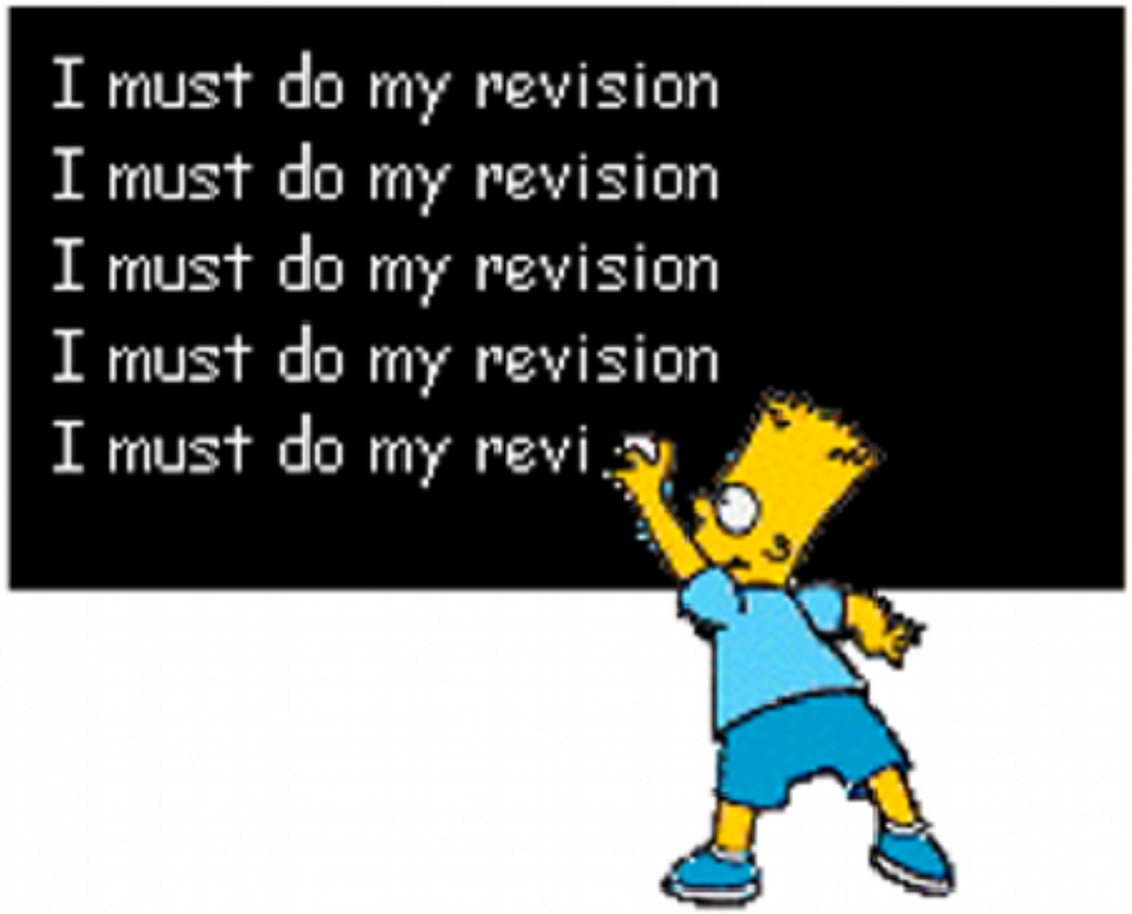
CO3091 - Computational Intelligence and Software Engineering

Lecture 28



Machine Learning Revision

Leandro L. Minku

Image from: http://www.literaturedelights.com/wp-content/uploads/2016/03/revision-simpsons.jpg.

Overview

- Algorithms:
  - k-NN
  - Decision trees
  - Naive bayes
  - Ensembles
  - Class imbalance learning
  - Continuous learning
  - Wrapper feature selection

- Software engineering problems:
  - Software effort estimation
  - Software defect prediction

