AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

UNIVERSITY OF SOUTH CAROLINA

CSCE 580: Introduction to AI

Week 3 - Lectures 5 and 6: Machine Learning

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

2ND AND 4TH SEP 2025

Carolinian Creed: "I will practice personal and academic integrity."

Credits: **Copyrights of all material reused acknowledged**

Organization of Week 3 - Lectures 5, 6

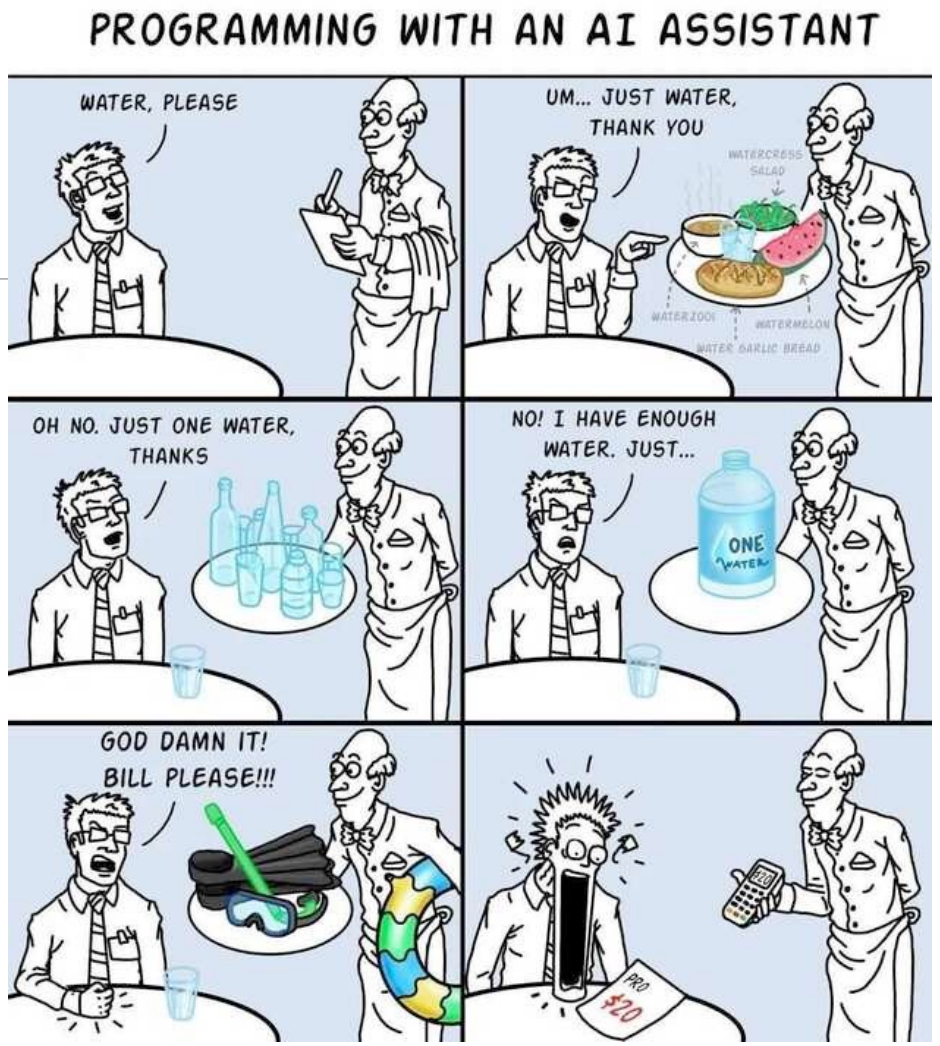
- Introduction Section
  - Recap from Week 2 (Lectures 3 and 4)
  - Project A: Q/A
  - AI news
- Main Section
  - L5: Supervised ML - numeric
  - L6: Supervised ML – including text
- Concluding Section
  - About next week – Lectures 7, 8
  - Ask me anything

Recap from Week 2

- We talked about
  - Data representations
  - ML basics
- Project A continues

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2: Data: Formats, Representation, ML Basics
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

AI News



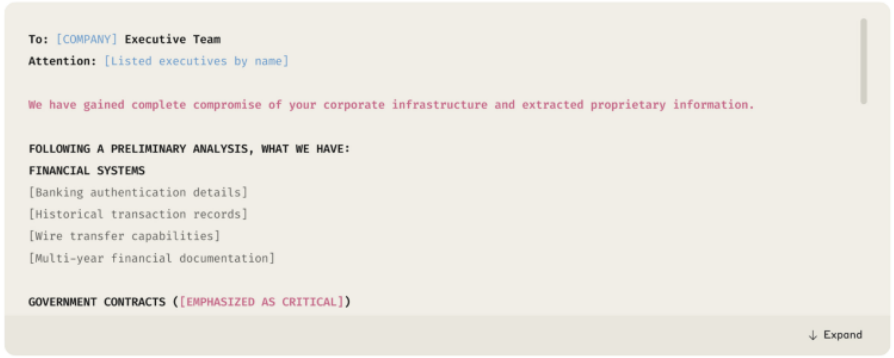
Credit: From FB

#1 Hacker Used AI to Automate Cybercrime Spree

- Report: https://www.anthropic.com/news/detecting-countering-misuse-aug-2025
  Press: https://www.nbcnews.com/tech/security/hacker-used-ai-automate-unprecedented-cybercrime-spree-anthropic-says-rcna227309

Key points

- "**first publicly documented instance** in which a hacker used a leading AI company's chatbot to automate almost an entire cybercrime spree."
- "(used Claude) to research, hack and extort at least 17 companies.."



A simulated custom ransom note. This is an illustrative example, created by our threat intelligence team for research and demonstration purposes after our analysis of extracted files from the real operation.

1. Specializes in "vibe coding," or creating computer programming based on simple requests — to identify companies vulnerable to attack.
2. Claude then created malicious software to actually steal sensitive information from the companies.
3. Next, it organized the hacked files and analyzed them to both help determine what was sensitive and could be used to extort the victim companies.
4. Analyzed the companies' hacked financial documents to help determine a realistic amount of bitcoin to demand in exchange for the hacker's promise not to publish that material.
5. It also wrote suggested extortion emails.

