Key Machine Learning

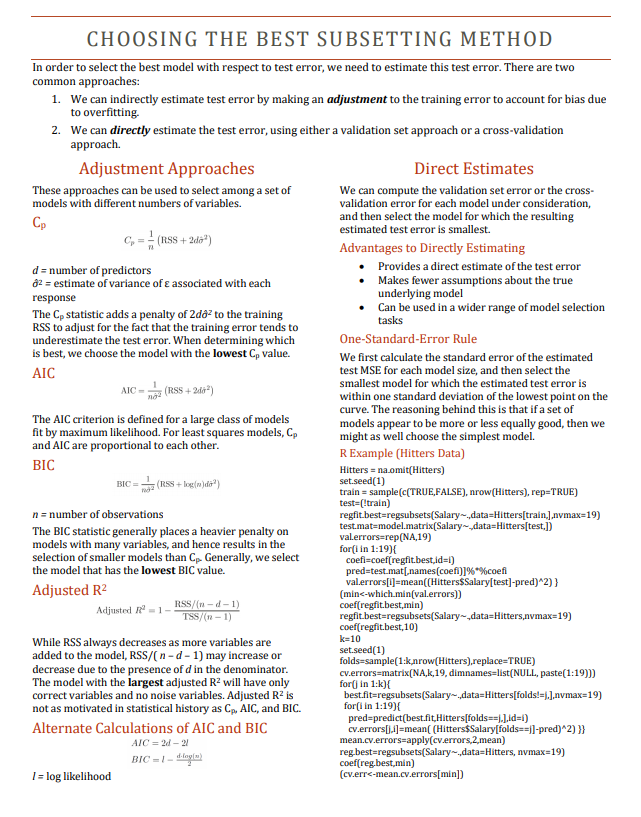
* In regression method, the most commonly used measure is the mean squared error. MSE is small if the predicted responses are very close to the true responses and will be large if some of the observations, the predicted and true responses differ substantially.
* Expected MSE: refers to the average test MSE that we would obtain if we repeatedly estimated f using a large number of training sets and tested each x0
* Variance refers to the amount by with f hat would change if we estimated it using a different training set. If a method has a high variance then small changes in the training data can result in large changes in f hat. More flexible statistical methods have higher variance
* Bias: error that is introduced by approximating a real-life problem. More flexible methods result in less bias.

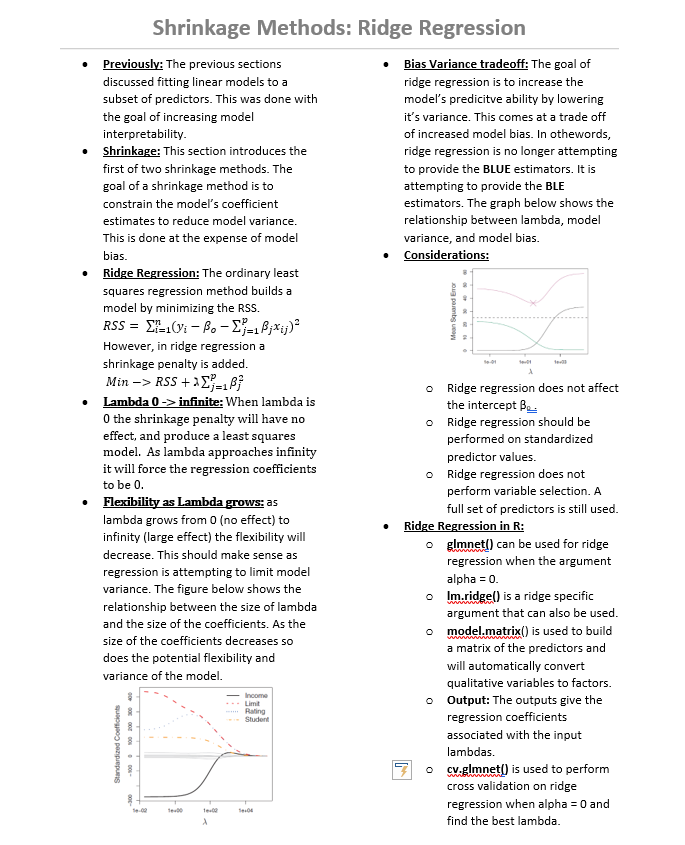
K-Nearest Neighbors

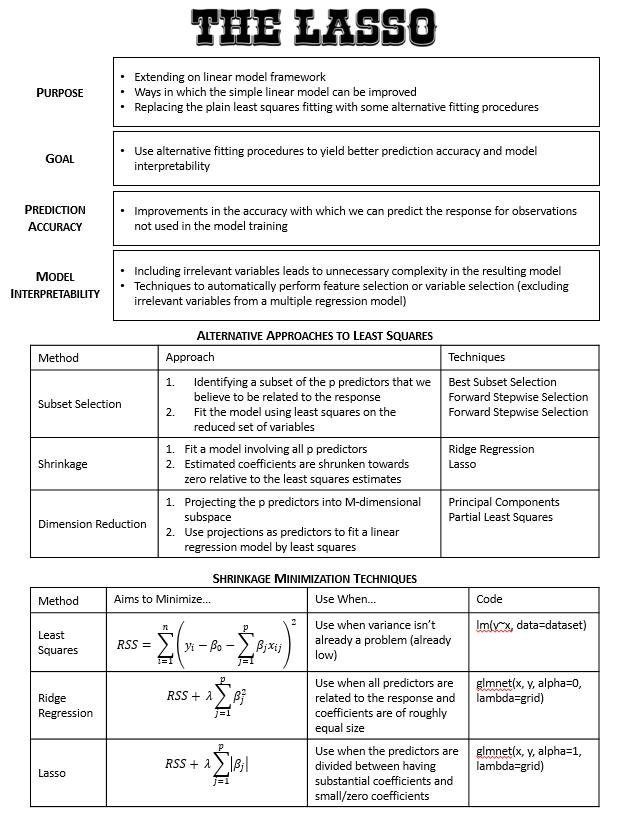
* When K = 1, the decision boundary is overly flexible and finds patterns in the data that don’t correspond to the Bayes decision boundary. Has low bias, but very high variance. As K grows, the method becomes less flexible and produces a decision boundary that is close to linear. Low variance but high bias
* Need to install “class” package
* The K-nearest neighbors (KNN) classifier is an example of a classification method that attempts to estimate the Bayes conditional distributions, and then classify a given observation to the class with highest estimated probability
* The knn() function requires 4 parameters:
  + 1. A matrix containing the predictors (the X's) associated with the training data
  + 2. A matrix containing the predictors (the X's) associated with the data for which we wish to make predictions, for the validate set, for the training set
  + 3. A vector containing the class labels (the Y's) for the training observations
  + 4. A value for K, the number of nearest neighbors to be # used by the classifier.

Linear Regression

* Residual – Difference between the ith observed response value and the ith response value that is predicted by the model
* RSS – Residual Sum of Squares
* Null hypothesis – There is no relationship between X and Y
* Alternative hypothesis – There is a relationship between X and Y
* If we see a small p-value then we can infer that there is an association between the predictor and the response – we reject the null hypothesis – that is, we declare a relationship to exist between X and Y – if the p-value is small enough.
* RSE – Residual Standard Error – is an estimate of the standard deviation of the error – roughly speaking, it is the average amount that the response will deviate from the true regression line. It is considered a measure of the lack of fit of the model.
* R-Squared - provides an alternative measure of fit. It takes the form a proportion – the proportion of variance explained.
* TSS – Total Sum of Squares – measures the total variance in the response Y – can be thought of as the amount of variability inherent in the response before the regression is performed. In contrast, RSS measures the variability that is left unexplained after performing the regression.
* F statistic – when there is no relationship between the response and predictors, one would expect the F-statistics to take on a value close to 1.
* Forward Selection – begin with the null model – the model that contains the intercept but no predictors. We then fit p simple linear regressions and add to the null model the variable that results in the lowest RSS. We then add to that model the variable that results in the lowest RSS for the new 2-variable model. This approach is continued until some stopping rule is satisfied.
* Backward selection – start with all variables in the model, and remove the variable with the largest p-value – which is the variable with the that is the least statistically significant. This continues until a stopping rule is reached
* Backward selection cannot be used if p > n
* Potential problems when we fit a linear regression model to a particular data set:
  + Non-linearity of the response-predictor relationships
  + Correlation of error terms
  + Non-constant variance of error terms
  + Outliers
  + High-leverage points
  + Collinearity
    - To assess, the best way is the compute the variance inflation factor (VIF)
    - The smallest possible value for VIF is 1, which indicates the complete absence of collinearity
    - A VIF that exceeds 5 or 10 indicates a problematic amount of collinearity
* In R:
  + lm(y ~ x, data) used to fit a simple linear regression model
    - Where y is the response, x is the predictor, and data is the data set in which these variables are kept.
  + If you do summary()
    - Gives the p-values, and standard errors for the coefficients. In addition, gives the R-Squared and F-statistic for the model
  + Confidence interval is done by confint()
  + predict() can be used to produce confidence intervals and prediction intervals (see bottom of page 111)
  + To draw a line: abline() and if want to draw a line with intercept a and slope b, type abline(a, b)
  + To increase the width of the regression line: lwd = 3 will increase it by a factor of 3
  + To create difference plotting symbols: use the pch option
  + par(mfrow = c(2,2)) divides the plotting window into a window of 2x2
  + For multiple linear regression, we also use the lm() function

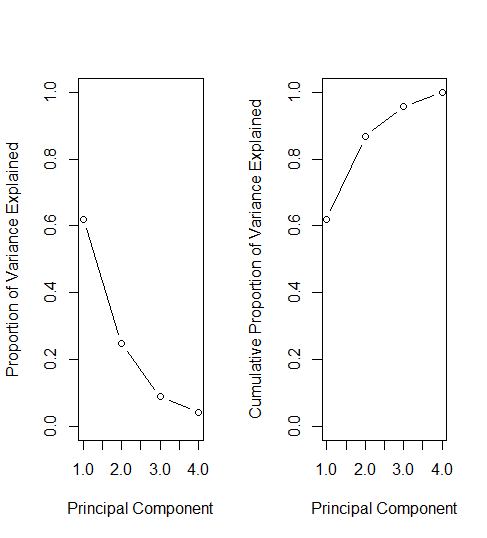


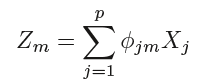


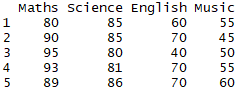


# PCA - Principal Component Analysis (Team 9)

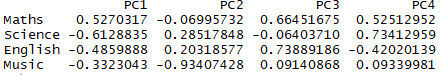
* **What is Dimension Reduction?** – reducing the number of variables in a data set to a more manageable amount (m) while preserving the majority of the data’s variability
* **What does PCA do?** – reduce the dimensions/variance of a dataset while retaining as much model variability as possible
* **When do you use PCA?** – when you have a *large set of correlated predictors* with a *high variance*
* **What to keep in mind** – in using PCA, the created coefficient set (m) has a reduction in variance, but at the cost of introducing bias by constraining the coefficients (increases as p increases relative to n)
* **1st Principal Component** – The vector along which the data varies the most
  + 2nd Principal component is that in which the data varies next most, orthogonal to 1st (uncorrelated) and so forth
    - Each additional PC adds less explanation of variation than the previous, summing to 100%



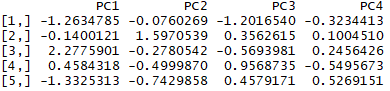
* **PC Scores** - 
  + Where *zm*is the principal component score from a non-negative eigenvector from the covariance matrix
  + ɸim and ɸi+1m such that ɸ2im + ɸ2i+1m  = 1
  + Remember, m < p because of *dimension reduction*
* **R (the important stuff)**
  + prcomp*(dataframe, scale = T)*
    - does PCA for you
    - Note: if you scale data prior, set scale = F
  + prcomp**$x**
    - Gives PC scores
  + prcomp**$rotation**
    - gives **ɸ’s**
  + prcomp**$summary**
    - amount of variation explained by each PC
* **Example** – 5 Students, 4 Variables