Overview

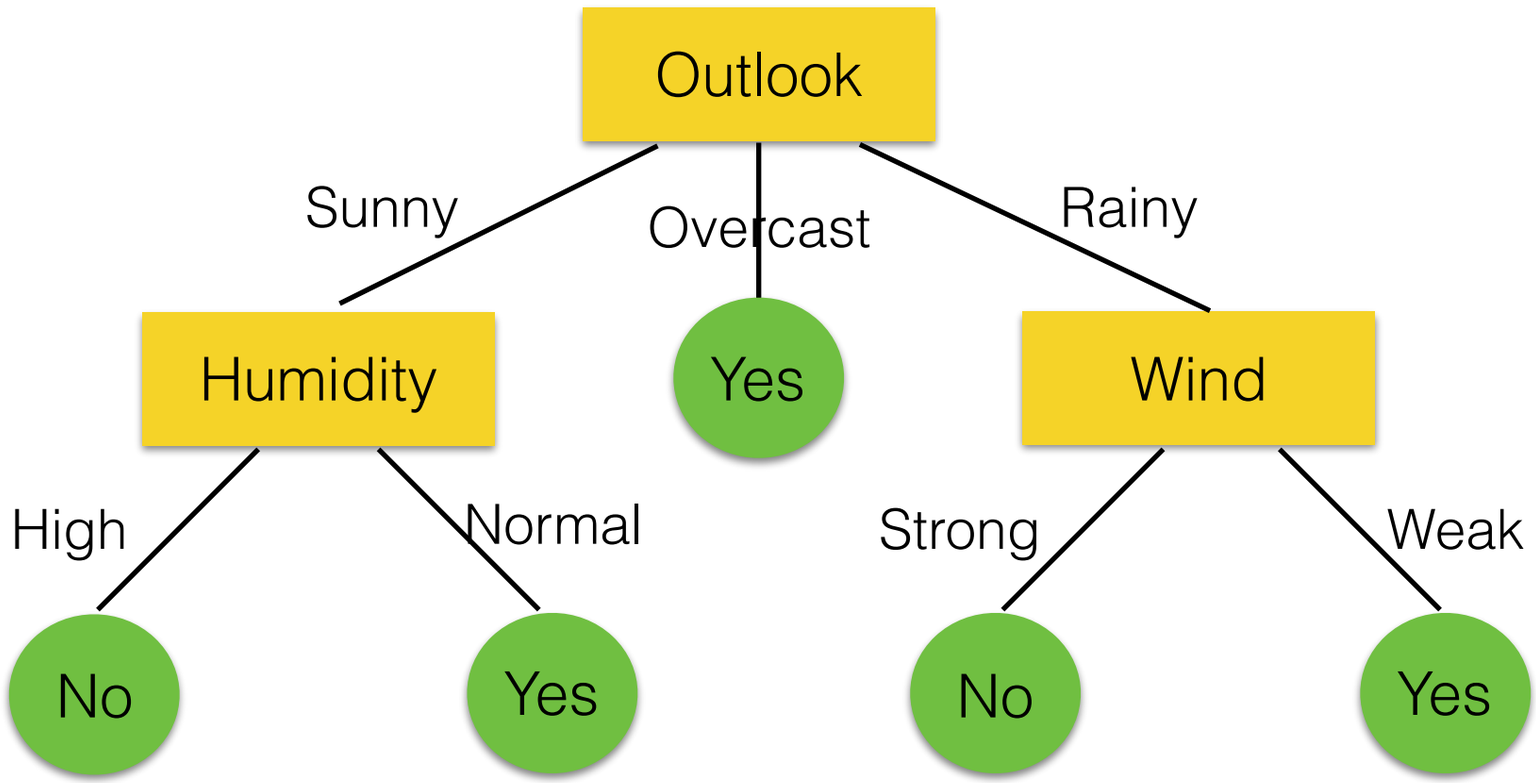
- Algorithms:
  - k-NN
  - **Decision trees**
  - **Naive bayes**
  - **Ensembles**
  - **Class imbalance learning**
  - **Continuous learning**
  - **Wrapper feature selection**

- Software engineering problems:
  - **Software effort estimation**
  - **Software defect prediction**

Optional Lab Session

- Individual emails to be sent this afternoon.

