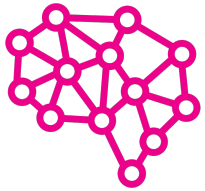
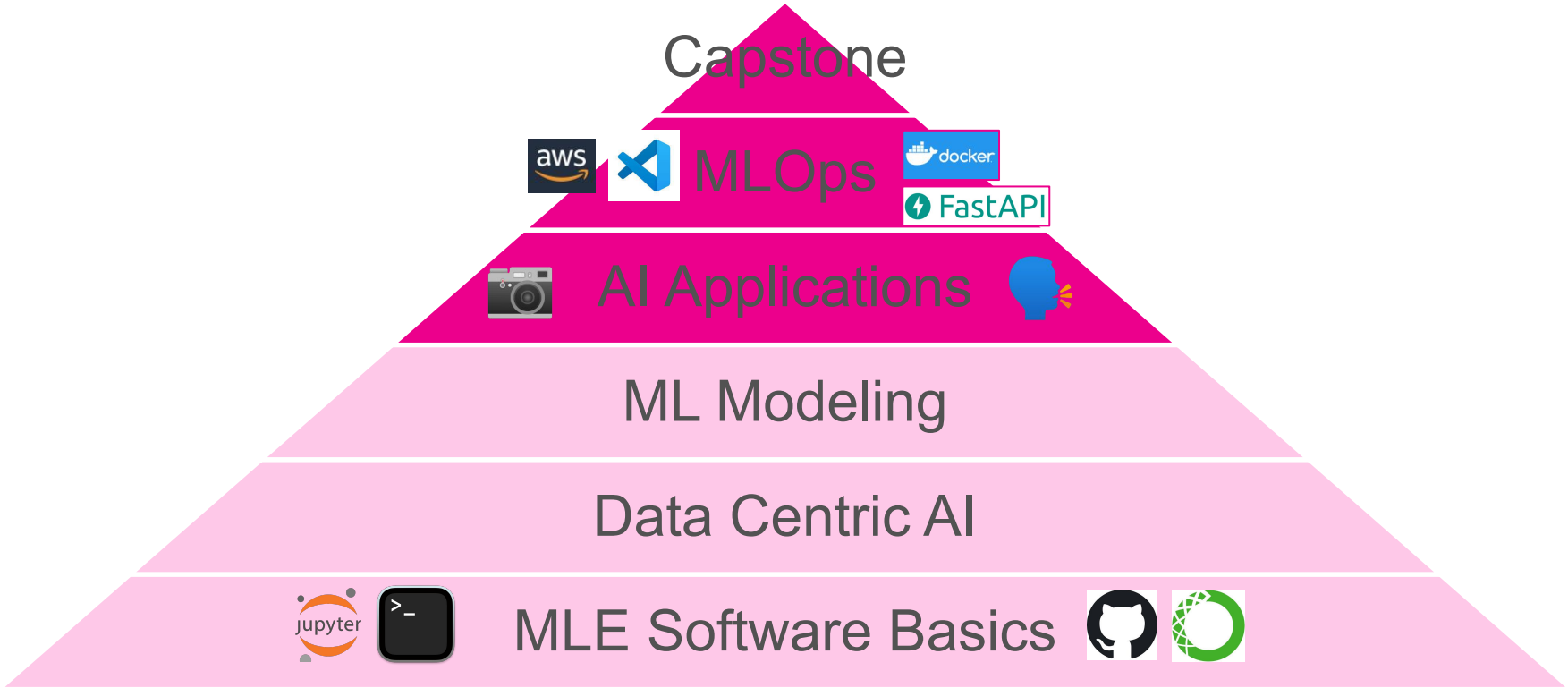


FourthBrain

MLE Program, Cohort 11 (MLE11)

Week 6: Regression, Classification, Class Imbalance and Metrics

Becoming a Machine Learning Engineer



Our Updated Curriculum!

1. ML Project Scoping
2. Real, Live Data Streams
3. Data Wrangling & Exploratory Analysis
4. Big Data

DATA CENTRIC AI

5. Supervised ML
6. Deep Learning & AutoML
7. Unsupervised, Semi- & Self-supervised Learning

ML MODELING

8. Computer Vision
9. Natural Language Processing
10. Transformers & Fine Tuning Pre-Trained Networks

AI APPLICATIONS

11. Building ML Web Apps
12. Containerization
13. Model Serving
14. Machine Learning in Production

MLOps

Last Session

Concepts

- Big Data Processing
- Big Data Tools
- Good Data over Big Data

Hands on

- Exploring and wrangling a big data set to predict customer/client behavior using an ML pipeline

🤖 This Week!

Concepts

- Regression
- Classification
- Data Imbalance
- Accuracy Metrics
- AI Explainability

Hands on

- Predicting customer conversion using an explainable ML pipeline

What questions do you have?

Industry Note

Today's Class Agenda

- Capstone Presentations (1 hr)
- Break (5 min)
- Supervised ML (2.5 hrs)
- Break (30 min)
- Coding Assignment (2 hrs)

Capstones

- In class Presentation this and last Saturday
  - this is the initial Capstone presentation
  - use this template
  - focus on the manageable timeline
  - 5 minutes per team - 5 min presentation + 2 min QA
    - the presentations will be in the Main Room
  - make sure to include Ethics and Architecture Design slide
  - About Capstone deliverables

- NOTE - all the useful Capstone links in Canvas at the bottom of Modules page!

Capstone Presentations

1. Protein
2. Booz Allen
3. GroupBy
4. Auto Speech Recognition
5. Booz Allen Turbine RL Forecast
6. Spotify2
7. NLP

Don’t fear math!

Vocab 1

- Optimization - min or max or some function by altering *x*.
  - this function is called objective function, cost function or loss function as well
  - by altering *x* is denoted with argmin or argmax in academic publishing
- We will get into details on optimization next week - specifically into gradient descent.

Supervised learning

- Given a set of **training examples**: $x_i =< x_{i1}, x_{i2}, \ldots, , x_{in}, Y >$
- x_{ij}: jth feature of the ith training example
- y_i: desired output for the ith example

- Goal: want to learn a function f: $X_1 \times X_2 \times X_3 \times \cdots \times X_N \rightarrow Y$ which maps the input variables onto the output variable
- Formally, f is called the hypothesis.
- Output Y can have different types:
    - If Y=$\mathbb{R}$ the problem is regression

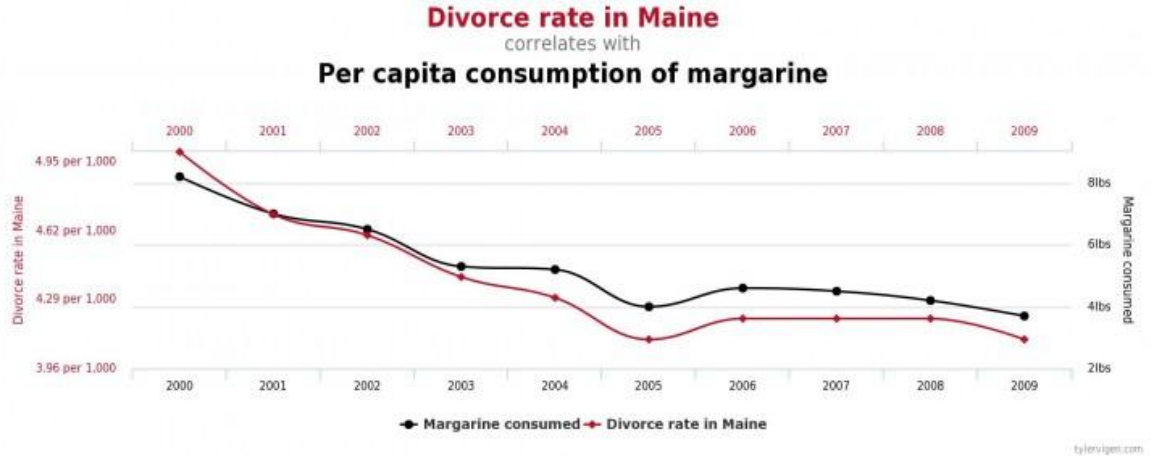
Vocab 2

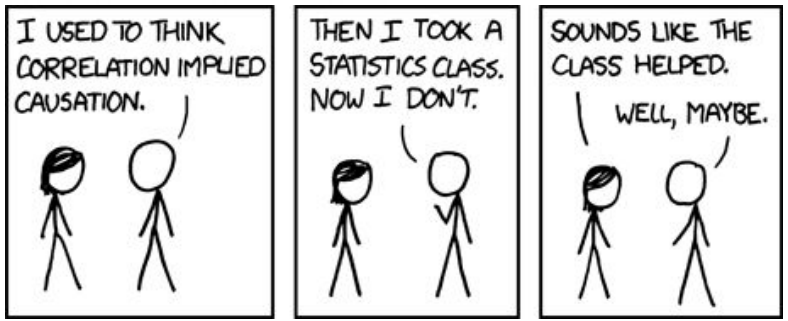
- Supervised Learning - target variable is provided
- Classification - predicting which of the categories some input belongs to.
- Regression - predicting a numerical value given some input

Linear Models

Pre-work for coding

Correlation doesn't imply Causation





Regression problem

Suppose Y is a linear function of X:

$$f_w(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m = w_0 + \Sigma w_j x_j$$

The w_j are called parameters or weights

To simplify notation, we add an attribute $x_0 = 1$ to m other attributes (also called bias term or intercept)

How should we pick the *weights*?