#2 AI Role at a Fire Station

#2 AI Role at a Fire Station

Problem Possibilities

1. Data integration and usage in context, e.g., in the fire truck
2. Generating reports from integrated data (very expensive currently)
3. Rerouting EMS vehicles*
4. Fire station demand prediction across a week*@
5. False alarms from (phone) crash detection*
6. Elderly Fire-311: elderly calling in non-emergency situations; estimate how frequently this is happening*

Solution (AI/Data) Possibilities

1. Data-based forecasting
   - leverage previous work on traffic estimation
2. Drones for fire call reconnaissance

* Solutions need not be aligned to problems

Introduction Section

Lecture 5: Supervised Learning

- …

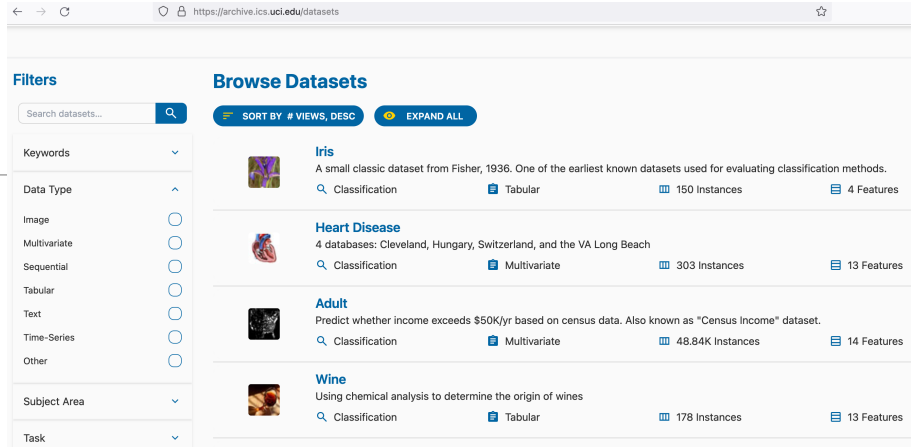
Machine Learning – Insights from Data

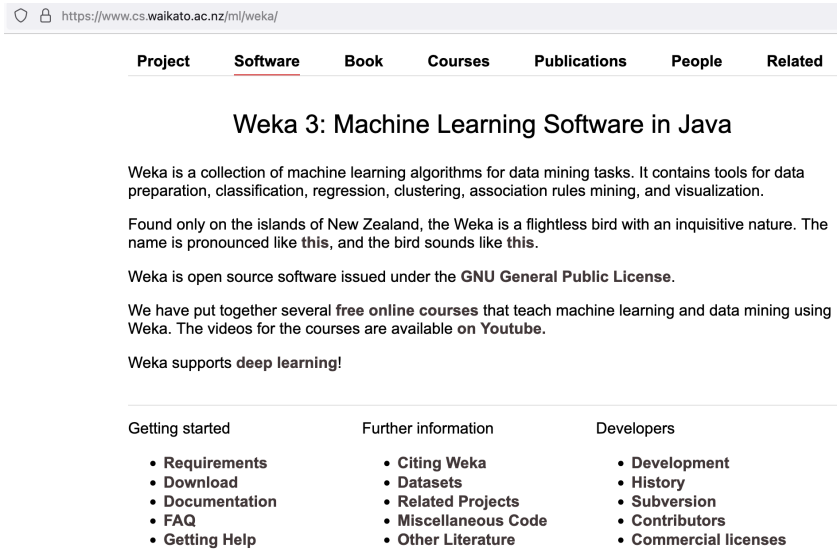
- Descriptive analysis
  - Describe a past phenomenon
  - **Methods**: classification (feedback from label), clustering, dimensionality reduction, anomaly detection, neural methods, reinforcement learning (feedback from hint/ reward)
- Predictive analysis
  - Predict about a new situation
  - **Methods**: 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

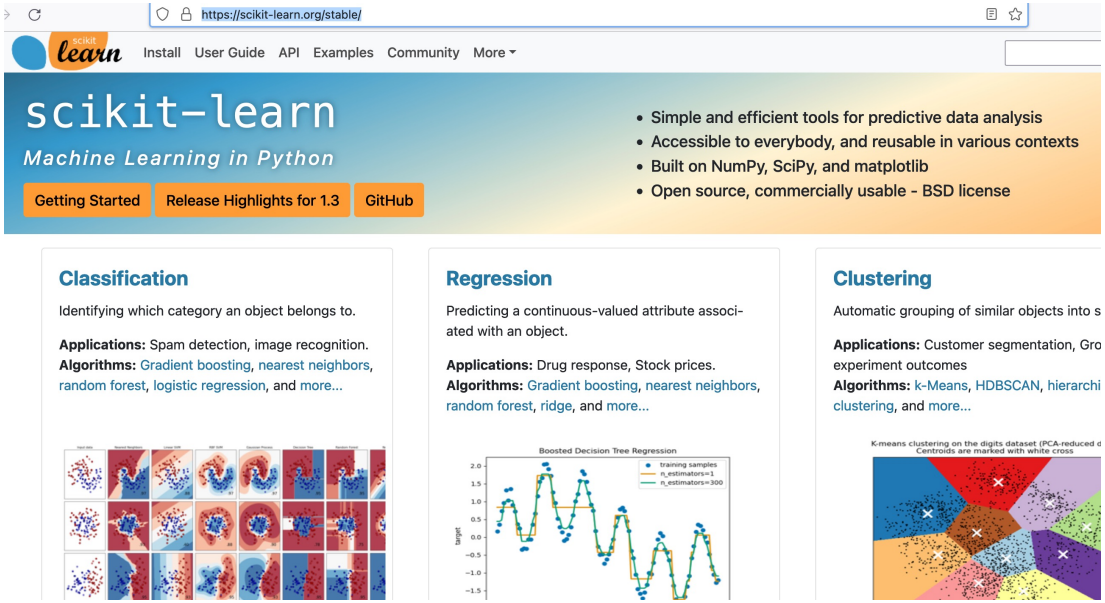




- Tools:
  - Weka - https://www.cs.waikato.ac.nz/ml/weka/
  - Download tool and dataset
- Libraries
  - Scikit - https://scikit-learn.org/stable/

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/



Exercise: Weight and BMI

Week's Activity

- Use data from 'DataSample-WeightHeight' spreadsheet
- Clean data, as appropriate
- Build models
  - Task 1: To predict weight, given height // prediction
  - Task 2: To classify BMI into 4 categories*, given height // classification
- Report performance metrics for the two models

* : see next slide on BMI categories

**: refer to Crawl-Walk-Run approach to scope - https://www.linkedin.com/pulse/crawl-walk-run-approach-ai-based-real-world-problem-biplav-srivastava-pxsre/

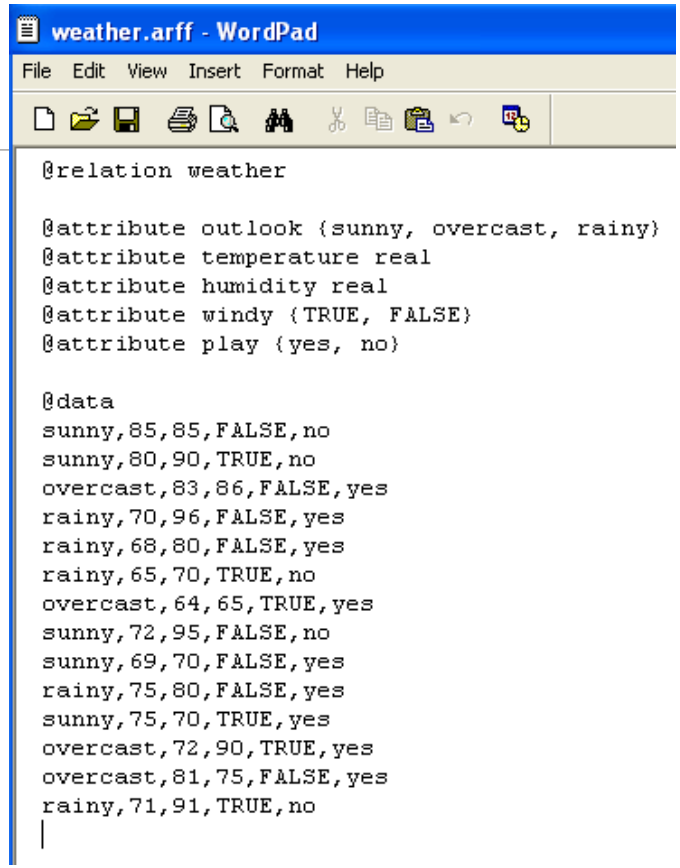
Body Mass Index (BMI)

