

CSCE 580: Introduction to AI

Lecture 13: Machine Learning - Classification

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

1ST OCT 2024

Carolinian Creed: “I will practice personal and academic integrity.”

Credits: Copyrights of all material reused acknowledged

Organization of Lecture 13

- Introduction Segment
 - Recap of Lecture 12
- Main Segment
 - The variety of methods for classification
 - Logistic Regression
 - Decision trees
 - Random forest
 - Naïve Bayes
 - Boosting
 - Metrics – AUC / ROC
 - Discussion: Choosing a method that works
- Concluding Segment
 - Course Project Discussion
 - About Next Lecture – Lecture 14
 - Ask me anything

Introduction Section

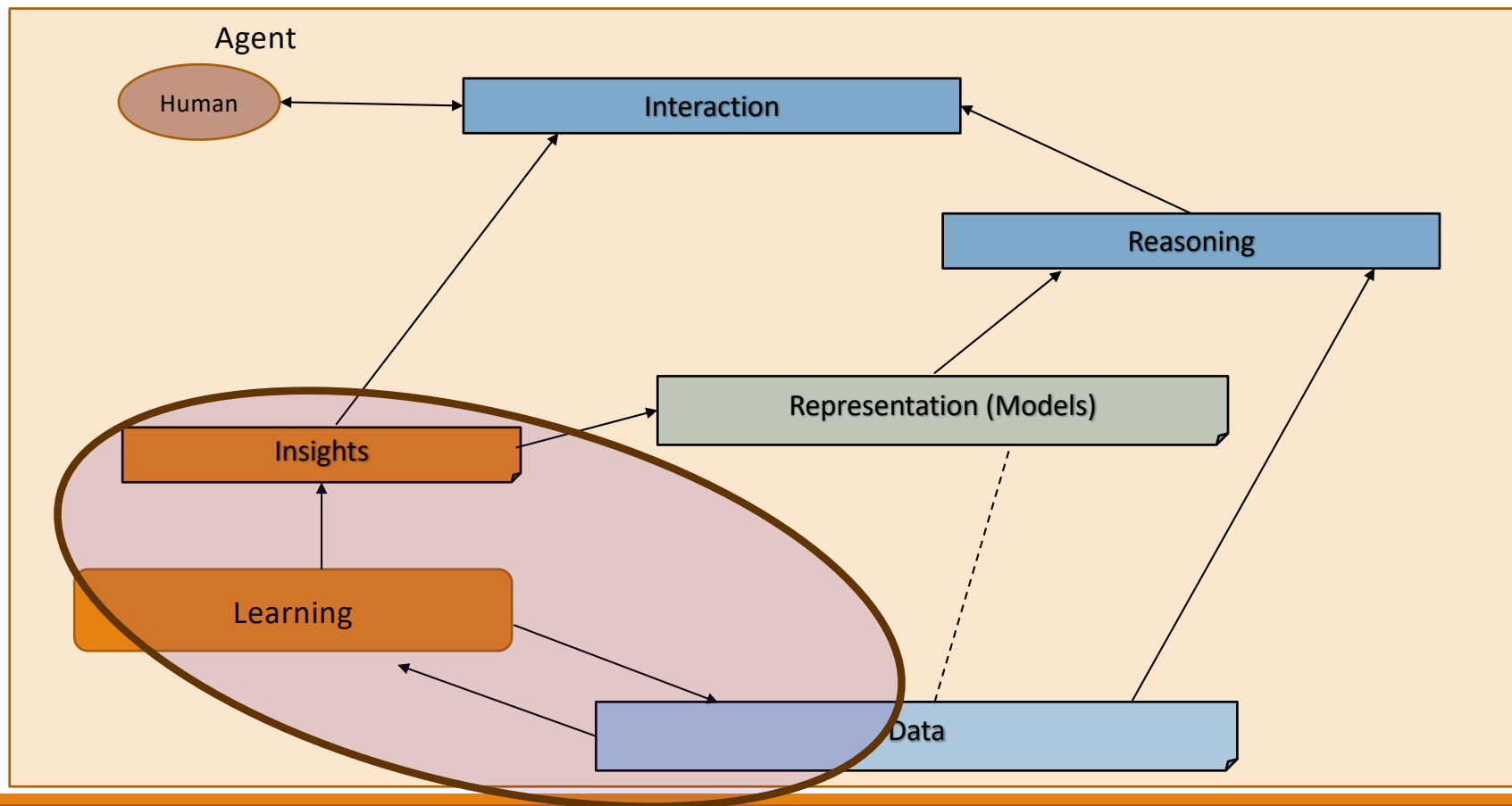
Recap of Lecture 12

- Problem Settings
- Data preparation and feature engineering
- Solving classification problems

Intelligent Agent Model



Relationship Between Main AI Topics



Where We Are in the Course

CSCE 580/ 581 – In This Course

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 4-5: Search, Heuristics - Decision Making
- Week 6: Constraints, Optimization – Decision Making
- Week 7: Classical Machine Learning – Decision Making, Explanation