Vocab 3

- Model weights - determining how each feature affects the prediction
- Gradient - simply put it is the name for a derivative in vector based calculus
- Residuals = error

Least-squares solution method

$$f_w(X) = w_0 + w_1 x_1 + \cdots + w_m x_m = w_0 + \Sigma w_j x_j$$

Most common choice is to find the w that minimizes:

$$Err(w) = \Sigma(y_i - w^T x_i)^2$$

We want to solve for w:

$$X^T(Y - Xw) = 0$$

$$X^T Y = X^T X w$$

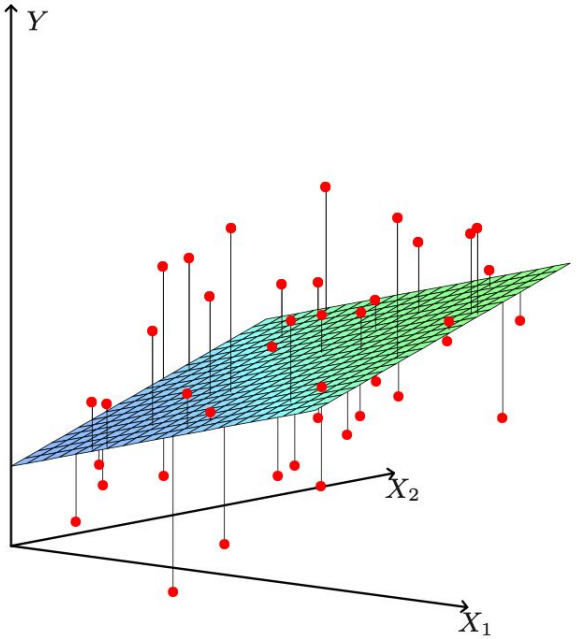
$$\hat{w} = (X^T X)^{-1} X^T Y \text{ where } \hat{w} \text{ are estimated}$$

$$\text{Prediction: } \hat{Y} = X\hat{w} = X(X^T X)^{-1} X^T Y$$

alternative for minimizing mean-squared error (MSE)

$$Err(w) = (Y - Xw)^T (Y - Xw)$$

$$\frac{\sigma Err(w)}{\sigma(w)} = 2(X^T X w - X^T Y)$$



Jolle Pineau, COMP-551

Gradient Descent Solutions

Why use Gradient Descent when Normal Equations can be solved?

Ans: To solve normal equations,

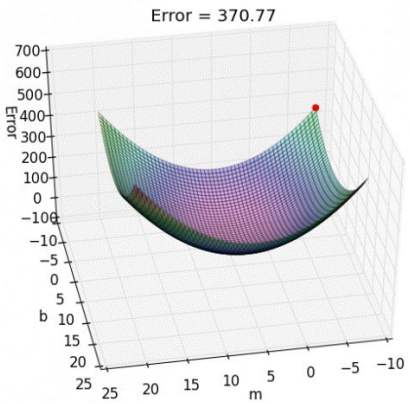
$$\theta = (X^T X)^{-1} X^T Y$$

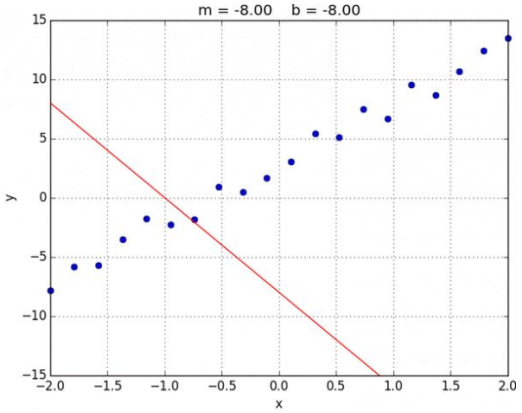
-Computational complexity is $O(n^3)$, for *n* dimensions.

-As *n* grows, complexity explodes.

-Iterative solutions in batches is better for optimization.

The gradient error is a vector indicating the direction to minimum point





Source:
https://medium.com/@savannahar68/getting-started-with-regression-a39aca03b75f

Instead of directly finding the minimum (using closed form eq) we can take small steps toward the minimum.

Some Applications of Regression...

- *Business Optimization Use Cases.*
  - ***Easy:*** *To* understand the *relationship* between oven temperature and shelf-life of baked cookies.
  - To know the *relationship* between wait times of callers and number of complaints.
  - ***Not Easy***: To find the leading *cause* for faulty part production and eliminate it.
  - The HR Department, intends to identify divisions with most churn to redesign bonus.
- *Medical Use Cases:*
  - ***Easy:*** To find *relationships* between body weight and diabetes, blood pressure etc.
  - Find the *relationship* between household salary, no. of hospital visits etc. for insurance premiums.
  - ***Not Easy:*** To allocate a *risk factor* towards flu-like diseases based on pre-existing conditions.
  - A pharma company intends to find *relative risk* between pre-existing conditions and new drug.

4 assumptions of linear regression

Data

Check for:

1. **Linear relationship between x and y**
2. **Independence of the residuals**
3. **Normally distributed residuals**
4. **Heteroscedasticity or constant variance or the residuals**

In probability theory, we consider a set of random variables independent and identically distributed if each random variable has the same probability distribution as the others and all are mutually independent

Computational Complexity

- Gradient Descent: $O(kn^2)$, for n-features, k-iterations.
- Normal equations: $O(n^3)$, for matrix inversion.
- Generalized Linear Models: $O(n^3 + mn^2)$
- Decision Tree: $O(m.mn)=O(m^2n)$ [number of voters*complexity per tree]

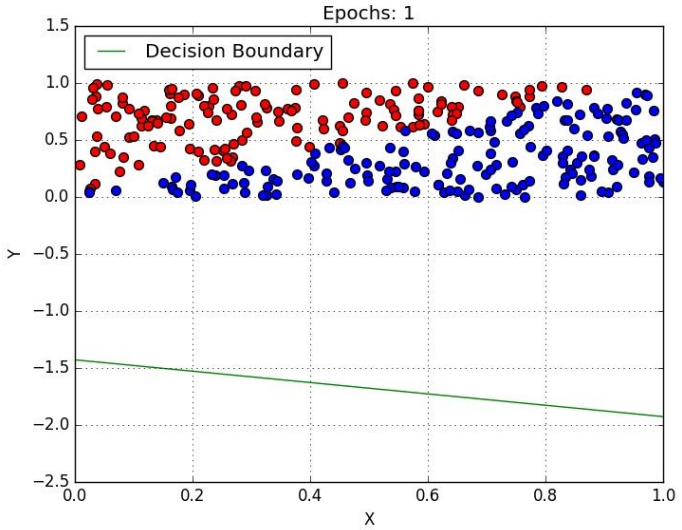
More on Regression

- **Decision Tree Regressor (link)**
- **Random Forest Regressor (link)**
- **MLP Regressor, Neural Networks (link)**
- **Support Vector Regressor(link)**
- **Gradient Boosting Regressor(link)**

- **Learn more about confidence of interval (link)**

Classification Definition

In the context of Machine Learning, "*classification* is considered an instance of supervised learning, i.e., learning where a training set of correctly identified observations is available."