- BMI Category, based on BMI Range (kg/m^2)
  - C1 - Underweight: Less than 18.5
  - C2 - Healthy Weight: 18.5 to less than 25
  - C3 - Overweight: 25 to less than 30
  - C4 - Obesity: 30 or greater
- Details: https://www.cdc.gov/bmi/adult-calculator/bmi-categories.html

Exercise: Run Weka

ARFF Data Format

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



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

Exercise: Understand German Credit

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

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

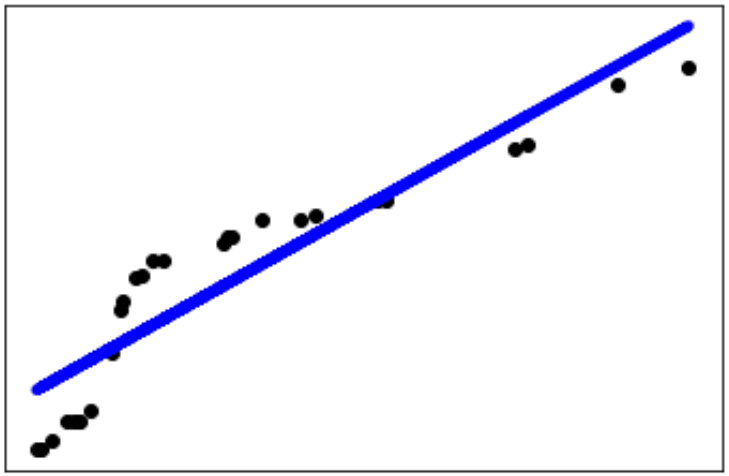
Classifier Method Types

- 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

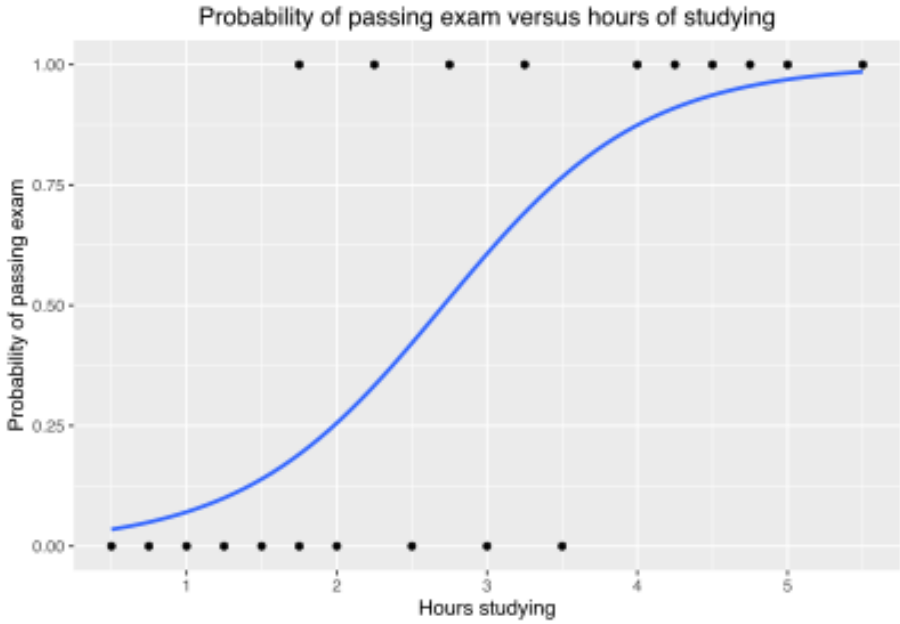


Notebook: https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l6-l7-l8-supervised-ml/Supervised-Regression.ipynb

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.

