CMPUT 466
Machine Learning: Day 4

Professor: Bailey Kacsmar
kacsmar@ualberta.ca

Winter 2024

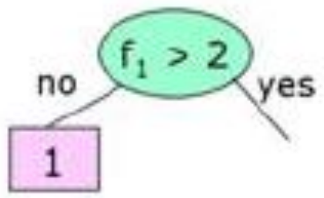
Many of these slides are derived from Alona Fyshe, Alex Thomo. Thanks!

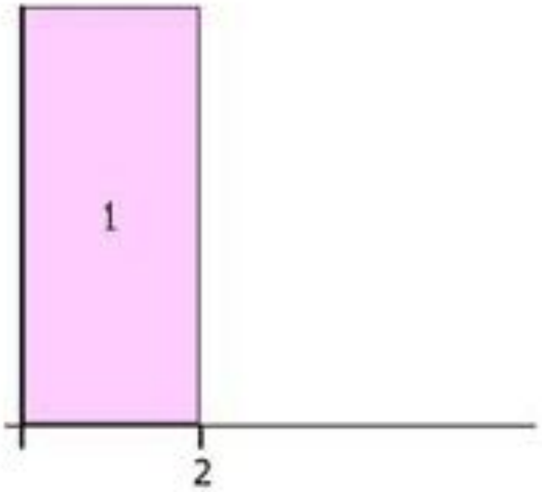
On Evaluation and Performance…

(with the help of decision trees)

Numerical attributes revisited

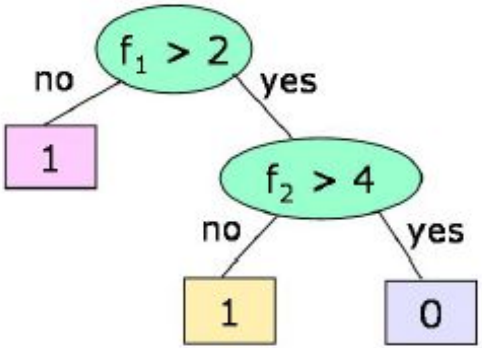
- Tests in nodes are of the form $f_i >$ constant
- Divides the space into rectangles.

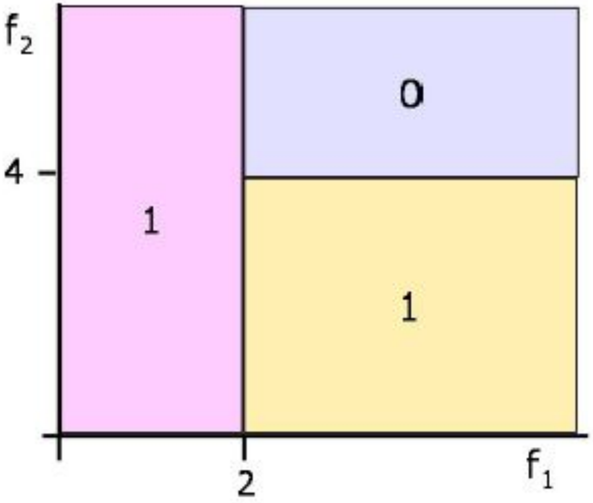




Numerical attributes

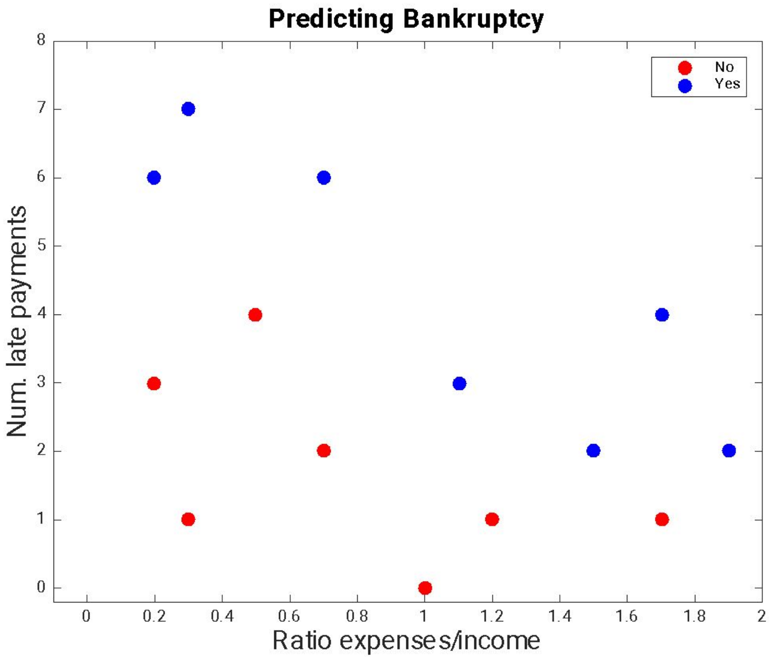
- Tests in nodes are of the form f_i > constant
- Divides the space into rectangles.





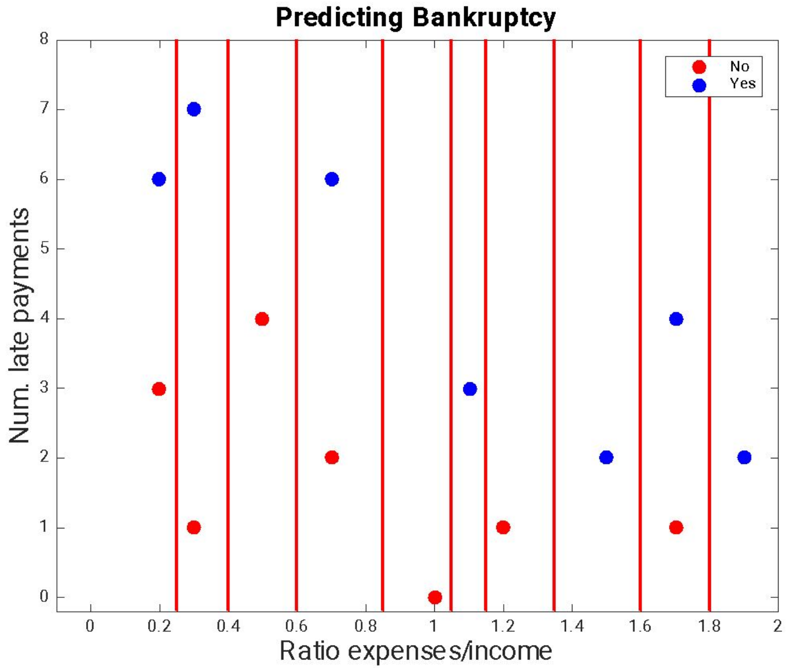
Example: Predicting Bankruptcy

Late	Ratio	Bankruptcy?
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



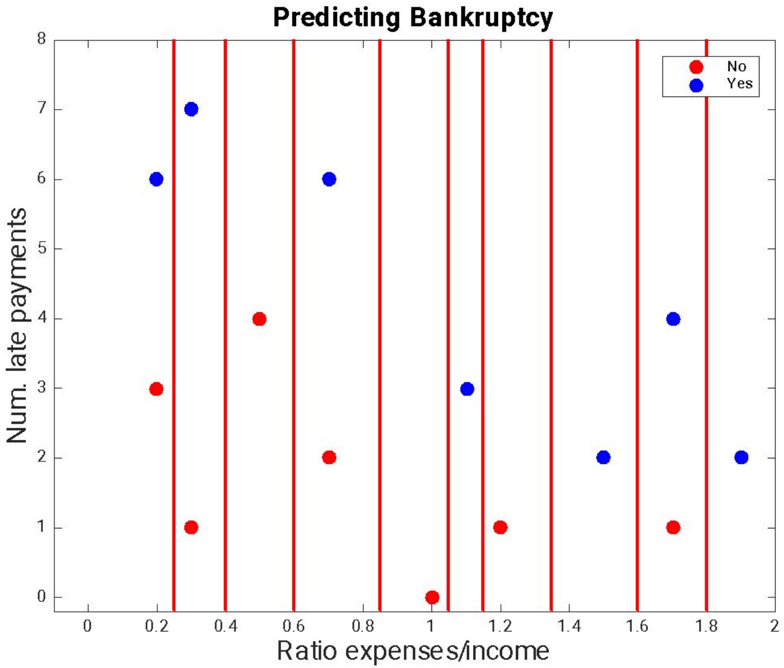
Considering splits

- Consider ‘splitting between each data point in each “attribute”



Considering splits

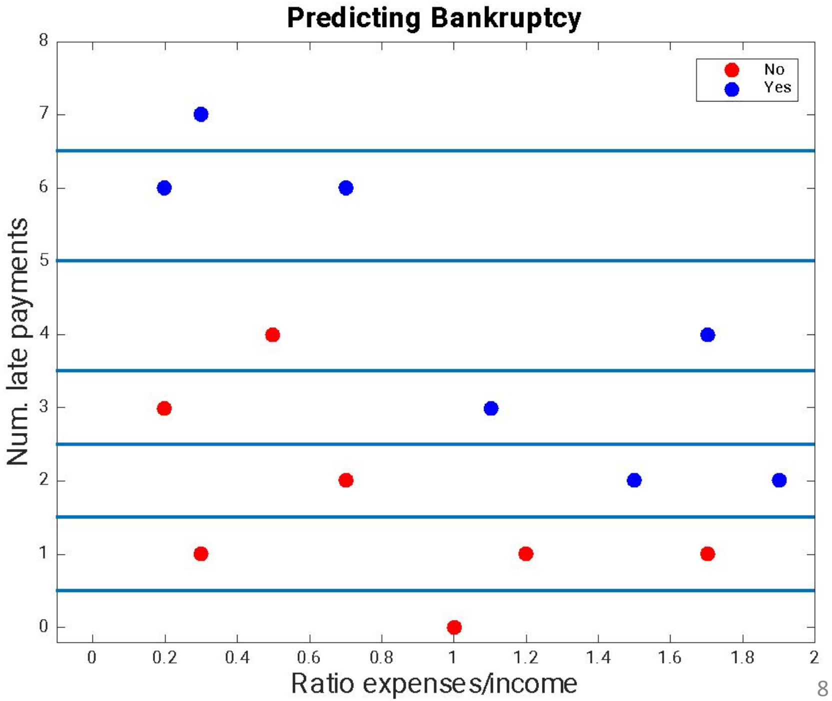
- Consider splitting between each data point in each “attribute”



- So, here we'd consider **9** different splits in the **ratio dimension**

Considering splits II

- And there are another **6** possible splits in the **late payments dimension**



Recall: Entropy

- H(X) = E(I(X)) **Expected** value of the **information** in X
- Expected value: $\mathrm{E}(\mathrm{f}(\mathrm{X})) = \sum_i P(x_i) * f(x_i)$
- Information: $I(x_i) = -\log_2 P(x_i)$
- Entropy: $H(X) = E(I(X)) = \sum_i P(x_i) I(x_i) = -\sum_i P(x_i) \log_2 P(x_i)$

Explicitly Consider: Information Gain

Information Gain is calculated as:

IG(T,a) = H(T) - H(T|a)

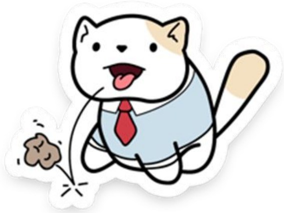
IG(T,a) = (Entropy of the parent node) – (average entropy of the child nodes)

= H(T) - [Pr(leftChildren)*H(leftChildren)+ Pr(rightChildren)*H(RightChildren)]

https://en.m.wikipedia.org/wiki/Information_gain_(decision_tree)

Act.