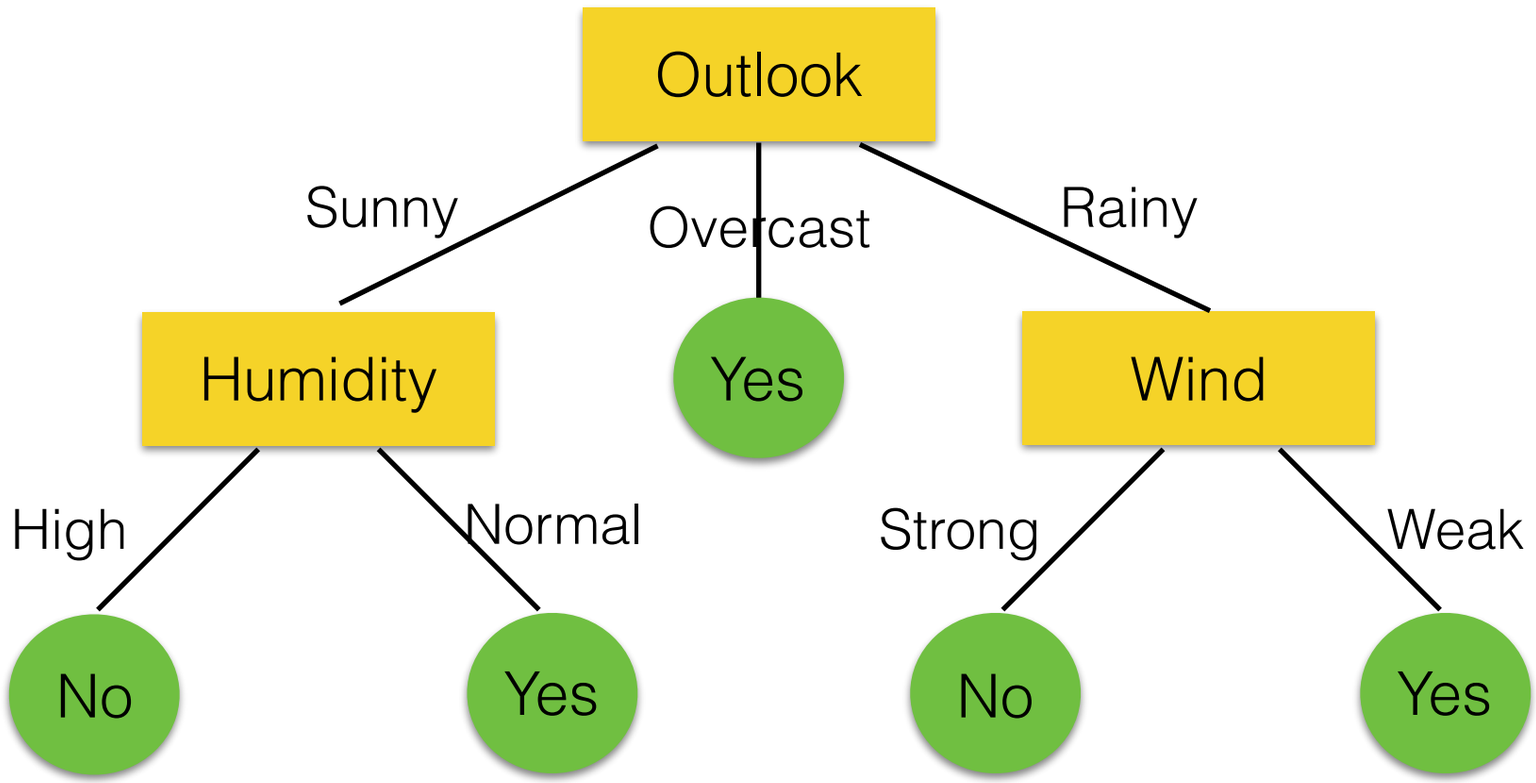
Decision Trees



What would be the prediction for an instance
[outlook=rainy, temperature=hot, humidity=normal, wind=strong, y=?]

Core idea — find the attributes whose splits will best separate the data into homogeneous sets.

Decision Trees

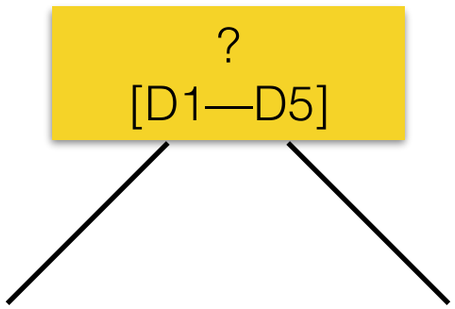


What would be the prediction for an instance
[outlook=rainy, temperature=hot, humidity=normal, wind=strong, y=?]

Core idea — find the attributes whose splits lead to he highest information gain or reduction in variance.

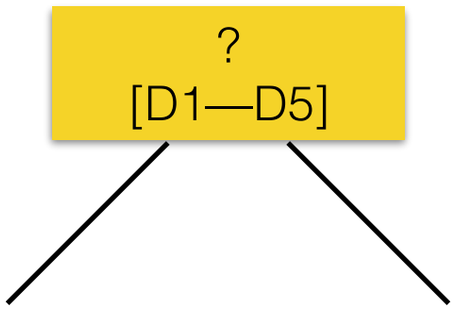
Choosing an Attribute to Split On — Information Gain or Variance Reduction?