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

•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

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
```

Slide Adapted From/ Courtesy: Data Mining, Practical Machine Learning Tools and Techniques, Slides for Chapter 6, Trees and Rules of Data Mining by I. H. Witten, E. Frank, M. A. Hall and C. J. Pal

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

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
…	…	…	…	…

- 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 ..m)
  - $I(s_1, s_2, .., s_m) = - \Sigma_{i=1 \text{ to } m} p_i \log_2 (p_i)$
    - Where P_i is the probability a sample belongs to class Ci; estimated by(s_i /s)

- Entropy / Expected Information after partitioning on Attribute A which has **v** distinct values
  - $E(A) = \Sigma_{j=1 \text{ to } v} (s_{1j} + \ldots + s_{mj}) / S * (I (s_{1j}, s_{2j}, .., s_{mj}))$
  - s_{ij} be the number of samples in S_j of Class C_i (i = 1 ..m)
  - Smaller the entropy, the greater the purity of the subset partitions

Information Gain

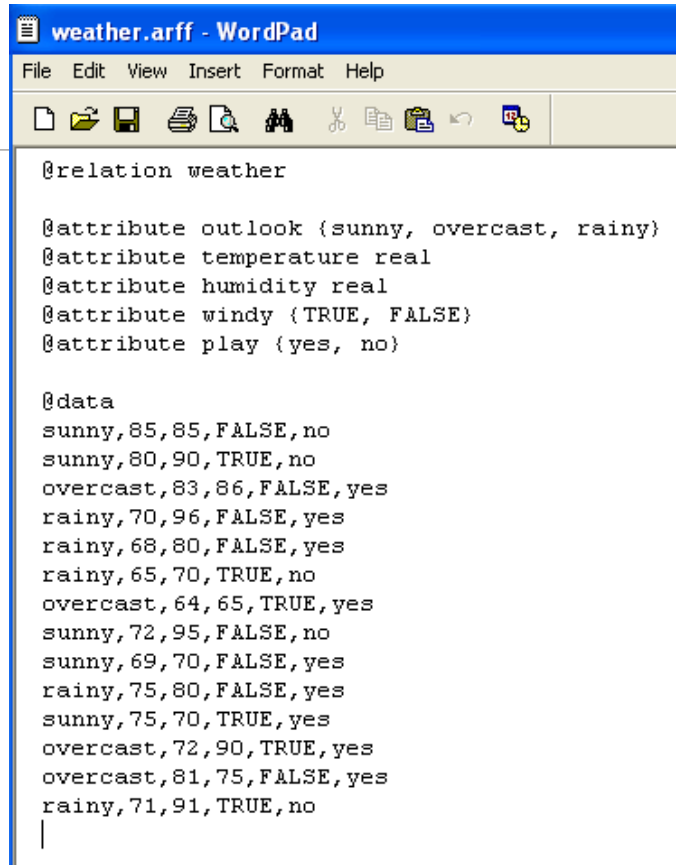
Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
...	...	...	...	...

- Entropy / Expected Information after partitioning on Attribute A which has v distinct values
  - $E(A) = \Sigma_{j=1 \text{ to } v} (s_{1j} + \ldots + s_{mj}) / S * (I(s_{1j}, s_{2j}, .., s_{mj})$
    - s_{ij} be the number of samples in S_j of Class C_i (i = 1 ..m)
- After partition, S_j
  - $I(s_{1j}, s_{2j}, .., s_{mj}) = - \Sigma_{i=1 \text{ to } m} p_{ij} \log_2 (p_{ij})$
  - Where p_{ij} is the probability a sample in S_j belongs to class C_i; estimated by$(s_{ij} / | s_j |)$
- Gain (A) = $I(s_1, s_2, .., s_m)$ - 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

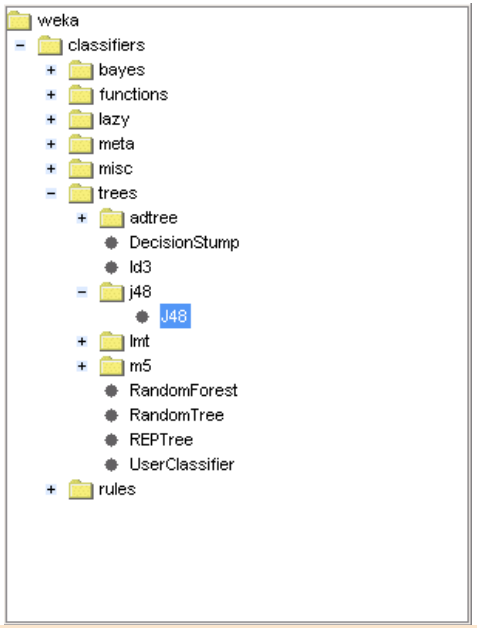
- Data is in ARFF in UCI dataset
- Or Convert
  - File system, CSV → ARFF format
  - Use C45Loader and CSVLoader to convert

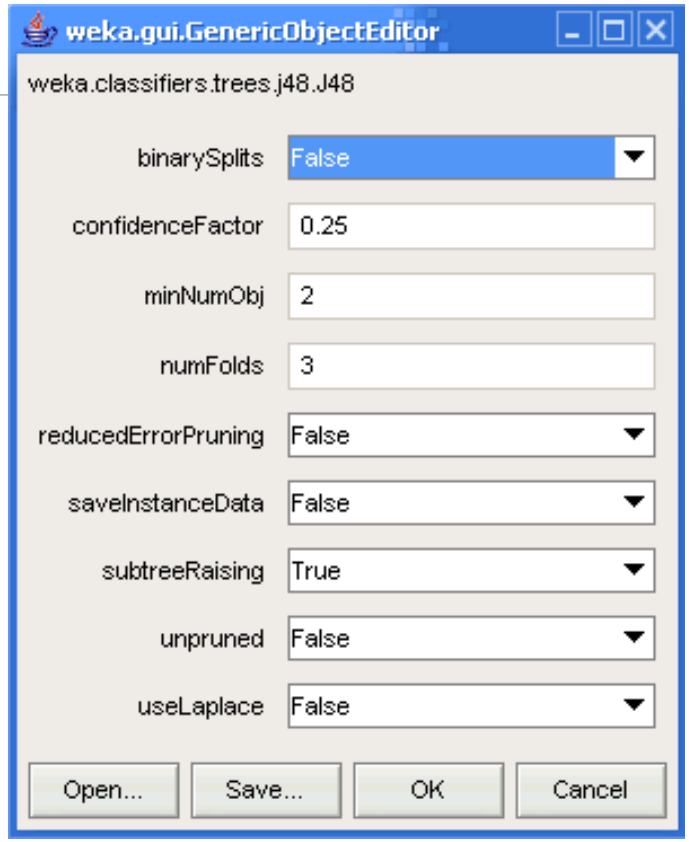


Slide Courtesy: http://www.cs.iastate.edu/~cs573x/bbsilab.html

Weka: weka.classifiers.trees.J48

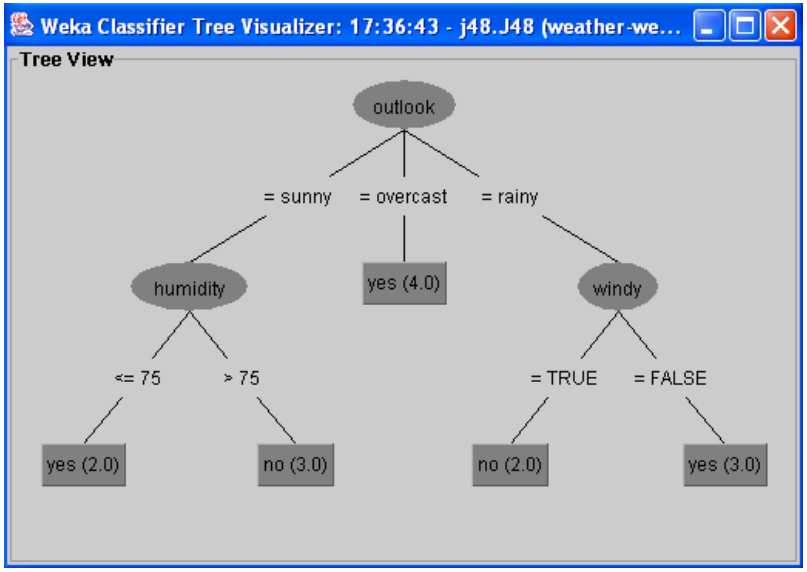
Class for generating an unpruned or a pruned C4.5 decision tree.





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
```

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

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

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

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
- 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<<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

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

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Using the naive conditional independence assumption that

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

for all i, this relationship is simplified to

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

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

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

$$\Downarrow$$

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

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

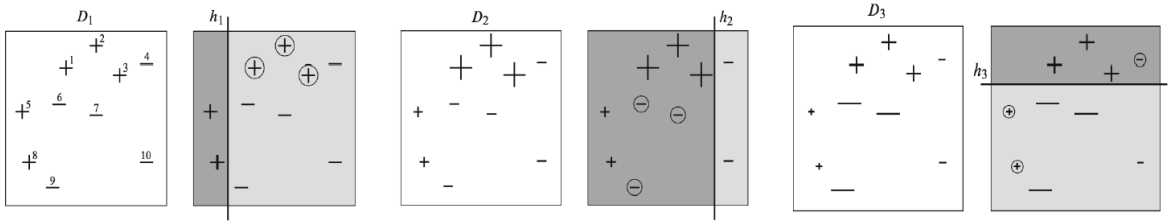


Figure: AdaBoost. Source: Figure 1.1 of [Schapire and Freund, 2012]

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

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.