• Week 8: Machine Learning - Classification

• Week 9: Machine Learning - Classification – Trust Issues and

Mitigation Methods

• Topic 10: Learning neural network, deep learning, Adversarial attacks

• Week 11: Large Language Models – Representation, Issues

• Topic 12: Markov Decision Processes, Hidden Markov models -

Decision making

• Topic 13: Planning, Reinforcement Learning – Sequential decision making

• Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Main Section



Credit: Retrieved from internet

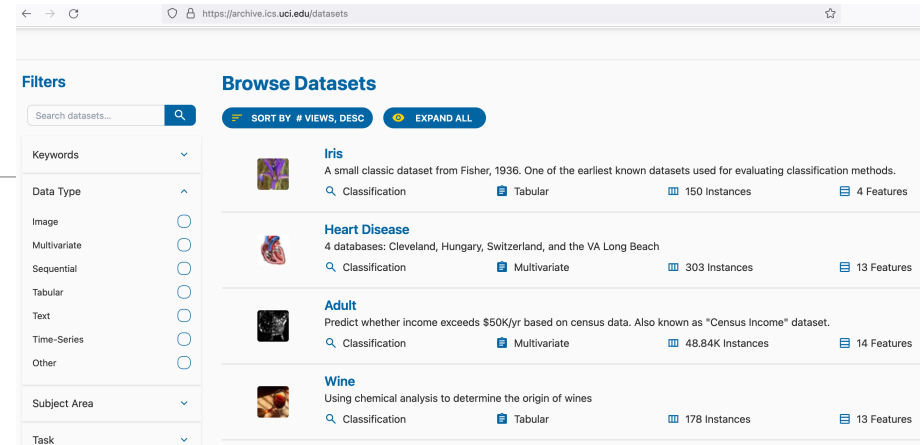
Machine Learning – Insights from Data

- Descriptive analysis
 - Describe a past phenomenon
 - **Methods:** classification (feedback from label), clustering, dimensionality reduction, anomaly detection, neural methods, reinforcement learning (feedback from hint/ reward)
- Predictive analysis
 - Predict about a new situation
 - **Methods:** time-series, neural networks
- Prescriptive analysis
 - What an agent should do
 - **Methods:** simulation, reinforcement learning, reasoning
- New areas
 - Counterfactual analysis
 - Causal Inferencing
 - Scenario planning

Reference and Demo

- Data: UCI Datasets

- <https://archive.ics.uci.edu/datasets>
- Browse or search



Weka 3: Machine Learning Software in Java

Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization.

Found only on the islands of New Zealand, the Weka is a flightless bird with an inquisitive nature. The name is pronounced like this, and the bird sounds like this.

Weka is open source software issued under the GNU General Public License.

We have put together several free online courses that teach machine learning and data mining using Weka. The videos for the courses are available on Youtube.

Weka supports deep learning!

Getting started	Further information	Developers
<ul style="list-style-type: none">• Requirements• Download• Documentation• FAQ• Getting Help	<ul style="list-style-type: none">• Citing Weka• Datasets• Related Projects• Miscellaneous Code• Other Literature	<ul style="list-style-type: none">• Development• History• Subversion• Contributors• Commercial licenses

- Tools:

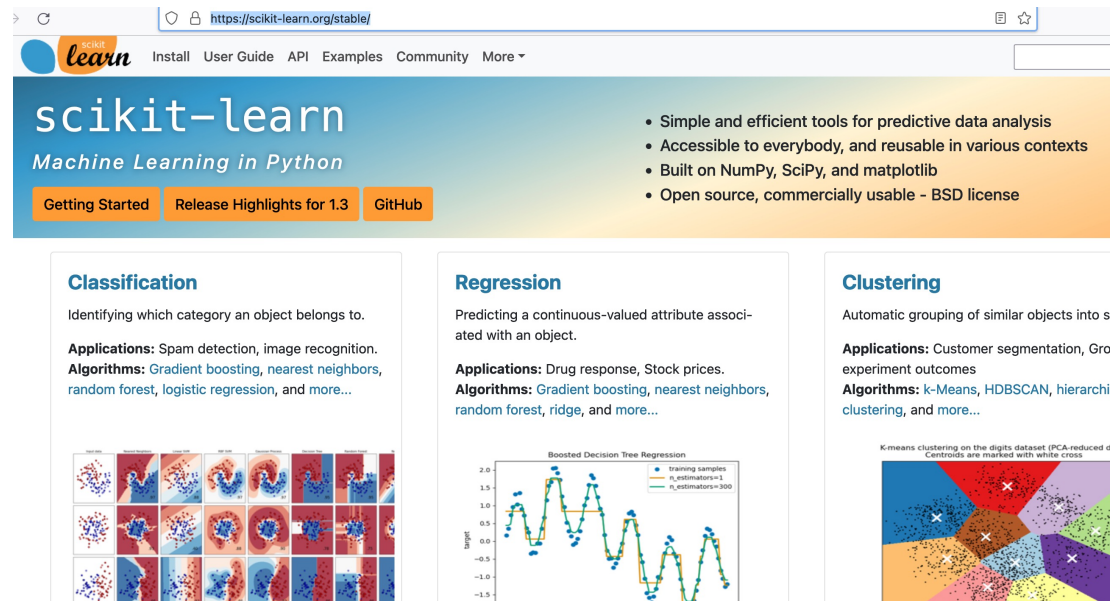
- Weka - <https://www.cs.waikato.ac.nz/ml/weka/>
- Download tool and dataset

- Libraries

- Scikit - <https://scikit-learn.org/stable/>

Reference and Demo

- Data: UCI Datasets
 - <https://archive.ics.uci.edu/datasets>
 - Browse or search
- Tools:
 - Weka - <https://www.cs.waikato.ac.nz/ml/weka/>
 - Download tool and dataset
- Libraries
 - Scikit - <https://scikit-learn.org/stable/>



The screenshot shows the scikit-learn website at <https://scikit-learn.org/stable/>. The page features a navigation bar with links for Install, User Guide, API, Examples, Community, and More. The main header includes the scikit-learn logo and the tagline "Machine Learning in Python". Below the header, there are three buttons: "Getting Started", "Release Highlights for 1.3", and "GitHub". To the right, a list of features highlights the library's simplicity, accessibility, built-in dependencies (NumPy, SciPy, matplotlib), and open-source nature (BSD license).

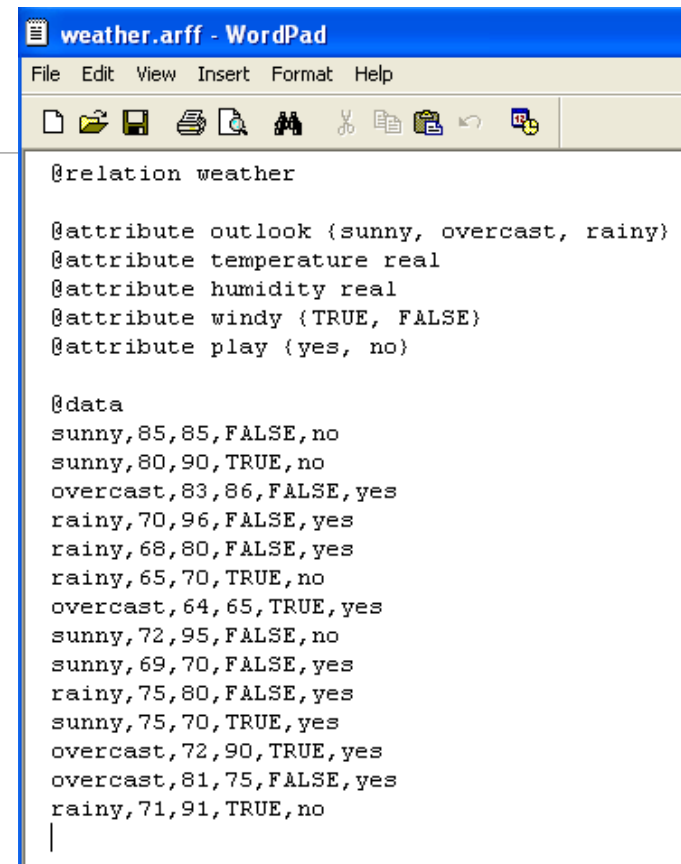
The page is divided into three main sections:

