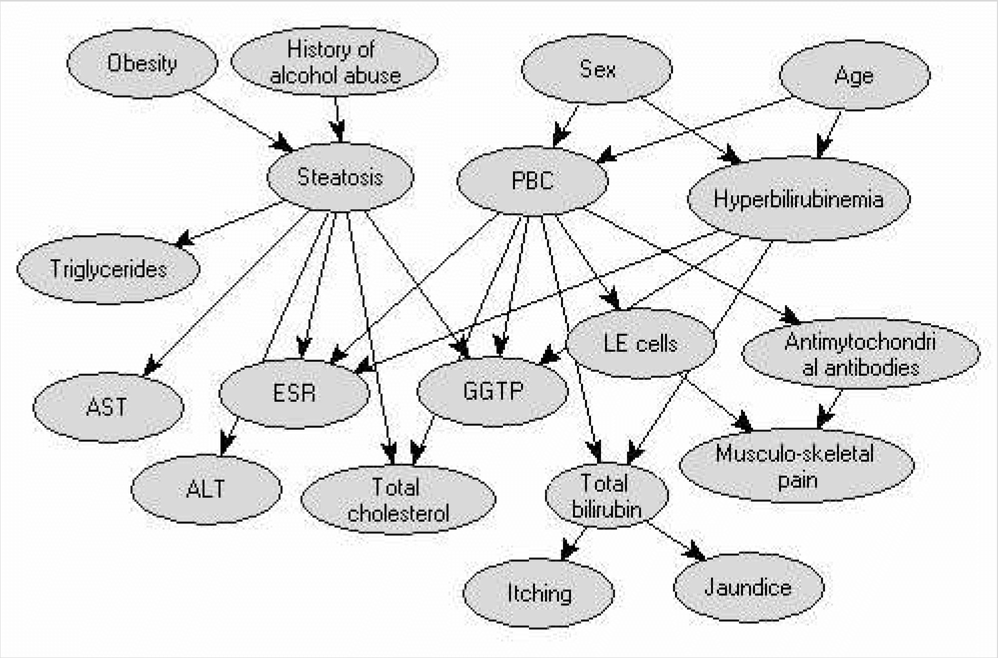
K NEAREST NEIGHBORS

Paul Speaker

Bayesian Wrap-up

- We have only scratched the surface of Bayesian Classification Models
- Steps in complexity
  - Naïve Bayes (no dependence)
  - LDA and QDA → $P(X_1 | X_2)$
- Next steps: further dependencies
  - $P(X_1 | (X_3 | X_2))$ for example
  - **Bayesian Networks**
  - Many applications, particularly NLP



$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

K-Nearest Neighbors

- K nearest neighbors (KNN) is another machine learning algorithm used for classification.
- Basic idea: KNN works by identifying the K “nearest” data points to a given test data point and using them to determine the class or value of the test data point.
  - Has to use a train/test split
  - No model to build with train dataset
  - Simply make predictions on test dataset with k-nearest from training dataset
- Non-parametric—no explicit parameters for model
- Of all classification models, hardest to interpret probabilistically

“Nearest”

- “Nearest” is a more flexible concept than you might think at first
  - First requirement—”nearest” means only use continuous variables
  - Sometimes, binary categorical variables can be converted to 0/1 for purposes of KNN
- Problem: different X’s have different scales
  - If one X has much larger spread (standard deviation) then others, it will dominate the distance calculation
  - Solutions: re-scaling X’s
  - Normalized X → $X' = (X - \mu)/\sigma$
    - (Note: subtracting mean does not really affect KNN, but it’s often a standard part of scaling algorithms)
- Another potential issue—multiple ways of measuring distance
  - Standard—square root (sum of squares of differences) – like RMSE
  - “Taxicab”—sum of (absolute value of differences) – like MAE

KNN in R

- Command to do KNN is…knn
- Format:
  - knn(train = <train dataset>, test = <test dataset>, cl = <class field in train>, k=<number>)
- Scaling X's can be done with scale() command
  - before putting into knn,
    - train_scale = scale(train) test_scale = scale(test)
    - Or scale before split
- Other options:
  - prob = <TRUE/FALSE>
    - Default is FALSE
    - If TRUE, returns proportion of K nearest that are one class or other (sort of a probability)
  - use.all → handling of ties (for binary target, best to use odd K)
  - l = (deviations from majority rule, like a classification threshold)

Advantages and Limitations

- Advantages of KNN
  - Intuitive (I’ve never seen KNN rejected because results didn’t make sense)
  - Works especially well for multi-valued (more later)
  - Non-parametric means highly nonlinear relationships can be captured

- Limitations of KNN
  - Suffers particularly hard from curse of dimensionality
    - Struggles more than most models with large numbers of inputs
  - Multivalued categorical inputs not really useful
  - Computationally expensive for larger datasets (search for nearest is hard)
  - Also struggles more than most with imbalanced datasetws

Enhancements

- Most important enhancement to KNN is weighted KNN
- Idea is to weight closer point more heavily
  - Even if 4 out of 5 closest points are one category, it is possible to classify as second category if 5th point is close enough
- Calculate with weighted average
  - Weights are usually reciprocal of the distance
- Another option: “all within”
  - Rather than pick K and computing the closest, you pick a distance and pick all points within the fixed distances
  - Still can use distance weighting
  - My experience: often outperforms KNN
    - In R: frNN (does not weight by distance)
    - Python: sklearn.neighbors.RadiusNeighborsClassifier (can be weighted or not)