Outlook	**Temp**	**Humidity**	**Windy**	**Play**
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



Entropy (for Outlook)

$$\text{entropy}(5/14,9/14) = -5/14*\log_2(5/14) -9/14*\log_2(9/14)$$
$$= 0.94\ldots$$

Information Gain: Attribute "Outlook"

outlook=sunny

entropy(2/5,3/5) = -2/5*log2(2/5) -3/5*log2(3/5) = .971

outlook=overcast

entropy(4/4,0/4) = -1*log2(1) -0*log2(0) = 0

outlook=rainy

entropy(3/5,2/5) = -3/5*log2(3/5)-2/5*log2(2/5) = .971

AE = .971*(5/14) + 0*(4/14) + .971*(5/14) = **.693**

IG = 0.94-.693 = 0.247

Act.

Information Gain

Information Gain is calculated as:

$$IG(T,a) = H(T) - H(T|a)$$

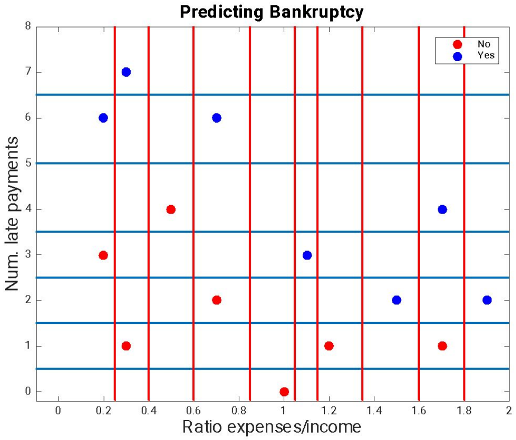
$$IG(T,a) = (\text{Entropy of the parent node}) - (\text{average entropy of the child nodes})$$

$$= H(T) - [Pr(\text{leftChildren}) * H(\text{leftChildren}) + Pr(\text{rightChildren}) * H(\text{RightChildren})]$$

Calculate the information gain from Tuesdays (Day 3 slides) example for each of: outlook (in class), humidity|overcast, temperature|rainy, and Windy|rainy

https://en.m.wikipedia.org/wiki/Information_gain_(decision_tree)

Bankruptcy: Late Payments



thresh	Neg. less	Pos. less	Neg. greater	Pos. greater	AE
6.5	**7**	**6**	**0**	**1**	**0.92**
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	**1**	**0**	**6**	**7**	**0.92**

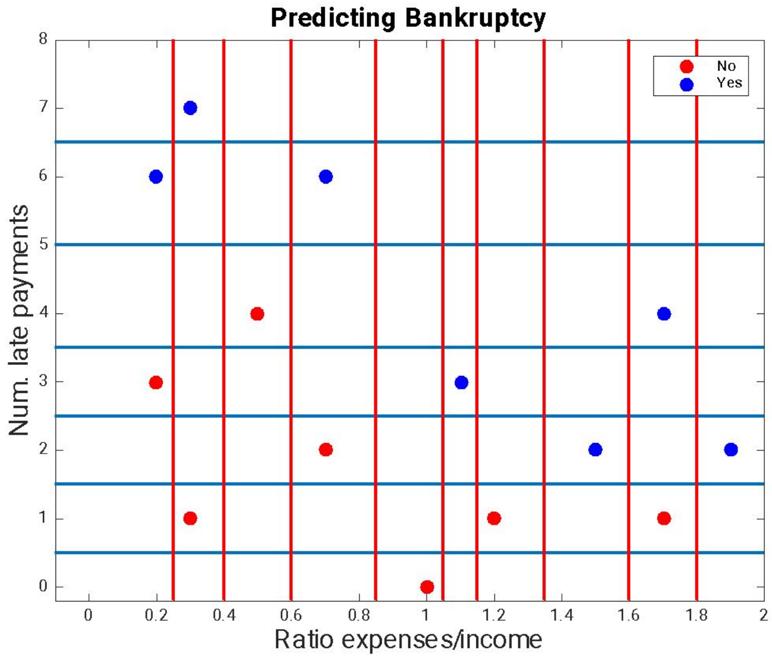
For the first and last rows:

$H([7,6]) = (-(7/13)*\log_2(7/13)-(6/13)*\log_2(6/13)) = .9957$

$H([0,1]) = (-(0/1)*\log_2(0/1)-(1/1)*\log_2(1/1)) = 0$

$H([7,6],[0,1]) = .9957*13/14 + 0*1/14 = .92$

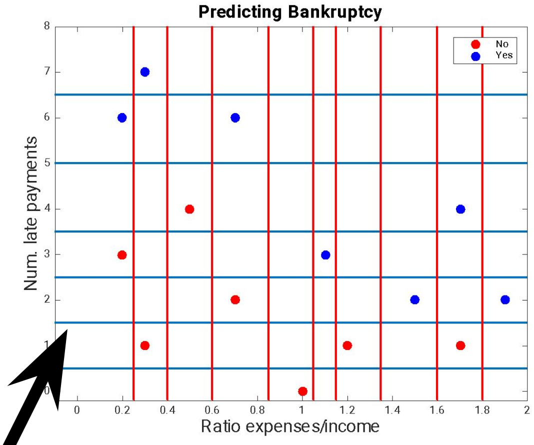
Bankruptcy Ratio

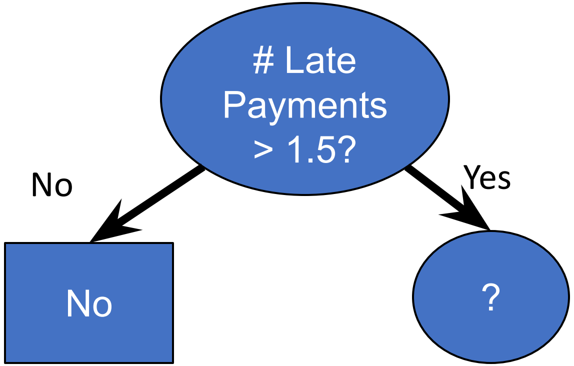


thresh	0.25	0.4	0.6	0.85	1.05	1.15	1.35	1.6	1.8
AE	1	1	0.98	0.98	0.94	0.98	0.92	0.98	0.92

Bankruptcy Example

thresh	Neg. less	Pos. less	Neg. greater	Pos. greater	AE
6.5	7	6	0	1	0.92
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.92

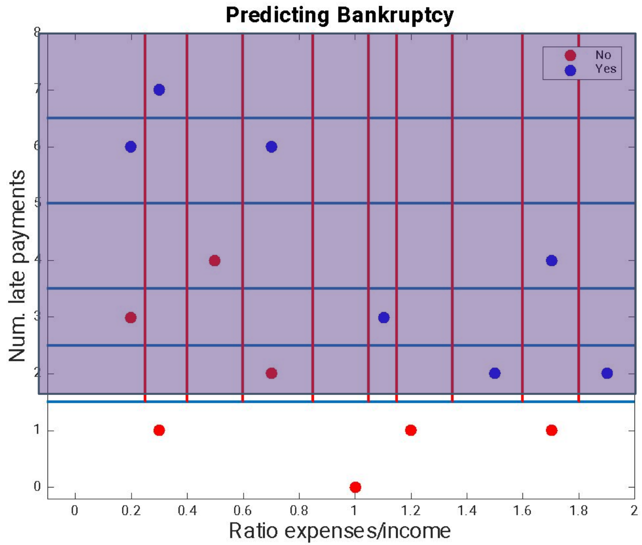




Bankruptcy Example

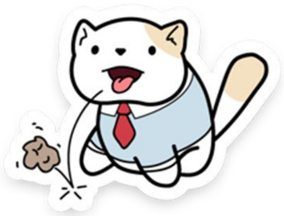
- Now, recurse on data points with num. late payments >= 1.5
- Can we just reuse these tables?

thresh	Neg. less	Pos. less	Neg. greater	Pos. greater	AE
6.5	7	6	0	1	0.92
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.92



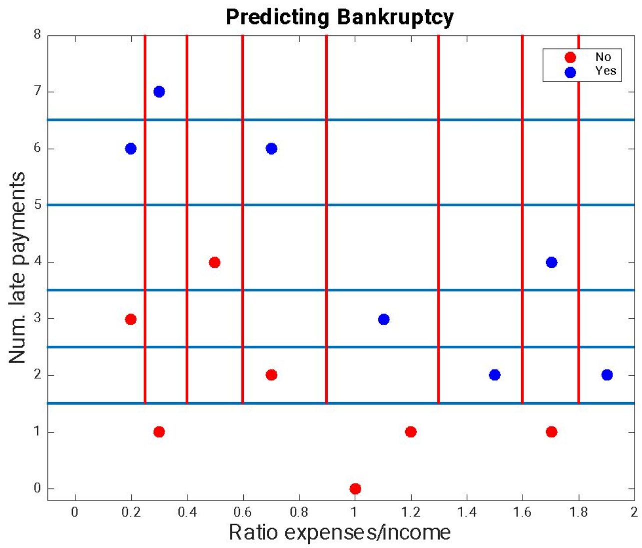
thresh	0.25	0.4	0.6	0.85	1.05	1.15	1.35	1.6	1.8
AE	1	1	0.98	0.98	0.94	0.98	0.92	0.98	0.92

Bankruptcy Example



- Have to make new tables

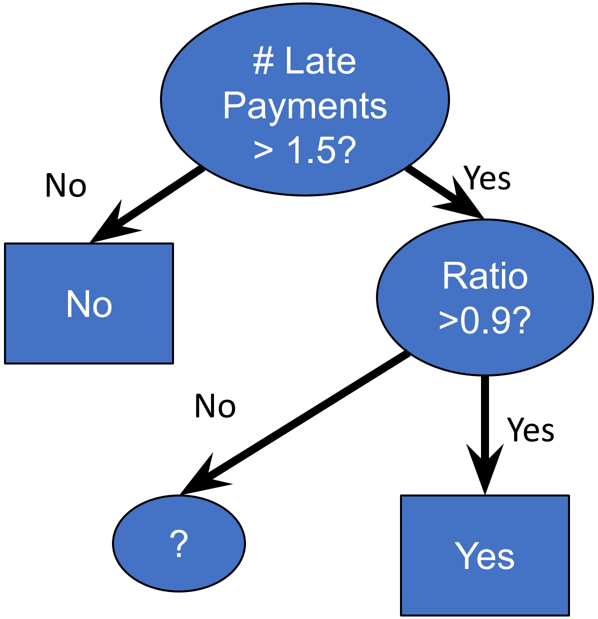
thresh	Neg. less	Pos. less	Neg. greater	Pos. greater	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88