- Classification**: Identifying which category an object belongs to. Applications include spam detection and image recognition. Algorithms listed are Gradient boosting, nearest neighbors, random forest, logistic regression, and more.
- Regression**: Predicting a continuous-valued attribute associated with an object. Applications include drug response and stock prices. Algorithms listed are Gradient boosting, nearest neighbors, random forest, ridge, and more.
- Clustering**: Automatic grouping of similar objects into groups. Applications include customer segmentation and Gro experiment outcomes. Algorithms listed are k-Means, HDBSCAN, hierarchical clustering, and more.

Exercise: Run Weka

ARFF Data Format

- Attribute-Relation File Format
- Header – describing the attribute types
- Data – (instances, examples) comma-separated list



```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
|
```

Slide Courtesy: <http://www.cs.iastate.edu/~cs573x/bbsilab.html>

Exercise: Run Kaggle Example

Weather classification:

<https://www.kaggle.com/code/ihabsherbiny/weather-classification-with-3-models>

```
import data
Exploratory Data Analysis (EDA)
target column
Split dataframe into X and y
split dataframe into train and test
Random Forest
SVM
Decision Tree Classifier
Models scores
```

Exercise: Understand German Credit

- Check in UCI
- Look at variants
 - <https://archive.ics.uci.edu/dataset/573/south+german+credit+update>

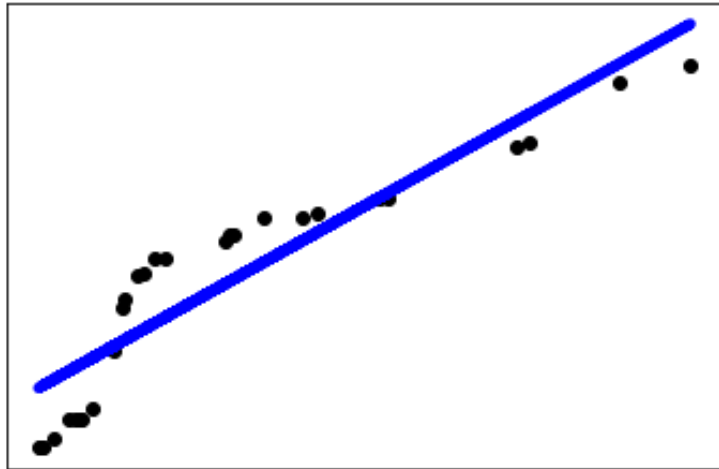
Classifier Method Types

- Individual methods
 - Logistic Regression
 - Decision Tree
 - Naïve Bayes
- Ensemble
 - Bagging: Aggregate classifiers (“bootstrap aggregation” => bagging)
 - Random Forest
 - Samples are chosen with replacement (bootstrapping), and combined (aggregated) by taking their average
 - Gradient Boosting: aggregate to turn weak learners into strong learners
 - Boosters (aggregators) turn weak learners into strong learners by focusing on where the individual weak models (decision trees, linear regressors) went wrong
 - Gradient Boosting

Source:

- Data Mining: Concepts and Techniques, by Jiawei Han and Micheline Kamber
- <https://towardsdatascience.com/getting-started-with-xgboost-in-scikit-learn-f69f5f470a97>

Linear Regression



Notebook: <https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l6-l7-l8-supervised-ml/Supervised-Regression.ipynb>

Logistic Regression

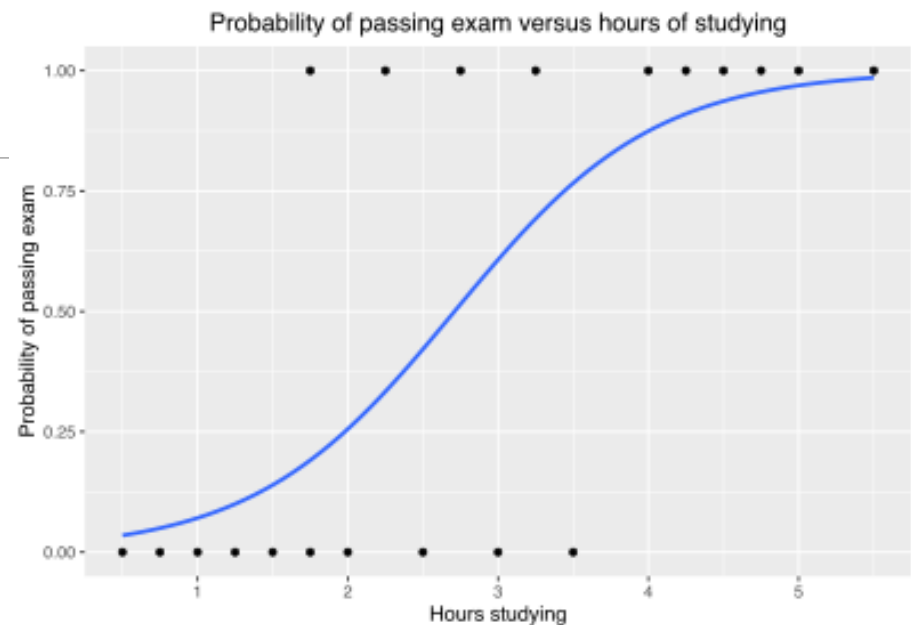
- Classification setting – labels are discrete
 - the **logistic model** (or **logit model**) is a [statistical model](#) that models the [probability](#) of an event taking place by having the [log-odds](#) for the event be a [linear combination](#) of one or more [independent variables](#)

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

where μ is a [location parameter](#) (the midpoint of the curve) and s is a [scale parameter](#).