* + **ɸ’s**

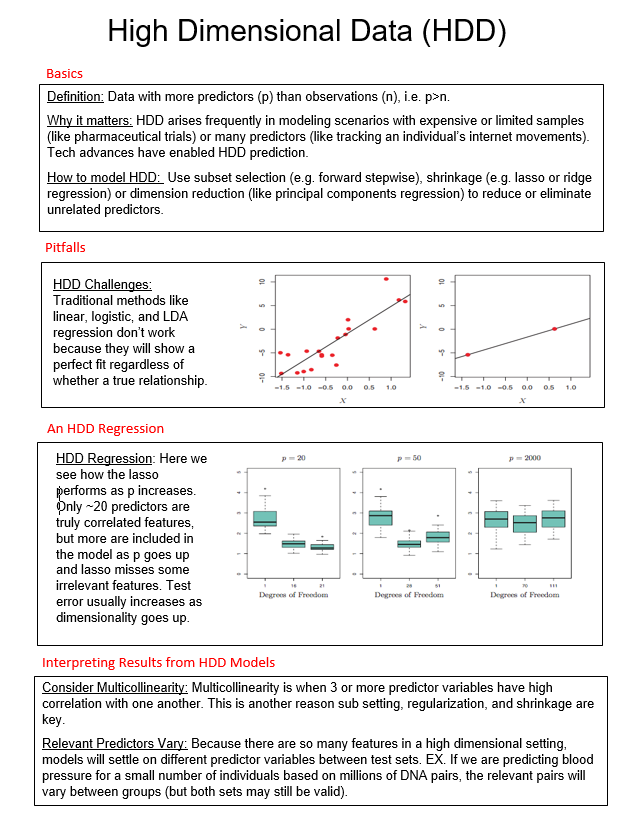


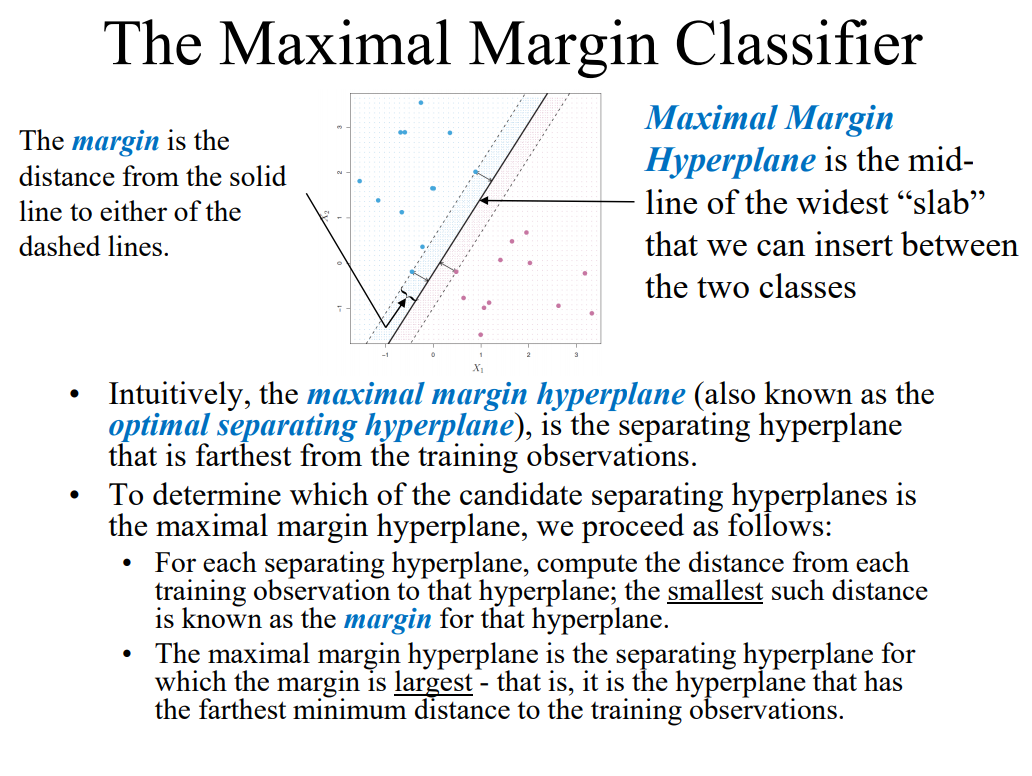
* + **Scores**

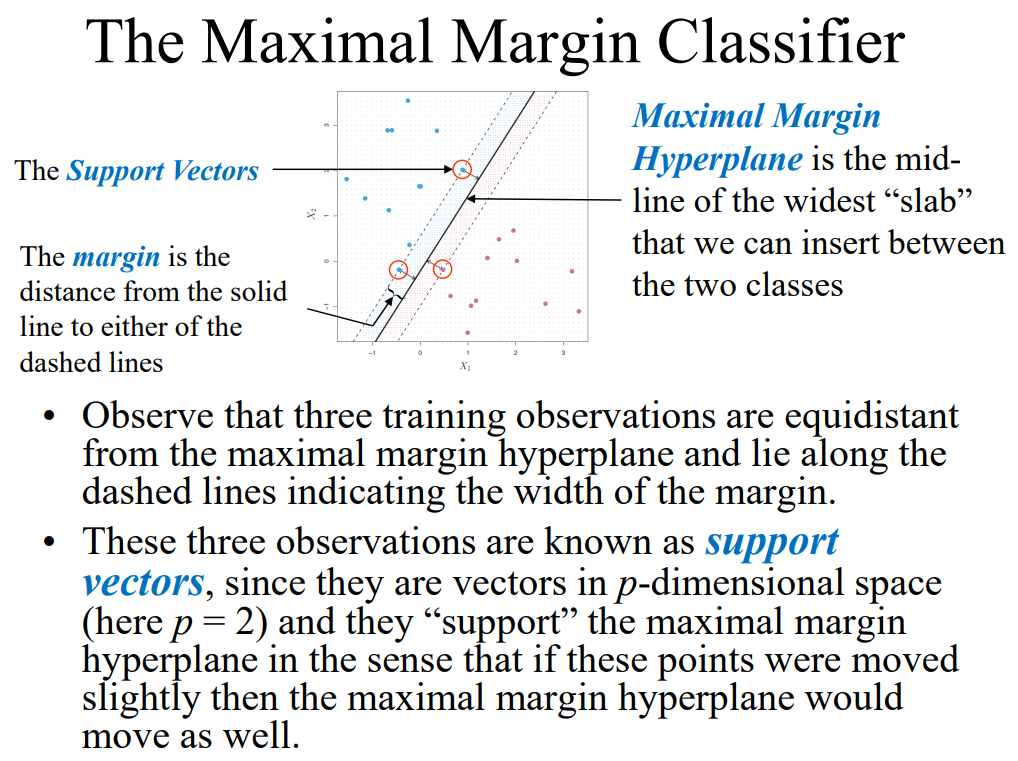


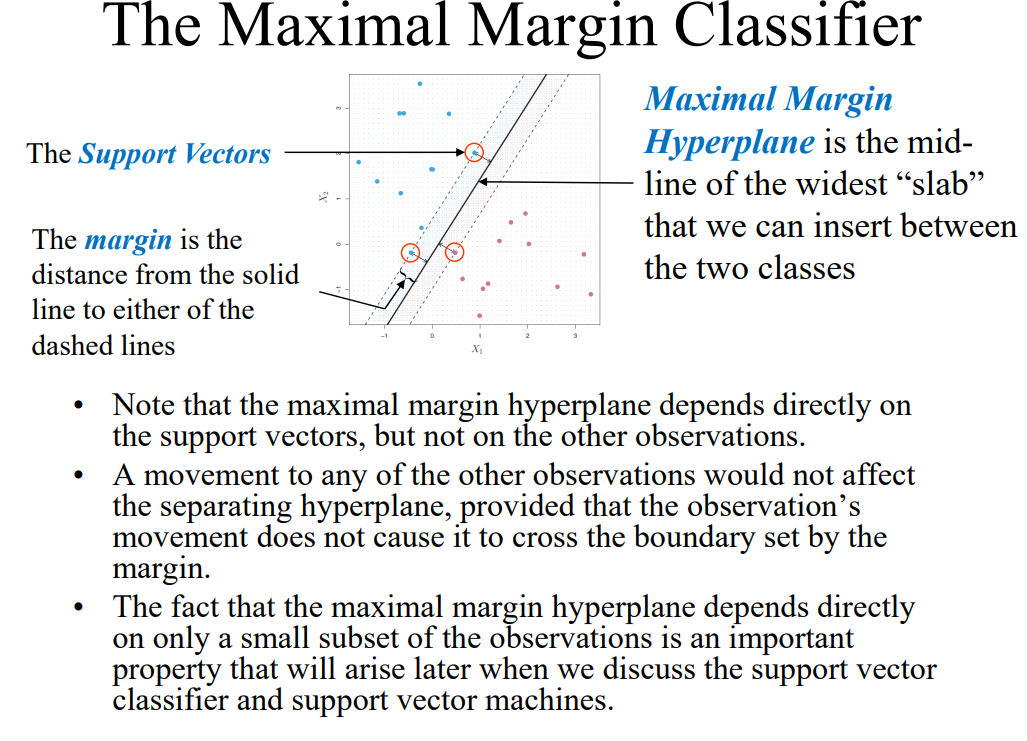
Dimension Reduction Methods Furthered: PCR and PLS

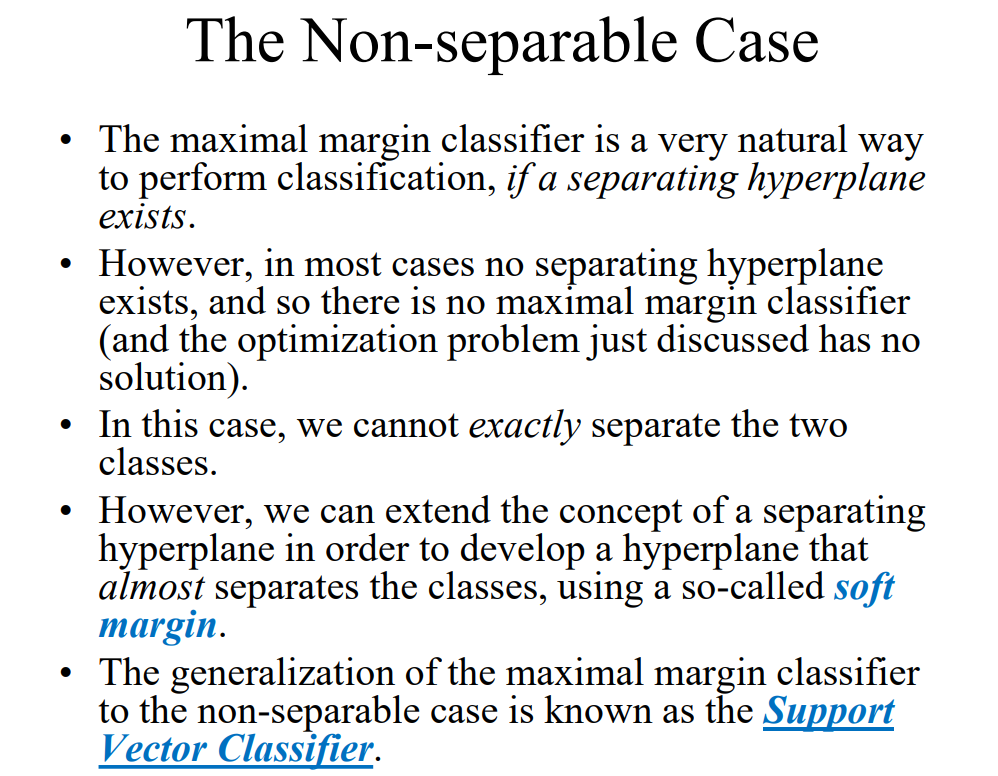
|  |  |
| --- | --- |
| **Review: PCA** | * Reduces Dimensions and variance of the data while retaining model variability * Use when dealing with a large set of correlated predictors, high variance |
| **PCR** | * Simplification of the Model * Use subset of principle components that explain the variance * Construct first M principle components, Z, using as predictors |
| A close up of a map  Description generated with very high confidence | * Small number of Principle Components explain most variability * Fit a Least squares model to Z means better results than simple linreg * In right graph you can see relation to only 2 of the predictors |
| **PLS (Partial Least Squares)** | * A supervised alternative to PCR * Looks at Y’s as opposed to variance in PCR * A dimension Reduction method |
| **The Code**  **rm(list=ls())**  **require(pls)**  **#Need to do this first:**  **Hitters<-na.omit(Hitters)**  **names(Hitters)**  **x<-model.matrix(Salary~.,Hitters)[,-1]**  **y<-Hitters$Salary**  **# Lab starts here**  **set.seed(1)**  **train<-sample(1:nrow(x), nrow(x)/2)**  **test<-(-train)**  **y.test<-y[test]**  **set.seed(2)** | * Has the potential to reduce bias, but also has the potential to increase variance * More dependent on Training Y’s * Can cancel out benefit of supervision |
| **set.seed(1)**  **pls.fit<-plsr(Salary~., data=Hitters,subset=train,scale=TRUE, validation="CV")**  **summary(pls.fit)**  **validationplot(pls.fit,val.type="MSEP")**  **pls.pred<-predict(pls.fit,x[test,],ncomp=2)**  **mean((pls.pred-y.test)^2)**  **pls.fit<-plsr(Salary~., data=Hitters,scale=TRUE,ncomp=2)**  **summary(pls.fit)**  **require(glmnet)**  **library (glmnet )**  **grid =10^ seq(10,-2, length =100)** | **ridge.mod =glmnet(x,y,alpha =0, lambda =grid)**  **ridge.mod =glmnet(x[train ,],y[train], alpha =0, lambda =grid, thresh =1e-12)**  **ridge.pred=predict(ridge.mod ,s=4, newx=x[test ,])**  **mean(( ridge.pred -y.test)^2)**  **lasso.mod =glmnet(x[train ,],y[train],alpha =1, lambda =grid)**  **set.seed (1)**  **cv.out =cv.glmnet(x[train ,],y[train],alpha =1)**  **bestlam =cv.out$lambda.min**  **lasso.pred=predict(lasso.mod ,s=bestlam ,newx=x[test ,])**  **mean(( lasso.pred -y.test)^2)** |