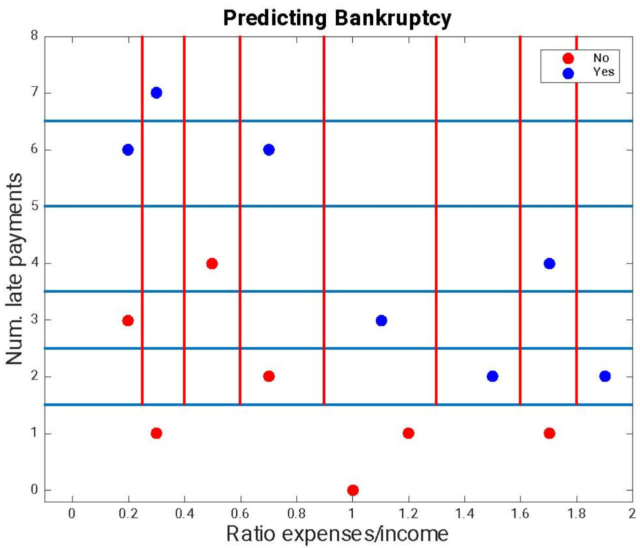


thresh	0.25	0.4	0.6	0.9	1.3	1.6	1.8
AE	0.85	0.88	0.79	0.6	0.69	0.76	0.83

Bankruptcy Example

- Have to make new tables

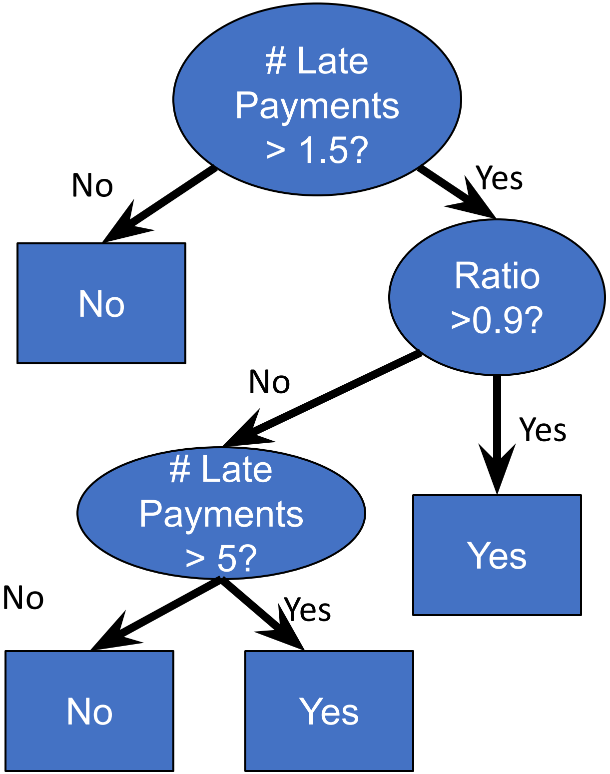


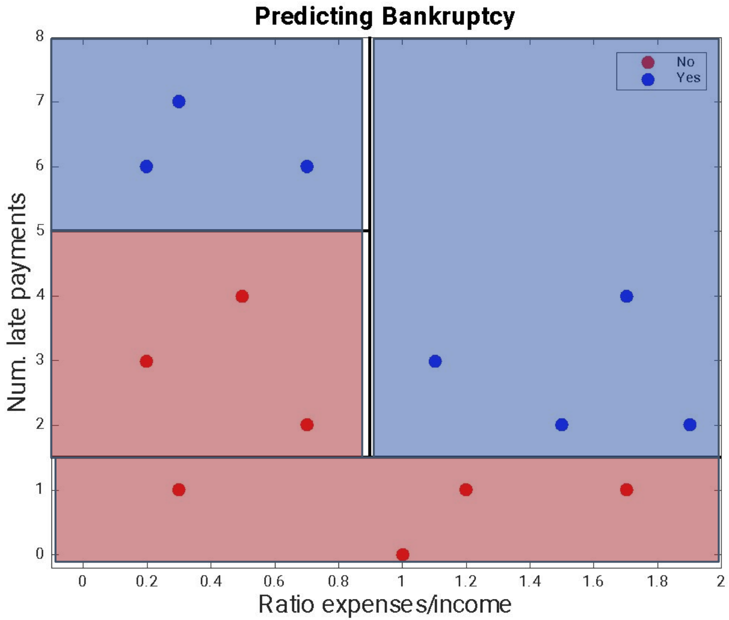


thresh	0.25	0.4	0.6	0.9	1.3	1.6	1.8
AE	0.85	0.88	0.79	0.6	0.69	0.76	0.83

Bankruptcy Example

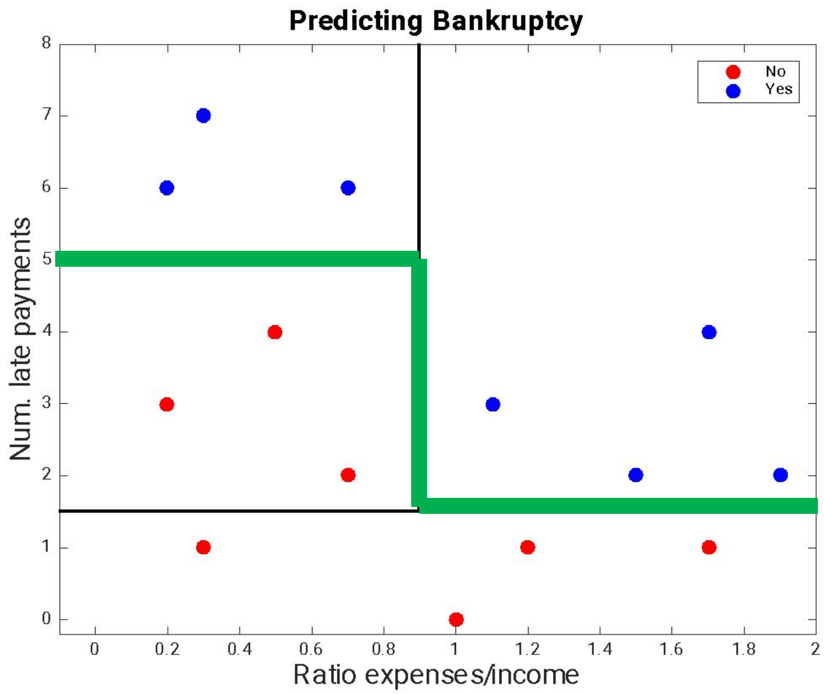
- Continue to obtain this tree:





Decision trees learn non-linear decision boundaries

- Can perform well when a non-linear boundary is required
- Can also overfit (we will talk more about this shortly)



What is classification?

What is classification?

That’s enough of it.

Classification vs Regression

- So far we have described classification
  - predicting one of a discrete set of labels
    - play tennis? yes/no
    - Neighbor’s behavior: walk/drive
    - Car type: luxury, mini, sports, van

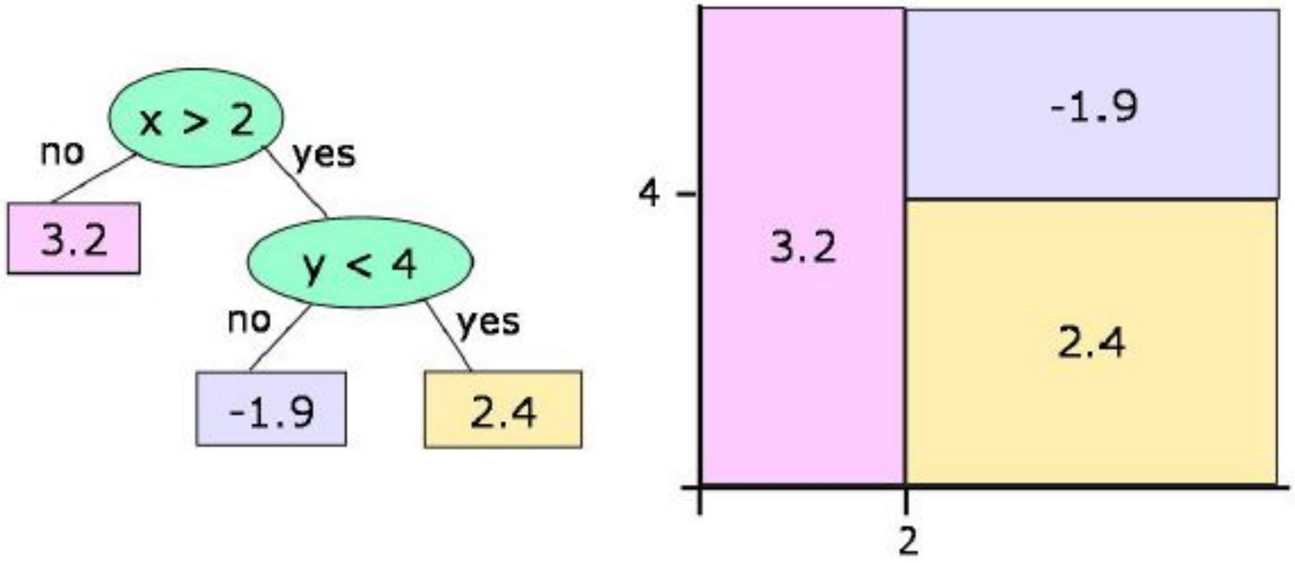
Classification vs Regression

- So far we have described classification
  - predicting one of a discrete set of labels
    - play tennis? yes/no
    - Neighbor’s behavior: walk/drive
    - Car type: luxury, mini, sports, van

- Sometime we want to predict a number in a range
  - E.g. age, forecast temperature, etc…
  - That is called ***regression*** (predicting a **real** number)

Regression Trees

- Like decision trees, but with **real-valued constant outputs** at the leaves.



Things to consider

- Prediction is a real number
  - Thus training labels are real numbers
  - Instead of {yes, yes, yes, no} at a leaf, we will have something like {0.1, 0.5, 1.3, 0.8}

Things to consider

- Prediction is a real number
  - Thus training labels are real numbers
  - Instead of {yes, yes, yes, no} at a leaf, we will have something like {0.1, 0.5, 1.3, 0.8}
- How to evaluate a candidate split?
  - How do we measure “purity” in real-valued numbers?
  - **How pure is {0.1, 0.5, 1.3, 0.8}?**

More things to consider

- What to predict based on the instances in a leaf node?
  - Since labels are continuous, most datapoints won’t have the same label

More more things to consider

- When to stop splitting?
  - If datapoints don’t have exactly the same label, if we split to perfect purity we’ll end up with only one training example in each node
    - We’ll end up with 4 leaf nodes: {0.1}, {0.5}, {1.3}, {0.8}
  - May not be good for **generalization**

More more things to consider

- When to stop splitting?
  - If datapoints don't have exactly the same label, if we spilt to perfect purity we'll end up with only one training example in each node
    - We'll end up with 4 leaf nodes: {0.1}, {0.5}, {1.3}, {0.8}
  - May not be good for **generalization**
- What could we measure to decide when to stop splitting?
  - Hint: What did we do previously with classification trees?