•Sources:

- <https://www.ibm.com/topics/logistic-regression>
- <https://developers.google.com/machine-learning/crash-course/logistic-regression/>
- https://en.wikipedia.org/wiki/Logistic_regression
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html



$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Source: Wikipedia

Decision Tree

Problem: Classify Weather Data

Input

Outlook	Temperature	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
...

Class Label

Output
(Informal)

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

Which Variable to Learn to Create Rules On?

- What do we want?
 - Compact model (e.g., set of rules)
 - High accuracy / low error
- Find the most discriminating variable
 - But how do we measure this
- Corner cases
 - If all the samples are the same, the decision tree is a ?
 - Leaf node with the only class
 - If there are no attributes in the dataset, the decision tree is?
 - A node with most common class

Outlook	Temperature	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
...

Expected Information/ Entropy

- Concept: Expected Information

- Let

- Class label has m distinct values (i.e., m distinct classes)
 - s_i be the number of samples of S of Class C_i ($i = 1 \dots m$)

- $I(s_1, s_2, \dots, s_m) = - \sum_{i=1 \text{ to } m} p_i \log_2(p_i)$

- Where P_i is the probability a sample belongs to class C_i ; estimated by (s_i / s)

Outlook	Temperature	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
...

- Entropy / Expected Information after partitioning on Attribute A which has v distinct values

- $E(A) = \sum_{j=1 \text{ to } v} (s_{1j} + \dots + s_{mj}) / S * (I(s_{1j}, s_{2j}, \dots, s_{mj}))$

- s_{ij} be the number of samples in S_j of Class C_i ($i = 1 \dots m$)

- Smaller the entropy, the greater the purity of the subset partitions

Information Gain

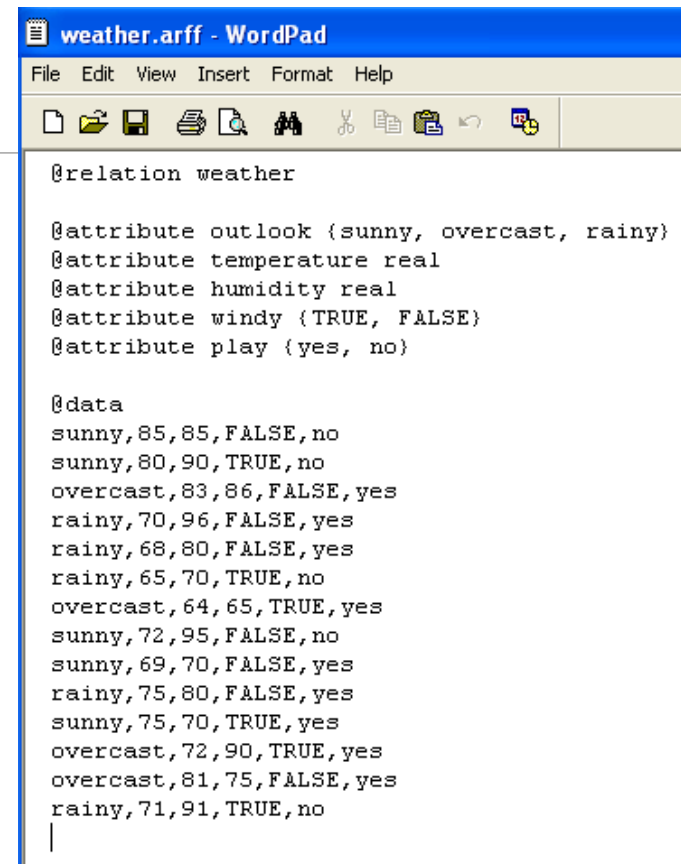
Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
...

- Entropy / Expected Information after partitioning on Attribute A which has v distinct values
 - $E(A) = \sum_{j=1 \text{ to } v} (s_{1j} + \dots + s_{mj}) / S * (I(s_{1j}, s_{2j}, \dots, s_{mj}))$
 - s_{ij} be the number of samples in S_j of Class C_i ($i = 1 \dots m$)
- After partition, S_j
 - $I(s_{1j}, s_{2j}, \dots, s_{mj}) = - \sum_{i=1 \text{ to } m} p_{ij} \log_2(p_{ij})$
 - Where p_{ij} is the probability a sample in S_j belongs to class C_i ; estimated by $(s_{ij} / |s_j|)$
- Gain (A) = $I(s_1, s_2, \dots, s_m) - E(A)$
 - Is the expected reduction in entropy by knowing the value of Attribute A
- **Method:** Split on the attribute which leads to the highest information gain

Weka Exercise

ARFF Data Format

- Data is in ARFF in UCI dataset
- Or Convert
 - File system, CSV → ARFF format
 - Use [C45Loader](#) and [CSVLoader](#) to convert



```
@relation weather

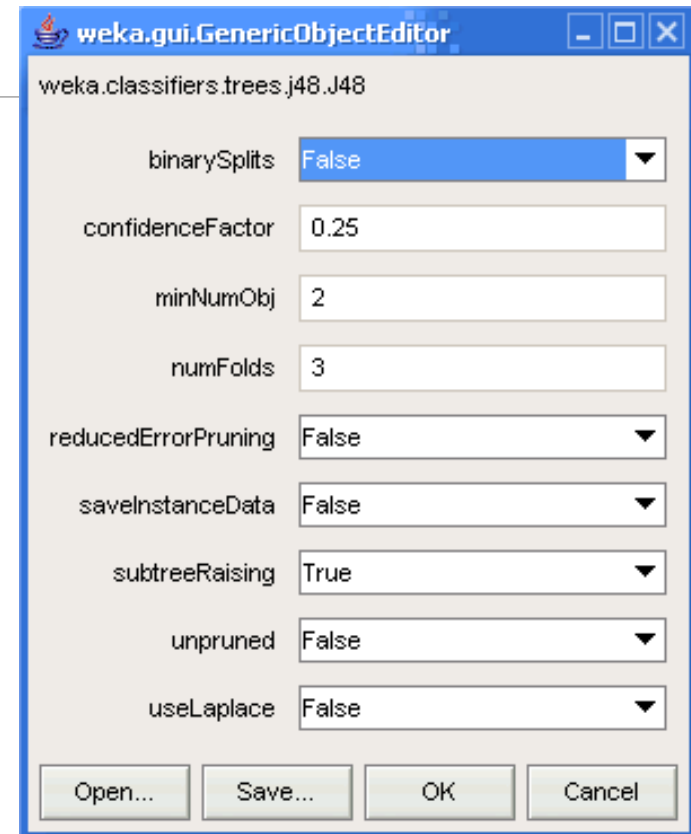
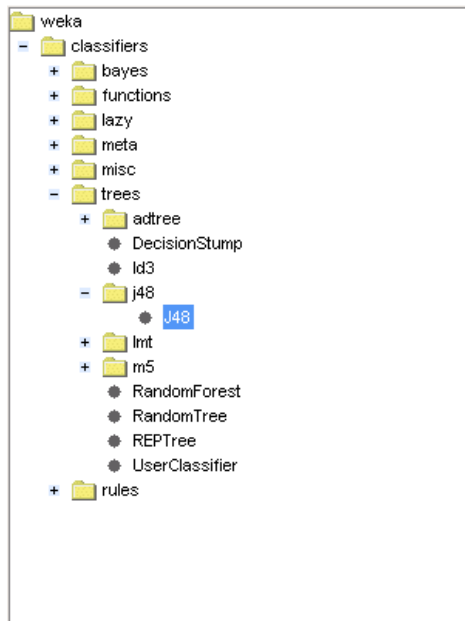
@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
|
```

