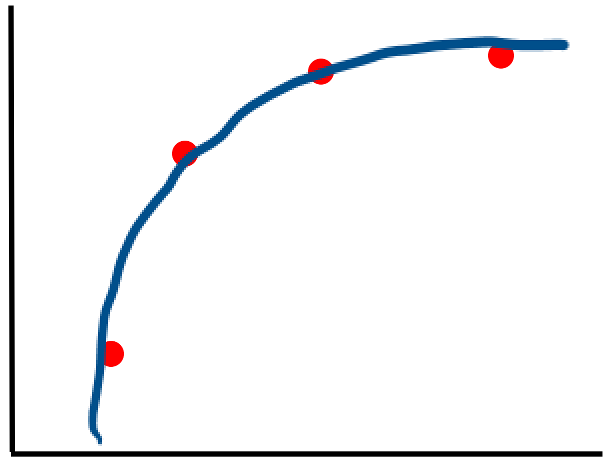
selected features $\left\{ \begin{array}{l} \text{years of experience} \\ \text{Educational level} \\ \text{country} \end{array} \right\}$ enough

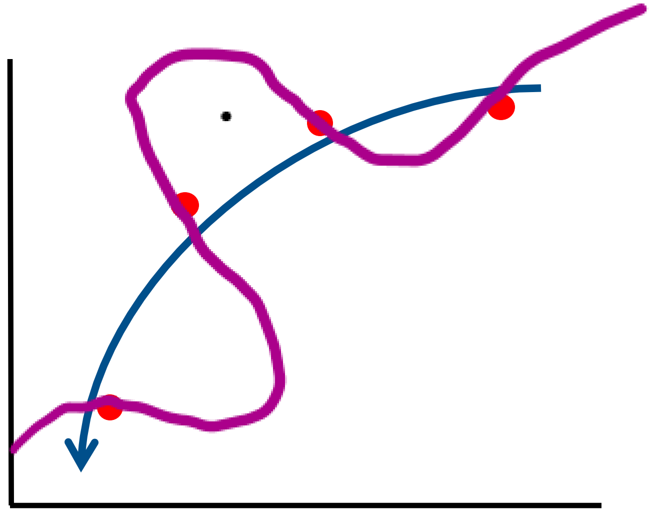
x_1, x_3, x_{50}

Further reading: Andrew Ng

Regularization: Intuition

Reduce the Size of Parameters w_j





$$w_1 x + w_2 x^2 + \underset{\approx 0}{\cancel{w_3}} x^3 + \underset{\approx 0}{\cancel{w_4}} x^4 + b$$

make w_3, w_4 really small (≈ 0)

$$\min_{\vec{w},b} \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \cancel{1000}\, \underset{0.002}{\underbrace{w_3^2}} + \cancel{1000}\, \underset{0.004}{\underbrace{w_4^2}}$$

Further reading: Andrew Ng

Regularization: Intuition

n features

Years of experience	Age	Educaltional level	Position	Skill level	Company size	. . .	Diversity factor	Geographical location	Salary
x_1	x_2	x_3						x_{50}	y

Small values $w_1, w_2, \dots, w_n, b$ Less prone to overfit

$w_3 \approx 0$
$w_4 = 0$

$w_1, w_2, \dots, w_{50}, b$

$$j(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)} - y^{(i)})^2 + \overbrace{\frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2}^{\text{regularization term}}$$

$\lambda > 0$

regularization parameter

Further reading: Andrew Ng

Regularization

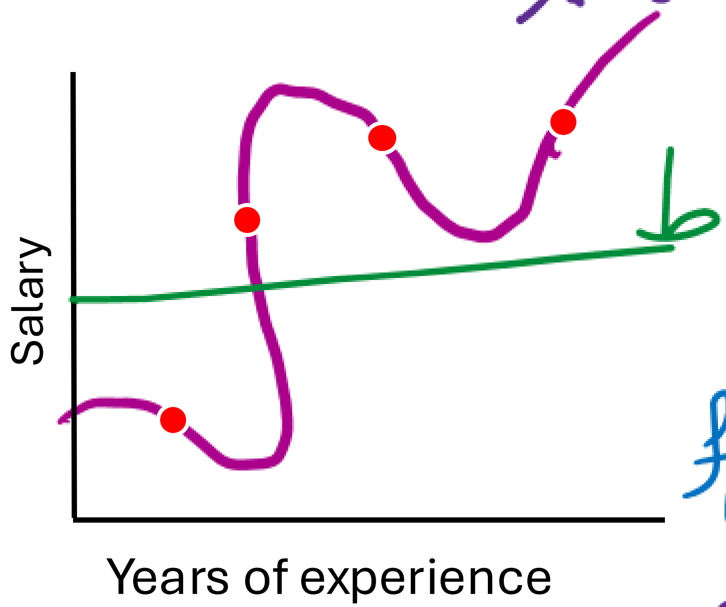
$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b}\left[\underbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2}_{\text{MSE}} + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}_{\text{regularization term}}\right]$$

fit data

Keep wj Small

λ balances both goals

$\lambda = 0$



$\lambda = 10^{20}$

$$f_{\vec{w},b}(\vec{x}) = \underset{\approx 0}{\cancel{w_1}}x + \underset{\approx 0}{\cancel{w_2}}x^2 + \underset{\approx 0}{\cancel{w_3}}x^3 + \underset{\approx 0}{\cancel{w_4}}x^4 + b$$