Source: https://www.cs.toronto.edu/~frossard/post/classification/

https://en.wikipedia.org/wiki/Statistical_classification

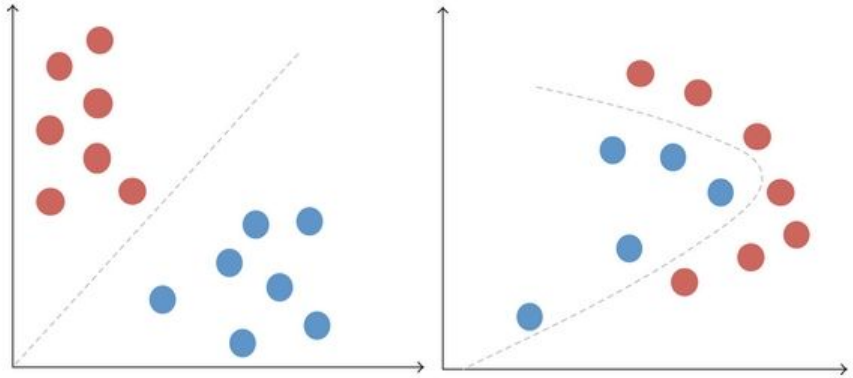
Classification Models

Linear:

- ★ Logistic Regression
- ★ Linear SVM

Non Linear:

- ★ Kernel SVM
- ★ Neural Networks
- ★ Decision Trees
- ★ K-Nearest Neighbors



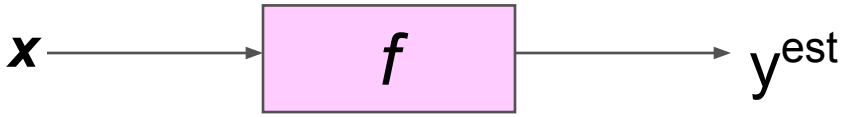
Linear Classification

Non-Linear Classification

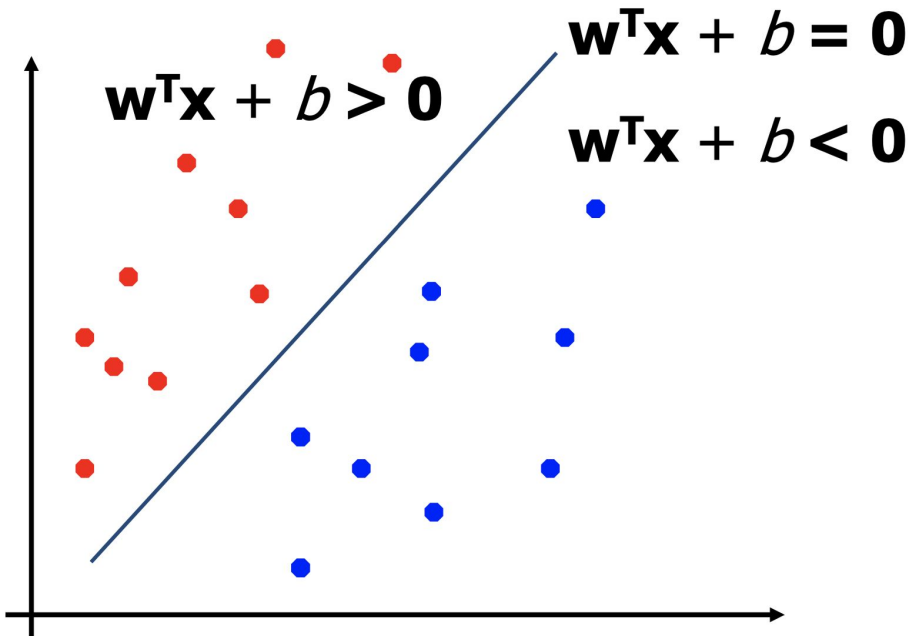
https://www.researchgate.net/figure/Linear-versus-nonlinear-classification-problems_fig4_279274803

Linear Classifiers

Estimation:



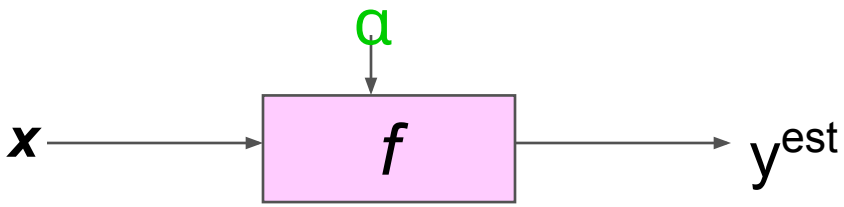
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



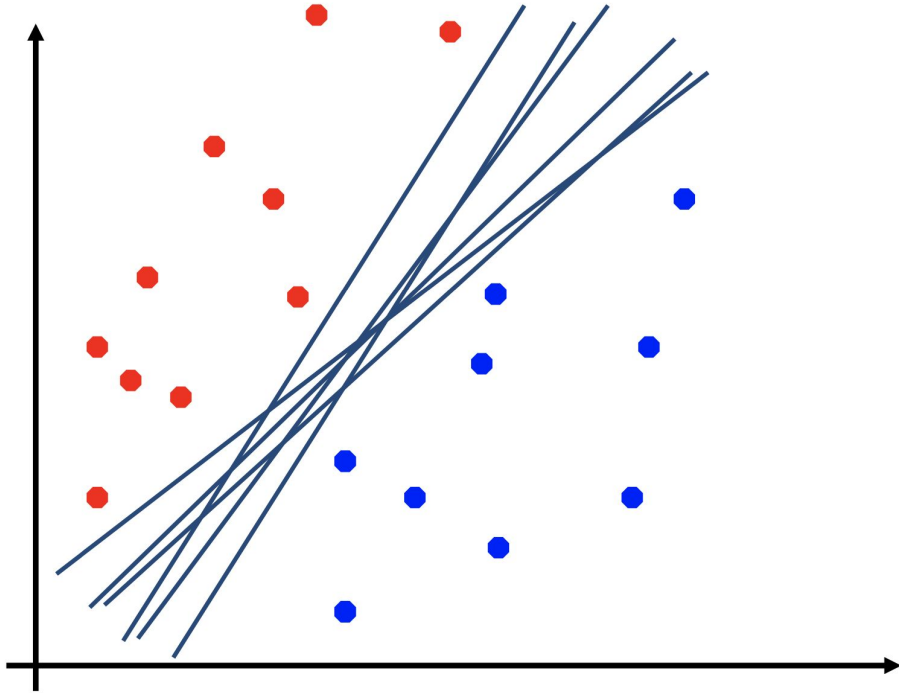
w: weight vector
x: data vector

How would you classify this data?

Linear Classifiers



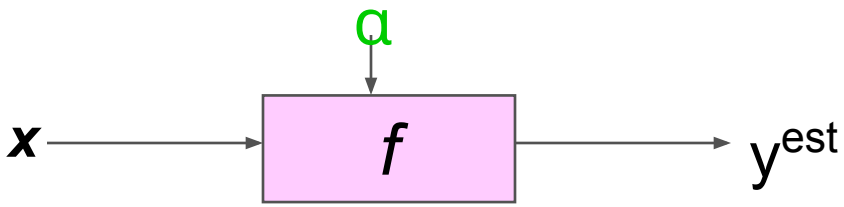
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



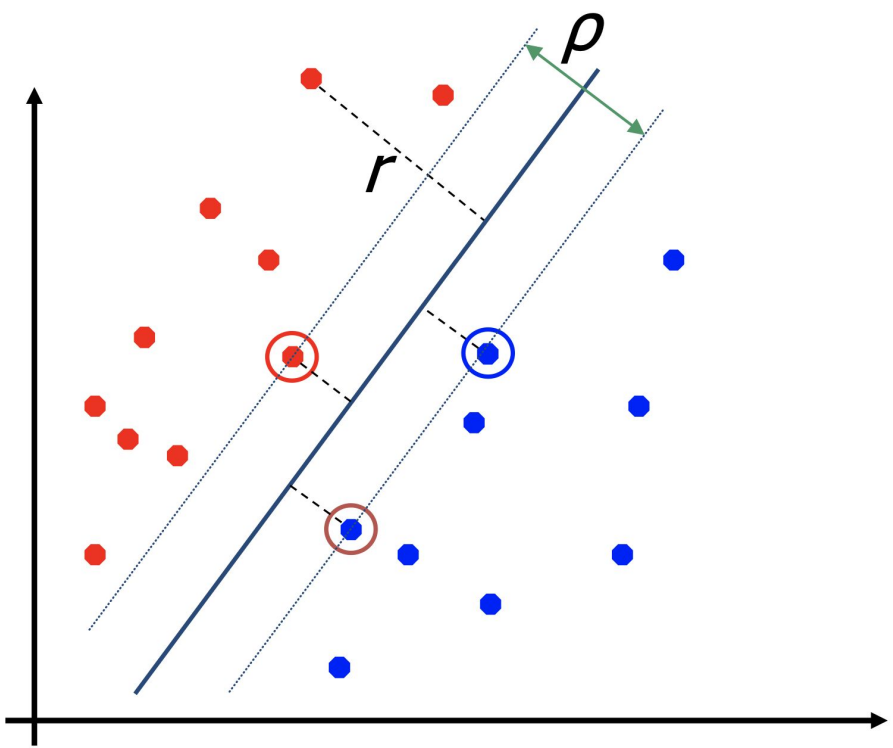
Any of these would be fine..

..but which is best?

Maximum Margin



$f(x,w,b) = sign(w.\ x - b)$



The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Support Vectors are those datapoints that the margin pushes up against

Distance from example x_i to

$$r = \frac{w^T x_i + b}{\|w\|}$$

Extension to Non-linear Decision Boundary

How to generalize it to become nonlinear?