Day	x_1 (Wind)	x_2 (Humidity)	y (Play)
D1	Strong	High	No
D2	Strong	High	No
D3	Weak	High	No
D4	Strong	Normal	Yes
D5	Strong	Normal	Yes



Choosing an Attribute to Split On — Information Gain

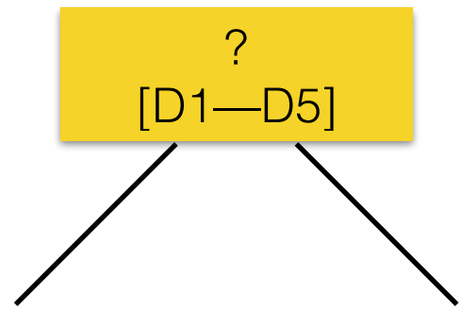
Day	x_1 (Wind)	x_2 (Humidity)	y (Play)
D1	Strong	High	No
D2	Strong	High	No
D3	Weak	High	No
D4	Strong	Normal	Yes
D5	Strong	Normal	Yes



1. Calculate information gain when splitting on Wind
2. Calculate information gain when splitting on Humidity
3. Choose attribute with the largest information gain

Numeric Attributes

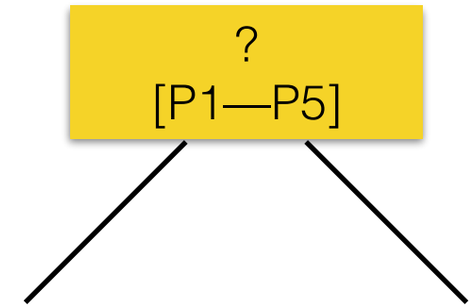
Day	x_1 (Wind)	x_2 (Humidity)	y (Play)
D1	Strong	100	No
D2	Strong	90	No
D3	Weak	70	No
D4	Strong	40	Yes
D5	Strong	20	Yes



1. Calculate information gain when splitting on Wind
2. Calculate information gain when splitting on Humidity and threshold 95
3. Calculate information gain when splitting on Humidity and threshold 80
4. Calculate information gain when splitting on Humidity and threshold 55
5. Calculate information gain when splitting on Humidity and threshold 30
6. Choose attribute (+threshold) with the largest information gain

Choosing an Attribute to Split On — Reduction in Variance

Project	x_1 (Size)	x_2 (Team Expertise)	y (Effort)
P1	Small	High	1
P2	Small	High	2
P3	Medium	High	3
P4	Medium	Normal	4
P5	Large	Normal	10



1. Calculate reduction in variance when splitting on Size
2. Calculate reduction in variance when splitting on Team Expertise
3. Choose attribute with the largest reduction in variance

Information Gain and Reduction in Variance

$$\text{InfoGain}(\text{examples},A) = \text{Entropy}(\text{examples}) - \sum_{v_i \in \text{Values}(A)} \frac{|\text{examples}_{vi}|}{|\text{examples}|} \text{Entropy}(\text{examples}_{vi})$$

$$\text{VarRed}(\text{examples},A) = \text{Variance}(\text{examples}) - \sum_{v_i \in \text{Values}(A)} \frac{|\text{examples}_{vi}|}{|\text{examples}|} \text{Variance}(\text{examples}_{vi})$$

Entropy and variance are measures of how heterogeneous a set of examples are in terms of their output values.

Naive Bayes Predictive Model

Training Set

Person	x_1 (Flowers)	x_2 (Hair)	y (Gender)
P1	Likes	Long	Female
P2	Likes	Long	Female
P3	!Like	Long	Female
P4	!Like	Short	Male
P5	!Like	Short	Male

Model

Frequency Table for Flowers	Gender = Female	Gender = Male	Total:
Likes	2	0	2
!Like	1	2	3
Total:	3	2	5

Frequency Table for Hair	Gender = Female	Gender = Male	Total:
Long	3	0	3
Short	0	2	2
Total:	3	2	5

Making Predictions

Model

Frequency Table for Flowers	Gender = Female	Gender = Male	Total:
Likes	2	0	2
!Like	1	2	3
Total:	3	2	5

Frequency Table for Hair	Gender = Female	Gender = Male	Total:
Long	3	0	3
Short	0	2	2
Total:	3	2	5

$$P(C|F_1,\ldots,F_n) = P(C) \prod_{i=1}^{n} P(F_i|C)$$

Calculate $P(C|F_1,\ldots,F_n)$ for each class.