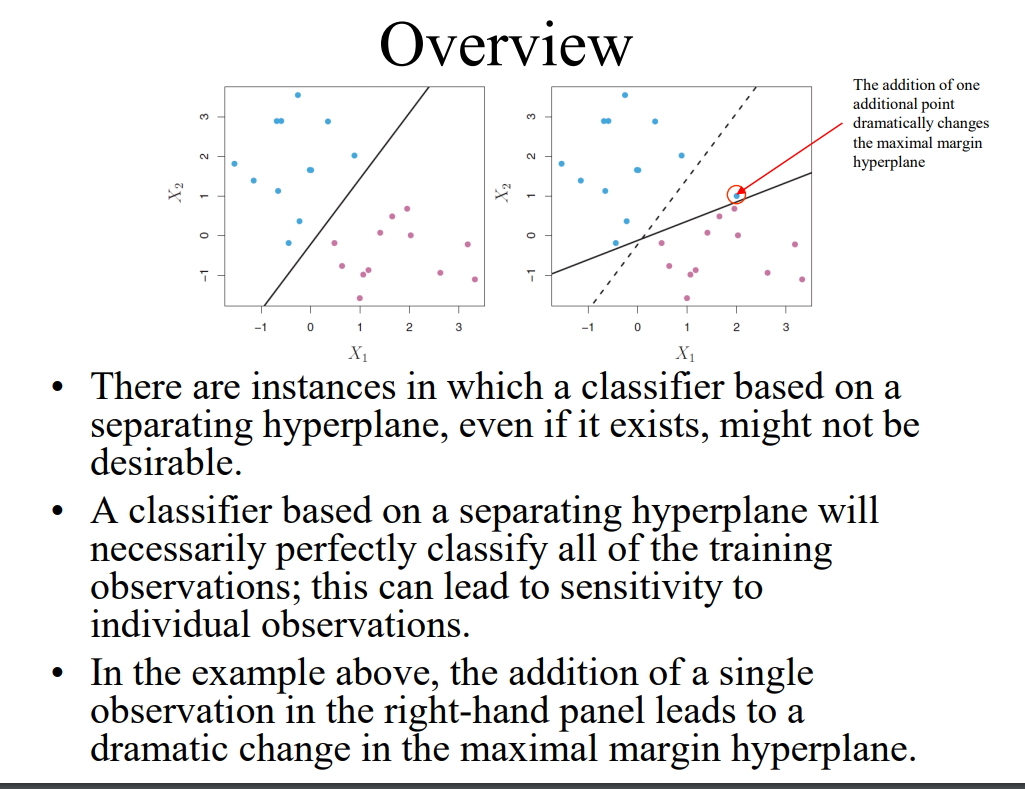


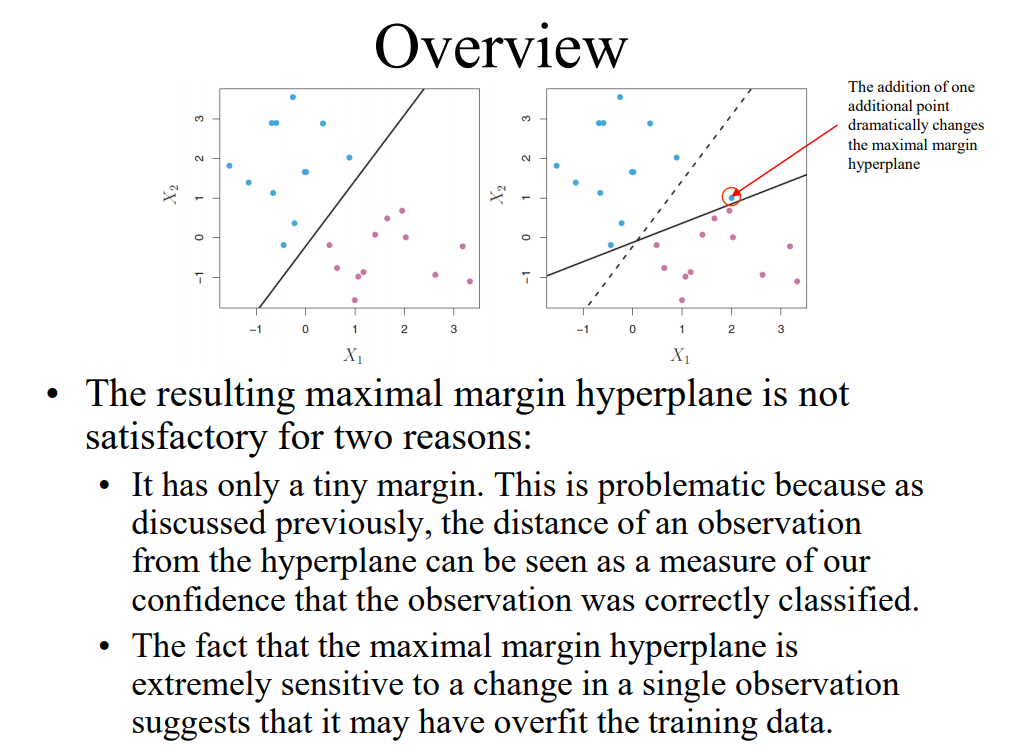


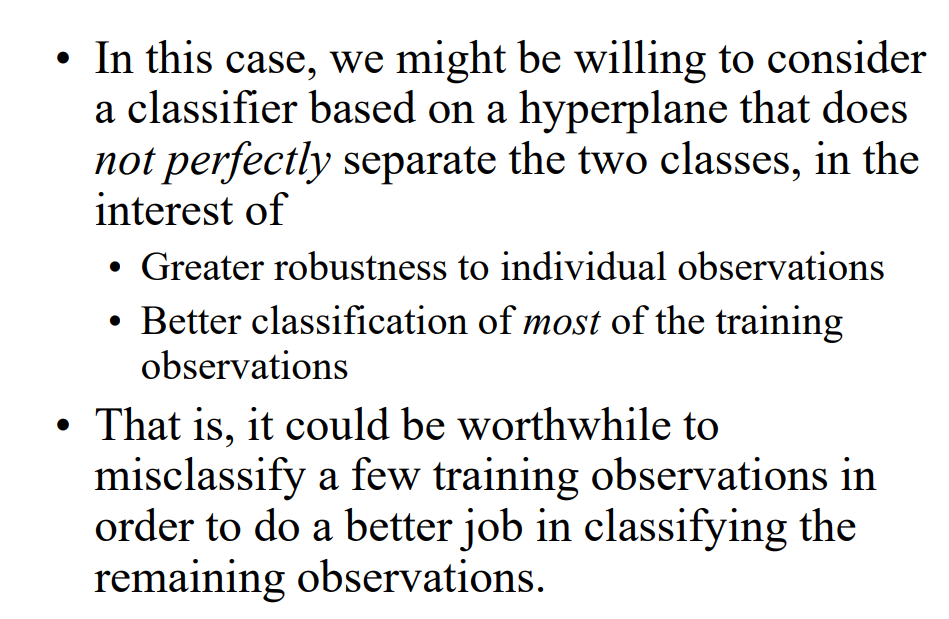


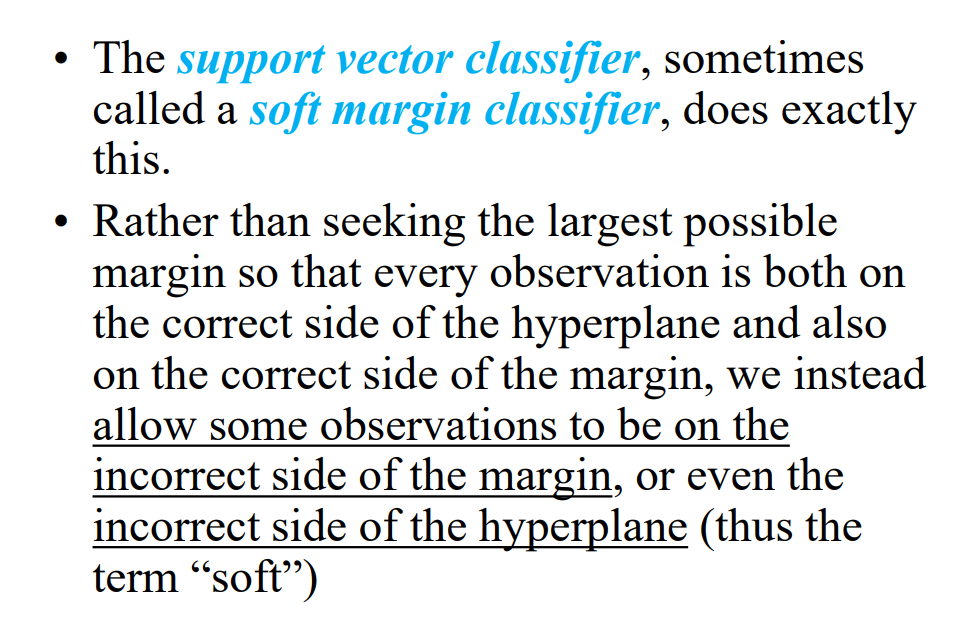


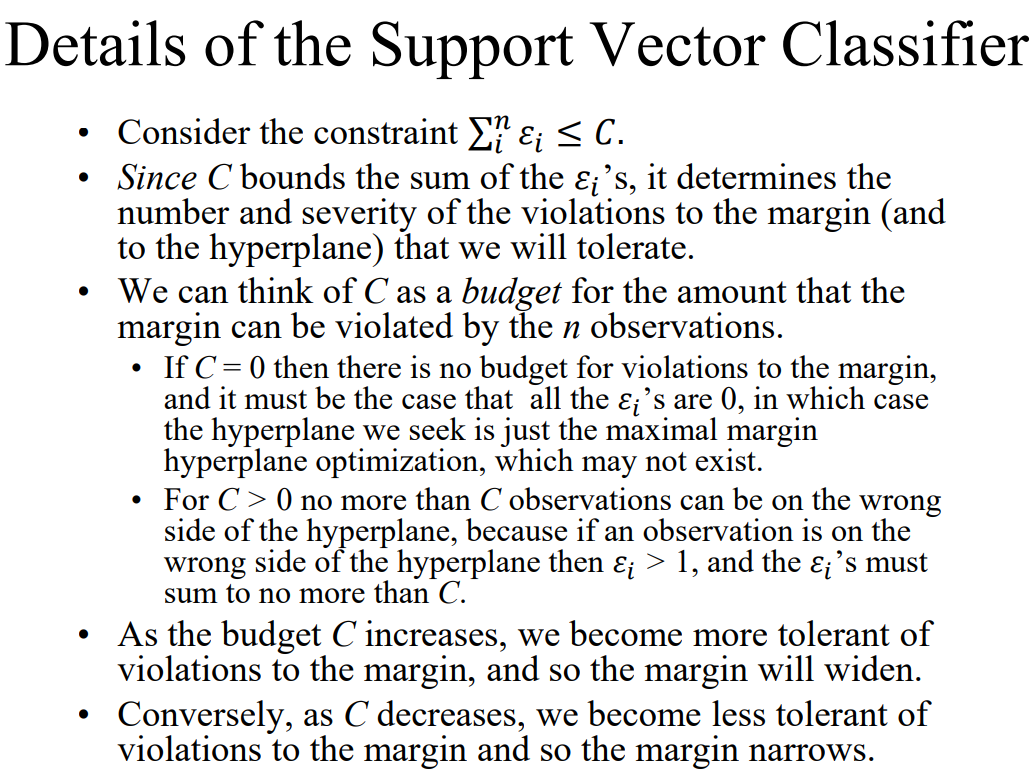




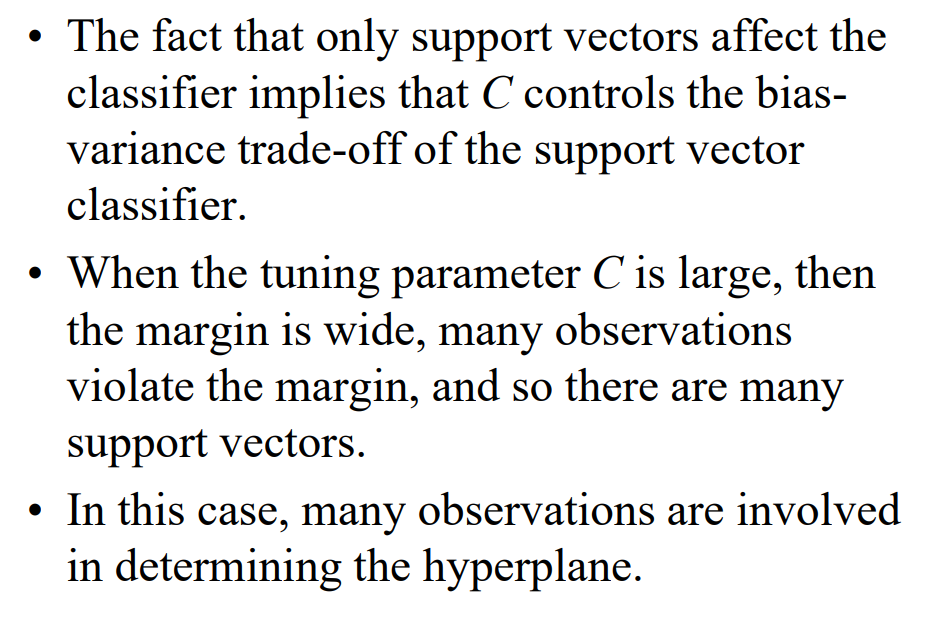


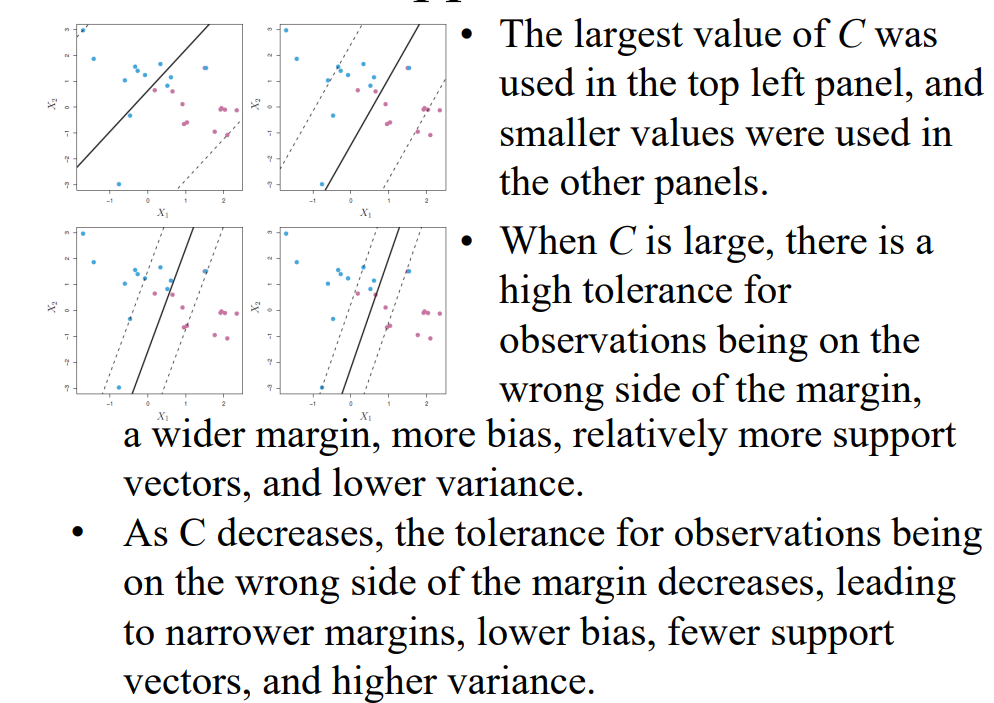