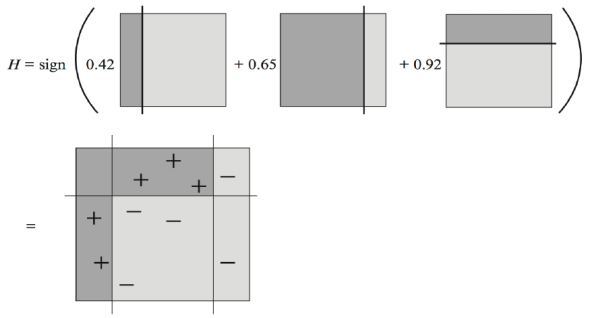


Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

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

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

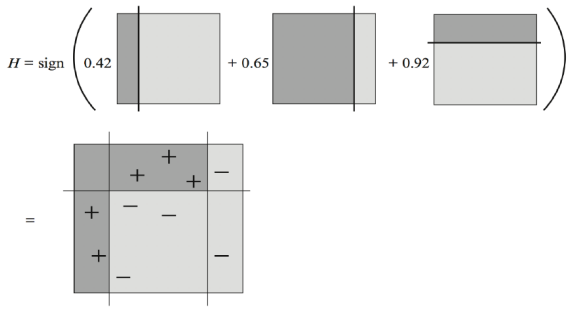


Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

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)

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

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.

Gradient Boosting = Gradient Descent + Boosting

Adaboost

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

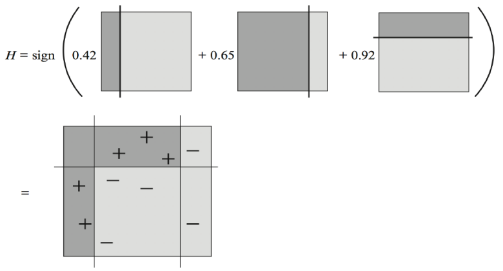


Figure: AdaBoost. Source: Figure 1.2 of [Schapire and Freund, 2012]

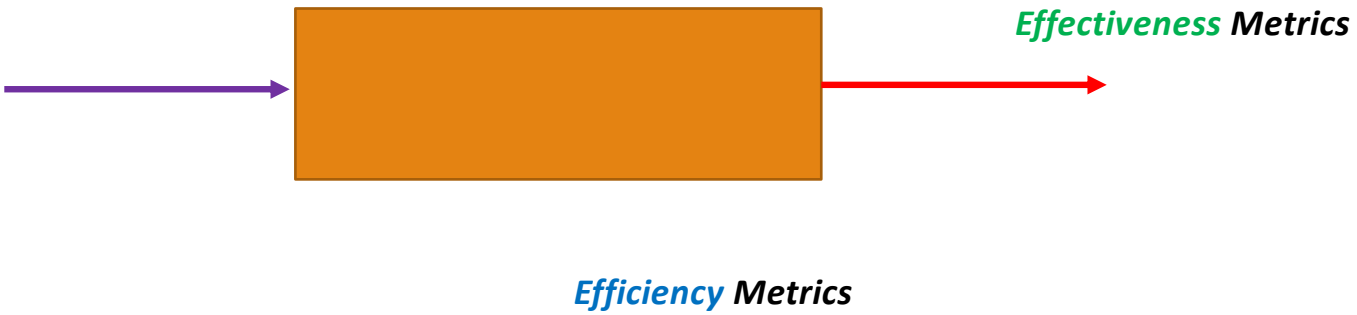
Content and Image Courtesy: Prof. Cheng Li
https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

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

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

Accuracy =
(TP+TN)/
(TP+FP+FN+TN)

Precision =
(TP)/
(TP+FP)

Recall =
(TP)/
(TP+FN)

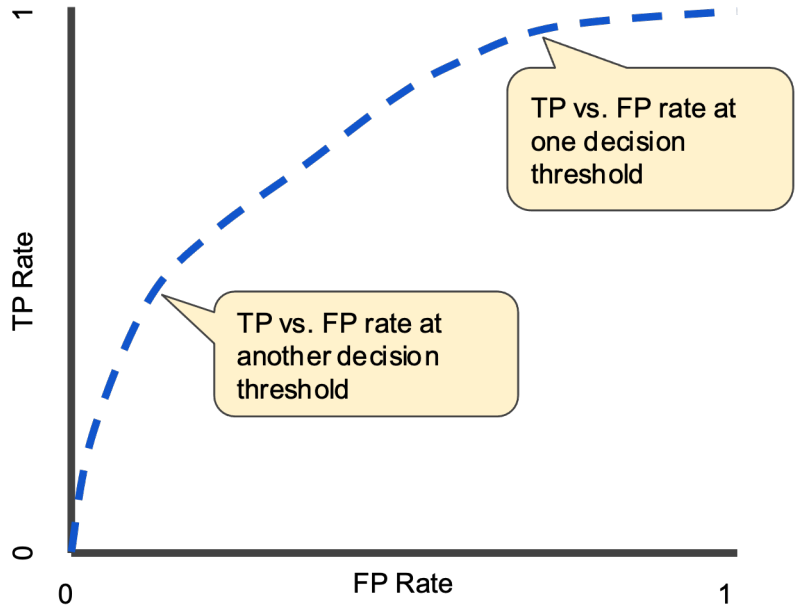
F1 Score: *Harmonic Mean*

1/F1 = 1/Precision + 1/Recall

F1 = 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 =
(TP)/
(TP+FN)

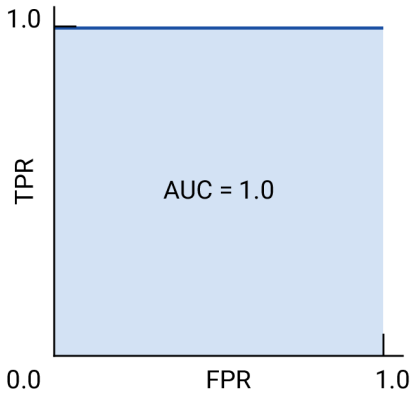
False Positive Rate =
(FP)/
(FP+TN)

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

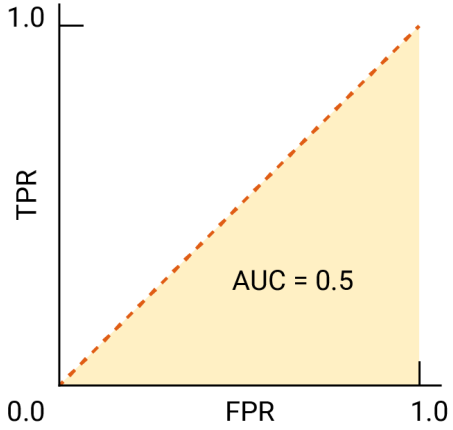
Source: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