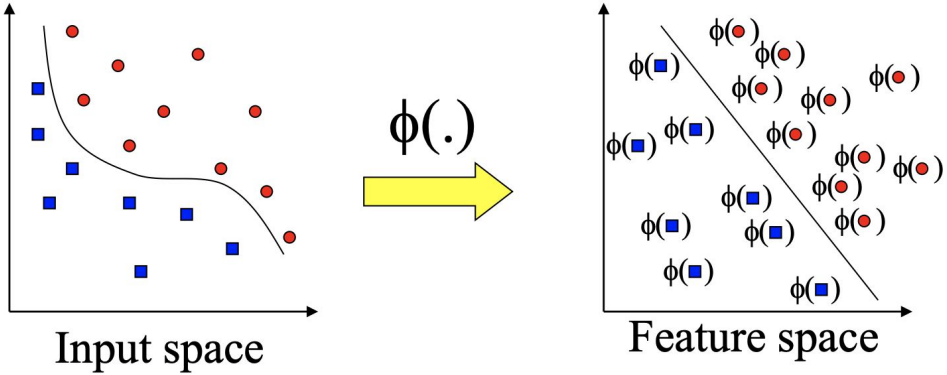
Key idea: transform $\boldsymbol{x}_i$ to a higher dimensional space

- Input space: the space the point $\boldsymbol{x}_i$ are located
- Feature space: the space of $\boldsymbol{\varphi}(\boldsymbol{x}_i)$ after transformation

Why transform?

- Linear operation in the feature space is equivalent to non-linear operation in input space
- Classification can become easier with a proper transformation. For example, adding a new

Extension to Non-linear Decision Boundary



Optimization problem

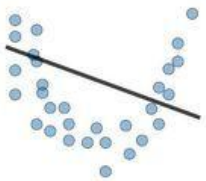
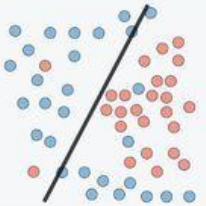
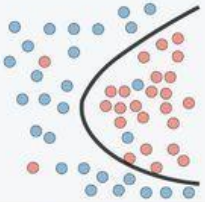
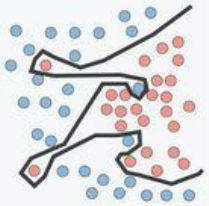
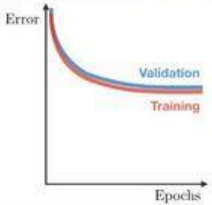
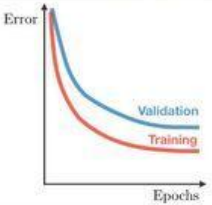
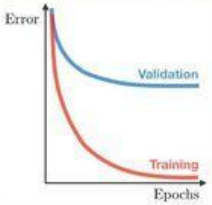
$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j, \mathbf{K}(\mathbf{x_i}, \mathbf{x_j})$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$h(\mathbf{x}) = sign\left(\sum_{i=1}^{l} \alpha_i \cdot y_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b \right)$$

Want to learn more? (link)

Bias Variance Tradeoff

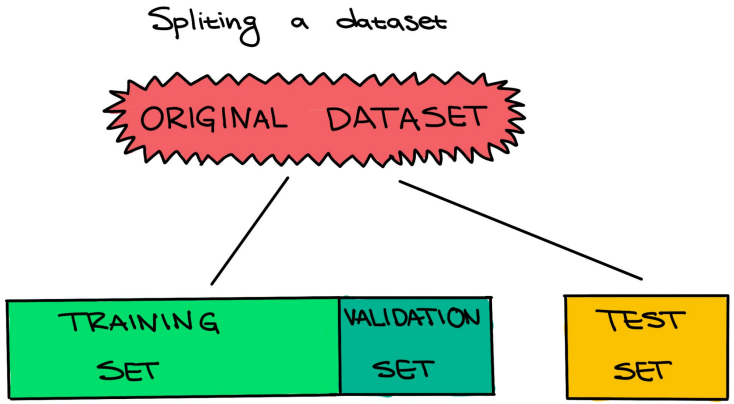
	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	• Complexify model • Add more features • Train longer		• Perform regularization • Get more data

https://www.pinterest.com/pin/618893173771969641/

Hyperparameters and Regularization

- hyperparameters are model settings or model preferences that might improve prediction measurement.
  - validation set - this is where we can play around with hyperparameters tunings

- regularizing a model means adding a penalty to cost function - the goal is to boost the prediction metrics on test dataset
  - helps overfitting or underfitting
- Splitting the data - training, test, validation



source: https://carpentries-incubator.github.io/ml4bio-workshop/02-T-cells/index.html

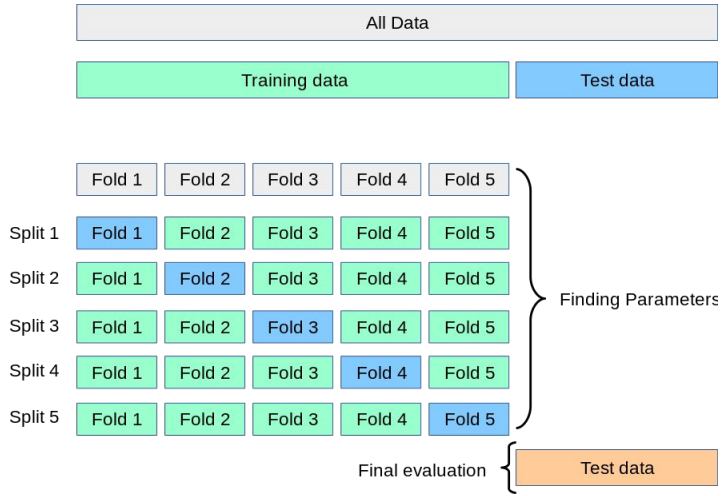
Cross validation

For one specific data and objective, we can use various methods such as Logistic Regression or K-nearest Neighbors, or SVM.

How to decide which one to use?

Cross-validation allows us to compare different machine learning methods and get a sense of how well they perform in practice.

1) Data to train the ML model
2) Data to test the ML model



source: https://scikit-learn.org/stable/modules/cross_validation.html

Steps so far

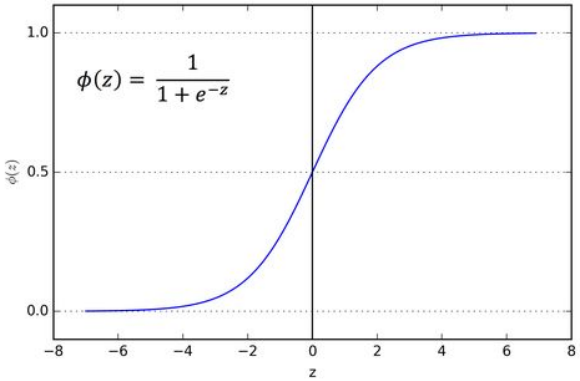
1. Data Preprocessing
2. Splitting the data
3. Pick a model for a baseline model

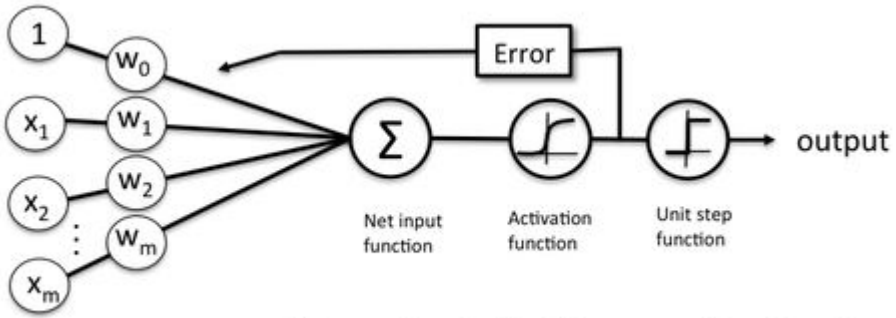
Usually we then benefit from creating the model registry or using an already existing service to enable easy model iteration.

What questions do you have?