Slide Courtesy: <http://www.cs.iastate.edu/~cs573x/bbsilab.html>

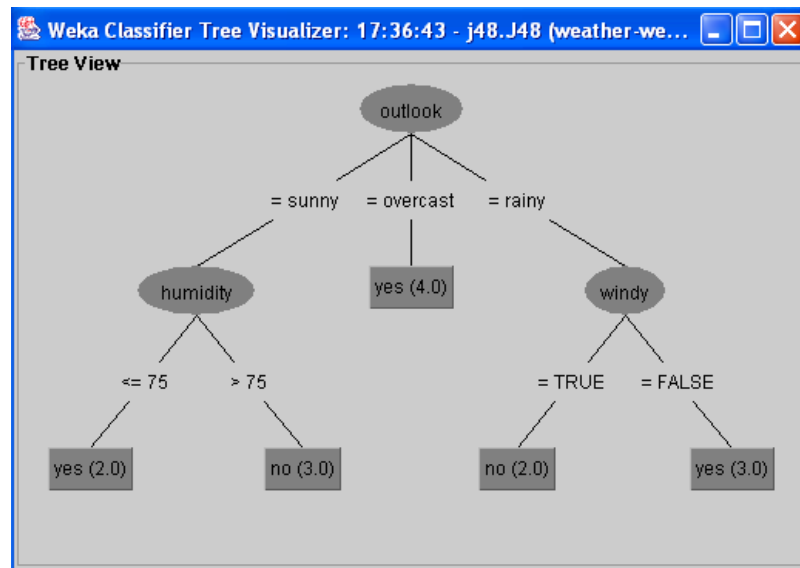
Weka: weka.classifiers.trees.J48

Class for generating an unpruned or a pruned C4.5 decision tree.



Slide Courtesy: <http://www.cs.iastate.edu/~cs573x/bbsilab.html>

Understanding Output



Slide Courtesy: <http://www.cs.iastate.edu/~cs573x/bbsilab.html>

Weka: Decision Tree Output

J48 pruned tree

outlook = sunny

| humidity = high: no (3.0)

| humidity = normal: yes (2.0)

outlook = overcast: yes (4.0)

outlook = rainy

| windy = TRUE: no (2.0)

| windy = FALSE: yes (3.0)

Number of Leaves : 5

Size of the tree : 8

=== Summary ===

Correctly Classified Instances	7	50	%
Incorrectly Classified Instances	7	50	%
Kappa statistic	-0.0426		
Mean absolute error	0.4167		
Root mean squared error	0.5984		
Relative absolute error	87.5	%	
Root relative squared error	121.2987	%	
Total Number of Instances	14		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.556	0.6	0.625	0.556	0.588	0.633	yes
	0.4	0.444	0.333	0.4	0.364	0.633	no
Weighted Avg.	0.5	0.544	0.521	0.5	0.508	0.633	

=== Confusion Matrix ===

a b <-- classified as
5 4 | a = yes
3 2 | b = no

Test Options

- Percentage Split (2/3 Training; 1/3 Testing)
- Cross-validation
 - Estimating the generalization error based on resampling when limited data
 - averaged error estimate.
 - Cross-fold validation (10-fold)
 - Leave-one-out (Loo)
 - Stratified

Random Forest

- An ensemble method
- Credits
 - Ideas introduced by Tin Kam Ho in 1995, https://en.wikipedia.org/wiki/Tin_Kam_Ho
 - Matured by Leo Breiman and Adele Cutler at Berkeley (https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#intro)
 - History: Khaled Fawagreh, Mohamed Medhat Gaber & Eyad Elyan (2014) Random forests: from early developments to recent advancements, Systems Science & Control Engineering, 2:1, 602-609, DOI: [10.1080/21642583.2014.956265](https://doi.org/10.1080/21642583.2014.956265)
- Main steps (Input: data, N= number of trees)
 - If the number of cases in the training set is N, sample N cases at random - but *with replacement*, from the original data. This sample will be the training set for growing the tree.
 - If there are M input variables, a number $m \ll M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
 - Each tree is grown to the largest extent possible. There is no pruning.