E.g., (Flowers=!Like, Hair=Short, Gender=?)

P(Gender=Male | Flowers=!Like, Hair=Short)

P(Gender=Female | Flowers=!Like, Hair=Short)

Predict the class with the highest $P(C|F_1,\ldots,F_n)$.

Problem: P(Hair=Short | Gender = Female) = 0/3

Laplace Smoothing

Training Set

Person	x_1 (Flowers)	x_2 (Hair)	y (Gender)
P1	Likes	Long	Female
P2	Likes	Long	Female
P3	!Like	Long	Female
P4	!Like	Short	Male
P5	!Like	Short	Male

Laplace smoothing is used to compute $p(F_i|C)$.

Model

Frequency Table for Flowers	Gender = Female	Gender = Male	Total:
Likes	2+1	0+1	4
!Like	1+1	2+1	5
Total:	5	4	9

Frequency Table for Hair	Gender = Female	Gender = Male	Total:
Long	3+1	0+1	5
Short	0+1	2+1	4
Total:	5	4	9

Naive Bayes Model — Categorical + Numerical Input Attributes

Training Set

Person	x_1 (Flowers)	x_2 (Height)	y (Gender)
P1	Likes	1.65	Female
P2	Likes	1.85	Female
P3	!Like	1.70	Female
P4	!Like	1.70	Male
P5	!Like	1.90	Male
P6	Like	1.95	Male

Probability density function is used to provide $p(F_i|C)$.

Model

Frequency Table for Flowers	Gender = Female	Gender = Male	Total:
Likes	2+1	1+1	5
!Like	1+1	2+1	5
Total:	5	5	10

Parameters Table for Height	Gender = Female	Gender = Male
μ	1.73	1.85
δ^2	0.00723	0.01167

Ensembles of Learning Machines

Ensembles of learning machines combine base models so that:

- correct predictions of some models compensate the wrong predictions of others, or
- underestimations compensate for overestimations and vice-versa.

For them to work well we need diversity!

Class Imbalance

Class imbalance occurs when at least one class is a minority in comparison to at least one other class.

We can use strategies such as oversampling and undersampling to help.

Original training set

Module id	x1 = branch count	x2 = LOC	x3 = halstead	…	y = defective?
1	18	1000	1	…	No
2	30	900	10	…	Yes
3	20	5000	3	…	Yes
4	25	100	3	…	No
5	50	500	4	…	No
6	4	30	3	…	No
7	25	2000	2	…	No
8	28	3000	5	…	No
9	13	1000	10	…	No
10	25	500	12	…	No

Continuous Learning Algorithms

- Online machine learning algorithms:
  - Update predictive models whenever a new training example is received.
    - Can learn additional data.
    - Can be faster than offline algorithms.
- Strategies to deal with changes:
  - Increase in error can be used as an indicator of change.
  - Ensembles can add new models to learn new situations.
  - Ensembles can keep old models to preserve potentially useful old knowledge.

Wrapper Feature Selection

Curse of dimensionality

We can use genetic algorithms for feature selection.

Applying Machine Learning to a New Problem

- Talk with the experts to determine what the problem is and what data are available that may be relevant.
- Choose a feature selection method.
- Based on the data / problem, select a set of machine learning approaches to investigate.
  - Certain pre-processing of the data may be necessary, e.g., normalisation.
  - Use a validation procedure to select an approach, parameter values [and possibly input features].
  - Test the selected approach and present results to clients.

Applying Machine Learning to an Instance of a Known Problem

- Research existing literature to check which input attributes and approaches are most likely to perform well.
- Still perform some experiments for your problem instance, as results can vary across problem instances.

Software Effort Estimation

- Estimation of the effort required to develop a software project.
  - Effort is measured in person-hours, person-months, etc.