Logistic Regression

- Linearly separable classes
- binary or multiclass classification
- predicts the probability that a certain observation belongs to a class
- logistic sigmoid function, or S-shaped function - intro to neural networks
    - logit could be thought of as a one layer neural network
    - sigmoid function often used as an activation function in nn hidden layer



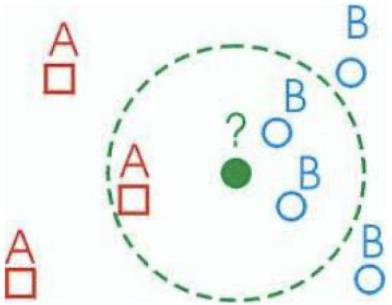


Schematic of a logistic regression classifier.

source: https://sebastianraschka.com/faq/docs/logisticregr-neuralnet.html

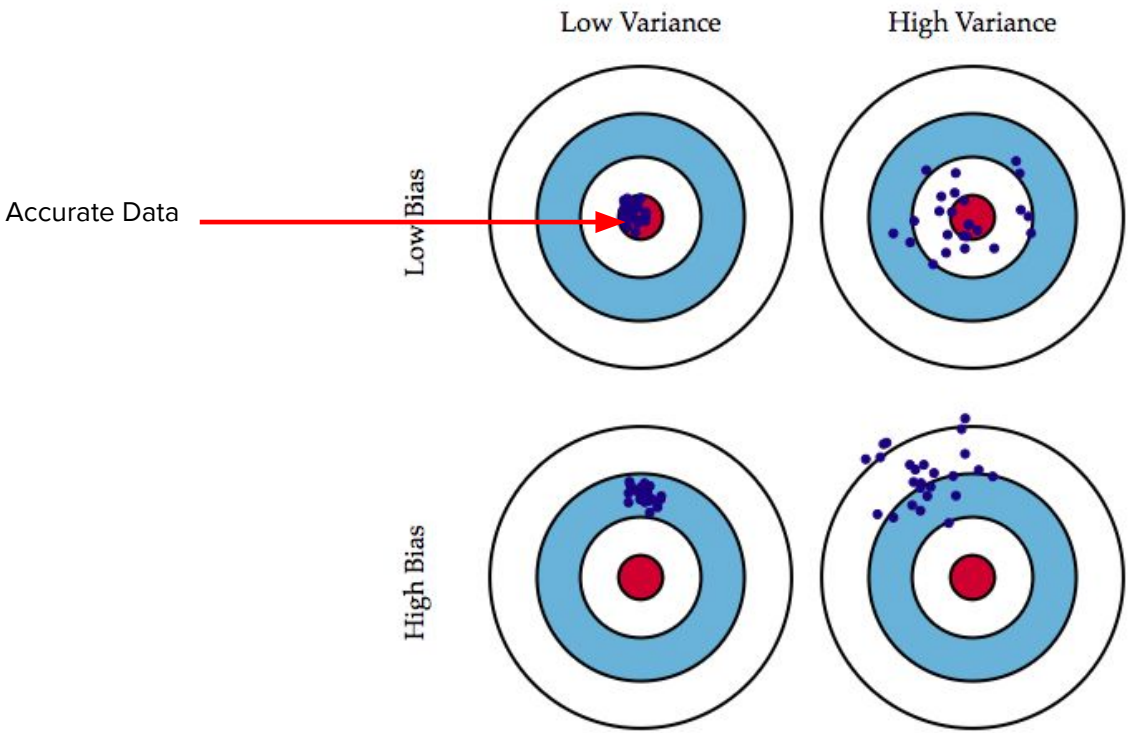
K-nearest Neighbors

- Simple, but a very powerful classification algorithm
- Classifies based on a **similarity measure**
- Non-parametric
- Lazy learning
- Does not “learn” until the test example is given
- Whenever we have a new data to classify, we find its K-nearest neighbors from the training data



Classifies by “Majority votes” for its neighbor classes

Bias Variance tradeoff k-NN



https://medium.com/30-days-of-machine-learning/day-3-k-nearest-neighbors-and-bias-variance-tradeoff-75f84d515bdb

Time Complexity of Classification Models

For ‘m’ samples with ‘n’ dimensions

- K-NN: O(kmn)

Looping to find distances in ‘n’ dimensions for ‘m’ samples.

- Logistic Regression: O(mn)

Time to solve the optimization problem

- SVM: $O(m^2)$ to $O(m^3)$

Quadratic programming can be similar to inverting a matrix of size m.

- Decision Tree: O(mlog(m)n)

Finding the best decision point per feature needs to traverse all samples.

- Random Forest: O(mlog(m)nk), for k-trees

Classification Output Metrics

Consider a case for Binary Classification data set with 100 samples (88/12% distribution).
Classification outcome is below:

All samples are either TP, TN, FP, FN

Predicted-> Actual | V	Not Sick	Sick
Not Sick	78 (TN)	10 (FP)
Sick	7 (FN)	5 (TP)

Precision=?
Recall=?
Accuracy=?
F1 score=?

Precision=TP/(TP+FP)

Recall=TP/(TP+FN)

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

F1 Score= TP/ (TP+1/2(FP+FN))
F1 Score = 2 X (Precision x Recall / Precision + Recall)

Specificity – TN/(TN+FP)

Preference of Recall vs F1 Score?

Classification Output Metrics

Consider a case for Binary Classification data set with 100 samples (88/12% distribution).
Classification outcome is below:

All samples are either TP, TN, FP, FN

Predicted-> Actual | V	Not Sick	Sick
Not Sick	78 (TN)	10 (FP)
Sick	7 (FN)	5 (TP)

Precision=TP/(TP+FP)

Recall=TP/(TP+FN)

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

F1 Score= TP/ (TP+1/2(FP+FN))
F1 Score = 2 X (Precision x Recall / Precision + Recall)

Specificity – TN/(TN+FP)

Precision=?
Recall=?
Accuracy=?
F1 score=?

Precision=33%
Recall=42%
Accuracy=83%
F1 score=37%

Preference of Recall vs F1 Score?

F1-score and measuring AUC

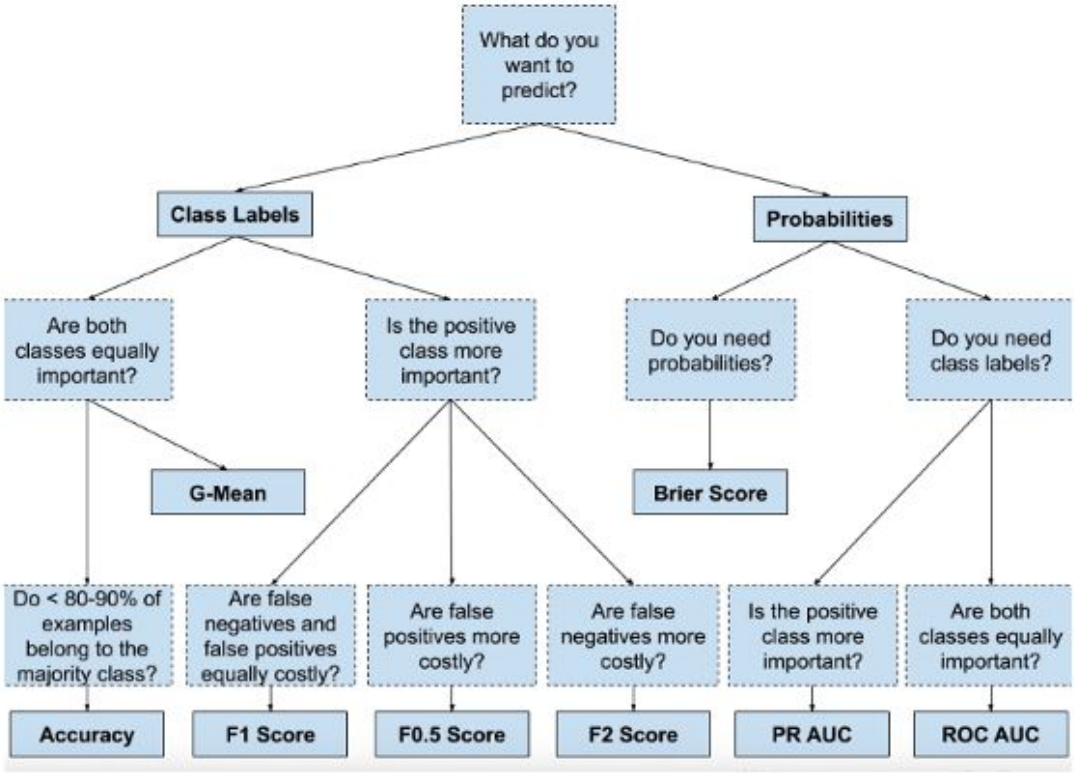
- F1 score of a class is given by the harmonic mean of precision and recall

  (2×precision×recall / (precision + recall))

  high recall + high precision : the class is perfectly handled by the model.
  The model detects the class well and is also highly trustable when it does.

- AUC

Framework for choosing metrics



What questions do you have?

