* **SUPERVISED LEARNING**
  + Given X(inputs), Y(target)
  + Try to learn to predict target given input
  + 2 main functions: train(X,Y) and predict(X)
  + 2 Types
    - Classification
      * Labels are discrete
        + Whether or not it will rain tomorrow
        + Whether or not Google’s stock price will rise or fall tomorrow
    - Regression
      * Labels are real numbers
      * Trying to predict mm of rainfall tomorrow
      * Trying to predict the value of Google’s stock price
* Unsupervised Learning
  + Given X only
  + Try to learn structure of the data, density estimation(p(X)), clustering
* Data Types and Shapes
  + X = matrix of shape NxD
    - N = number of samples
    - D = number of features
  + Y = vector of shape N X 1
    - Regression: will contain float values
    - Classification: will contain integers from 0…K-1 where K = # classes
* Generalization
  + Generalization: predict accurately not only for data we trained on but new data we haven’t seen before
  + Usually split data into train/test sets to get an idea of how well a mode will generalize
* K-NEAREST NEIGHBORS CONCEPTS
  + Basic Premise: To make a prediction, use closest known data points
  + When can KNN fail?
* NAÏVE BAYES
  + Grounded in probability, which can be powerful
  + We will purposely make this model not powerful (‘naïve’)
  + Example
    - We want to determine if an email is spam
    - Can look at words like: free, pills, money
    - We want to find
      * P(‘money’|spam)
    - P(‘money’|not spam)
  + What makes this Naïve?
    - Consider p(‘cash’|spam)
    - Is it correlated with p(‘money’|spam)?
    - Probably but we assume independence
    - With naïve Bayes we assume all of the features are independent
    - P(all words | spam) = p(word1 |spam) \* p(word2|spam)
* NON-NAÏVE BAYES
  + Usually we just call it ‘Bayes Classifier’
  + More generally we can have a ‘Bayes Model’
    - Can either do classification or regression
* DESCRIMINATIVE VS GENERATIVE CLASSIFIERS
* Discriminative
  + Classifiers like logistic regression model this directly (discriminative)
  + We start with X, we get Y
* Generative
  + We start with Y(the class) and model X
  + Think of each class as a ‘data-making machine’
  + It ‘generates’ the data
  + Naïve Bayes
* DECISION TREE
  + Basically a bunch of nested if statements
  + What makes it ML is how we choose the conditions
  + Based on information theory
  + One key feature: We only look at one attribute at a time
    - Each condition checks only 1 column of X
  + Usually call these ‘input features’ but called ‘attributes’ when talking about decision trees
  + leaf Node = where you make the prediction
* INFORMATION ENTROPY
  + Related to variance
  + Wide variance: we don’t know much about the data we will get
  + Slim variance: we can be more confident about the data we will get
  + Entropy is a measure o much information we get from finding o the value of the random variable
* HYPERPARAMETERS
  + Key point: we want the classifier to perform well on data it hasn’t seen before (generalization)
    - Sometimes we call this data ‘test data’
  + If we are using it to choose hyper parameters or a model we might call it ‘validation data’
* K-fold Cross Validation
  + Popular method for choosing hyper parameters
  + Split data into K parts (typical values for K = 5, 8 ,10)
  + Loop K times
  + In each iteration, take 1 part out (use it for validation), use the rest for training
  + Returns K different scores (accuracies)
  + Can simply use the mean
  + Can also use statistical testing to check if one hyper parameter setting is ‘statistically significantly’ better than the other
* FEATURE EXTRACTION AND FEATURE SELECTION
  + Feature Extraction
    - Lesson: we require intimate knowledge of the data
    - Many times, we need ‘domain knowledge’
* Feature Selection
  + Suppose you are a domain expert, and you’ve extracted features
  + How to choose the best ones?
    - Keep too many > over fit
    - We want to keep just the most powerful and discriminatory
  + Greedy Method
    - Build a classifier for each individual feature, pick the best one via cross-validation
    - Build another set of classifier, all of which contain the first (best) feature, and one other feature. Pick the best via cross-validation. Now you have two features
    - Repeat.
* Automatic Extraction/Selection
  + Principal component Analysis
  + A dimensionality reduction technique
  + Automatic – doesn’t require domain knowledge
  + Several desirable properties
    - All outputs are uncorrelated (no redundancy)
    - Outputs are sorted by information contained (measure by variance)
    - We choose enough features such that we retain 95% or 99% (or some other threshold ) of the original variance (this is feature selection)
    - Disadvantage: it’s only a linear transformation
* FIT > PREDICT > SCORE
  + Fit aka Train
  + Two functions:
    - Fit(X,Y)
    - Predict(X)
* Whether supervised or unsupervised learning, we know that there’s a cost function to minimize
  + Squared Error (typical for regression)
* **BOILER PLATE CODE**
* Class MyModel:
* def predict(sself, X):
* pass
* def fit(self, X, Y, eta, T):
* pass
* X, Y = load\_my\_dataset()
* Model = MyModel()
* Model.fit(X,Y)
* HOW TO CODE INDEPEDENTLY
  + All data is the same – no matter the domain or industry
  + Interfaces to the algorithms are also conveniently all the same: fit/predict
  + Or fit/(transform) in the unsupervised case
  + Mix and Match: Any data, any algorithm