Slide Courtesy: Leo Breiman and Adele Cutler website

Random Forest in Action

- Code examples:
 - <https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l6-l7-supervised-ml/Supervised-RandomForest-Classification.ipynb>
- Scikit Library: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

Activity: Try Weka and Classifiers

- Naïve Bayes Method
- Gradient Tree Boosting
- Neural Network – MLP

Naïve Bayes Classifier

Notation:

Class variable y and
dependent feature vector x_1 through x_n

Using the naive conditional independence assumption that

$$P(y | x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)}$$

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),$$

for all i , this relationship is simplified to

$$P(y | x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1, \dots, x_n)}$$

Bayes assumption: given the
value of the class variable,
every pair of features are
conditionally independent

Since $P(x_1, \dots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y | x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i | y)$$

↓

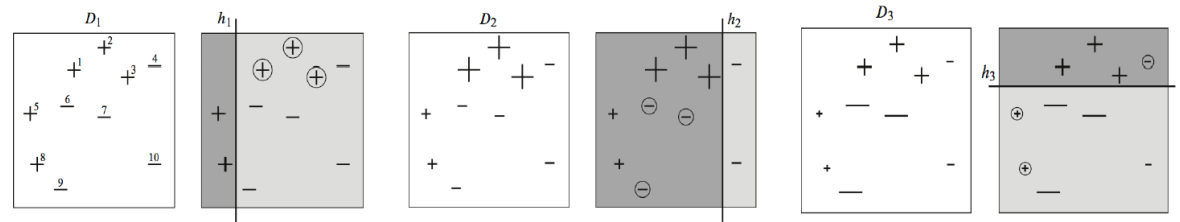
$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i | y),$$

Source: https://scikit-learn.org/stable/modules/naive_bayes.html

Boosting Methods

- Concepts

- **Weak learner:** a classifier that is only slightly correlated with the true classification
 - label examples better than random guessing
- **Strong learner:** a classifier that is (arbitrarily) well-correlated with the true classification.



- Boosting

- “Convert weak learners to strong learners”
- Adapt[at]ive Resampling and Combining algorithm

Figure: AdaBoost. Source: Figure 1.1 of [Schapire and Freund, 2012]

Source: [https://en.wikipedia.org/wiki/Boosting_\(machine_learning\)](https://en.wikipedia.org/wiki/Boosting_(machine_learning))

Image Courtesy: Prof. Cheng Li

Boosting Methods

Gradient Boosting = Gradient Descent + Boosting

Adaboost

$$H(x) = \sum_t \rho_t h_t(x)$$

- ▶ Fit an additive model (ensemble) $\sum_t \rho_t h_t(x)$ in a forward stage-wise manner.
- ▶ In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
- ▶ In Adaboost, “shortcomings” are identified by high-weight data points.

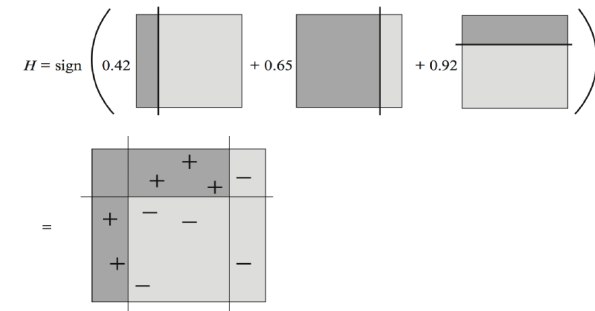


Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

Content and Image Courtesy: Prof. Cheng Li

https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

Boosting Methods

Gradient Boosting = Gradient Descent + Boosting

Adaboost

AdaBoost,

Illustration: for binary classification, images

1. Form a large set of simple features
2. Initialize weights for training images
3. For T rounds
 1. Normalize the weights
 2. For available features from the set, train a classifier using a single feature and evaluate the training error
 3. Choose the classifier with the lowest error
 4. Update the weights of the training images: increase if classified wrongly by this classifier, decrease if correctly
4. Form the final strong classifier as the linear combination of the T classifiers (coefficient larger if training error is small)

Source: [https://en.wikipedia.org/wiki/Boosting_\(machine_learning\)](https://en.wikipedia.org/wiki/Boosting_(machine_learning))

$$H(x) = \sum_t \rho_t h_t(x)$$

$$H = \text{sign} \left(0.42 \begin{array}{|c|c|} \hline \text{dark} & \text{light} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{dark} & \text{light} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{dark} & \text{light} \\ \hline \end{array} \right)$$

$$= \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

Image Courtesy: Prof. Cheng Li

https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

Boosting Methods

Gradient Boosting = Gradient Descent + Boosting
Adaboost

Gradient Boosting = Gradient Descent + Boosting
Gradient Boosting

- ▶ Fit an additive model (ensemble) $\sum_t \rho_t h_t(x)$ in a forward stage-wise manner.
- ▶ In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
- ▶ In Gradient Boosting, “shortcomings” are identified by gradients.
- ▶ Recall that, in Adaboost, “shortcomings” are identified by high-weight data points.
- ▶ Both high-weight data points and gradients tell us how to improve our model.

$$H(x) = \sum_t \rho_t h_t(x)$$

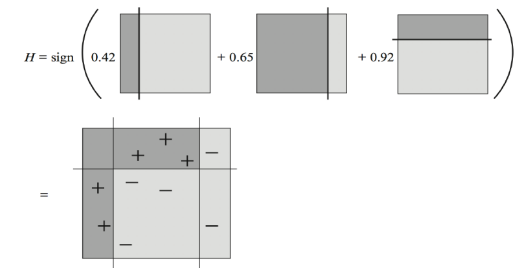


Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

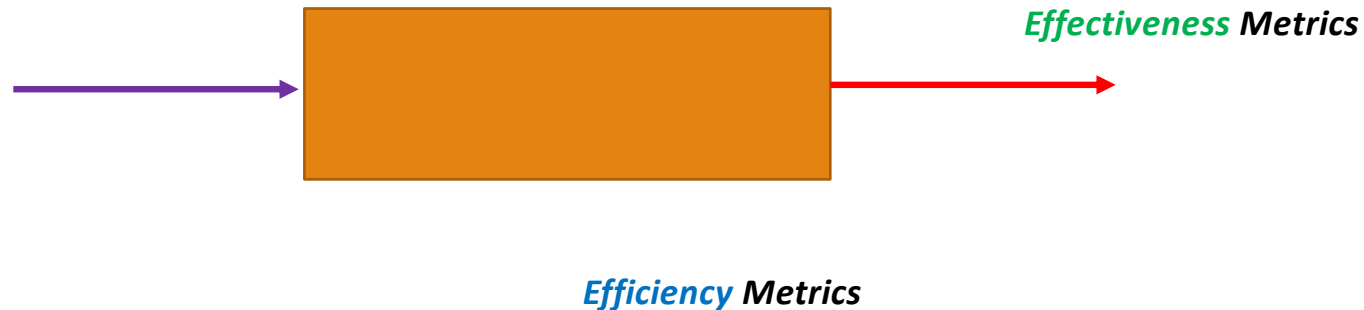
Content and Image Courtesy: Prof. Cheng Li
https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