Yup, Pruning

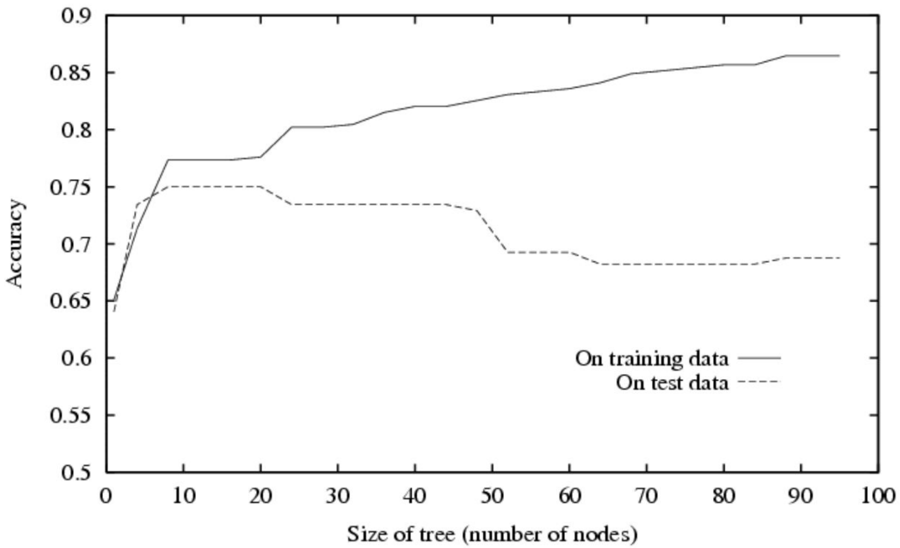
- How can we reduce overfitting in our decision trees?
  - make the trees smaller (less complex)

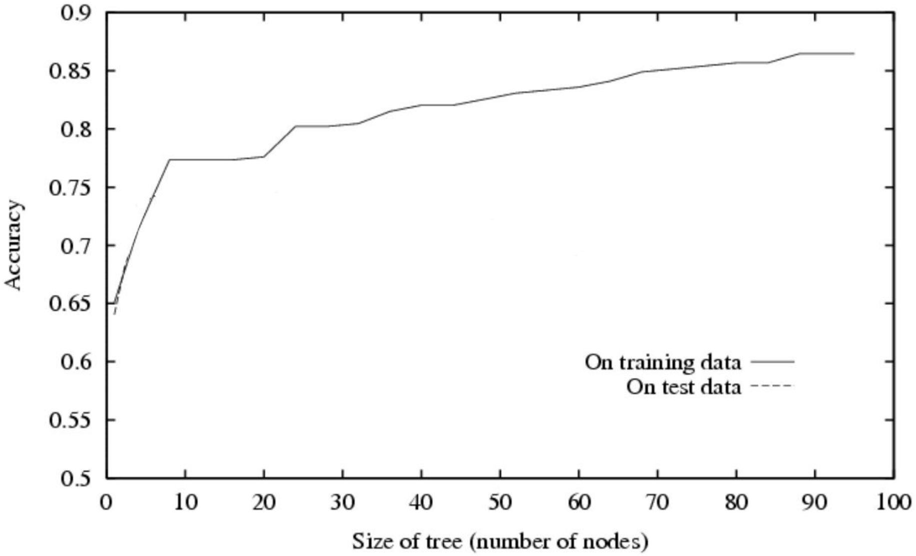
Time to Define: Overfitting

- What is overfitting?
- What does overfitting look like?

Overfitting

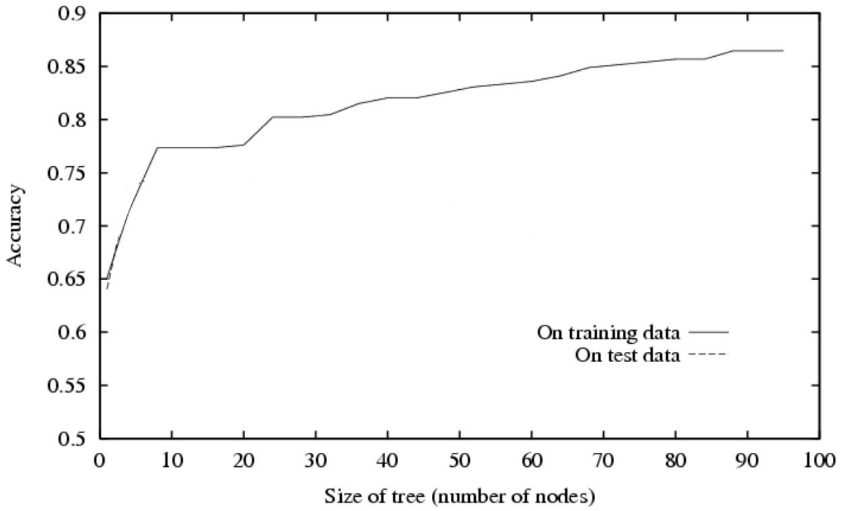
- What is overfitting?
- What does overfitting look like?

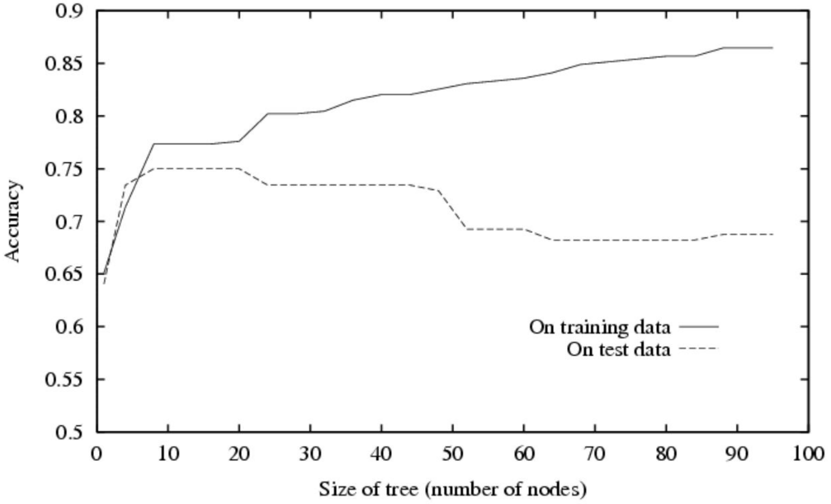


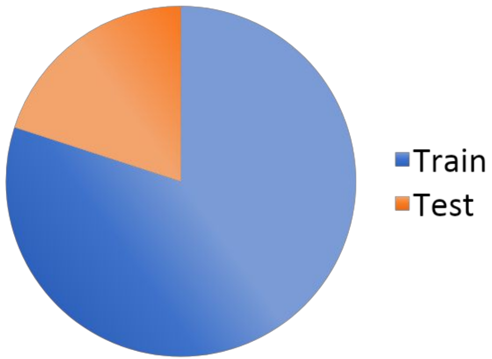


Overfitting

- What is overfitting?
- What does overfitting look like?







Why is it overfitting?

- Where does overfitting come from?
  - noise in labels
  - noise in features
  - too little data (lack of representative data)
  - model-data mismatch

Back to Decision Trees…

Avoiding overfitting?

Avoiding overfitting?

- Control the growth

ORRRR

- Trim the growth

Early Stopping (Pre-prune)

- Stop making splits when
  - **average entropy** doesn't change much
  - Predefined **# of training instances** reach leaf
  - Predefined **depth**

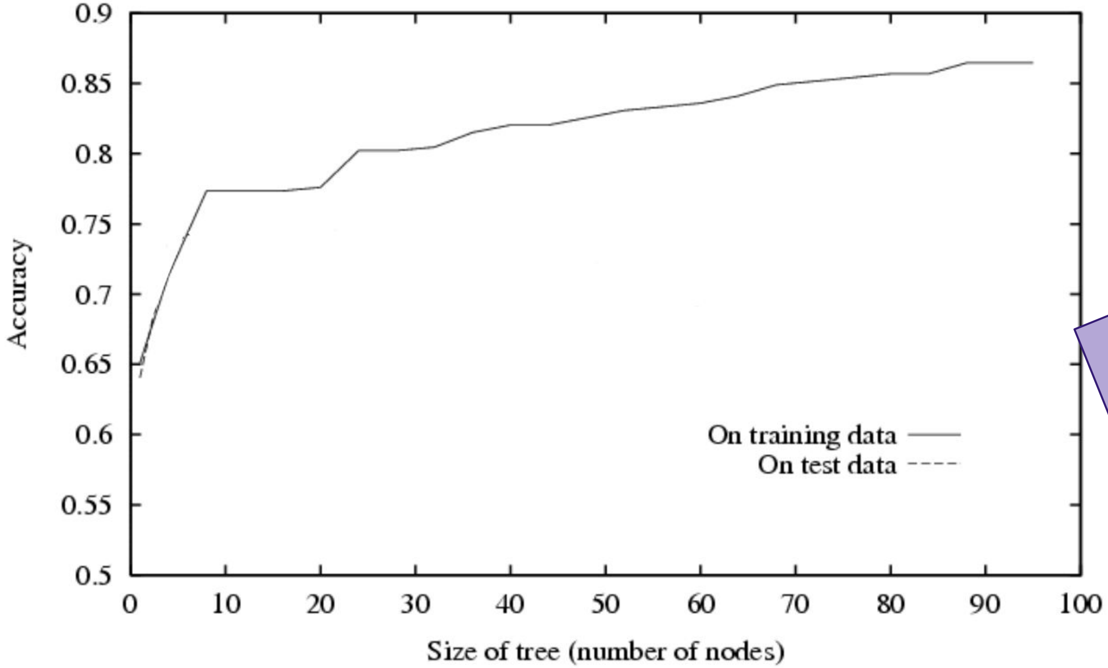
Early Stopping (Pre-prune)

- Stop making splits when
  - **average entropy** doesn’t change much
  - Predefined **# of training instances** reach leaf
  - Predefined **depth**

Control the trees “growth”