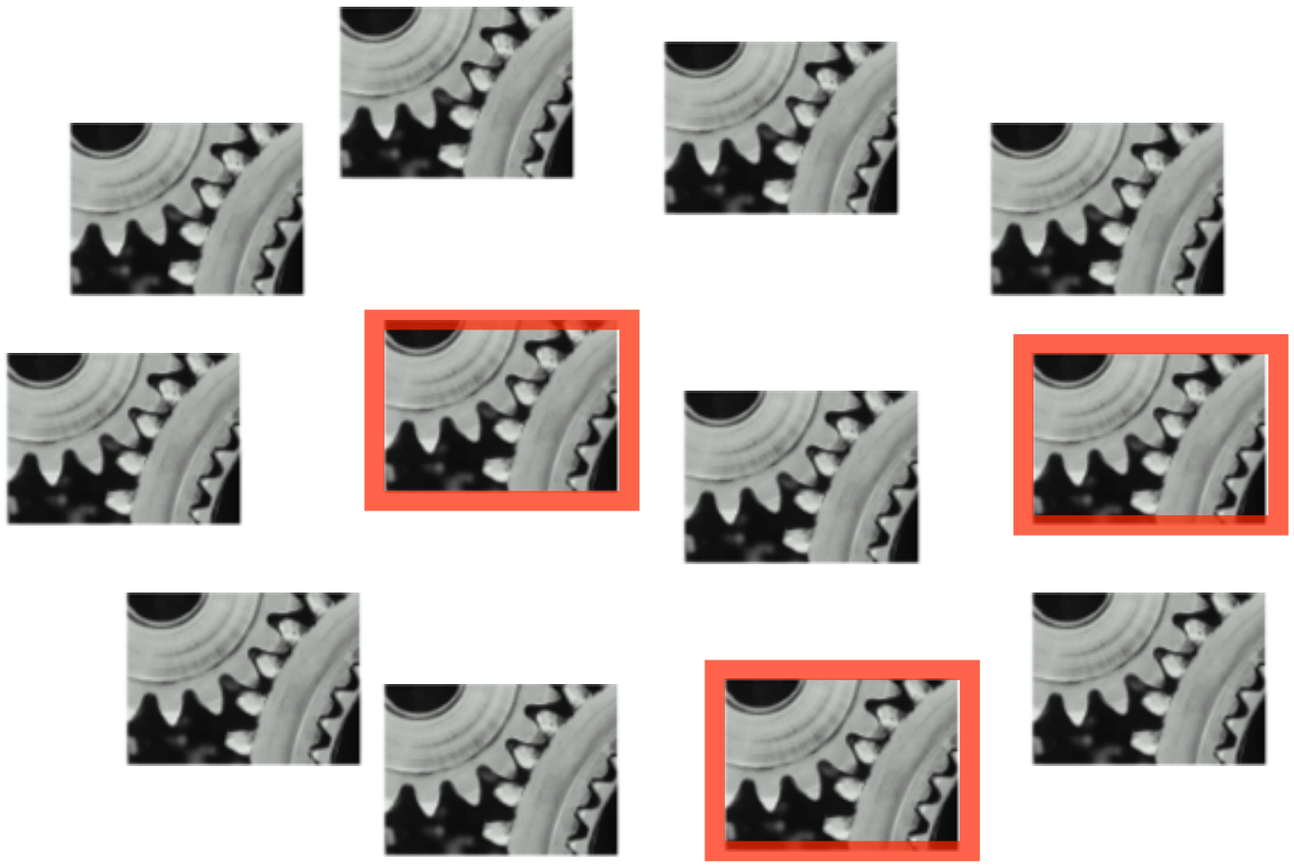
- Based on features such as programming language, team expertise, estimated size, development type, required reliability, etc.

x_1 = programming language	x_2 = team expertise	x_3 = estimated size	…	y = required effort
Java	low	1000	…	10 p-month
C++	medium	2000	…	20 p-month
Java	high	2000	…	8 p-month
…	…	…	…	…

- K-NN, decision trees, bagging of decision trees.

Software Defect Prediction

Prediction of which components are more likely to be defective (buggy), so that we can increase testing cost-effectiveness.



Software Defect Prediction

When we find bugs, we hire them to work for us!

Module id	x1 = branch count	x2 = LOC	x3 = halstead	…	y = defective ?
1	18	1000	1	…	No
2	30	900	10	…	Yes
3	20	5000	3	…	Yes
4	25	100	3	…	No
5	50	500	4	…	No
6	4	30	3	…	No
7	25	2000	2	…	No
8	28	3000	5	…	No
9	13	1000	10	…	No
10	25	500	12	…	No

Naive bayes + class imbalance techniques.

Overview

- Algorithms:
  - k-NN
  - **Decision trees**
  - **Naive bayes**
  - **Ensembles**
  - **Class imbalance learning**
  - **Continuous learning**
  - **Wrapper feature selection**

- Software engineering problems:
  - **Software effort estimation**
  - **Software defect prediction**