NBC and Boosting in Action

- Code examples:
 - <https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l6-l7-l8-supervised-ml/Supervised-NaiveBayes-GradientBoost-NN-Classification.ipynb>
- Scikit Library:
 - NBC: https://scikit-learn.org/stable/modules/naive_bayes.html
 - GradientBoost: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>

Metric Types

- **Effectiveness**: what the user of a system sees, primarily cares about
- **Efficiency**: what the executor in a system sees, primarily cares about



Metrics: Accuracy, Precision, Recall

Actual Class	Predicted class		
		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

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

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

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

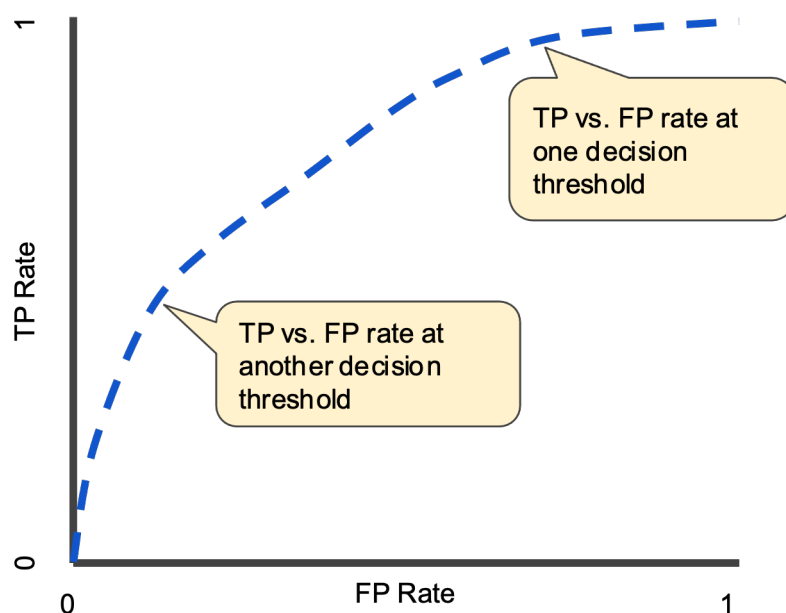
F1 Score: *Harmonic Mean*

$$1/F1 = 1/Precision + 1/Recall$$

$$F1 = \frac{2 * (Recall * Precision)}{(Recall + Precision)}$$

ROC – Receiver Operating Characteristic curve

An ROC curve plots TPR vs. FPR at different classification thresholds



True Positive Rate = Recall =
$$\frac{(TP)}{(TP+FN)}$$

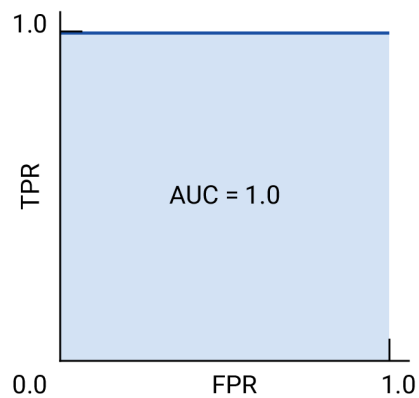
False Positive Rate =
$$\frac{(FP)}{(FP+TN)}$$

Actual Class	Predicted class	
	Class = Yes	Class = No
	Class = Yes	Class = No
Class = Yes	True Positive	False Negative
Class = No	False Positive	True Negative

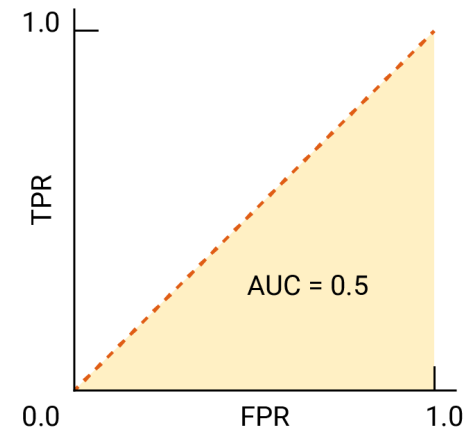
Source: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>

AUC/ ROC Examples

ROC and AUC of a perfect system



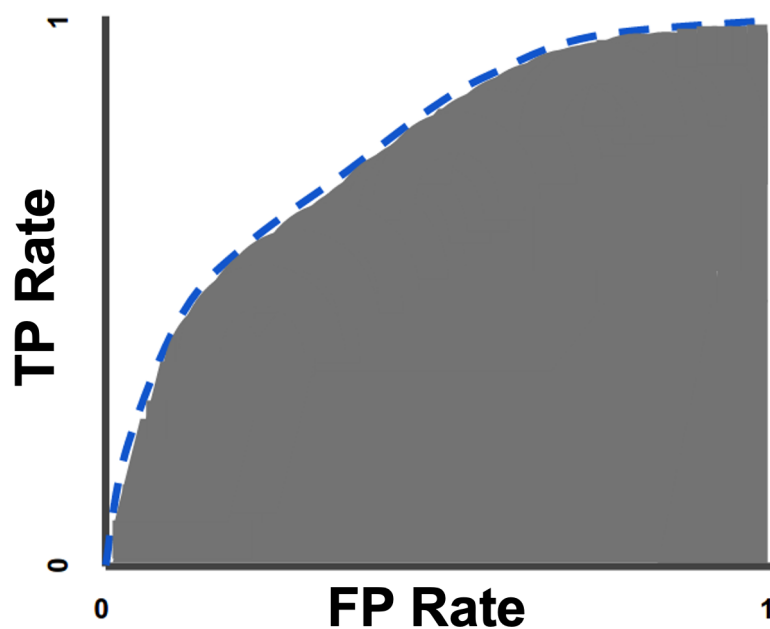
ROC and AUC of completely random guesses



The AUC is 0.5, representing a 50% probability of correctly ranking a random positive and negative example

Source: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>

AUC – Area Under the ROC Curve



- Aggregate measure of performance across all possible classification thresholds.
- Interpretation: probability that the model ranks a random positive example more highly than a random negative example

Not helpful when the cost of false negatives vs. false positives are asymmetric

Source: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>

Exercise and References

- Google:
<https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>
 - Take quiz
- Blogs: <https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/>