Post pruning

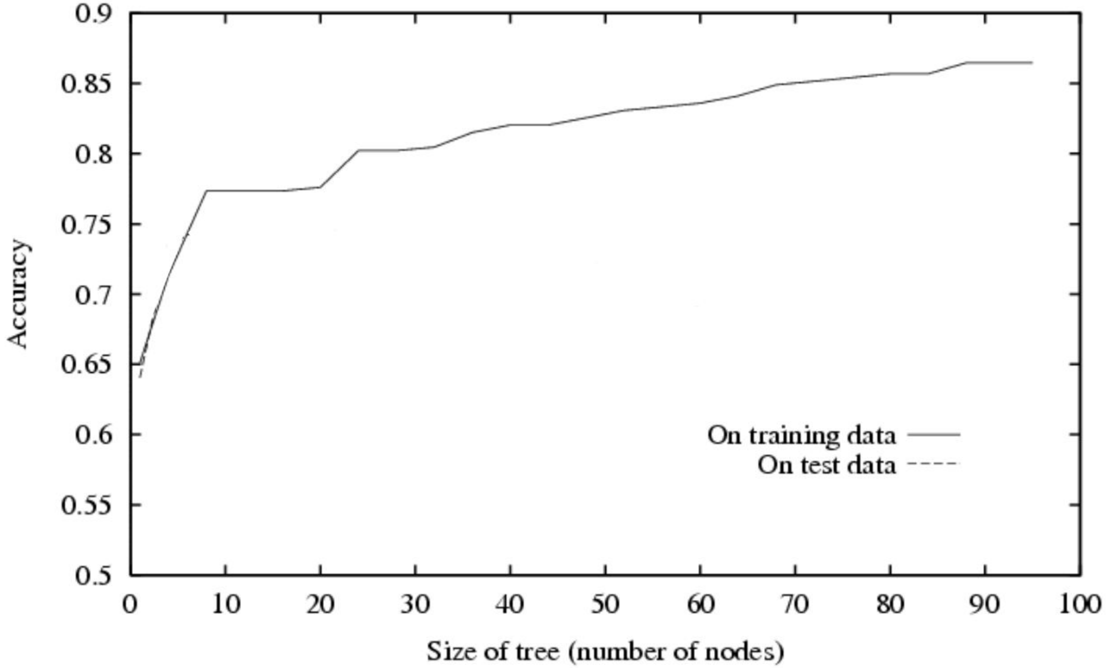
- Build a complex tree, then simplify
- It’s cheating to check test set accuracy during training/pruning

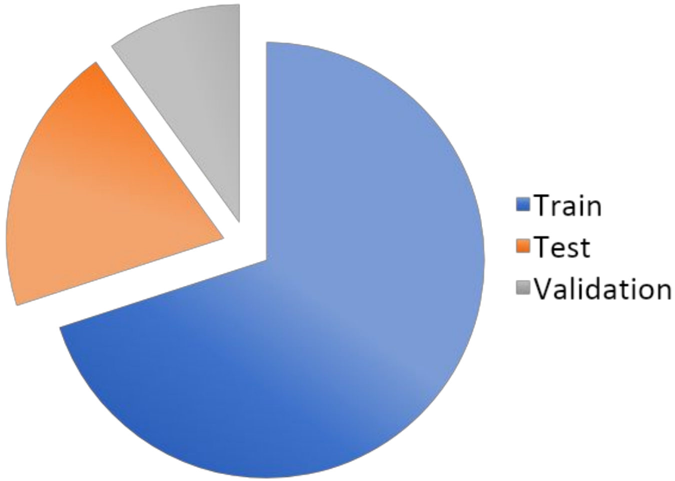


Trim it down

Post pruning: Better Split Another Set

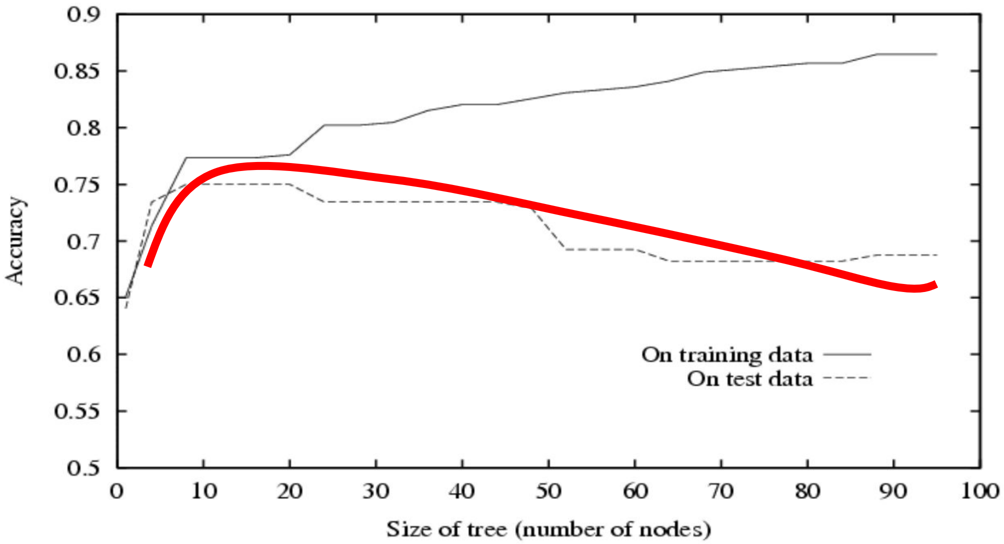
- Build a complex tree, then simplify
- It's cheating to check test set accuracy during training/pruning



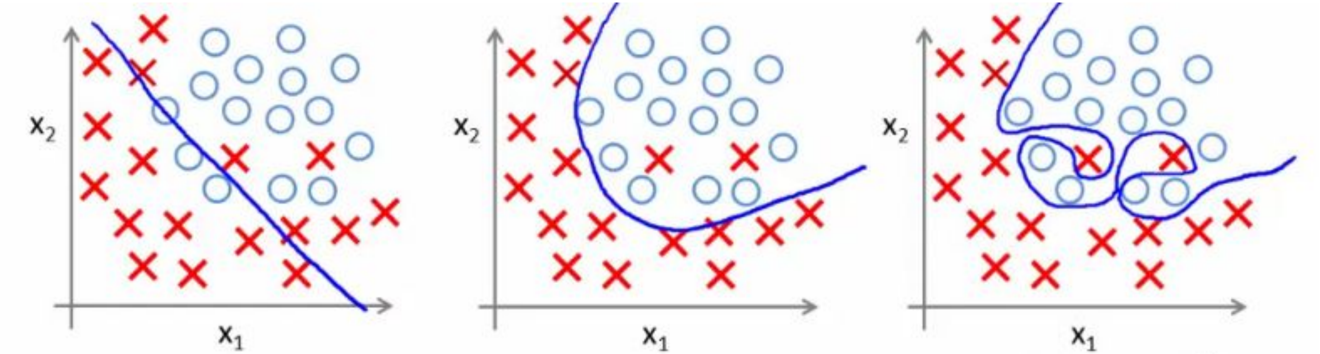


Description: Validation Set

- Select some part of your training data for validation (tuning)
    - **<u>Don't use it to train</u>**



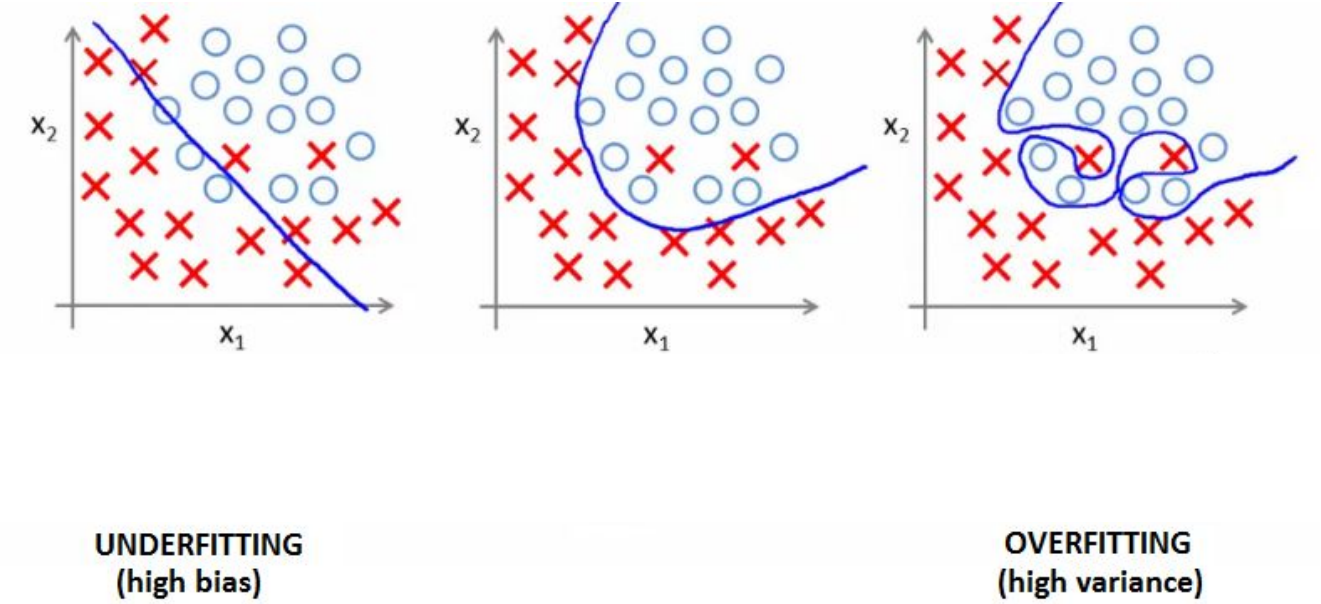
Over vs Under fitting



UNDERFITTING
(high bias)

OVERFITTING
(high variance)