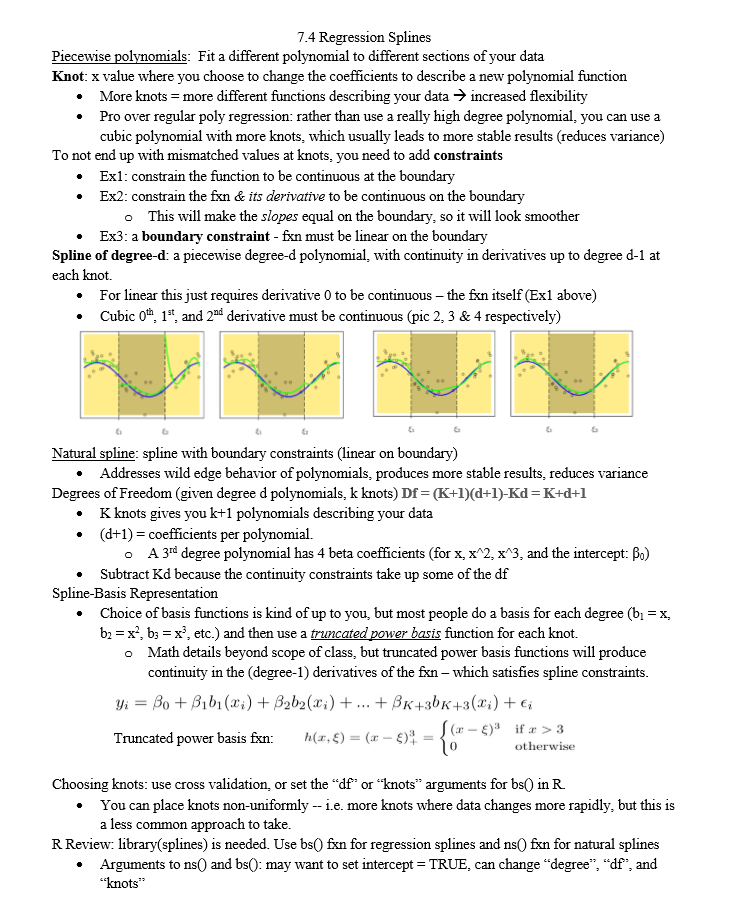




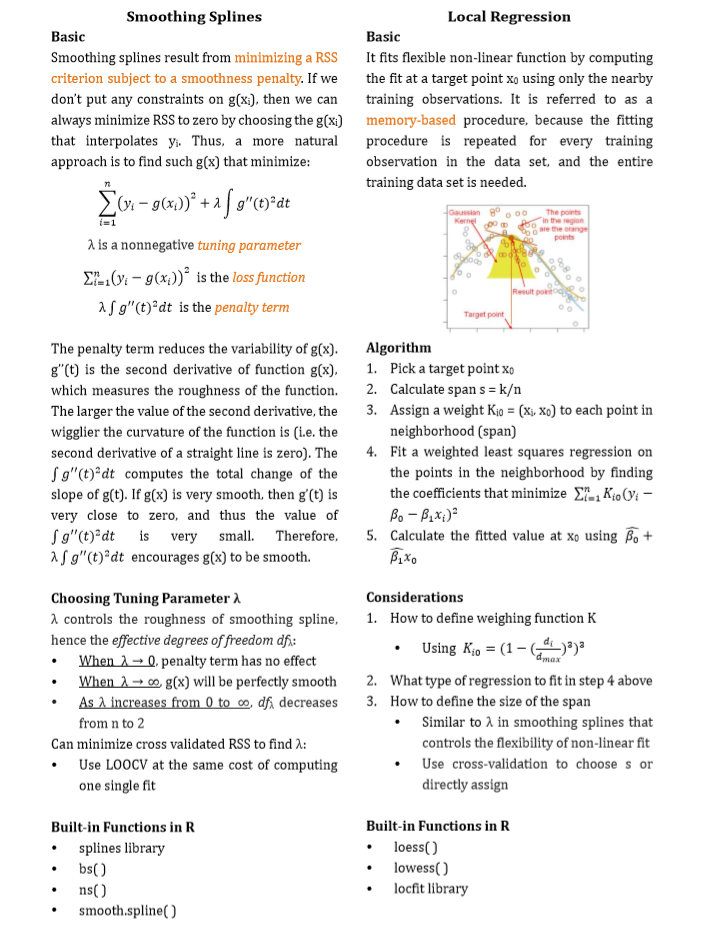




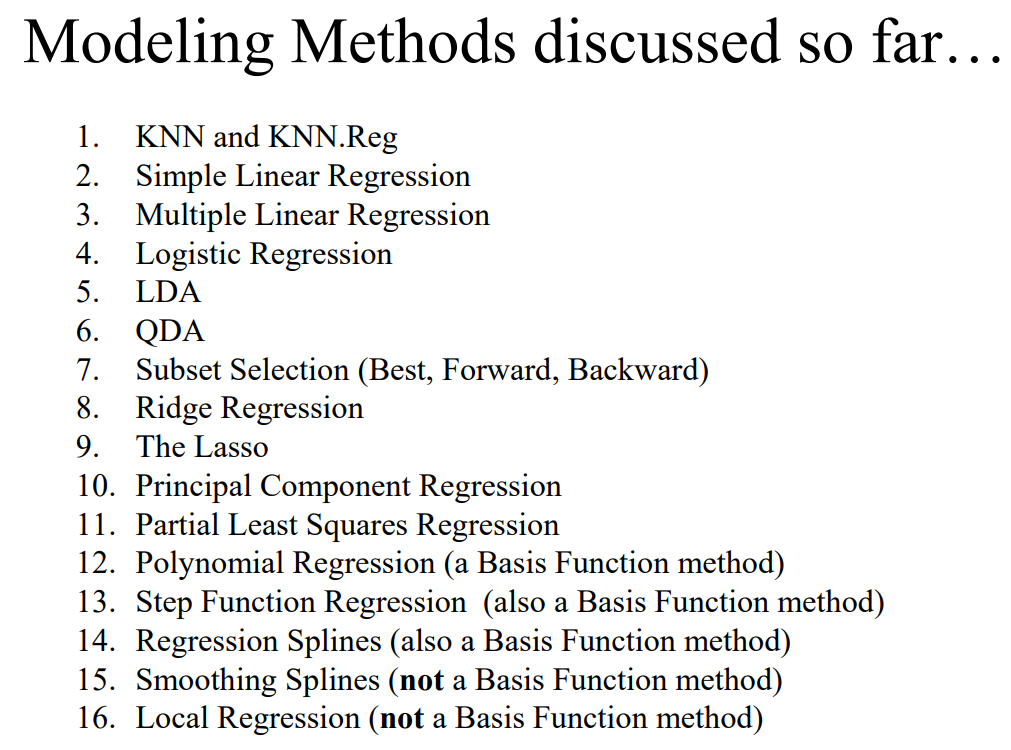
Regression Splines

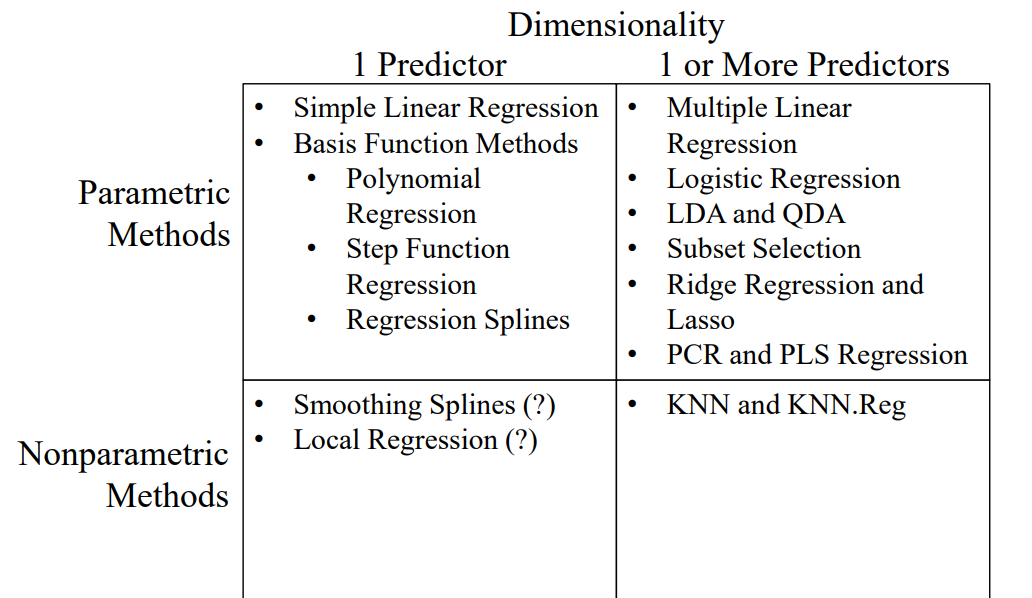


Smoothing Splines and Linear Regression

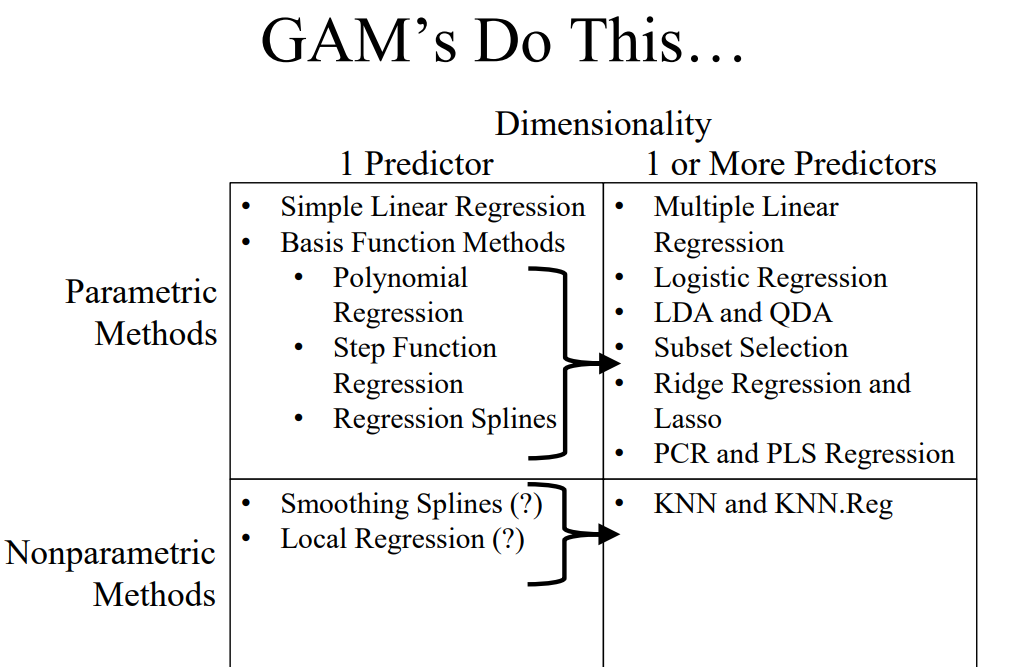