$f(x) = b$

Further reading: Andrew Ng

Approaches to regularization

Ridge regularization

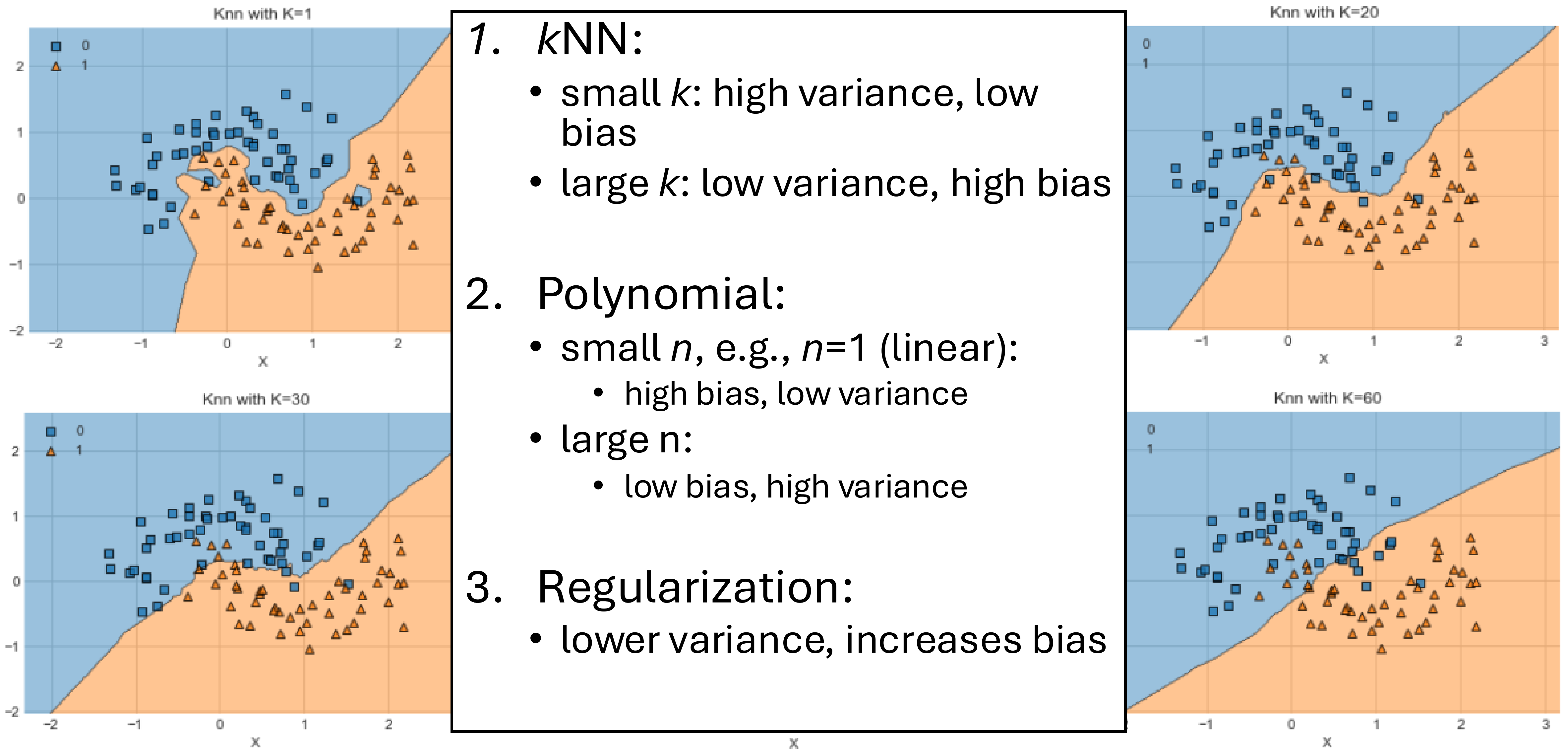
- The most common uses L2-distance
- Penalizes sum of squared coefficients
- Shrinks coefficients but keeps all of them
- Useful when you don't want to eliminate features
- $R(\boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{w} = \sum_{i=0}^{m} {w_i}^2$

Lasso regularization

- Uses L1
- Penalizes sum of absolute values of coefficients
- Retain only the most relevant variables
- Useful for feature selection
- $R(\boldsymbol{w}) = \sum_{i=0}^{m} |w_i|$

Elastic nets:

uses both L2 and L1
Balances sparsity and shrinkage
Useful for high-dimensional data



1. *k*NN:
   - small *k*: high variance, low bias
   - large *k*: low variance, high bias

2. Polynomial:
   - small *n*, e.g., *n*=1 (linear):
     - high bias, low variance
   - large n:
     - low bias, high variance

3. Regularization:
   - lower variance, increases bias

KNN visualization for the U-shaped dataset

https://towardsdatascience.com/knn-visualization-in-just-13-lines-of-code-32820d72c6b6

Concluding Insights

- Evaluate your model in an appropriate way

High variance

- Get more data
- Try different features
    - Using smaller sets of feature fix **high variance**
- Increase regularization when **the variance is high**
- Use Bagging

High bias

- Try different features
    - Adding features helps fix high bias
- Decrease regularization when **bias is high**
- Use Boosting

Questions

- Are we over-optimizing models based on narrow, abstract metrics rather than their actual meaningfulness and disclosure in real-world applications?
- Are we evaluating what truly matters?
- Are we improving performance or just reinforcing established norms?
- Can evaluation frameworks be made more transparent and context-aware rather than just more complex?
- What might evaluation hide during the evaluation process? (Biases, ethical concerns, real-world complexity, unintended consequences?
- Who defines what "good performance" means—and for whom? How does our choice of evaluation metric shape—or constrain—our notion of fairness? Do our evaluation methods reflect the social and ethical contexts in which models operate? How do our chosen evaluation metrics influence—or limit—our understanding of fairness?