Over vs Under fitting



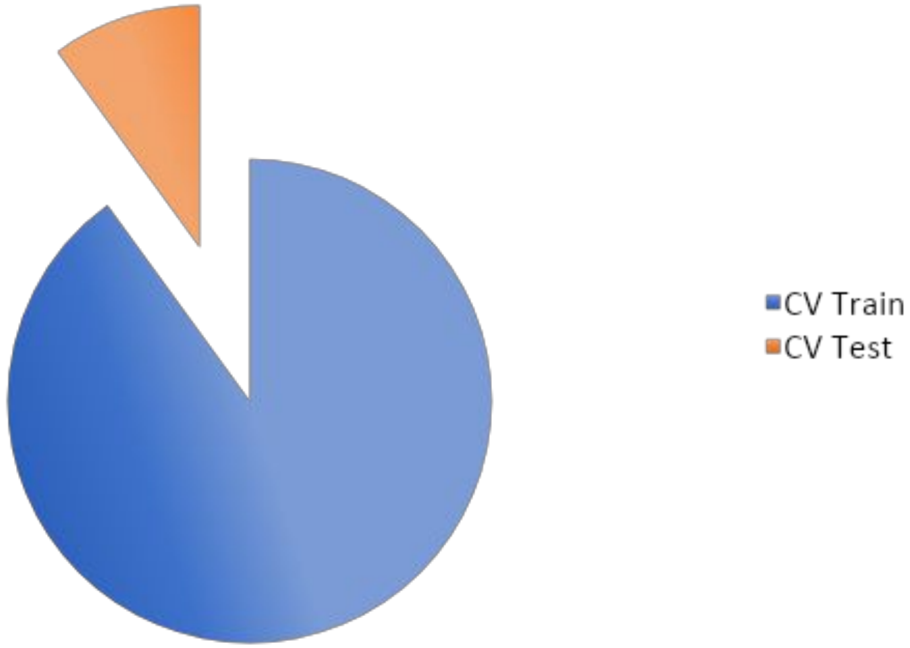
Bias: error of predict and train

Var: error of predict and test

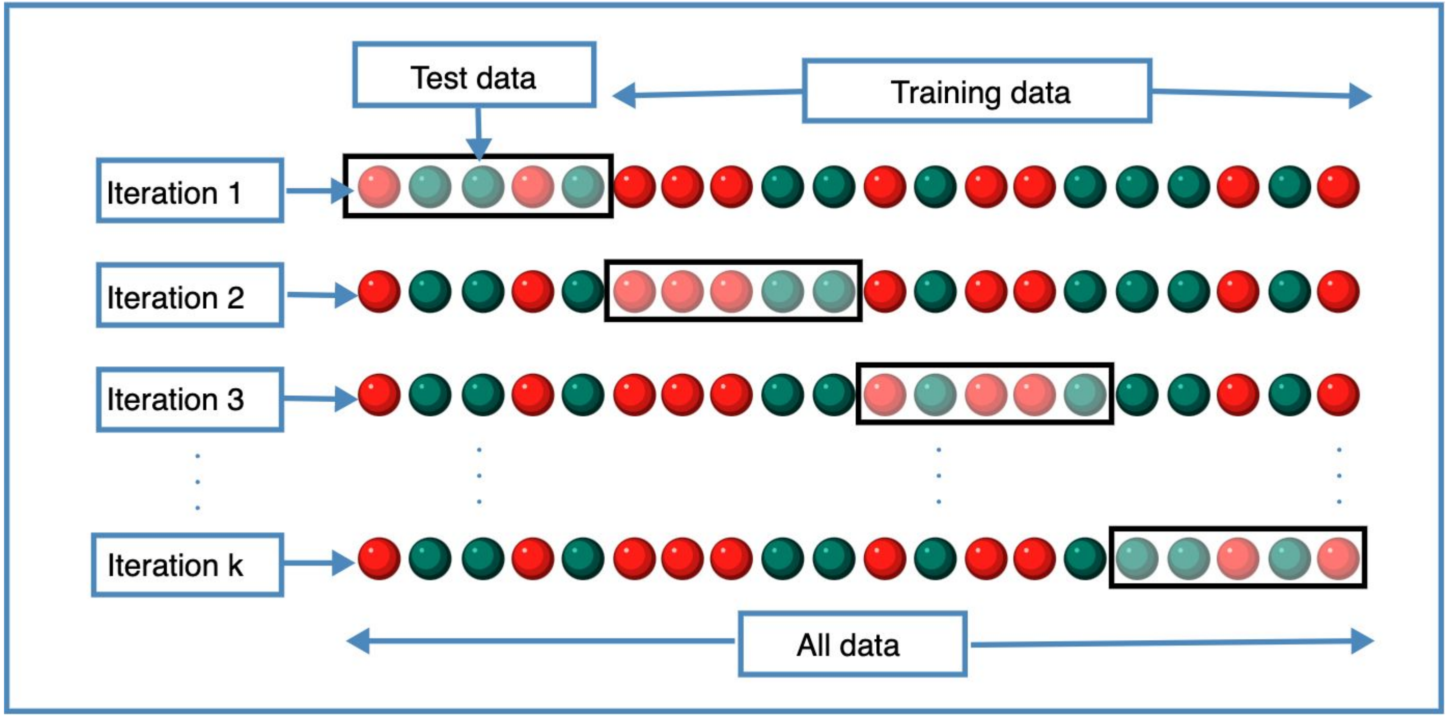
Model Selection

- Suppose we are trying to select among several different models for a learning problem.
- E.g.
  - Full tree vs. tree pruned to depth 5 vs. random forest?

Time to Define: Cross Validation



Cross Validation



Practical issues for CV

- How to big of a slice of the pie?
  - Commonly used K = 10 folds (thus each fold is 10% of the data)
  - Leave-one-out-cross-validation LOOCV (K=N, number of training instances)

Practical issues for CV

- How to big of a slice of the pie?
  - Commonly used K = 10 folds (thus each fold is 10% of the data)
  - Leave-one-out-cross-validation LOOCV (K=N, number of training instances)

- **One important point** is that (for a particular fold) the test data is never used for training, because doing so would result in overly (**indeed dishonest**) optimistic accuracy rates during the testing phase.

Practical issues for CV

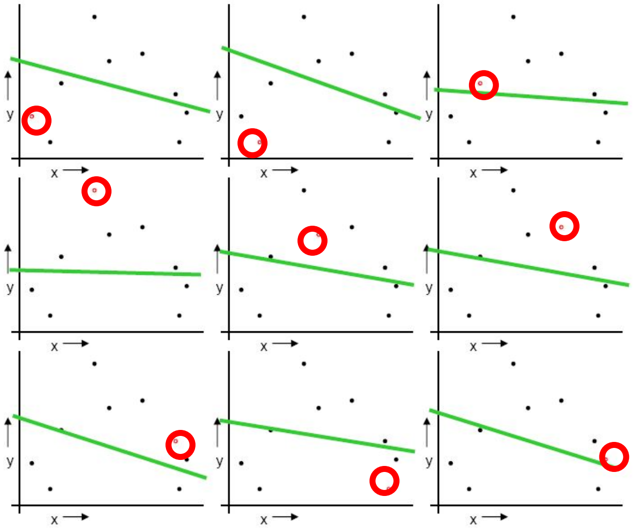
- How to big of a slice of the pie?
  - Commonly used K = 10 folds (thus each fold is 10% of the data)
  - Leave-one-out-cross-validation LOOCV (K=N, number of training instances)

- **One important point** is that (for a particular fold) the test never used for training, because doing so would result in o (**indeed dishonest**) optimistic accuracy rates during the tes

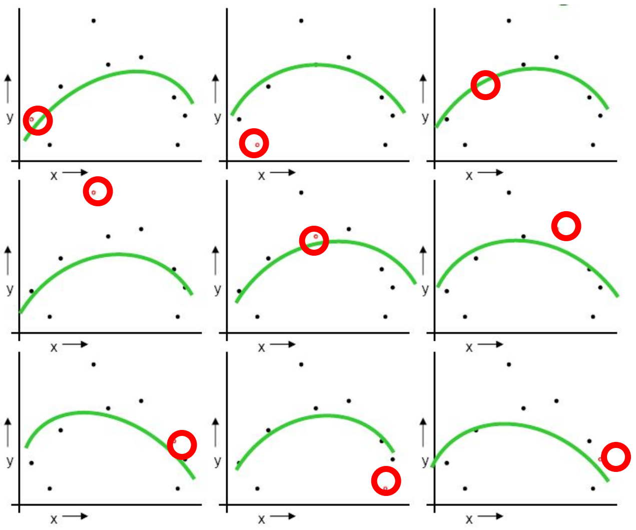
- Stratification – should you balance the classes across the folds?

Example:

- When $k=N$, the algorithm is known as Leave-One-Out-Cross-Validation (LOOCV)

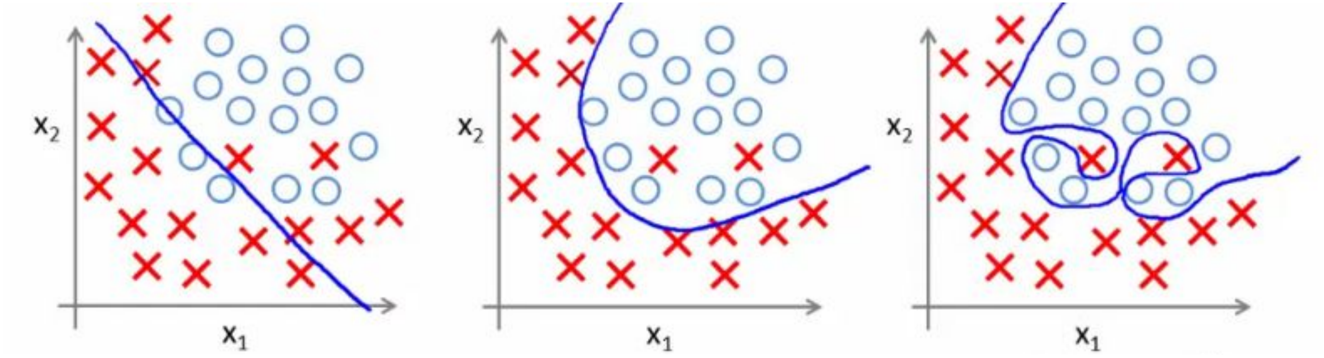


MSELOOCV(M_1)=2.12



MSELOOCV(M_2)=0.962

Why is CV so important?



UNDERFITTING
(high bias)

OVERFITTING
(high variance)

Measuring Performance

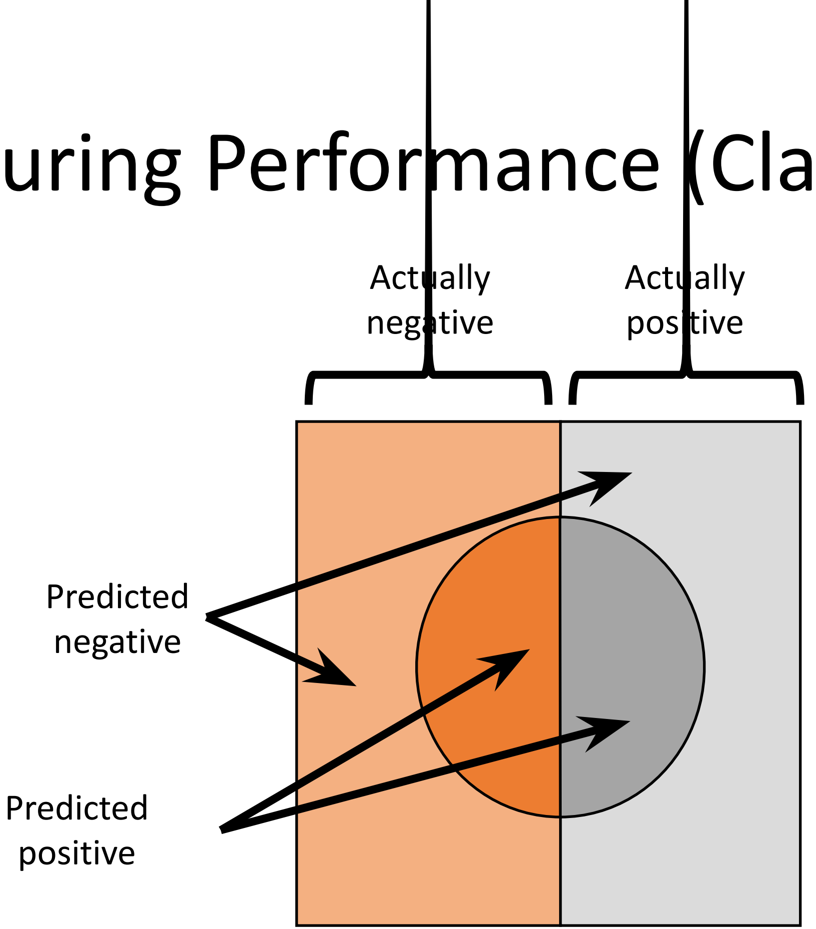
- We usually calculate performance on test data
- Calculating performance on training data is called resubstitution and is an *optimistic* measure of performance
  - why?
  - because it can’t detect overfitting

Performance/Evaluation?