Semester Summary Discussion

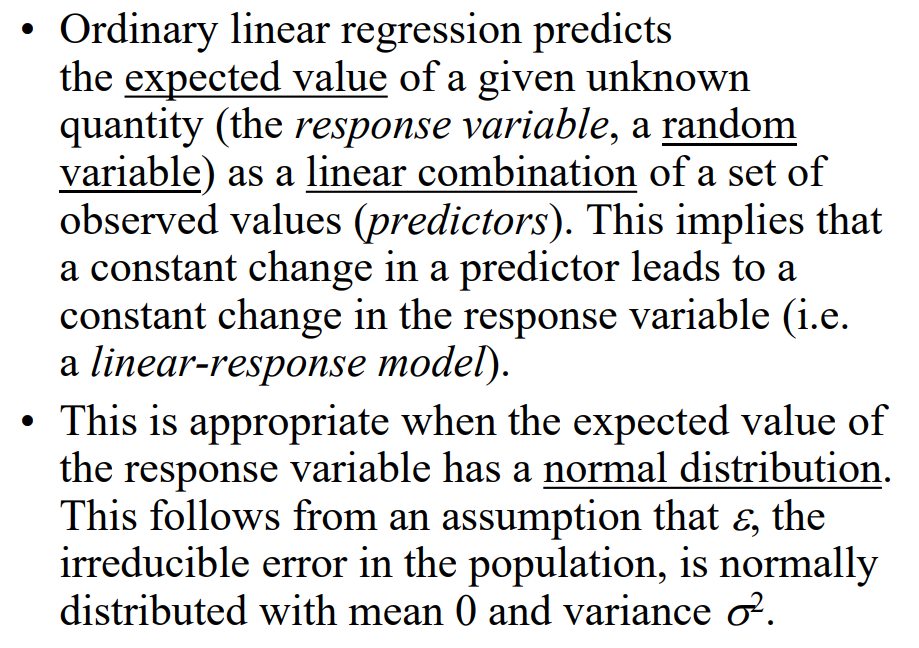


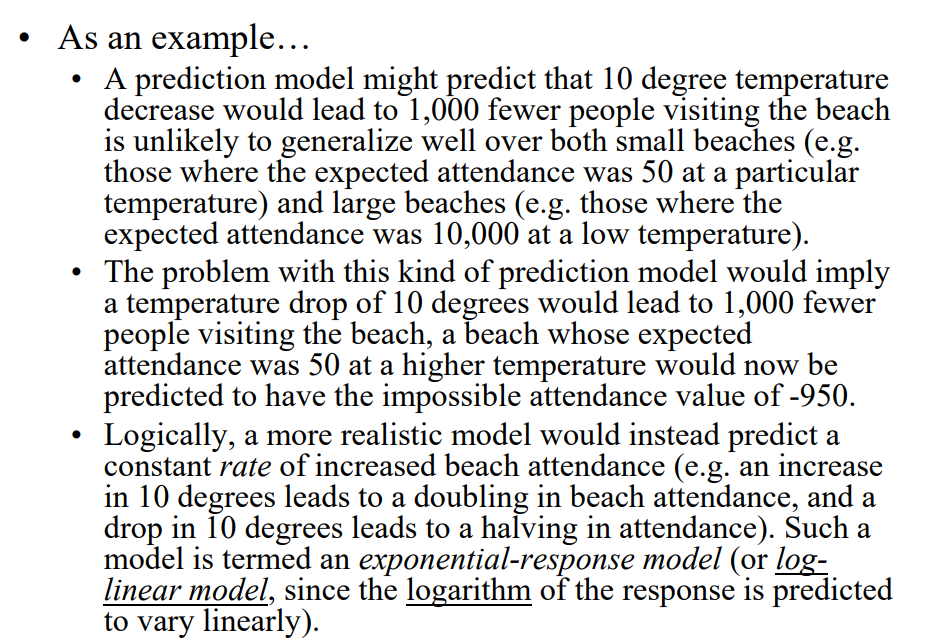


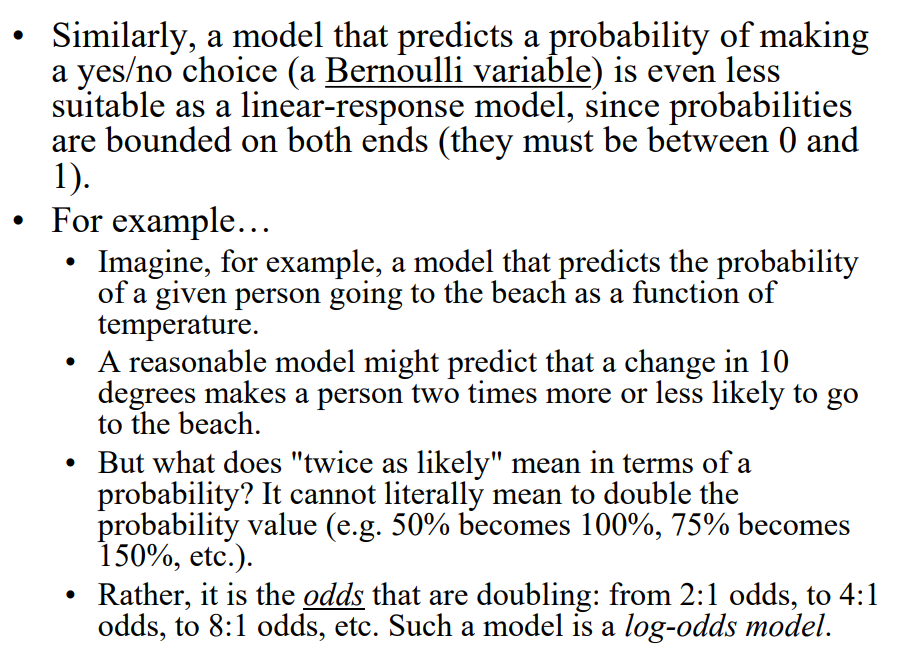
GAM’s

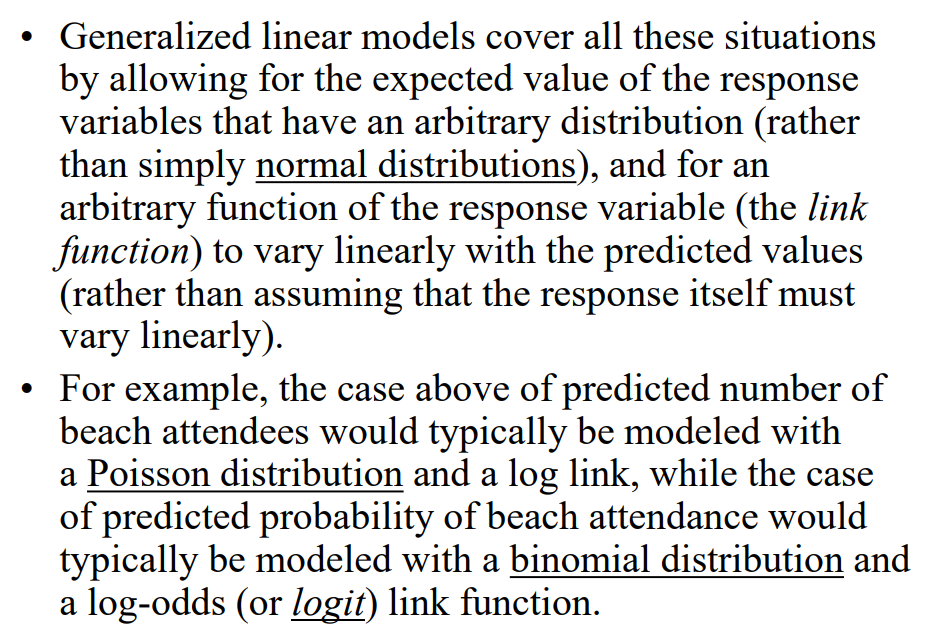


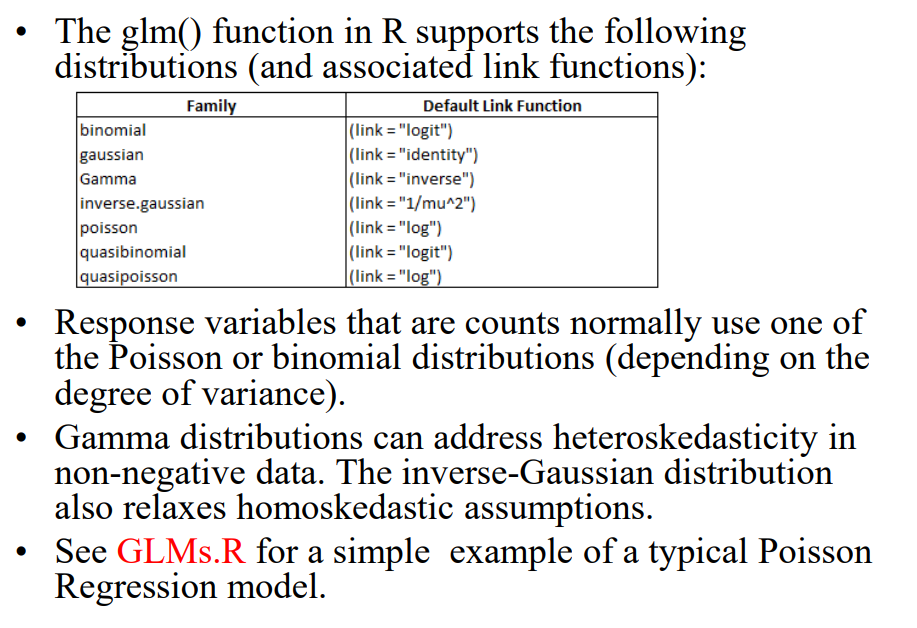
Generalized Linear Models (GLM’s)