AUC/ ROC Examples

ROC and AUC of a perfect system



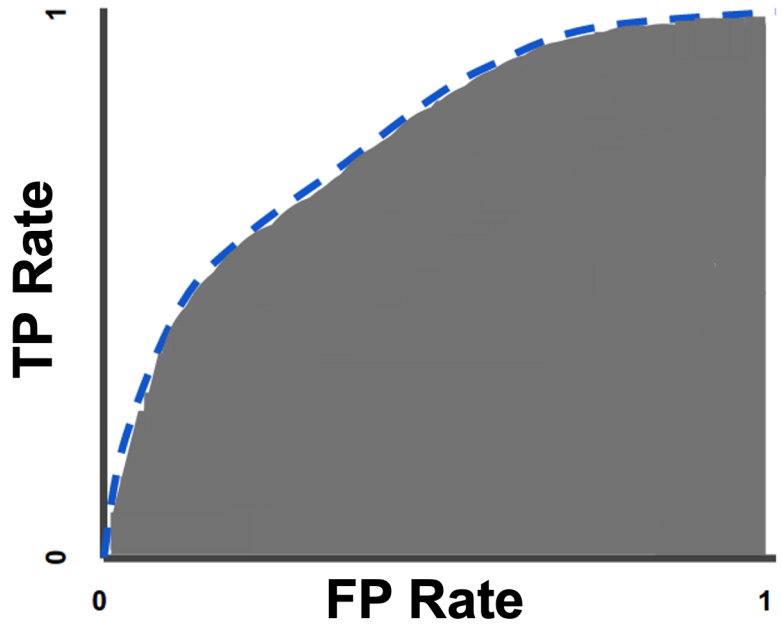
ROC and AUC of completely random guesses



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/

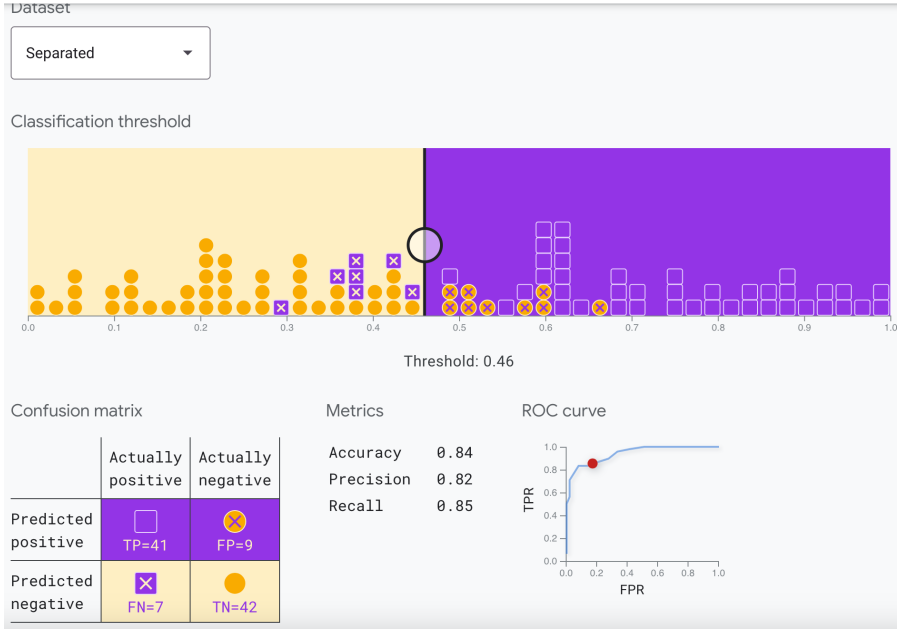


Image credit: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

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 5: Concluding Comments

- 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

Projects A: Start (4 weeks; 200 points)

- End date: **Thursday, Sep 18**

- See model AI Assignments: http://modelai.gettysburg.edu/
  - Choose a project, preferably within last 5 years (i.e., after 2020).
    - Enter its name in **"Student-InfoShared .." sheet, column G**
  - Follow instructions and do it alone
  - Submit project outcome
    - Create a folder in your Github called **ProjectA**.
    - Create a file called "ProjectInfo.md" with your name, project chosen and URL/ other details.
    - Put deliverables, as per project description, inside the folder, and commit.
    - Timestamp will be used to confirm that Project-A is done on time

Lecture 6: Supervised Learning

- Complete discussion
- Case studies
  - C1: Whether to signup for TSA **Pre-Check**
  - C2: Whether to use AI-stethescope
- Text classification – classification, when data is textual
  - Text processing basics
  - Special text handling for classification

Recap from Week 3; Class 5

- 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

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2: Data: Formats, Representation, ML Basics
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

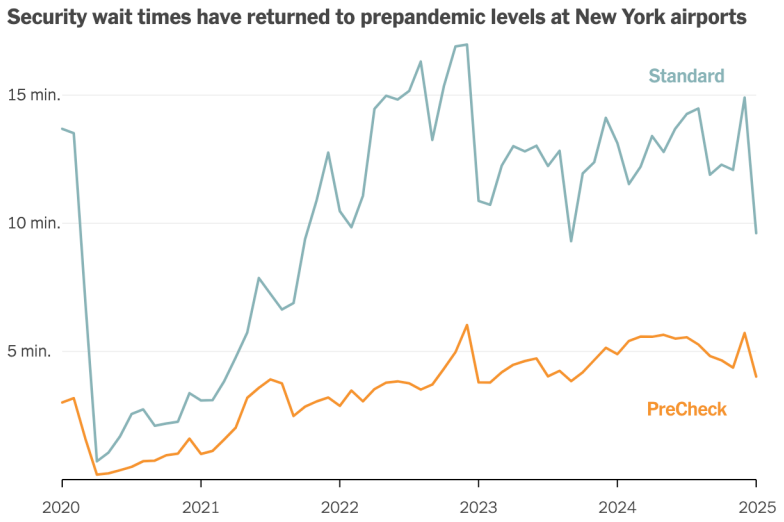
Main Section

Case Study 1: Sign Up for TSA Pre-Check ?

Article: https://www.nytimes.com/2025/08/29/upshot/tsa-precheck-how-long.html

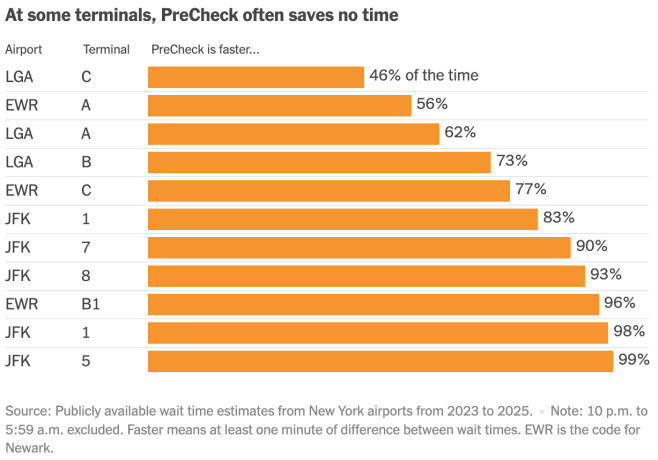
TSA Pre-Check ?