Measuring Performance (Classification)

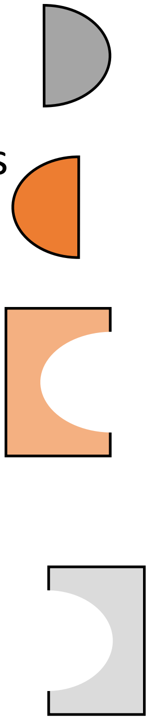
- **Accuracy**
  - (# test instances correctly labeled)/(# test instances)
- **Error**
  - 1- accuracy
  - (# test instances **in**correctly labeled)/(# test instances)

Measuring Performance (Classification)

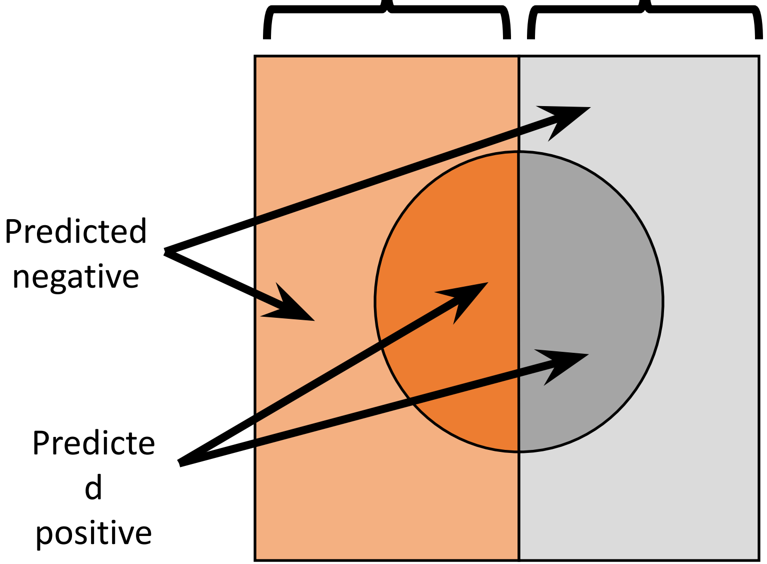


Measuring Performance (Classification)

- True positives
- False positives
- True neg
- False positive

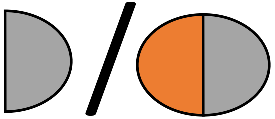


How to remember: the first word is whether the prediction is correct, the second word is what the prediction was.

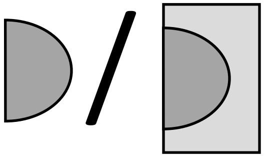


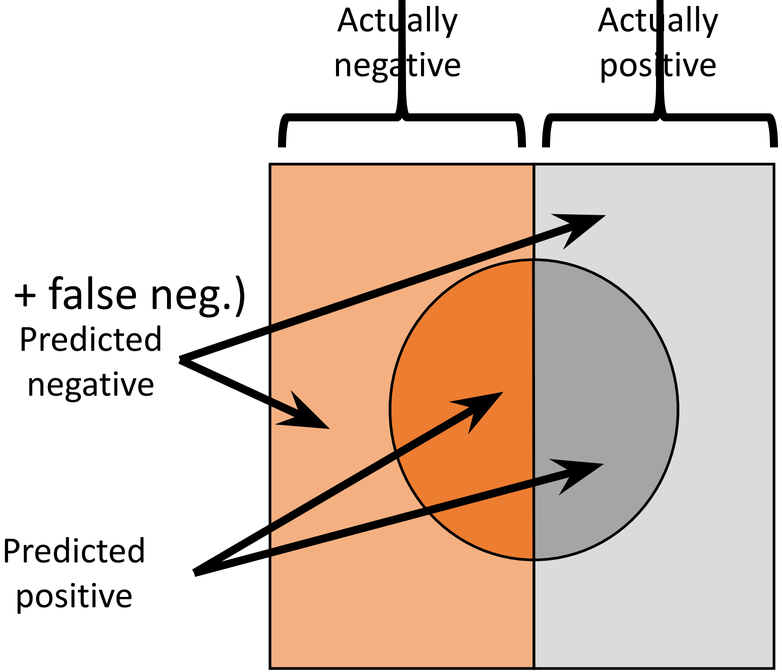
Terms to Measure Performance (Classification)

- **Precision**
  - true positives/(true pos. + false pos.)



- **Recall**
  - true positives/(true pos. + false neg.)





How to remember: the first word is whether the prediction is correct, the second word is what the prediction was.

Measuring Performance (Classification)

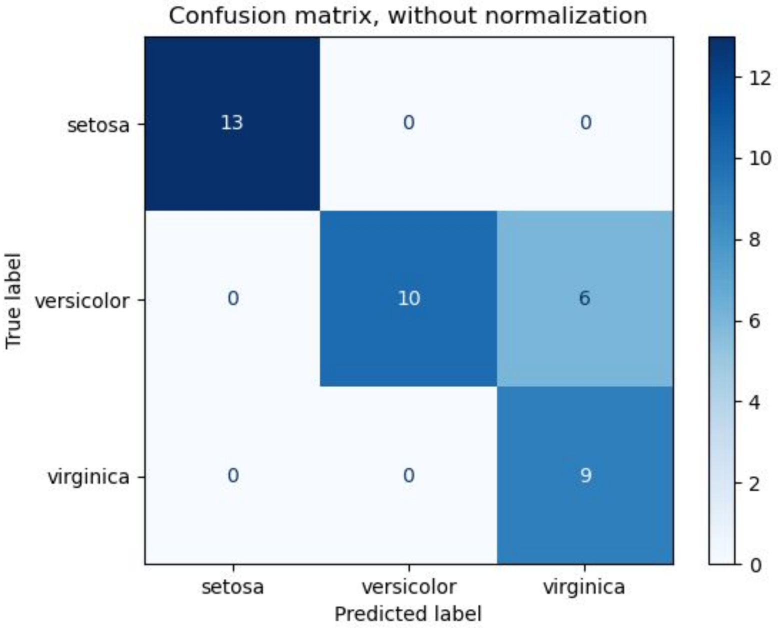
- F1
  - harmonic mean of precision and recall
  - 2*(p*r)/(p+r)

Where p = precision and r = recall.

Evaluating when >1 class

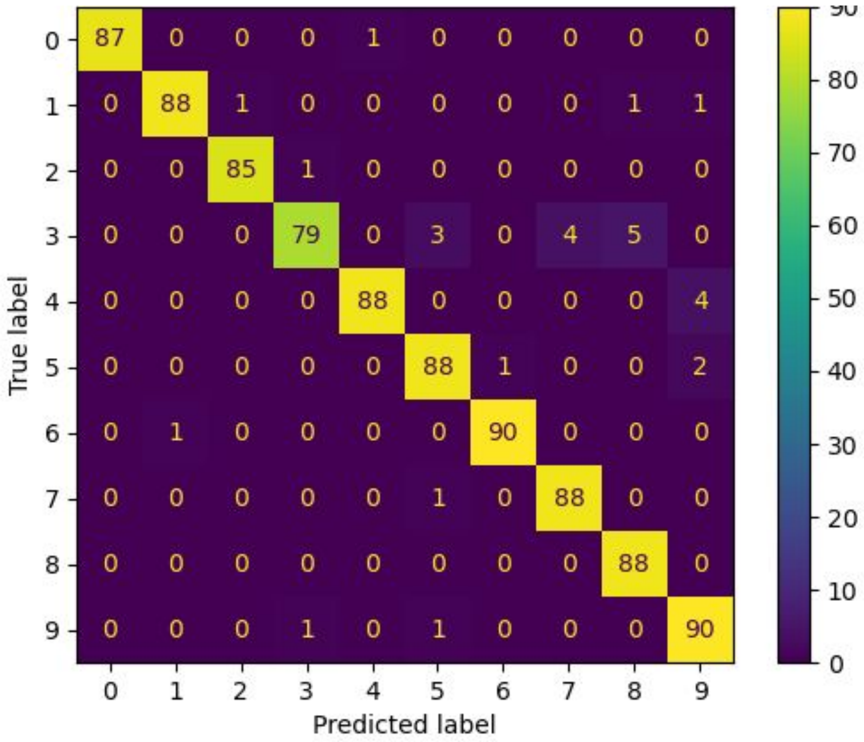
- Can still compute accuracy/error
- Can also compute per-class P, R, F1

Performance Eval. Tool: Confusion matrix



https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix

Confusion matrix (handwritten digits)



https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html#sphx-glr-auto-examples-classification-plot-digits-classification-py

Measuring Performance (Regression)

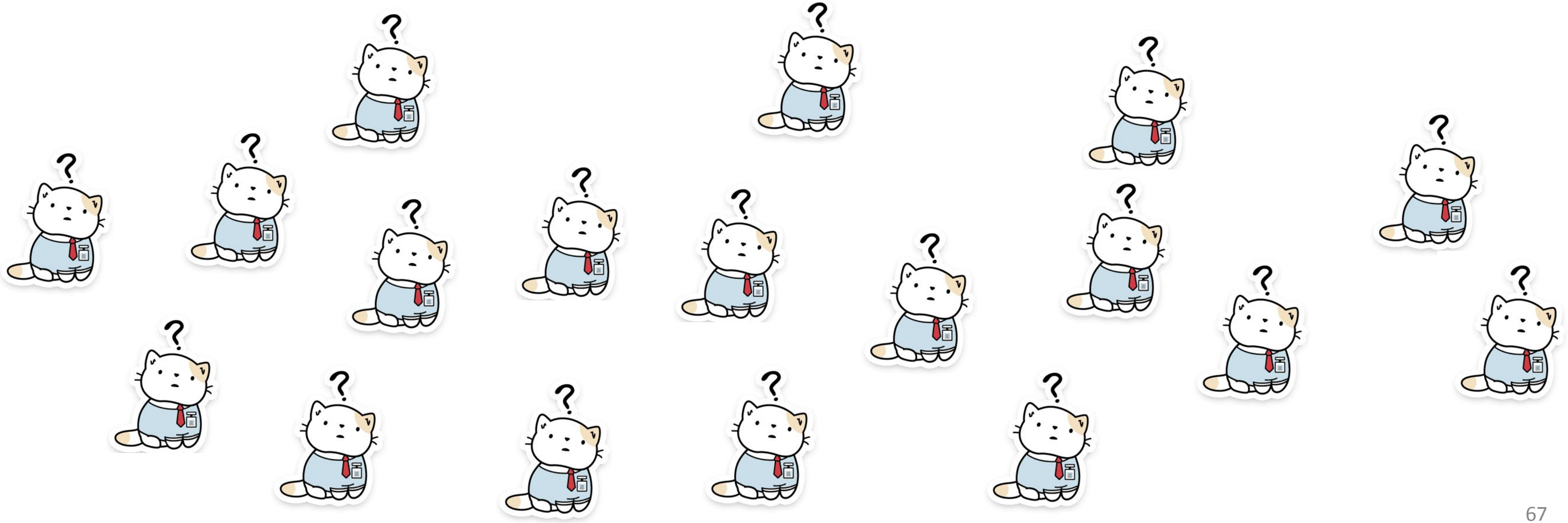
- Regression
  - predicting a real number

- Root Mean Squared Error (RMSE)
  - sometimes(mostly?) just MSE (no sqrt)

$$\sqrt{\frac{1}{N}\underbrace{\sum_{i=1}^{N}}_{\text{sum over all N test instances}}\underbrace{(\hat{y}_i - y_i)}_{\text{error}}^2}$$

Performance

- **Comparing performance of classifiers**
- How do you know if your accuracy number is “high” or error is “low”?



