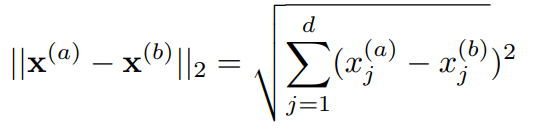
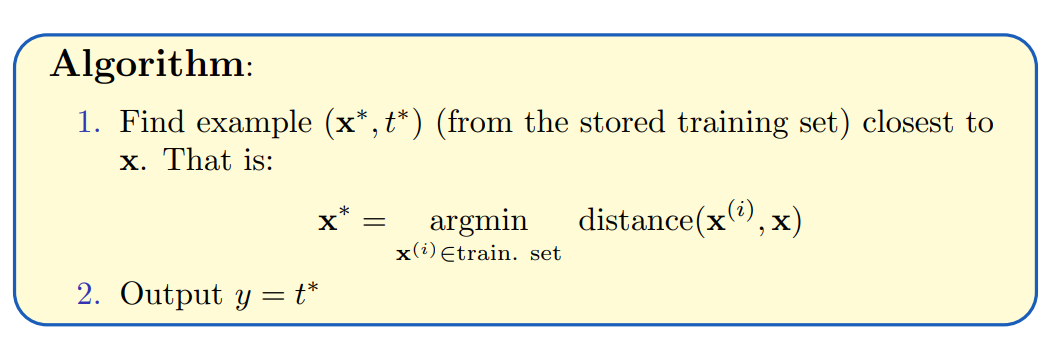
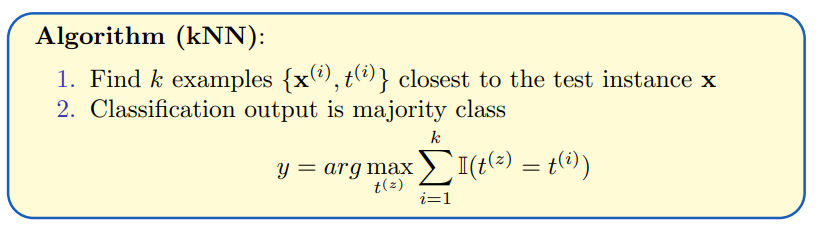
**Intro**: ML: an algorithm to automatically learn from data, or from experience, uncover patterns in data, building autonomous agents. Supervised, Semi-supervised, Reinforcement, Unsupervised learning

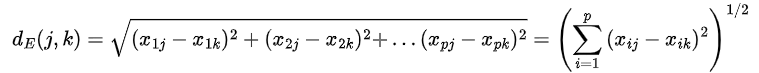
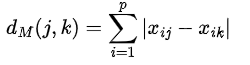
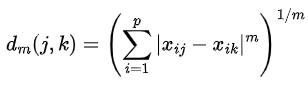
**Data**: Feature types: categorical feature and numerical feature

Accuracy = correct prediction / test instances

Error = 1- accuracy = incorrect prediction / test instances

**KNN:** a supervised classification algorithm that predicts the class of an output feature based on the class of other instances with the most similar, or "nearest," input features

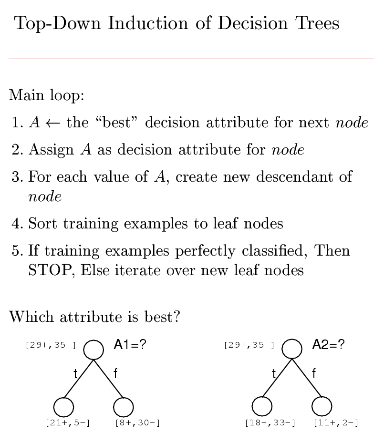
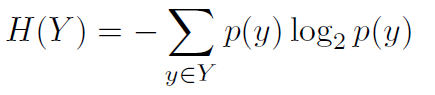
Euclidean: Manhattan distance: Minkowski distance: 

**Decision Tree:**

Information Gain of A is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A 

Entropy information theoretic measure that characterizes the impurity of a labeled set 𝑆.

𝐻 (𝑆) ≡ −𝑝⊕ log2 𝑝⊕ − 𝑝⊖ log2 𝑝⊖

the entropy of a discrete random variable Y

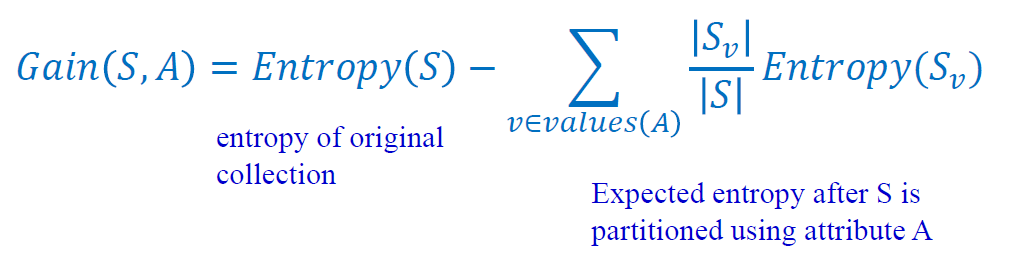
“High Entropy”:

* Variable has a uniform like distribution over many outcomes
* Flat histogram
* Values sampled from it are less predictable

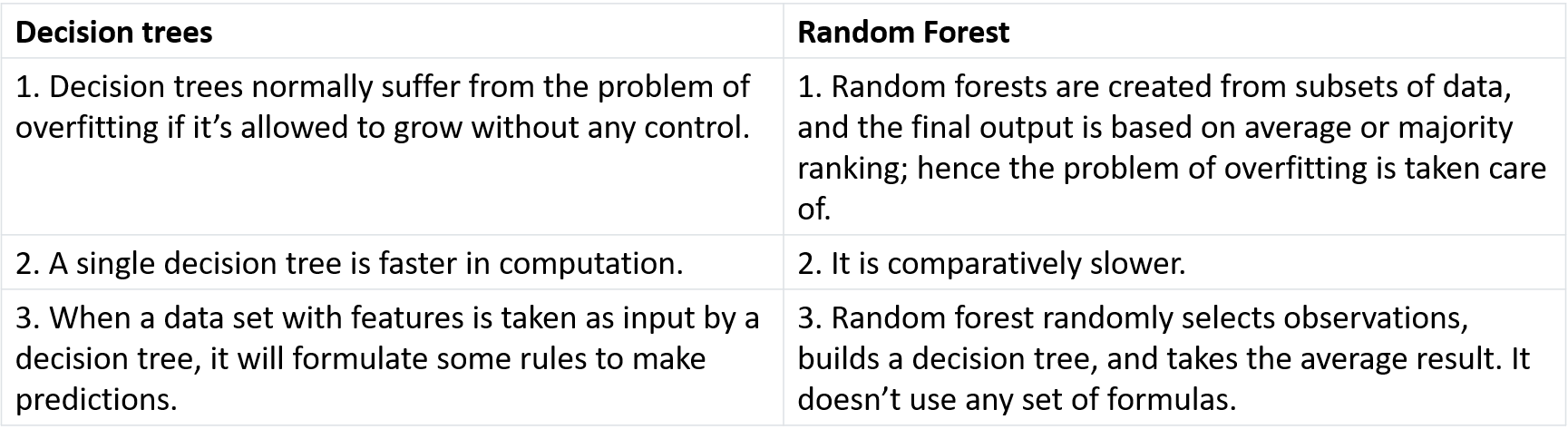
“Low Entropy”

* Distribution is concentrated on only a few outcomes
* Histogram is concentrated in a few areas
* Values sampled from it are more predictable

Gain(S,A) information provided about the target function, given the value of some other attribute A.

Conditional Entropy:

* H is always non-negative
* Chain rule: H(X, Y ) = H(X|Y ) + H(Y ) = H(Y |X) + H(X)
* If X and Y independent, then X does not affect our uncertainty about Y : H(Y |X) = H(Y )
* But knowing Y makes our knowledge of Y certain: H(Y |Y ) = 0
* By knowing X, we can only decrease uncertainty about Y : H(Y |X) ≤ H(Y )

IG(Y |X) in Y due to X, or the mutual information of Y and X: IG(Y |X) = H(Y ) − H(Y |X)

IG(split) = H(Y)-P(L)H(Y|L)-P(R)H(Y|R)

(Occam’s Razor: find the simplest hypothesis that fits the observations)

DT vs KNN:

-Simple to deal with discrete features, missing values, and poorly scaled data

-Fast at test time

-More interpretable

-Few hyperparameters

-Can incorporate interesting distance measures Advantages of neural nets over decision trees

-Able to handle attributes/features that interact in very complex ways

Amount of Overfitting: 𝑒𝑟𝑟𝑜𝑟𝑡𝑟𝑢𝑒(ℎ)− 𝑒𝑟𝑟𝑜𝑟𝑡𝑟𝑎𝑖𝑛(ℎ)

**Ensemble Learning:**

Ensembles combine classifiers to improve performance

Boosting: Train classifiers sequentially, each time focusing on training examples that the previous ones got wrong. (AdaBoost)

Boosting

* Reduces bias
* Increases variance (large ensemble can cause overfitting)
* Sequential
* High dependency between ensemble elements

Bagging

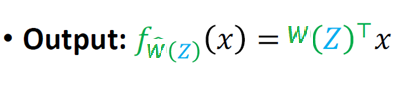
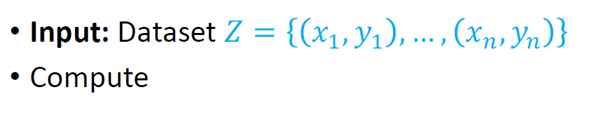
* Reduces variance (large ensemble can’t cause overfitting)
* Bias is not changed (much)
* Parallel
* Want to minimize correlation between ensemble elements.

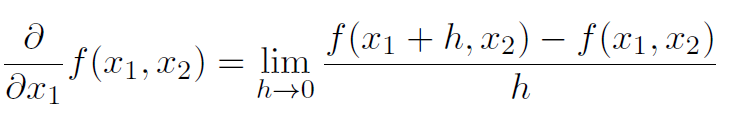
**Linear Regression**

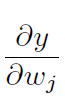
A loss function defines how bad it is.

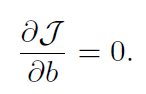
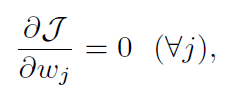
Squared error loss function:

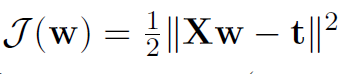
Mean Squared error = (1/n) \* Σ(yᵢ - ȳ)² loss function averaged over all training examples

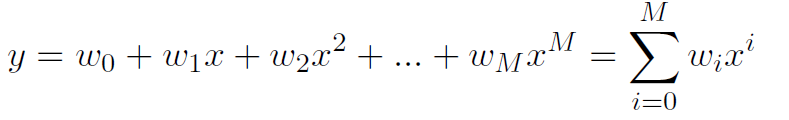


derivative of a multivariate function with respect to one of its arguments. 

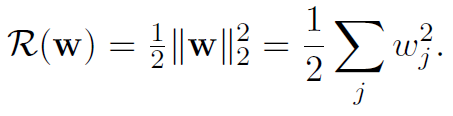


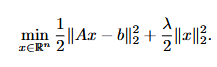
Partial derivative for 1 dimension

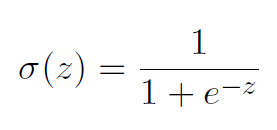
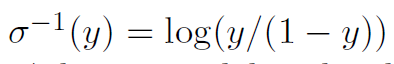
Partial derivative for d dimension  w=(XTX)−1 XTy

Polynomial Feature Mapping 

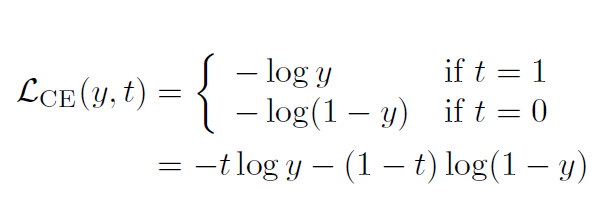
M=0, y=w0​ M = 1, y=w0​+w1​x M=3, y=w0​+w1​x+w2​x2+w3​x3

Regularizer: a function that quantifies how much we prefer one hypothesis vs. another 

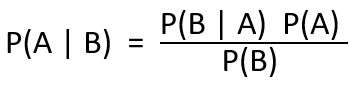
Regularized Least Squares 

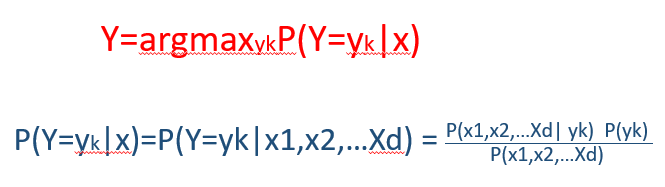
Logistic Activation Function  

A linear model with a logistic nonlinearity is known as log-linear:

Cross-entropy loss 

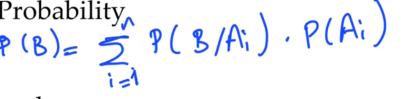
**Naïve Bayes**

Conditional probability: 

Naive Bayes classifier: 

Product Rule: P(A,B) = P(A|B)\*P(B) = P(B|A)\*P(A), P(A|B) = P(A,B)/P(B)

Sum Rule: P(A OR B) = P(A) + P(B)- P(A AND B)

Total Probability: 

Independent: P(A,B) = P(A)\*P(B)

conditional independent: P(A,B|C) = P(A|B,C)\*P(B|C) = P(A|C)\*P(B|C) = P(A|B,C)\*P(A|C)

P(Stolen) == P(Yes) \* P(Red|Yes) \* P(SUV|Yes) \* (Domestic|Yes)

P(Not Stolen) == P(No) \* P(Red|No) \* P(SUV|No) \* (Domestic|No)

**Evaluation Validation**

Confusion Matrix: a table that summarizes the combinations of predicted and actual values.

* A ***true positive (TP)***  correctly predicted as positive.
* A ***true negative (TN)***  correctly predicted as negative.
* A ***false positive (FP)***  is predicted as positive but is actually negative.
* A ***false negative (FN)***  is predicted as negative but is actually positive.

Accuracy: proportion of correct predictions. Acc= (TP + TN) / (TP + TN + FP + FN)

Precision :  the proportion of correct positive predictions

Precision = TP / (TP + FP)

Recall : the proportion of correctly predicted positive instances an

Recall = TP / (TP + FN)

F1-score: combine the precision and the recall into one measure

F1= 2 \* (Precision \* Recall) / (Precision + Recall)

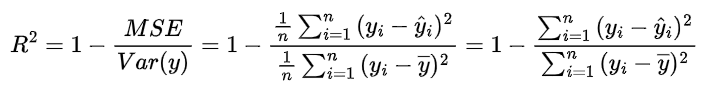
Regression Metrics: how close a model's predictions are to the actual values

Mean Absolute Error (MAE) = Σ|y\_true - y\_pred| / n

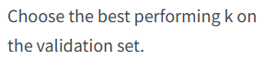
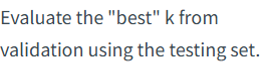
Mean Squared Error (MSE) = Σ(y\_true - y\_pred)² / n

Root Mean Squared Error (RMSE) = √MSE

R-squared (R²) = 1 - (SSres / SStot)



Model validation: the process of evaluating a model's initial performance and adjusting parameter estimates or hyperparameter settings if needed.

Validation on knn   

Preventing Overfitting

* Increase training data
* Feature selection/reduction
* Regularization techniques (L1, L2)
* Early stopping in iterative algorithms
* Ensemble methods

Addressing Underfitting

* Increase model complexity
* Add more relevant features
* Reduce regularization
* Train for more epochs (for neural networks)

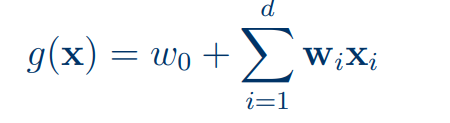
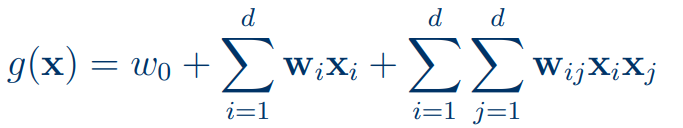
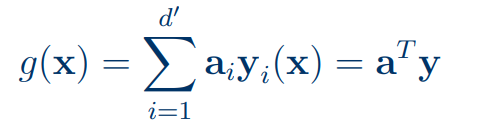
**Bayesian Networks:** conditional independence between subsets of variables

****

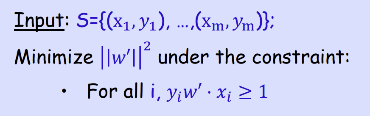
**SVM**

discriminant function :used to set a decision boundary between classes

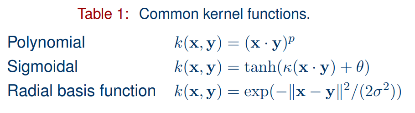
* , if g(x)>0 ‘+’, else ‘-‘

linear discriminant func quadratic discriminant func generalized linear discriminant func 

Perceptron Algorithm

* Set t=1, start with the all zero vector 𝑤1.
* Given example 𝑥, predict positive if 𝑤𝑡⋅𝑥≥0
* On a mistake, update as follows:
  + Mistake on positive, then update 𝑤𝑡+1←𝑤𝑡+𝑥
  + Mistake on negative, then update 𝑤𝑡+1←𝑤𝑡−𝑥
* Natural greedy procedure:
  + If true label of x is +1 and 𝑤𝑡 incorrect on x
    - we have 𝑤𝑡⋅𝑥<0, 𝑤𝑡+1⋅𝑥←𝑤𝑡⋅𝑥+𝑥⋅𝑥=𝑤𝑡⋅𝑥+𝑥2, so more chance 𝑤𝑡+1classifies x correctly.
  + Similarly for mistakes on negative examples.

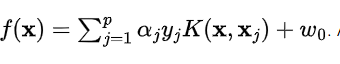
Geometric Margin: Large margin can help prevent overfitting.

A dataset is completely separated if all instances of each class fall on the same side of a hyperplane with no overlap.

A soft margin is a margin boundary that allows a few instances to be misclassified but with an associated penalty.

A support vector classifier categorizes instances using a hyperplane with a soft margin.

Support Vector Machines (SVMs): Directly optimize for the maximum margin separator

decision function for a new instance u is minimize 