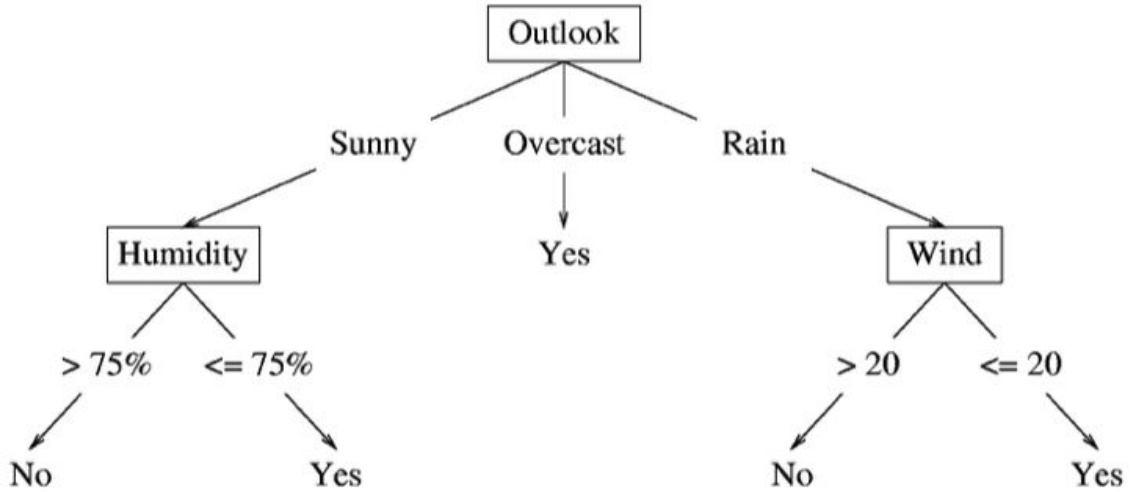
Decision trees

- Non-linear classifier
- Easy to use
- Easy to interpret
- Susceptible to overfitting but can be avoided.

In general, a decision tree makes a statement and then makes the decision based on whether or not the statement is True or False

Decision Tree

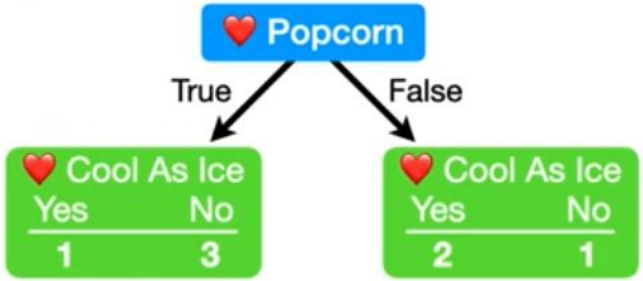
If features are continuous, internal nodes can test the value of the feature against a threshold



Basic Algorithm for Top-Down Induction of Decision Trees

1. A <- The best decision attributes for the next node
2. Assign A as a decision attribute for the node
3. For each value of A, create a new descendant of the node
4. Sort training examples to leaf nodes
5. If the training examples are perfectly classified, stop.
   Else, recurse over new leaf nodes

Gini Impurity



Gini Impurity for a Leaf = 1 - (the probability of "Yes")^2 - (the probability of "No")^2

$$= 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = 0.375 \qquad = 1 - \left(\frac{2}{1+2}\right)^2 - \left(\frac{1}{1+2}\right)^2 = 0.444$$

Total **Gini Impurity** = weighted average of **Gini Impurities** for the **Leaves**

$$= \left(\frac{4}{4+3}\right) 0.375 + \left(\frac{3}{3+4}\right) 0.444 = 0.405$$

The attributes with lowest impurity at the top of the tree

StatQuest,2022

Information Gain

- Information gain is one criteria to decide on the attribute.

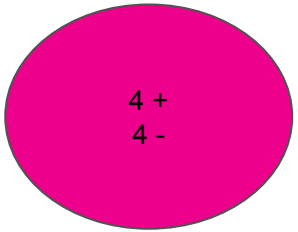
- Imagine:

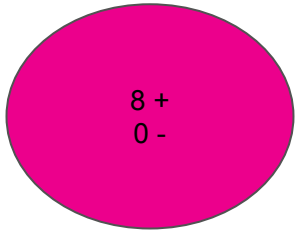
1. Someone is about to tell you your own name
2. You are about to observe the outcome of a dice roll
3. You are about to observe the outcome of a coin flip
4. You are about to observe the outcome of a biased coin flip

- Each situation have a different *amount of uncertainty* as to what outcome you will observe.

Entropy, purity

- Entropy measures the purity





The distribution is less uniform
Entropy is lower
The node is purer

Entropy

- The *expected amount of information* when observing the output of a random variable X

$$H(X) = E(I(X)) = \sum_i p(x_i) I(x_i) = -\sum_i p(x_i) \log_2 p(x_i)$$

If there X can have 8 outcomes and all are equally likely

$$H(X) == -\sum_i 1/8 \log_2 1/8 = 3 \quad \text{bits}$$

Information

- Information: reduction in uncertainty (amount of surprise in the outcome)

$$I(E) = \log_2 \frac{1}{p(x)} = -\log_2 p(x)$$

If the probability of this event happening is small and it happens the information is large.

- Observing the outcome of a coin flip is head → $I = -\log_2 1/2 = 1$
- Observe the outcome of a dice is → $I = -\log_2 1/6 = 2.58$

Conditional entropy and information gain

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i)$$

$$H(X \mid Y) = -\sum_j p(y_j) H(X \mid Y = y_j)$$

$$= -\sum_j p(y_j) \sum_i p(x_i \mid y_j) \log_2 p(x_i \mid y_j)$$

Reduction in uncertainty by knowing Y

Information gain:

(information before split) – (information after split)

$$IG(X,Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$

Which one do we choose?

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Information gain (X1,Y)= 0.61

Information gain (X2,Y)= 0.12

Pick the variable which provides the most information gain about Y

Pick X1

IG(X1,Y) = H(Y) – H(Y|X1)

H(Y) = - (5/10) log(5/10) -5/10log(5/10) = 1

H(Y|X1) = P(X1=T)H(Y|X1=T) + P(X1=F) H(Y|X1=F)

= 4/10 (1log 1 + 0 log 0) +6/10 (5/6log 5/6 +1/6log1/6)

= 0.39

Information gain (X1,Y)= 1-0.39=0.61

Overfitting

- You can perfectly fit to any training data
- Zero bias, high variance

- Two approaches:
- 1. Stop growing the tree when further splitting the data does not yield an improvement
- 2. Grow a full tree, then prune the tree, by eliminating nodes.

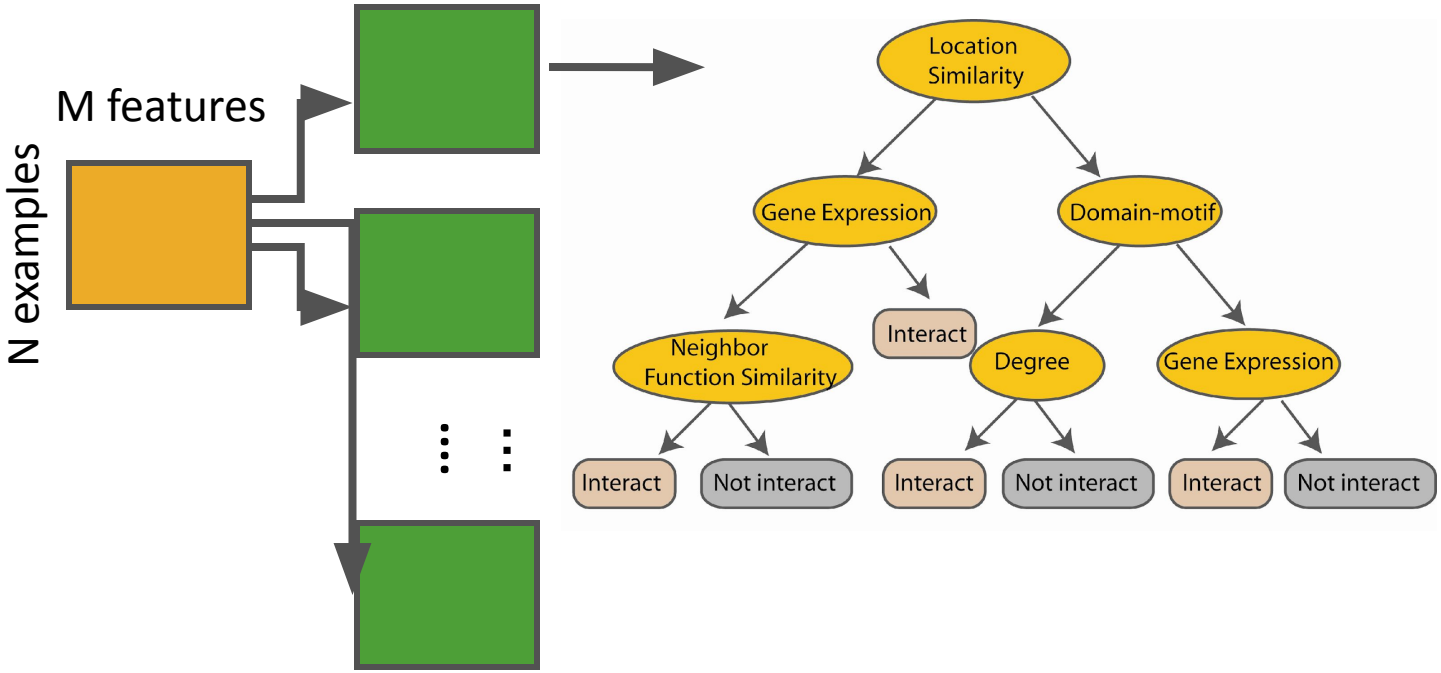
Bagging

- Bagging or *bootstrap aggregation* a technique for reducing the variance of an estimated prediction function.