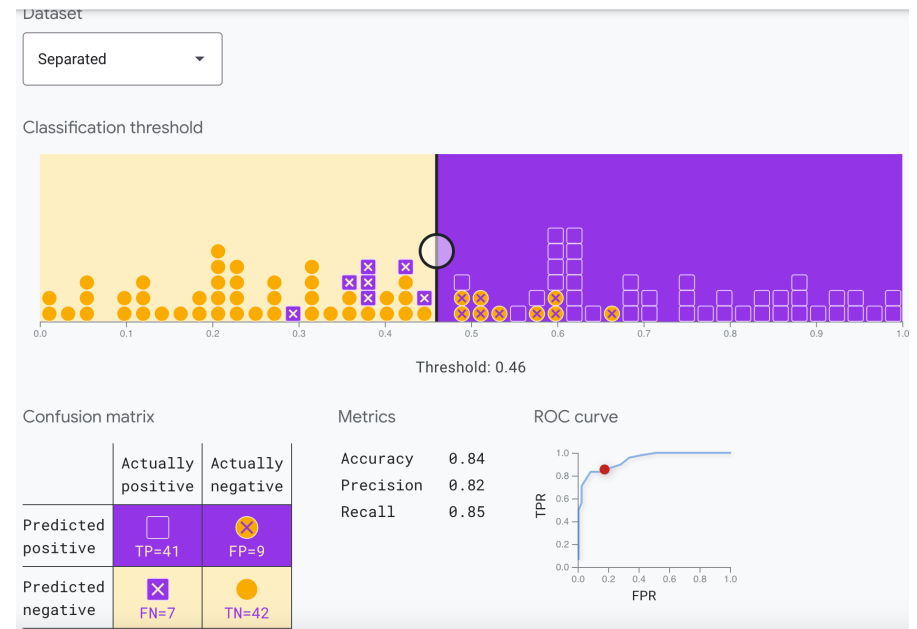


Image credit: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>

Which ML Classification Method to Choose?

- Reading material:
 - “Which ML to Use” with title: Data-driven advice for applying machine learning to bioinformatics problems
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5890912/>
 - “10 tips with title”: Ten quick tips for machine learning in computational biology
<https://biodatamining.biomedcentral.com/articles/10.1186/s13040-017-0155-3>

Discussion: 10 Tips Paper

- Access: <https://biodatamining.biomedcentral.com/articles/10.1186/s13040-017-0155-3>
- Chicco, D. Ten quick tips for machine learning in computational biology. *BioData Mining* **10**, 35 (2017). <https://doi.org/10.1186/s13040-017-0155-3>

The Tips

- Tip 1: Check and arrange your input dataset properly
- Tip 2: Split your input dataset into three independent subsets (training set, validation set, test set), and use the test set only once you complete training and optimization phases
- Tip 3: Frame your biological problem into the right algorithm category
- Tip 4: Which algorithm should you choose to start? The simplest one!
- Tip 5: Take care of the imbalanced data problem
- Tip 6: Optimize each hyper-parameter
- Tip 7: Minimize overfitting
- Tip 8: Evaluate your algorithm performance with the Matthews correlation coefficient (MCC) or the Precision-Recall curve
- Tip 9: Program your software with open source code and platforms
- Tip 10: Ask for feedback and help to computer science experts, or to collaborative Q&A online communities

Lecture 13: Summary

- We talked about
 - The variety of methods for classification
 - Logistic Regression
 - Decision trees
 - Random forest
 - Naïve Bayes
 - Boosting
 - Metrics – AUC / ROC
 - Discussion: Choosing a method that works

Concluding Section

Course Project

Discussion: Projects

- New: two projects
 - Project 1: model assignment
 - Project 2: single problem/ llm based solving / fine-tuning/ presenting result

Project Discussion

1. Go to Google spreadsheet against your name
2. Enter model assignment name and link from (<http://modelai.gettysburg.edu/>)

1. Create a private Github repository called “CSCE58x-Fall2024-<studentname>-Repo”. Share with Instructor (biplav-s) and TA (vishalpallagani)
2. Create Google folder called “CSCE58x-Fall2024-<studentname>-SharedInfo”. Share with Instructor (prof.biplav@gmail.com) and TA (vishal.pallagani@gmail.com)
3. Create a Google doc in your Google repo called “Project Plan” and have the following by next class (Sep 5, 2024)

Timeline

1. Title:
2. Key idea: (2-3 lines)
3. Data need:
4. Methods:
5. Evaluation:
6. Milestones
 1. // Create your own
7. Oct 3, 2024

Reference: Project 1 Rubric (30% of Course)

Assume total for Project-1 as 100

- **Project results** – 60%
 - Working system ? – 30%
 - Evaluation with results superior to baseline? – 20%
 - Went through project tasks completely ? – 10%
- **Project efforts** – 40%
 - Project report – 20%
 - Project presentation (updates, final) – 20%
- **Bonus**
 - Challenge level of problem – 10%
 - Instructor discretion – 10%
- **Penalty**
 - Lack of timeliness as per your milestones policy (right) - up to 30%

Milestones and Penalties

- Project plan due by Sep 5, 2024 [-10%]
- Project deliverables due by Oct 3, 2024 [-10%]
- Project presentation on Oct 8, 2024 [-10%]

Report Format

1. Title:
2. Key idea: (2-3 lines)
3. Data need:
4. Methods:
5. Screen shot (as applicable)
6. Evaluation:
7. Experience: *what learnt, anything special to discuss with class*

Presentation Format

2 minute video

Screen Shot

1. Title:
2. Key idea: 1 line summary
3. Data need:
4. Effort and Result
 1. What was done (scope)
 2. What was not done (decided not to, couldn't)
 3. Result

Experience

About Next Lecture – Lecture 14

Lecture 14: Machine Learning

- Unsupervised learning/ Clustering

9	Sep 17 (Tu)	Local search
10	Sep 19 (Th)	Adversarial games and search
11	Sep 24 (Tu)	Constraints & optimization
12	Sep 26 (Th)	Machine Learning - Basics
13	Oct 1 (Tu)	Machine Learning – Classification – Decision Trees, Random Forest, NBC, Gradient Boosting, ML-Text
14	Oct 3 (Th)	ML – Unsupervised / Clustering
15	Oct 8 (Tu)	Student presentations - project
16	Oct 10 (Th)	ML – NN, Deep Learning