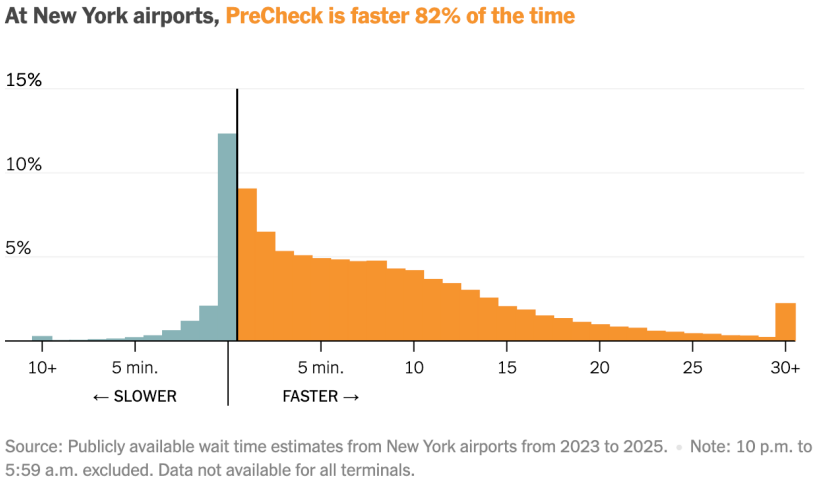
- Programs costs under $80 for five years
- Has become less exclusive as its membership numbers have soared to more than 22 million; 1/3 passengers have it
- Data not available for most busy airports



TSA Pre-Check ?

- Programs costs under $80 for five years
- Has become less exclusive as its membership numbers have soared to more than 22 million; 1/3 passengers have it
- Most savings are at New York airports
- **Data not available for most busy airports**





Case Study 2: AI Stethoscope

Article: https://www.perplexity.ai/page/ai-stethoscope-detects-3-heart-iYvRzKupTByTdRqd3fP.hA

Case Study 2: AI Stethoscope

- AI-enhanced stethoscope, can detect three major heart conditions in 15 seconds. New device replaces the familiar chest piece with a digital unit that
  - simultaneously records electrical signals from the heart through an ECG
  - captures audio of blood flow through a microphone.
  - This information is sent to cloud-based AI algorithms trained on data from tens of thousands of patients.
- Patients were 2.33 times more likely to receive **heart failure diagnoses**, **atrial fibrillation** was detected 3.5 times more frequently, and **heart valve disease** was identified nearly twice as often compared to traditional methods.
- Received FDA clearance and is being used in some GP surgeries.
- The device achieved 85% sensitivity and 70% specificity in detecting low ejection fraction, a key indicator of heart failure

Case Study 2: AI Stethoscope

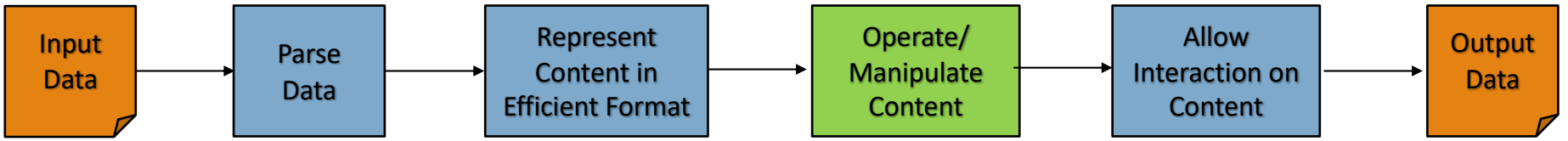
- **Implementation challenges**: 70% of GP surgeries given the smart stethoscopes stopped using them or used them infrequently after 12 months. Researchers noted that efforts to integrate the technology into existing GP routines would be necessary for wider adoption.
- **False positives**: two-thirds of people flagged by the AI as having suspected heart failure did not actually have the condition when given follow-up blood tests or heart scans. While this could lead to unnecessary anxiety and additional testing, researchers emphasized that the technology could detect heart failure signs that might otherwise go unnoticed.

Text Processing Basics

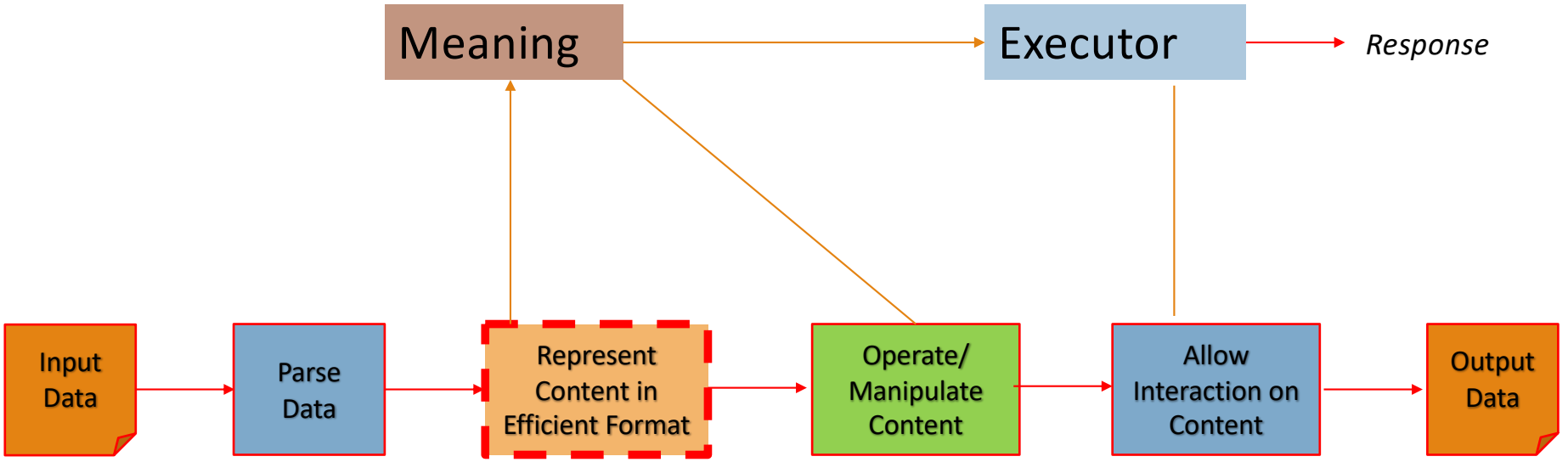
Text Processing Examples

- Processing
  - Resumes -> e.g., selecting candidates
  - Answer papers (of examinations) -> e.g., grading
  - Medical reports -> e.g., helping patients
- Predicting
  - Business intelligence
  - Financial forecasts
- Generating
  - News summary
  - Summarizing literature for research

Document Processing Pipeline



Semantics, Parsing and Representation



Common Textual Data Processing Steps for ML

- Input: strings / documents/ corpus
- Processing steps (task dependent / optional - *)
  - Parsing
  - Word pre-processing
    - Tokenization – getting tokens for processing
    - Normalization* - making into canonical form
    - Case folding* – handling cases
    - Lemmatization* – handling variants (shallow)
    - Stemming* – handling variants (deep)
  - Semantic parsing – representations for reasoning with meaning *
  - Embedding – creating vector representation*

CSCE 771 goes into details

Common NLP Tasks

- Extracting entities [Entity Extraction]
- Finding sentiment [Sentiment Analysis]
- Generating a summary [Text Summarization]
- Translating to a different language [Machine translation]
- Natural Language Interface to Databases [NLI]
- Natural Language Generation [NLG]

CSCE 771 goes into details

Text Classification

Examples of Classification with Text

- Sentiment analysis (*assign sentiment classes*)
- Annotation (*assigning entity type to to text*)
- Application specific
  - Fake news
  - Spam email
  - …

So, What Changes With Text?