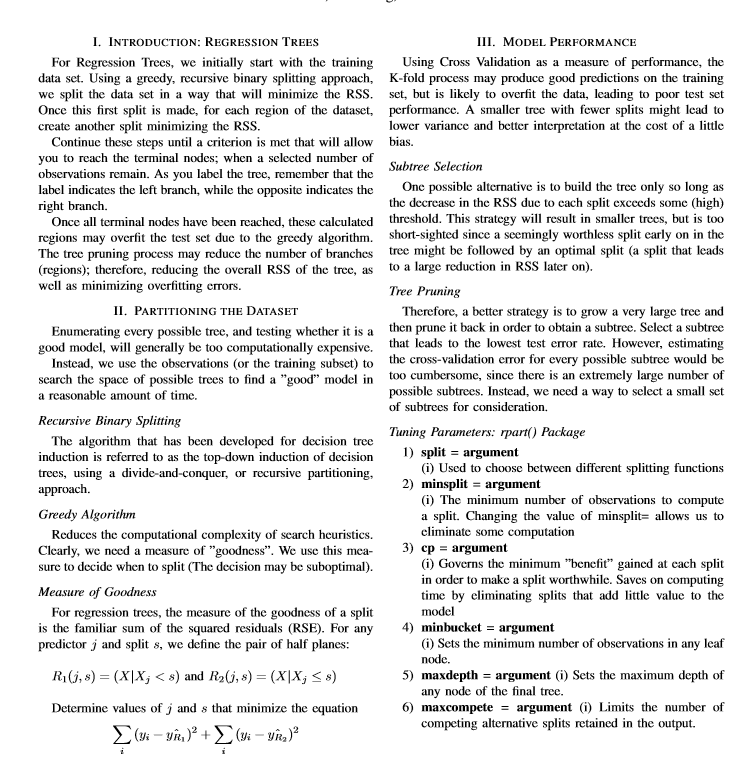




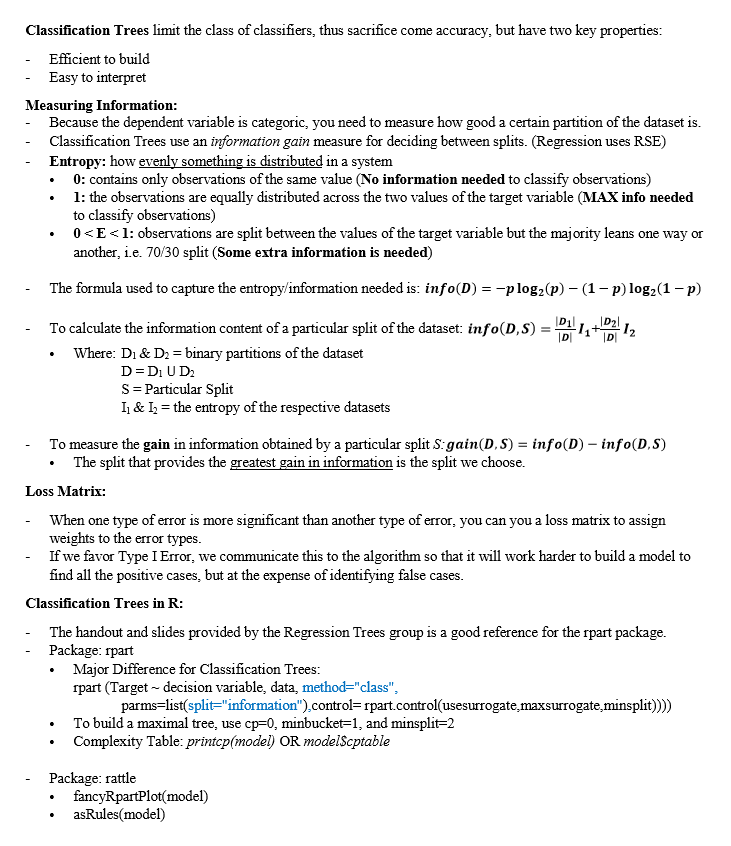




Regression Trees



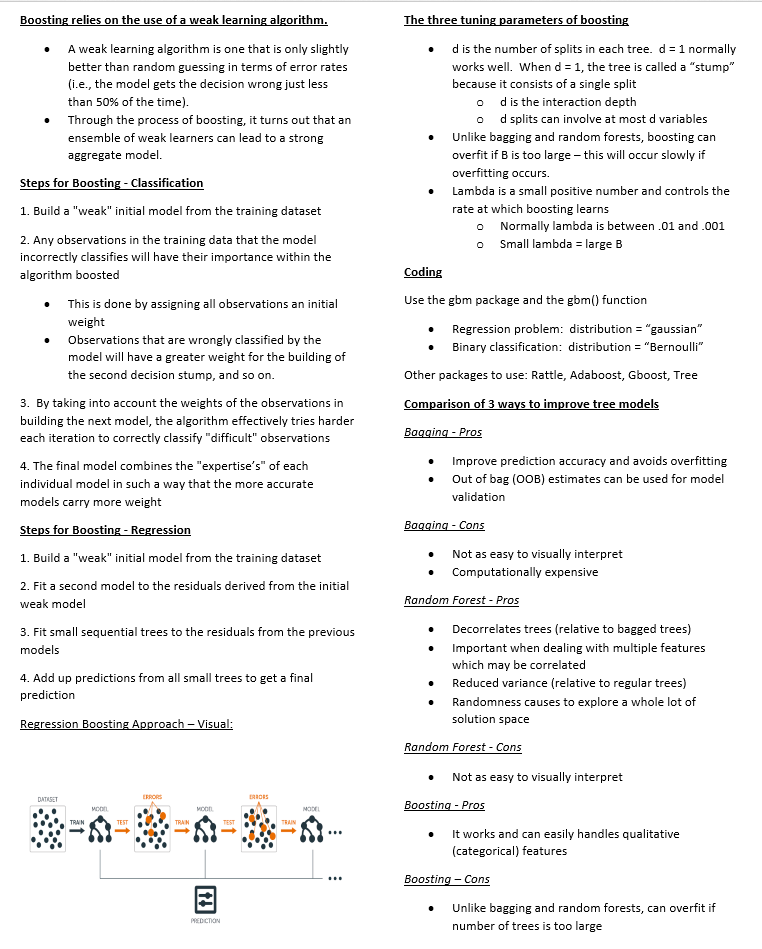
Classification Trees



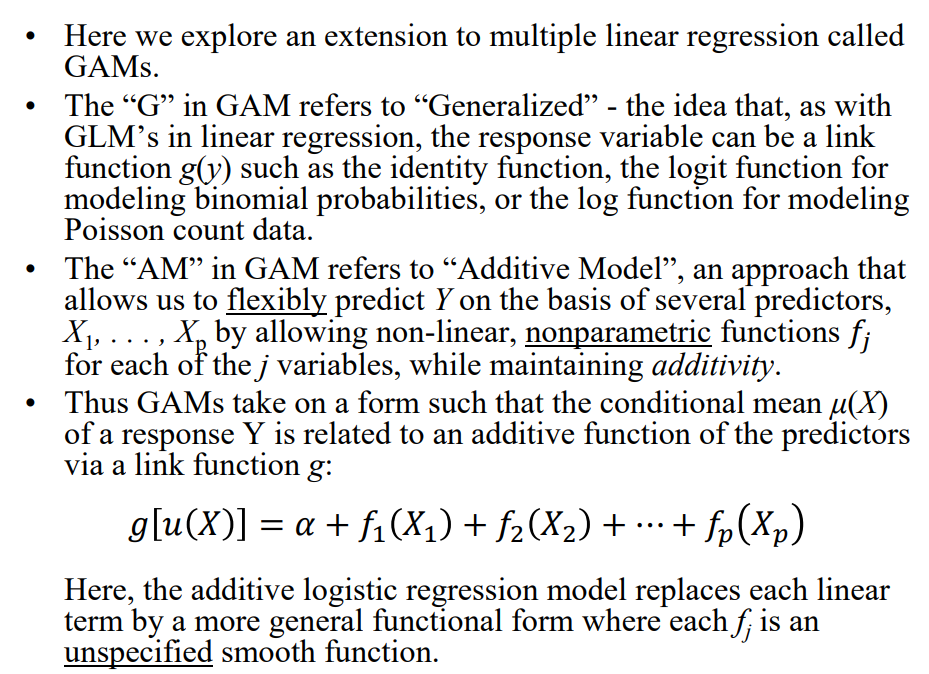
Bagging and Random Forests

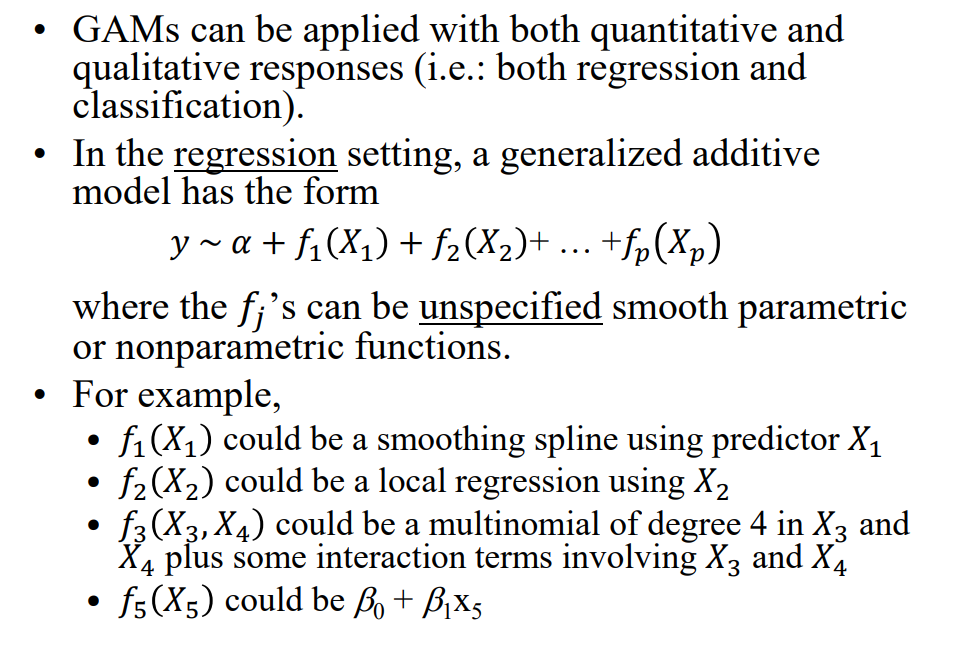


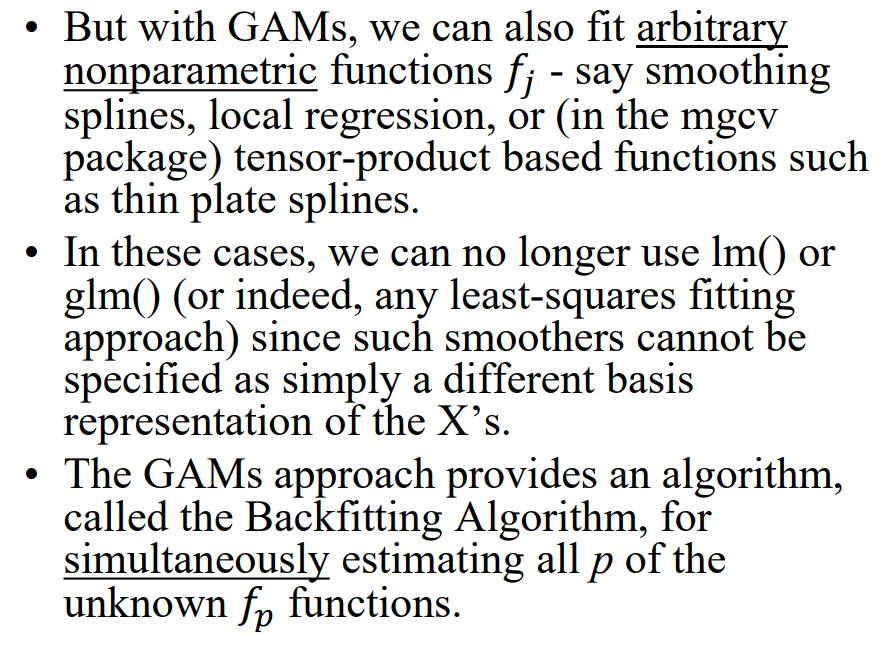
Boosting

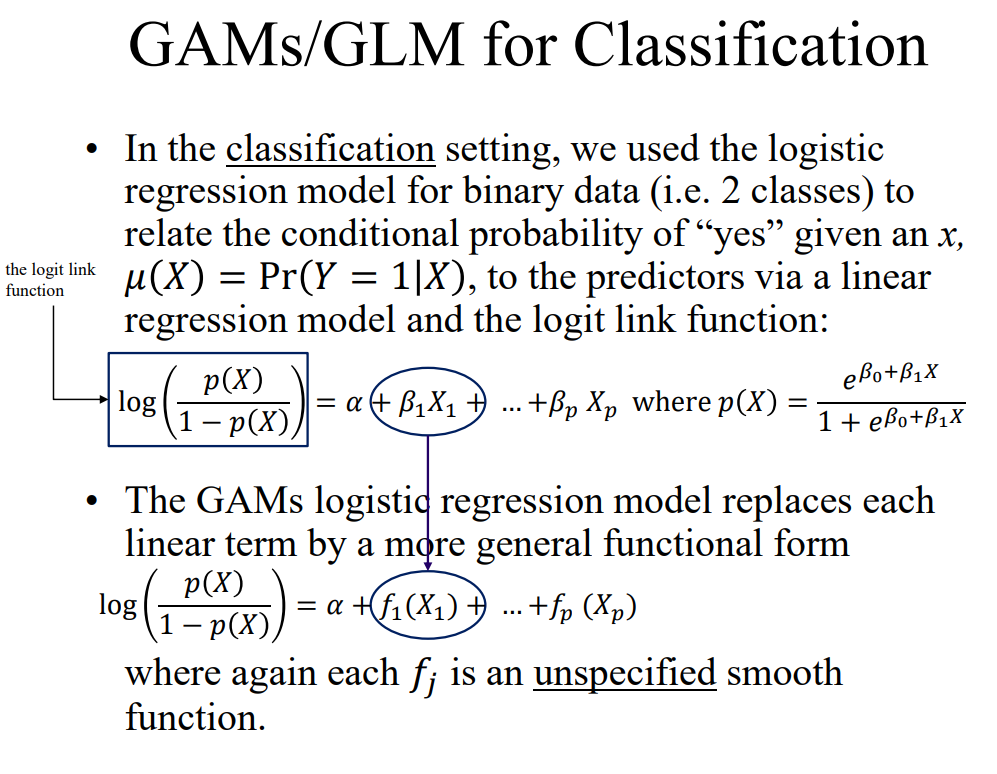
**

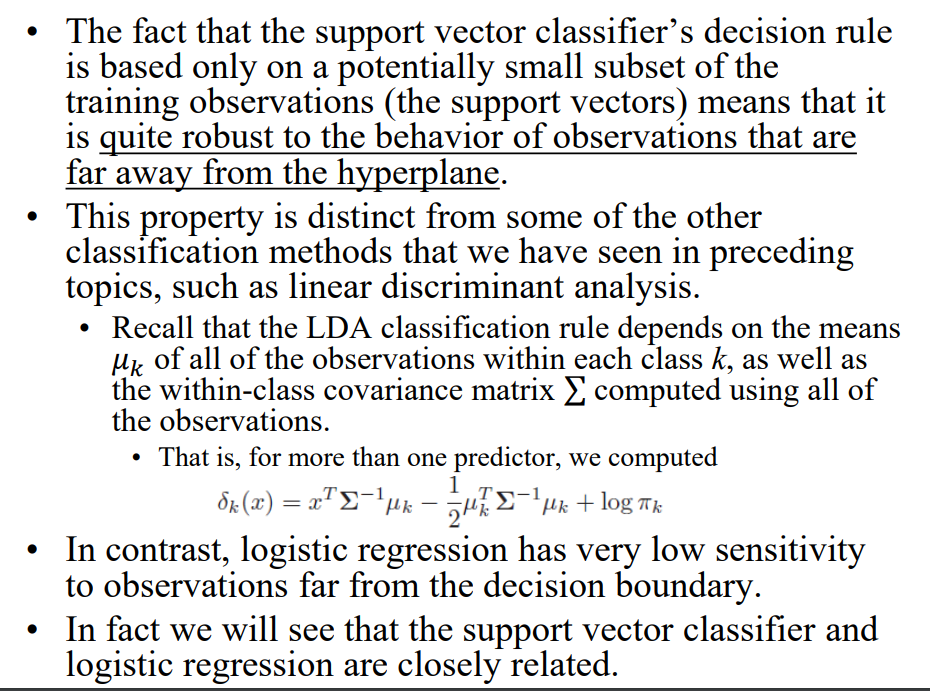
Generalized Additive Models



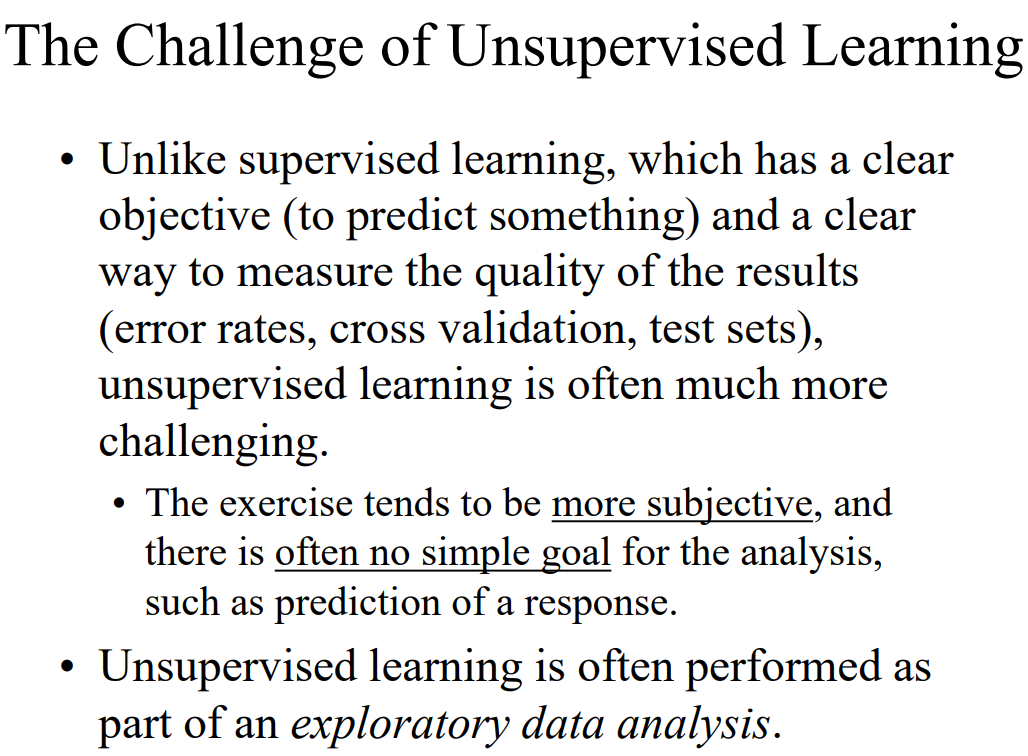


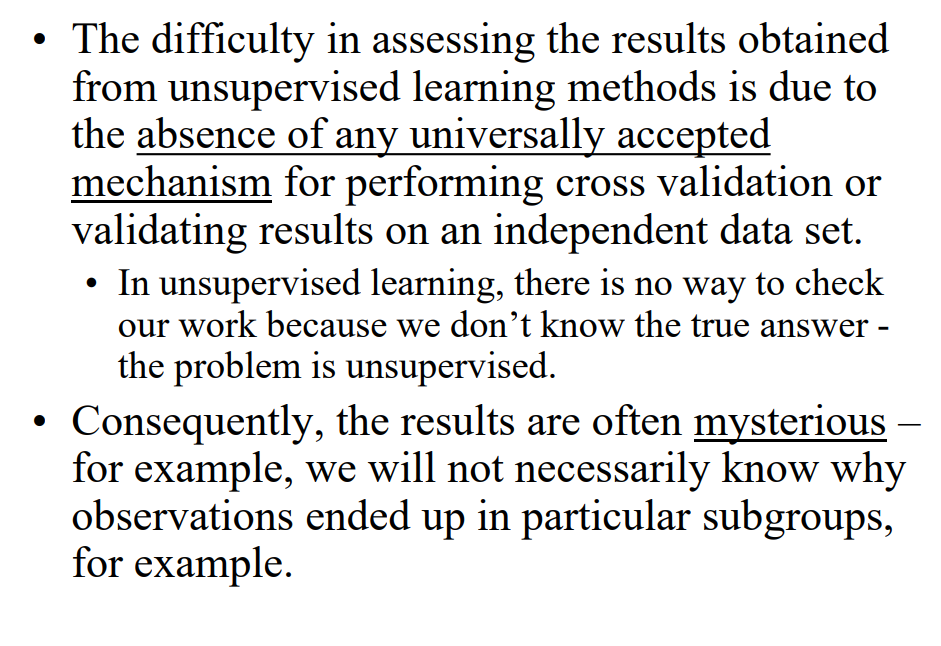


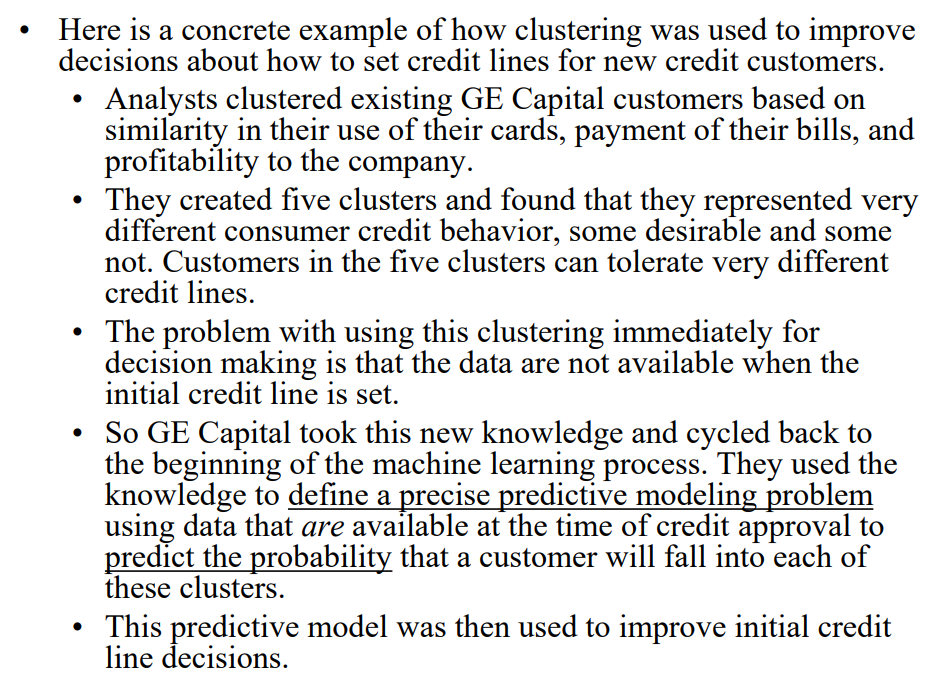




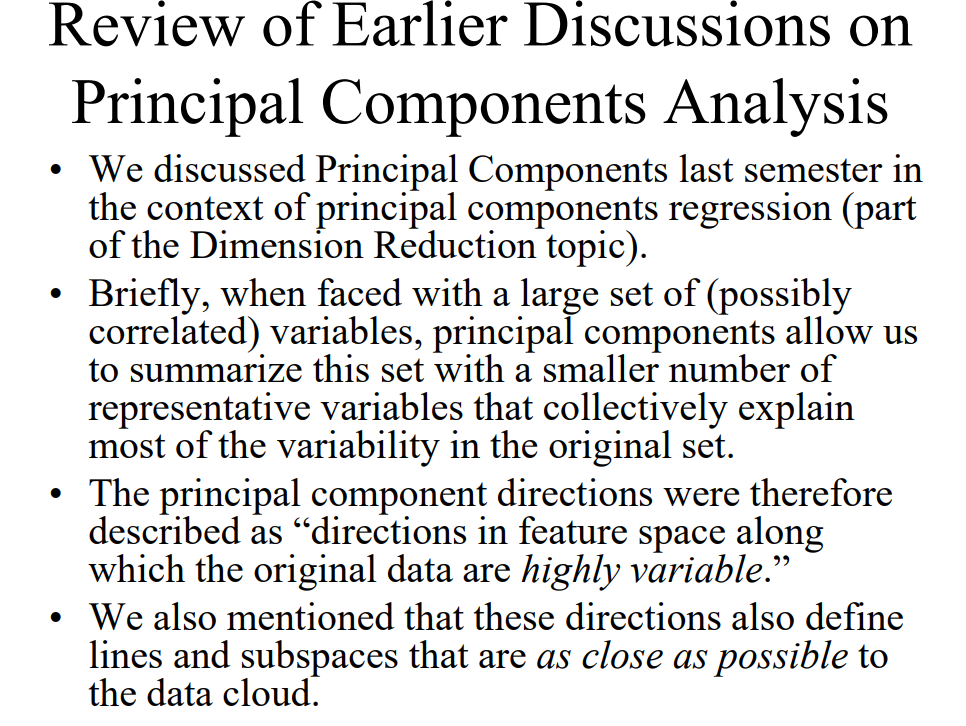
*Introduction to Unsupervised Learning*

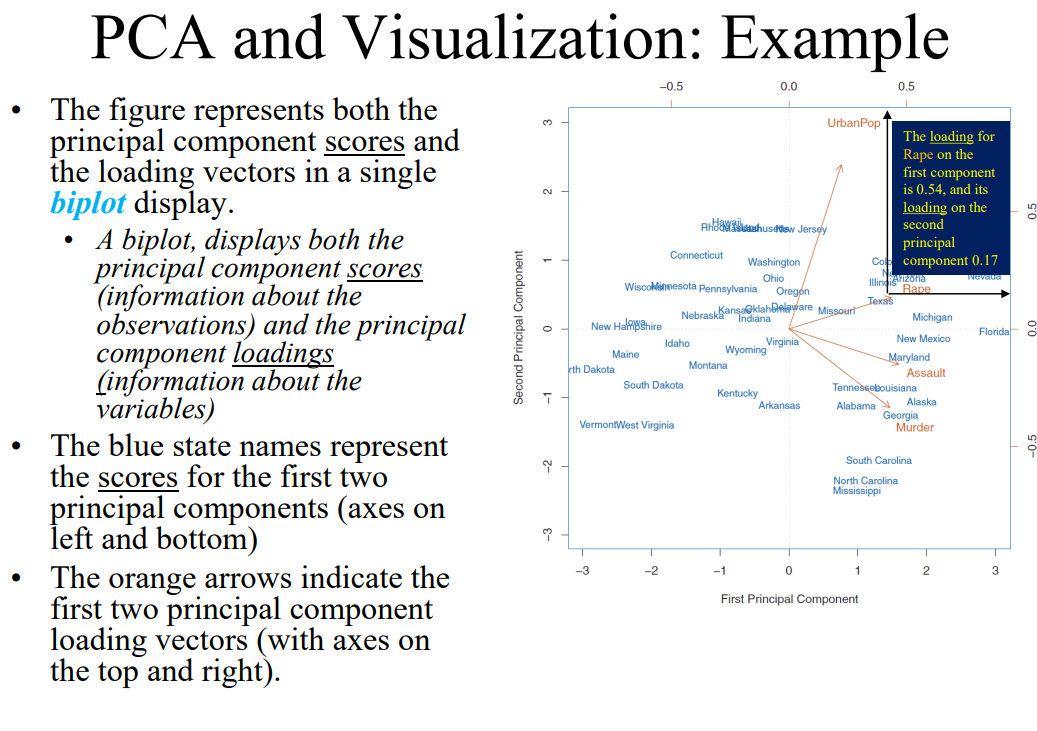


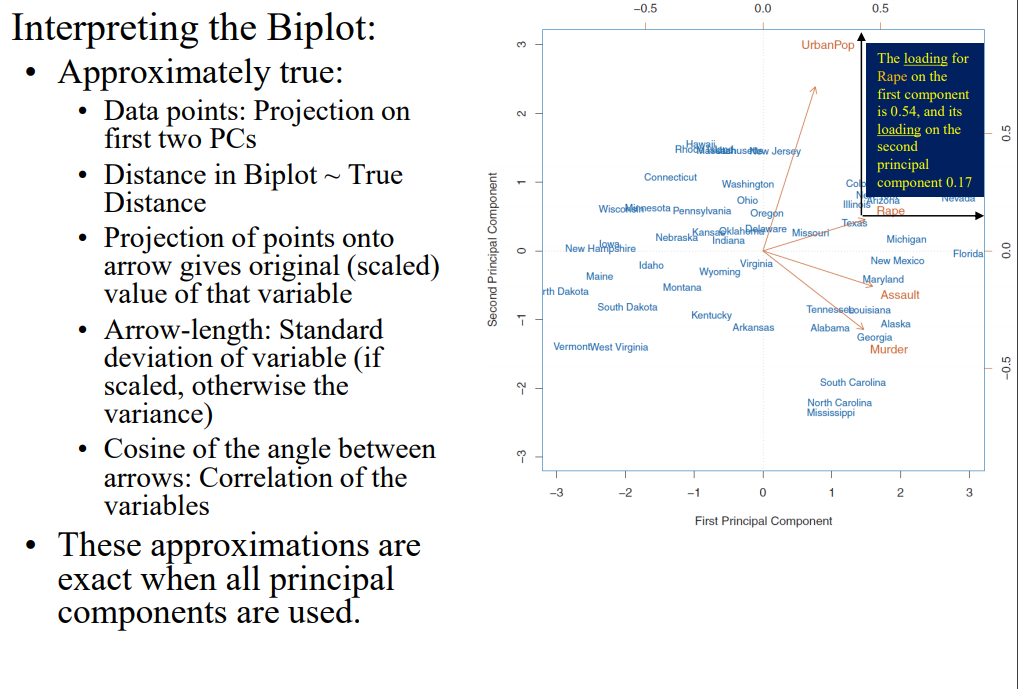
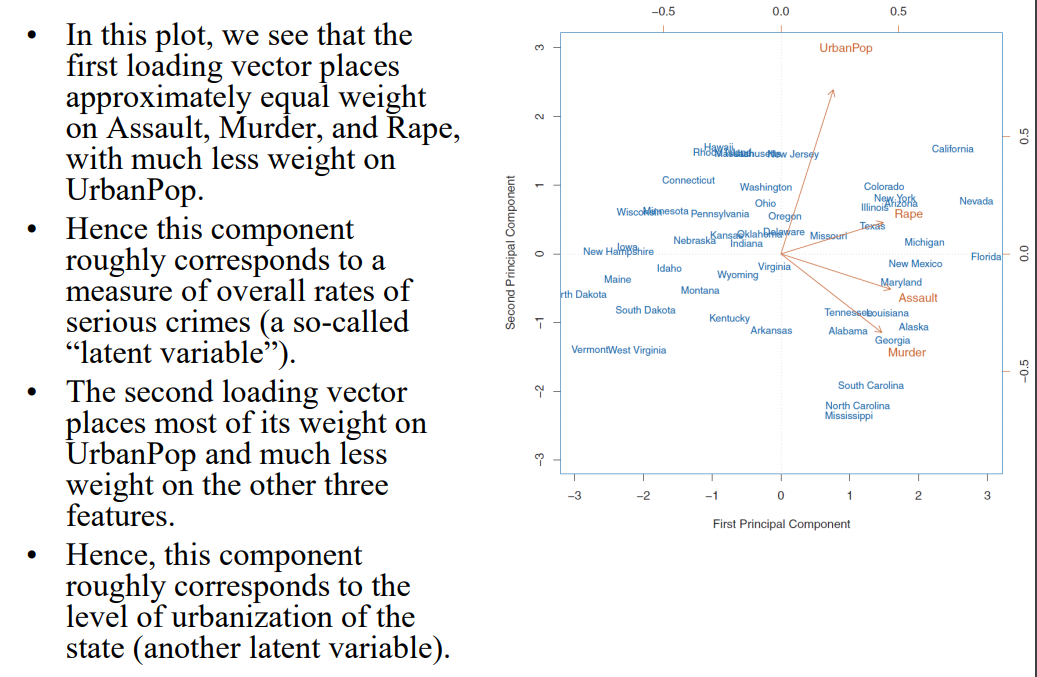


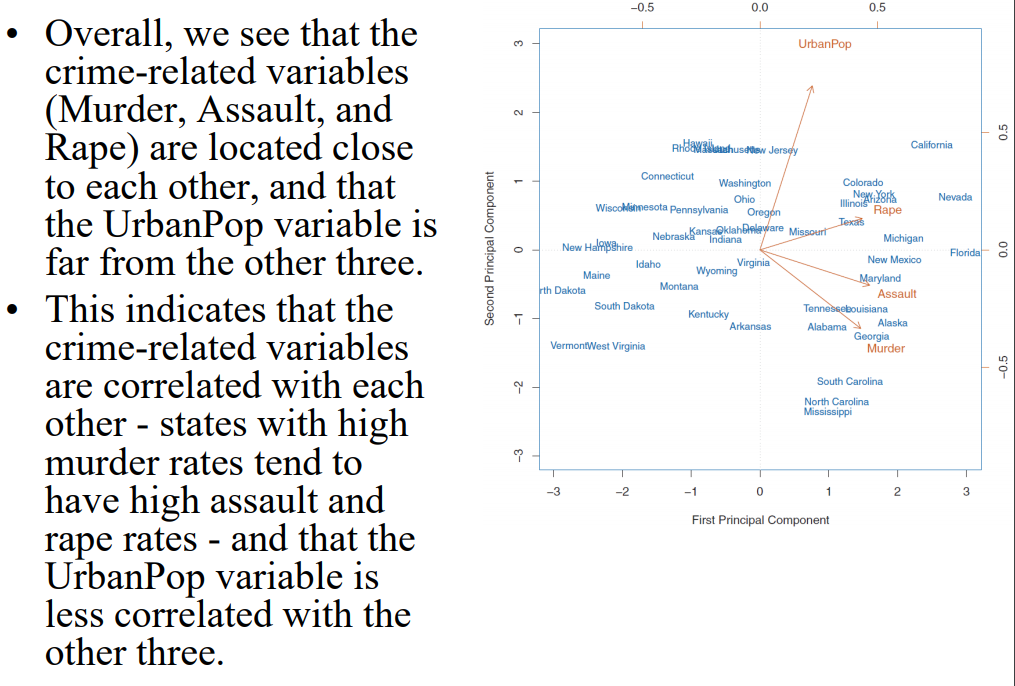