Exercise Z

Consider the task of building a classifier from random data, where the attribute values are generated randomly irrespective of the class labels. Assume the data set contains records from two classes, “+” and “−.” Half of the data set is used for training while the remaining half is used for testing.

(a) Suppose there are an **equal number** of positive and negative records in the data and the **classifier predicts every test record to be positive**. What is the expected error of the classifier on the test data?

(b) Repeat the previous analysis assuming that the classifier predicts each test record to be **positive class** with probability 0.8 and **negative class** with probability 0.2.

Exercise Z - answers

Consider the task of building a classifier from random data, where the attribute values are generated randomly irrespective of the class labels. Assume the data set contains records from two classes, “+” and “−.” Half of the data set is used for training while the remaining half is used for testing.

(a) Suppose there are an **equal number** of positive and negative records in the data and the **classifier predicts every test record to be positive**. What is the expected error of the classifier on the test data? **Answer: 50%.**

(b) Repeat the previous analysis assuming that the classifier predicts each test record to be **positive class** with probability 0.8 and **negative class** with probability 0.2. **Answer: 50%.**

Exercise Z - Part 2

Consider the task of building a classifier from random data, where the attribute values are generated randomly irrespective of the class labels. Assume the data set contains records from two classes, “+” and “−.” Half of the data set is used for training while the remaining half is used for testing.

(c) Suppose **2/3** of the data belong to the **positive** class and the remaining **1/3** belong to the **negative** class. What is the **expected error** of a classifier that **predicts every test record to be positive**?

(d) Repeat the previous analysis **assuming that the classifier predicts each test record to be positive class with probability 2/3** and negative class with probability **1/3**.

Exercise Z - Part 2 - answers

Consider the task of building a classifier from random data, where the attribute values are generated randomly irrespective of the class labels. Assume the data set contains records from two classes, “+” and “−.” Half of the data set is used for training while the remaining half is used for testing.

(c) Suppose **2/3** of the data belong to the **positive** class and the remaining **1/3** belong to the **negative** class. What is the **expected error** of a classifier that **predicts every test record to be positive**? **Answer: (2/3)*0+(1/3)*1 = 33%.**

(d) Repeat the previous analysis assuming that the classifier predicts each test record to be positive class with probability 2/3 and negative class with probability 1/3. **Answer: (2/3)*(1/3)+(1/3)*(2/3) = 44.4%.**

Exercise X

Consider a classifier X that has **Accuracy = 50%** on a (test) dataset with a class taking 2 possible values (A, B).

The distribution of the instances for each class value is:

A:50,   B:50.

How does X compare to a random classifier Y that outputs A, and B, 50%, 50% of the time, respectively.

Exercise X - Answer

Consider a classifier X that has **Accuracy = 50%** on a (test) dataset with a class taking 2 possible values (A, B).

The distribution of the instances for each class value is:

A:50,   B:50.

How does X compare to a random classifier Y that outputs A, and B, 50%, 50% of the time, respectively.

Answer:

- Y's accuracy: (50*50/100 + 50*50/100)/100 = 50%
- So, X performs the same (accuracy-wise) as Y.

Exercise W

Consider a classifier X that has **Accuracy = 50%** on a (test) dataset with a class taking 4 possible values (A, B, C, and D).

The distribution of the instances for each class value is

A:25, B:25, C:25, and D:25.

How does X compare to a random classifier Y that outputs A, B, C, and D 25%, 25%, 25%, and 25% of the time, respectively.

Exercise W - answer

Consider a classifier X that has **Accuracy = 50%** on a (test) dataset with a class taking 4 possible values (A, B, C, and D).

The distribution of the instances for each class value is

A:25, B:25, C:25, and D:25.

How does X compare to a random classifier Y that outputs A, B, C, and D 25%, 25%, 25%, and 25% of the time, respectively.

Answer:

- Y's accuracy: (25*25/100 + 25*25/100 + 25*25/100 + 25*25/100)/100 = 25%
- So, X does twice better than Y (accuracy-wise).

Exercise V

Distribution of the instances for each class value is
A:25, B:25, C:25, and D:25.

Random classifier Y outputs A, B, C, and D, 25%, 25%, 25%, and 25% of the time, respectively.

Precision and Recall (wrt A)?

Exercise V - Answer

Distribution of the instances for each class value is
A:25, B:25, C:25, and D:25.

Random classifier Y outputs A, B, C, and D, 25%, 25%, 25%, and 25% of the time, respectively.

Precision and Recall (wrt A)?

Answer:

- Y will say 25% of the time "A" and 75% of the time "not A".
- TP = 1/4*1/4, FP = 3/4*1/4, FN = 1/4*3/4
- Precision= TP/(TP+FP) = 25%
- Recall= TP/(TP+FN) = 25%

Exercise U

Distribution of the instances for each class value is
A:10, B:40, C:25, and D:25.

Random classifier Y outputs A, B, C, and D, 50%, 30%, 10%, and 10% of the time, respectively.

Precision and Recall (wrt A)?

Exercise U - answer

Distribution of the instances for each class value is
A:10, B:40, C:25, and D:25.

Random classifier Y outputs A, B, C, and D, 50%, 30%, 10%, and 10% of the time, respectively.

Precision and Recall (wrt A)?

Answer:

- Y will say 50% of the time "A" and 50% of the time "not A".

TP = ? FP = ? FN = ?

- Precision= TP/(TP+FP) = 1/10= 10%
- Recall= TP/(TP+FN) = 1/2= 50%

Tuning hyperparameters: Validation Set
(no cross validation, splits fixed)

- Split into Train/Validation/Test (e.g. 70,10,20%)
- Train on Training data, use validation data to set hyperparams or choose model type
- Report final accuracy by training on all of training data (with your final chosen parameters) and predicting on test data.

- Key idea: data used to tune hyperparams should **never** be used to report accuracy

Tuning hyperparameters: Nested Cross Validation

- Partition into k_1 sets and repeat k_1 times:

  I. For each set of $(k_1-1)/k_1$ Train, $1/k_1$ Test (e.g. 90,10%)

  - For each hyperparameter setting h
    - Partition into k_2 sets and repeat k_2 times:

      i. Take your Train set from step I (e.g. 90% of all data) and further split into $(k_2-1)/k_2$ *sub-Train*, $1/k_2$ *sub-Test* (e.g. 0.9*0.9=81% of all data, 0.1*0.9=9% of all data)

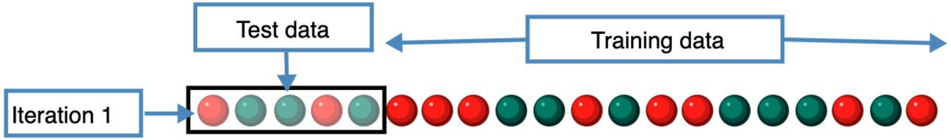
         i. Train on *sub-Train* from step i using hyperparams h

         ii. Test on *sub-Test* from step i

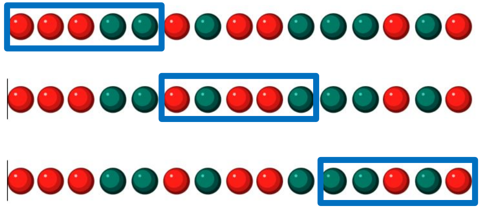
    - Calculate average performance across all k_2 splits for hyperparam h
  - Return hyperparam h' that maximizes performance

  III. Train on all Train data from step I using hyperparam h', test on Test data from step I. Record performance

- Report average performance across all k_1 folds of Train and Test

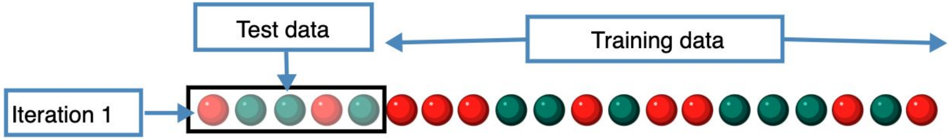


Run x-val on this train data, with each hyperparam



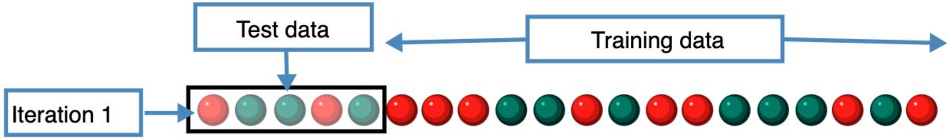
h1: 0.5, h2: 0.7, h3: 0.6, h4: 0.9   -> Best accuracy is with h4

Train on all of the training data using h4, test on test data, and save accuracy for this iteration

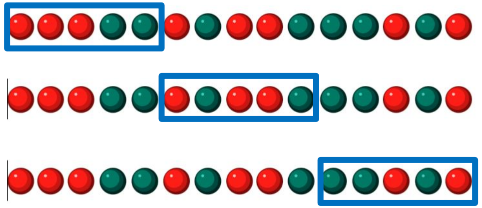


Repeat for all outer folds, then report average accuracy

Important: the data I use to choose a hyperparam is **not** used to calculate the **test** accuracy with that hyperparam in the **outer** loop!

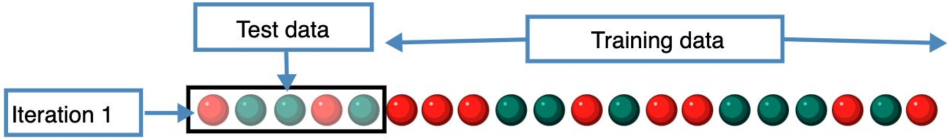


Run x-val on this train data, with each hyperparam



h1: 0.5, h2: 0.7, h3: 0.6, h4: 0.9   -> Best accuracy is with h4

Train on all of the training data using h4, test on test data, and save accuracy for this iteration



Repeat for all outer folds, then report average accuracy

ttps://commons.wikimedia.org/w/index.php?curid=82298768

Other Evaluation Methods

- Random subsampling / Monte Carlo cross validation
  - choose a test set randomly and repeatedly, without replacement
  - like cross-validation except test sets need not be disjoint
-

Other Evaluation Methods

- Random subsampling / Monte Carlo cross validation
  - choose a test set randomly and repeatedly, without replacement
  - like cross-validation except test sets need not be disjoint
- Bootstrap
  - choose a test set randomly with replacement
  - like random sampling, but with replacement
  - Pessimistic estimate, corrected with .632 bootstrap estimate

More info:
http://web.cs.iastate.edu/~jtian/cs573/Papers/Kohavi-IJCAI-95.pdf

- Also: https://mlfromscratch.com/nested-cross-validation-python-code/#/

Next…MLE and optimization