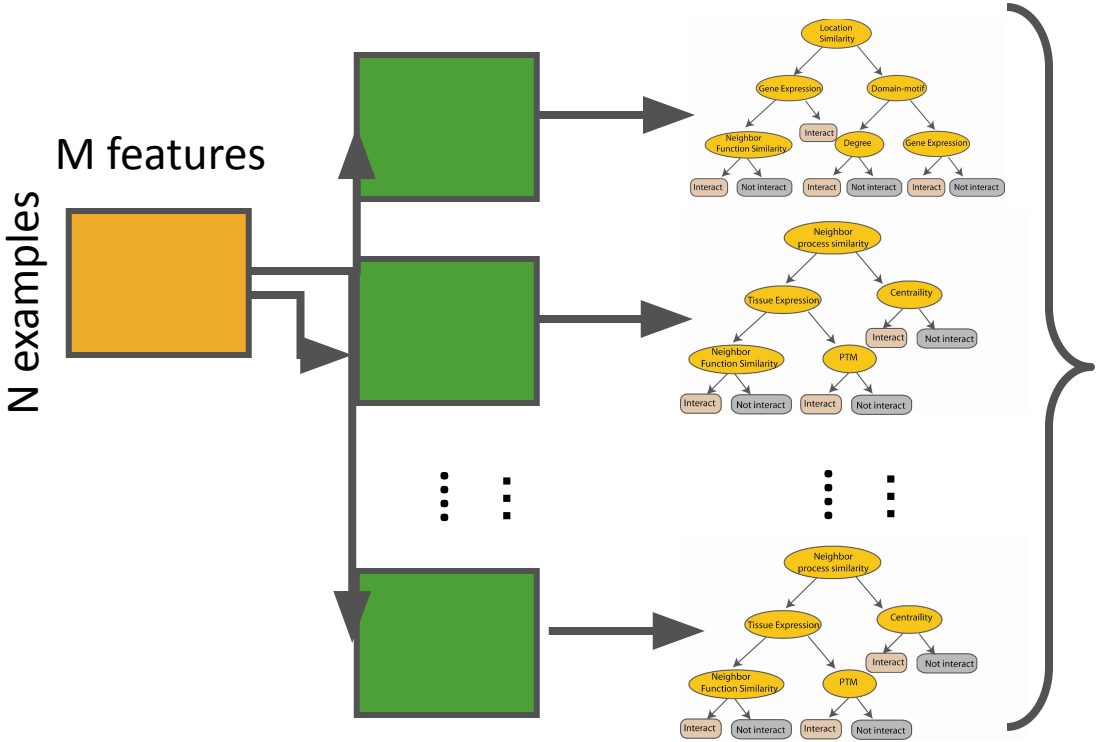
- For classification, a *committee* of trees each cast a vote for the predicted class.

Random Forest Classifier

At each node in choosing the split feature
choose only among $m<M$ features



Random Forest Classifier



Ensemble Methods

- Create an ensemble of models.
  Each uses a different set of 1,000 B and all 1,000 A

  Trained and validated separately.
  Take a majority vote of all these models.
  All data used!

Boosting

- used to reduce error by converting weak learning into a single strong learning model
- combines several weak treas and assigns weights to the outputs of the individual trees sequentially. The incorrect predictions from the first tree are given higher weights and fed into the next tree. Repeat the process into a single powerful rule.
- Types of boosting : adaptive boosting (AdaBoost), Gradient boosting, Extreme Gradient Boosting (XGBoost)
- vulnerable to outliers, hard to deploy

What is Class Imbalance?

- Training data - Detecting malignant or benign tumour
  Labels - Benign, Malignant

- 9:1 ratio!
  Leads to imbalance.

- Other cases - Fraud detection.

Undersampling

Undersample

Class A - 1000 samples
Class B - 100000 samples

Randomly pick 1000 out of class B set.

Why not undersample?

You just lost 9000 samples. Data should not go waste!

Oversampling

Generate more samples of your minority class.

- Create a generative model and then create new samples or

- Just pick existing samples with replacement (SMOTE, ADASYN)

Disadvantages of Oversampling:
Computationally expensive.
Algorithmically more involved.

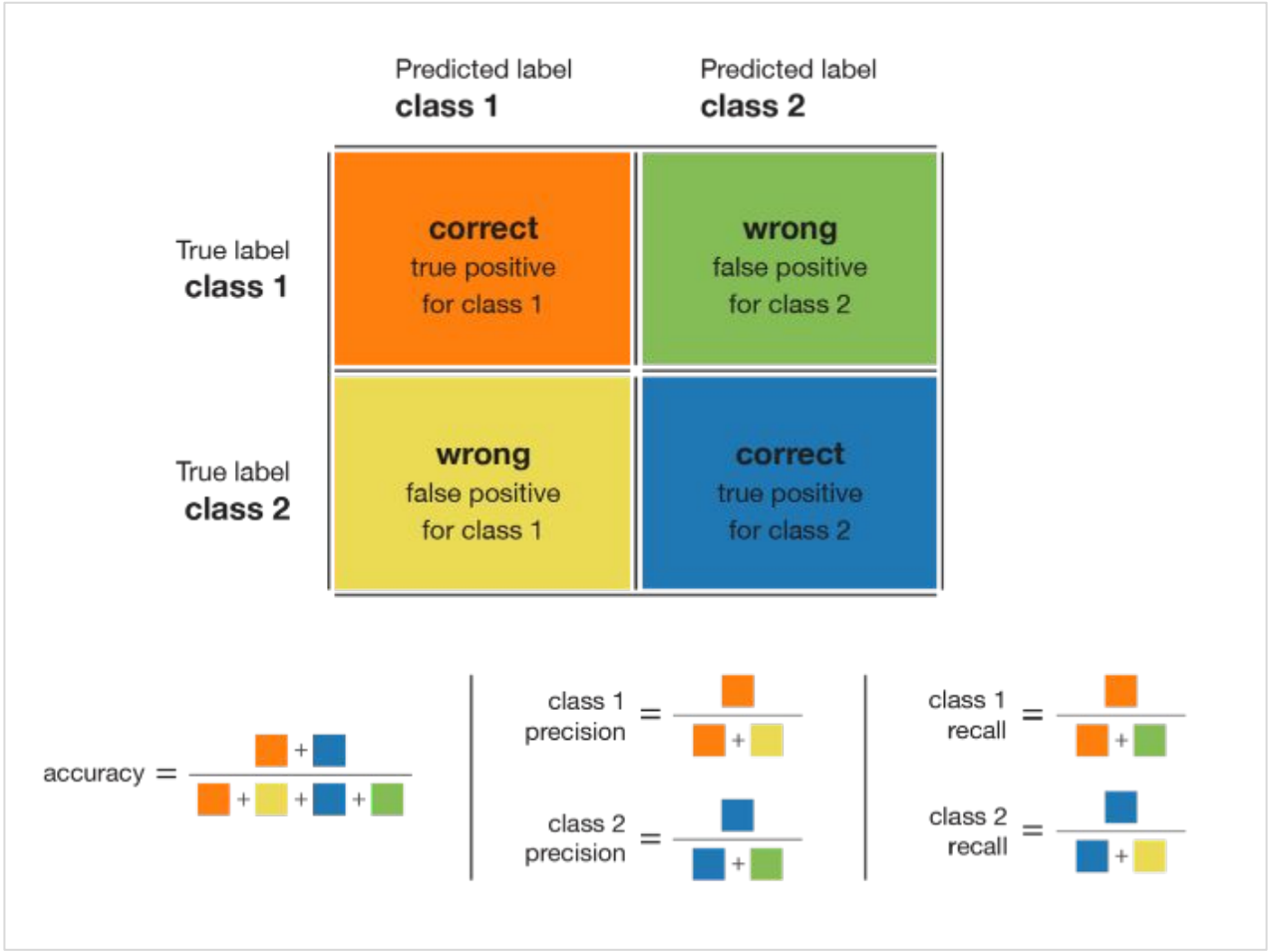
Using Hyperparameters to set weights

- Use weights as the great leveler.
- Give more weight to under-represented class.
- Specify weights in the hyperparameters of your ML algorithm.

How to evaluate a class imbalance model?

- Accuracy is a strict no no!