PCA Revisited

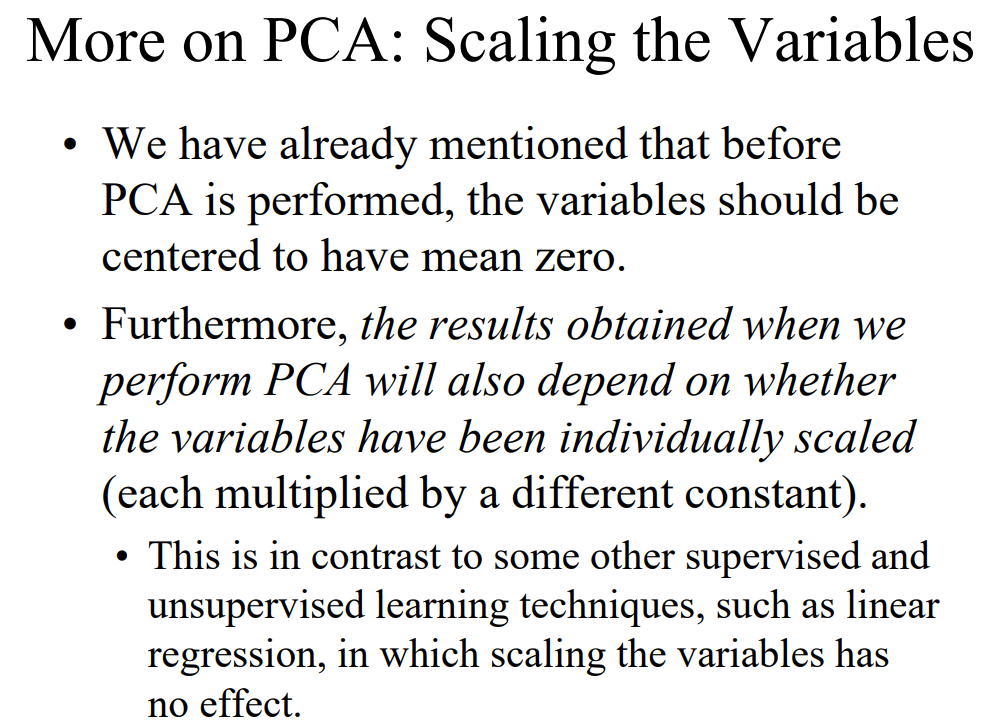


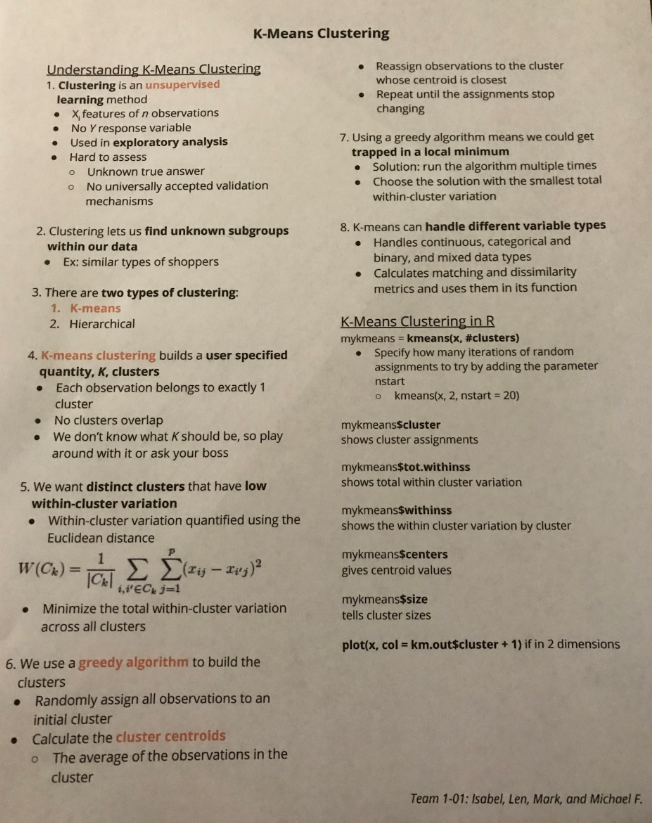




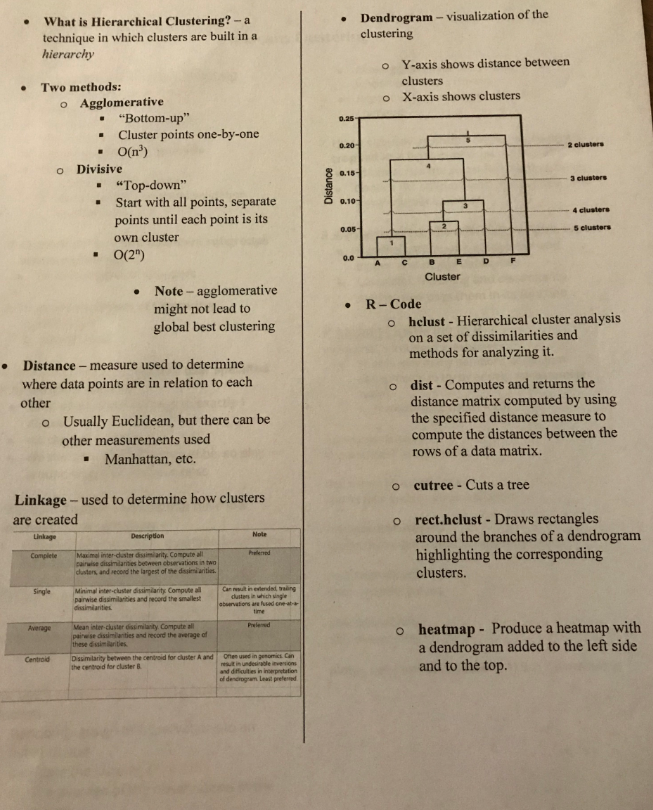






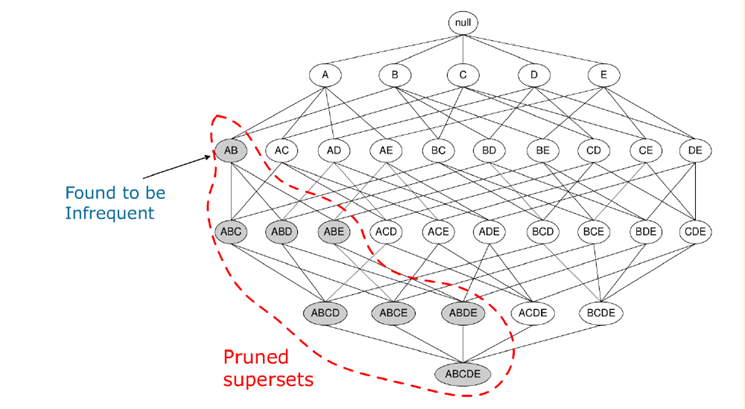
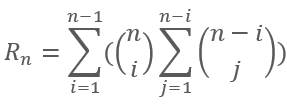


*Hierarchical Clustering*



**Association Analysis**

#### **The Apriori Search Algorithm**

* An algorithm for frequent itemset mining and association rule learning over transactional databases.
* **Phase I:** Generate ***frequent itemsets***
  + “Bottom up” approach - Start from individual item and extend one item at a time until no more itemset of the next level is frequent enough
  + ***Downward closure property***
    - All subsets of frequent itemset must also be frequent itemsets
    - No superset of an infrequent itemset can be a frequent itemset
  + ***Support*** - the frequency of itemsets appearing in all transactions
* **Phase II:** Build ***association rules*** (X → Y)
  + Generate all possible rules for each frequent itemset, and retain those with high ***confidence***
  + Number of association rules if n unique items are in a frequent itemset:
  + ***Confidence*** - conditional probability that Y (RHS of rule) appears in basket given that X (LHS) also appears

#### **Calculating Measures:**

* Support(X) = P(X), support(X U Y) = P(X U Y)
* Confidence( X → Y) = P(Y|X) = P(X U Y) / P(X) = support(X U Y) / support(X)
* Lift(X → Y) = support(X U Y) / support(X) \* support(Y) = confidence(X → Y) / support(Y)
  + *Note: “X U Y” represents transactions where X* ***and*** *Y both appear*

#### **R Package (“arules”):**inspect()

* apriori()
* Lhs() and rhs()
* sort(rules,by=)