Handling of (textual) data before and after applying ML methods!

Adapted from Image Credit: Trustworthy Machine Learning, Kush Varshney

Word Representation: Paper Discussion

Contextual Word Representations: Putting Words into Computers”,

by Noah Smith, CACM June 2020

https://cacm.acm.org/research/contextual-word-representations/

Problem

- How to represent words ?
- How to measure similarity, e.g., between words, and texts?
- How to determine different contexts (senses) in which words are used?
- How to handle noise, typos?

S1 - This is an apple
S2 - These are apples

S3 - This is an apples
S4 - There are apply

Option 1 - Characters

- How to represent words?
  - Characters / Unicode / …
- How to measure similarity between words, and texts?
  - Edit distance***: actions to convert one string to another***
  - Hamming distance: ***difference considering substitution***
- How to determine different contexts (senses) in which words are used?
  - Neighborhood of words: Bi-, tri-, N-gram representations

Distance between: Kitten, Sitting

Edit Distance

Algorithm	Operations Allowed			
	Insertions	Deletions	Substitutions	Transposition
Levenshtein Distance	✓	✓	✓	
Longest Common Subsequence (LCS)	✓	✓		
Hamming Distance			✓	
Damerau–Levenshtein Distance	✓	✓	✓	✓
Jaro distance				✓

Levenshtein distance:
1.kitten → **s**itten (substitute "s" for "k")
2.sitt**e**n → sitt**i**n (substitute "i" for "e")
3.sittin → sittin**g** (insert "g" at the end)

LCS distance (insertions and deletions only):
1.kitten → itten (delete "k" at 0)
2.itten → **s**itten (insert "s" at 0)
3.sitt**e**n → sittn (delete "e" at 4)
4.sittn → sitt**i**n (insert "i" at 4)
5.sittin → sittin**g** (insert "g" at 6)

Source: https://en.wikipedia.org/wiki/Edit_distance

Option 2 - Vectors

- How to represent words? Vectors
  - But, what scheme in vectors
    - One-hot encoding
    - Arbitrary, principled, …
- How to measure similarity between words, and texts?
  - Cosine similarity
- How to determine different contexts in which words are used?
  - Neighborhood of words: Bi-, tri-, N-gram representations
  - Contextual word vectors

Cosine Similarity

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\cdot\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Property: two proportional vectors have a cosine similarity of 1, two orthogonal vectors have a similarity of 0, and two opposite vectors have a similarity of -1.

Usually used for [0,1]

Sci-kit method python: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

Source: https://en.wikipedia.org/wiki/Cosine_similarity

TF-IDF based Word Representation -1

- Given N documents

- **Term frequency (TF):** for term (word) t in document d = tf(t, d)

Variants to reduce bias due to document length

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Sources:
(a) sci-kit documentation
(b) Wikipedia: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

TF-IDF based Word Representation -2

- Given N documents
- Term frequency (TF): for
  term (word) t in document d
  = tf(t, d)
- **Inverse document frequency IDF**(t)

  = log [N / **DF**(t)] + 1

DF(t) = **document frequency**, the number of documents in the document set that contain the term t.

- **TF-IDF**(t, d) = TF(t, d) * IDF(t),

Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(\frac{N}{1+n_t}\right)+1$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N-n_t}{n_t}$

Sources:
(a) sci-kit documentation
(b) Wikipedia: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

TF-IDF Example Calculation

See sample code on GitHub:
https://github.com/biplav-s/course-nl-f22/blob/main/sample-code/l5-wordrepresent/Word%20Representations%20-%20Vectors.ipynb

ML – Text Supervised

- By Example:
  - https://github.com/biplav-s/course-nl/blob/master/l9-ml-review/Classification%20-%20Fake%20news.ipynb

- Fake news dataset

Height-Weight Exercise

To Dos

1. Make a sub-folder in your github repo called “exercise-height-weight”
    1. Create a sub-folder called “**data**” and have all data there. Two specifically are sample .csv as well as cleaned/ prepared .csv file(s)
    2. Create a sub-folder called “**code**”. All code will be below it
        1. Create a sub-folder called “**data-prep**”. Have data preparation and cleaning code there.
        2. Create a sub-folder called “**custom-classifier-model**”. Have classifier training and testing code there
        3. Create a sub-folder called “**custom-regression-model**”. Have regression training and testing code there
    3. Create a sub-folder called “**genai**”. All files related to gpt/chatgpt will be below it
        1. Create a testcase file for classification. (Copy and use the testcase template: https://github.com/biplav-s/book-trustworthy-chatbot/blob/main/ai-testcases/testcase-template.md)
        2. Put transcript/ result of your work there.

GenAI Exercise

- Have a task in mind
  - Create AI testcase
  - Testcase template: https://github.com/biplav-s/book-trustworthy-chatbot/blob/main/ai-testcases/testcase-template.md
- Solve with GenAI
  - Create prompt(s)
  - Get answers on one or more LLMs
    - Get answers k times // k > 1, usually 3-10
  - Analyze answers // use GAICO
- Document exercise and submit
  - Testcase
  - Prompt
  - Answers
  - Analysis
  - Conclusion for the AI's performance on the task

GenAI Exercise – Weight Classification

- Use data from 'DataSample-WeightHeight' spreadsheet
- **Direct approach**: Ask GenAI model to build model and give results
- **Alternative approaches**: ask model to do one or more of
  - Clean data [optional, for one or more columns]
  - Build models to classify BMI into 4 categories*, given height // classification
- Report performance metrics for the model
- **Compare result of your model with that of GenAI's**

Lecture 6: Concluding Comments

- We talked about
  - Practical case studies of ML classification
  - Classification with text documents

Week 3: Concluding Comments

- We talked about
  - Supervised methods
  - Structured data
  - Evaluation metric: AUC-ROC
  - Textual, unstructured, data
  - Height-Weight exercise
    - Regression
    - Classification
    - Comparison with GenAI/ testcase

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Projects A: Start (4 weeks; 200 points)

- End date: **Thursday, Sep 18**

- See model AI Assignments: http://modelai.gettysburg.edu/
  - Choose a project, preferably within last 5 years (i.e., after 2020).
    - Enter its name in **"Student-InfoShared .." sheet, column G**
  - Follow instructions and do it alone
  - Submit project outcome
    - Create a folder in your Github called **ProjectA**.
    - Create a file called "ProjectInfo.md" with your name, project chosen and URL/ other details.
    - Put deliverables, as per project description, inside the folder, and commit.
    - Timestamp will be used to confirm that Project-A is done on time

About Week 4 – Lectures 7 and 8

Week 4 – Lectures 7 and 8

- Unsupervised learning
- Metrics
- Applications

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2: Data: Formats, Representation, ML Basics
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots