Lecture 12 Model Evaluation

- Confusion Matrix
  - Accuracy = (a+d)/(a+b+c+d)

	Predicted Class	
	Class = Yes	Class = No
Actual Class: Class = Yes	a (TP)	b (FN)
Actual Class: Class = No	c (FP)	d (TN)

  - TP = true positive
  - FP = false positive
  - FN = false negative
  - TN = True Negative
  - Accuracy can be misleading
    - If one particular class has a majority of the data set, a predictor that only predicts that class will yield a very high accuracy
- Cost Matrix

	Predicted Class	
	Class = Yes	Class = No
Actual Class: Class = Yes	C(Yes|Yes)	C(No|Yes)
Actual Class: Class = No	C(Yes|No)	C(No|No)

COST	Predicted Class	
	Yes	No
Actual Class: Yes	a	b
Actual Class: No	c	d

  - Precision = a /(a+c)
  - Recall = a/(a+b)
  - F-Measure = 2RP /(R+P)
  - Total Cost=TP·Cost(TP)+FP·Cost(FP)+FN·Cost(FN)+TN·Cost(TN)

Cost of Classification

COST	Predicted Class	
	Yes	No
Actual Class: Yes	-1	100
Actual Class: No	1	0

Model 1	Predicted Class	
	Yes	No
Actual Class: Yes	150	40
Actual Class: No	60	250

Accuracy = 80%
Cost = 3910

Model 2	Predicted Class	
	Yes	No
Actual Class: Yes	250	45
Actual Class: No	5	200

Accuracy = 90%
Cost = 4255

- Methods of Estimation
  - Goal: get a reliable estimate of performance of the model on unseen data
  - Holdout:
    - Split data into two sets a testing and a training set
    - Use ¼ of the dataset for testing and use ¾ for training
  - Cross Validaton
    - Split data into K equal sized folds (subsets)
    - Train the model on K-1 fols
    - Test the moden on the remaining 1 fold
      - Repeat this K times so each fold serves at a test set once
      - Average the performance metrics over all k runs
    - Partition into K disjoint subsets
    - K-fold: train on K-1 partitions , test on the remaining
    - K=n leave one out
      - Special case where K=n and n is the number of data points
      - For each example
        - Train on all other n-1 samples
        - Test on the one left out
          - This is very accurate but computationally expensive for large datasets

Method	Accuracy	Speed	Overfitting Risk
Holdout	Medium	Fast	Higher
K-Fold CV	High	Moderate	Lower
Leave-One-Out	Very High	Very Slow	Very Low

- Ensemble Methods
  - Ensembling in machine learning is the technique of combining multiple models (often called "learners" or "classifiers") to improve overall performance — usually in terms of accuracy, robustness, or generalization.
  - Reduces error
  - Increases stability
- Suppose you have 17 independent classifier each with an error rate of e = 0.20 and you take a majority vote to make a final decision
  - The majority needs to make a mistake and get at least 9/17 wrong
  - What is the change the ensemble gets it wrong
    - Binomial probability problem

$$P(X \geq 9) = \sum_{k=9}^{17} \binom{17}{k} (.2)^k (1 - .2)^{17-k} = 0.002581463$$

      -
      - This value is much smaller than 0.20
      - This shows how combining multiple weak learners can drastically reduce error if they are independent and better than random

- Bagging
  - Goal : reduce varible by training classifiers on different subsets of the data
  - Generate bootstrap samples by randomly sampling from the dataset to create multiple training sets
  - Train a separate model on each bootstrap sample
  - Combine their predictions (majority vote or average)
  - Good for unstable learners like decision trees
  - Random forests is bagging applied to decision trees
- Boosting
  - Reduce bias by focusing on errors made by previous models
    - Train the first classifier
    - Increase the weights of misclassified points
    - Train the next classifier to focus more on those errors
      - Repeat.
    - Combine all classifiers using a weight vote
  - Learners are not independent but work in sequence to correct each other