- Commonly, one can use the F1 score.



https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28

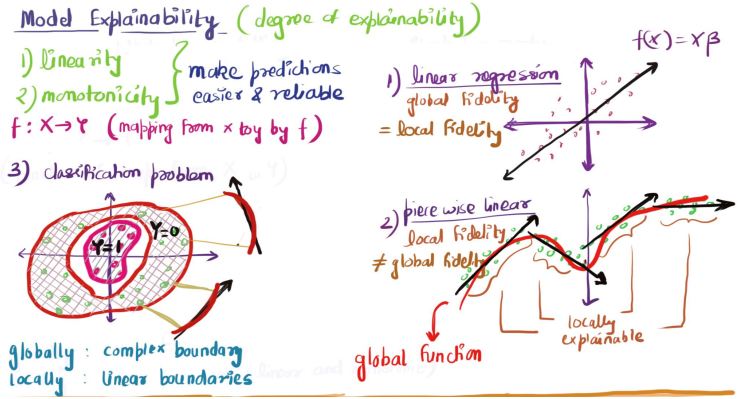
Some other ideas:

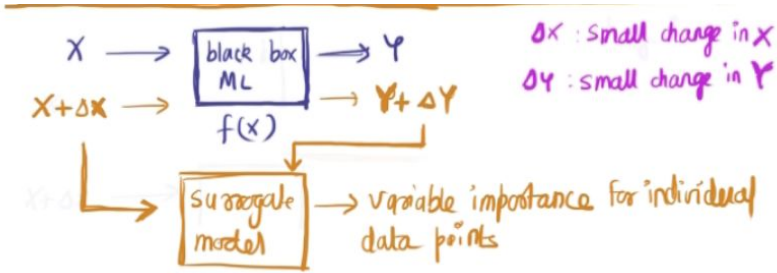
- Splitting the data into train and validation carefully
- Removing some classes in multi class classification where there are too few data points
- Check out this interesting medium article on the topic:

https://medium.com/ai-ml-at-symantec/ai-ml-security-pro-tips-class-imbalance-and-missing-labels-764fd18b7bf8

ML Model: Explain-ability vs Interpretability

"Explainability is the extent where the feature values of a sample are related to its model prediction in such a way that humans understand. In basic term, it is the understanding to the question *why is this happening*?".





https://towardsdatascience.com/idea-behind-lime-and-shap-b603d35d34eb

"Interpretability is defined as the amount of consistently predicting a model's result without trying to know the reasons behind the scene. It is easier to know the reason behind certain decisions or predictions if the interpretability of a machine learning model is higher." [Repeatable]

https://analyticsindiamag.com/explainability-vs-interpretability-in-artificial-intelligence-and-machine-learning/#:~:text=Explainability%20is%20the%20extent%20where,a%20way%20that%20humans%20understand.

Local Surrogate (LIME)

Local surrogate models are interpretable models that are used to explain individual predictions of black box machine learning models.

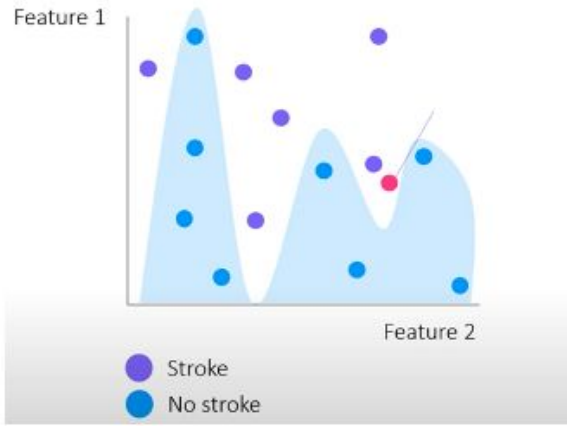
$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

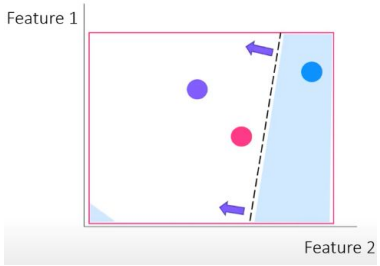
- Complex model → f
- Local model → g out of a set of interpretable models G

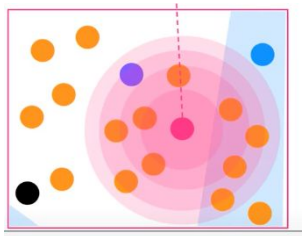
- First loss term→ approximation of the complex f by the sum of small local gs in the neighborhood of our datapoints π_x

- Second term → regularize the complexity of our simple surrogate model

Goal: good local approximation within each neighborhood







$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2$$

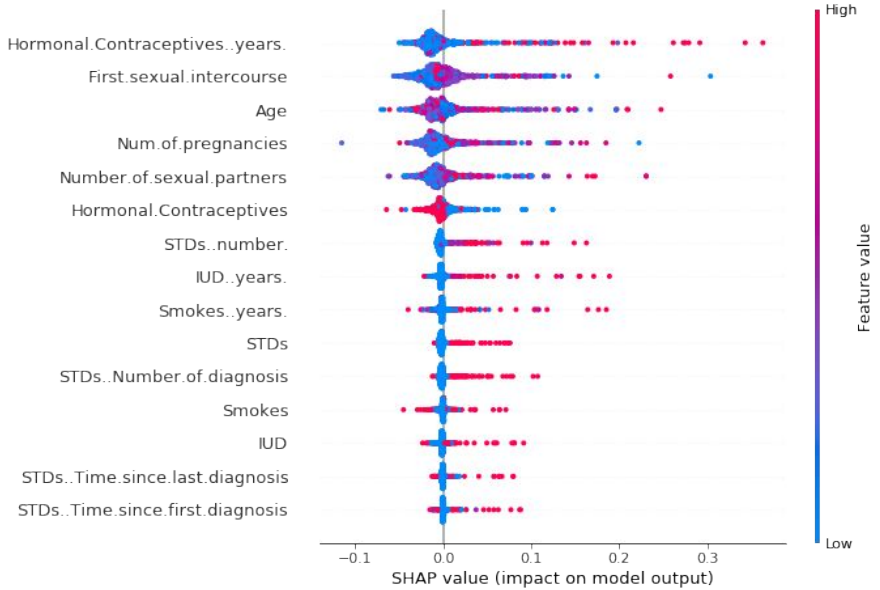
SHAP (Shapley Additive explanation) 💡

- a method to explain individual predictions. SHAP is based on the game theoretically optimal Shapley values.

- Complicated models are often required to model our natural settings, but have low interpretability. We have 2 choices to improve model interpretability:
    - reduce the complexity of the models, but also reduce model accuracy
    - develop improved, agnostic (for any model) model diagnostics

SHAP (Shapley Additive explanation)

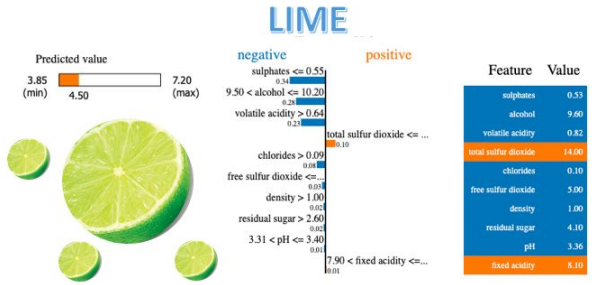
- Shapley values are adapted from game theory where the approach is used to calculate:
  - allocating resources between 'players' based on a summarization of marginal contributions to the game. Dividing up winnings between all players.
  - contribution of each predictor feature to push the response prediction away from the mean value of the response over training
  - an example of a marginal contribution includes $f(x_2, x_1) - f(x_2)$ and $f(x_1) - E[y]$ to quantify



LIME vs SHAP:

Local Interpretable Model-Agnostic Explanations (LIME)

- Shows how do we get global importance from the local importance (explainability) of a variable.
- It loosely means adding up the local weights.
- Fast and works on large batch sizes.
- Lacks stability and consistency (especially along variables with weight 0).

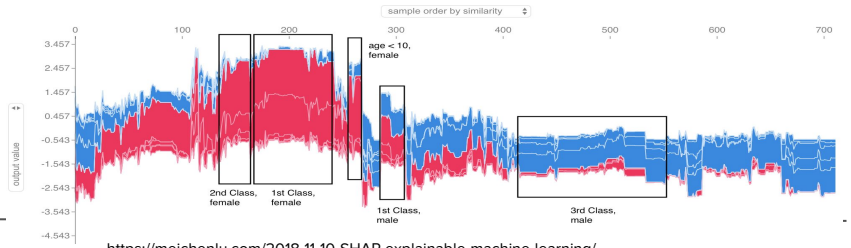


https://medium.com/analytics-vidhya/explain-your-model-with-lime-5a1a5867b423

Learn more?(Link)

SHapley Additive explanation

- Sum of SHAP values of all the variables for a data point is equal to the final prediction.
- In SHAP, we do not have to build a local model (like linear regression in LIME), rather the same function is used to calculate the Shapley values for each dimension.
- Guarantees fair contribution of variables.
- SHAP value is **NOT** the difference between the prediction with and without a variable, rather it is a contribution of a variable to the difference between the actual prediction and the mean prediction.
- Slow on large batch sizes.



https://meichenlu.com/2018-11-10-SHAP-explainable-machine-learning/

Break

Coding Sesh

Feedback on Lecture and Concepts?

See you next week!