**Discussion Questions:**

*You should always standardize your predictors for knn as a first step – why?*

**$**Distance metric used to make predictions

*How do you handle categorical predictors using knn?*

**$**No good way for knn to handle

*For 1-nearest neighbor, what would a plot of versus x look for the gas mileage example with only Displacement as a predictor?*

**$**g(x;): Step function

*How can you tell which predictors have the largest effect on the response in the K-nearest neighbors model?*

**$**Hardest to interpret

*What are the parameters of the fitted model (KNN)?*

**$**No parameters. ALE Plot to interpret model

*The Gaussian kernel weights decay more smoothly, but the other two “compact support” kernels have computational advantages. Why?*

**$**Because they only contain values within a certain window. Think of |x-|. Normal doesn’t have that.

*How would increasing change the predictor plotted three slides earlier?*

**$**It would make the window larger; ||

*Regarding the omnipresent bias/variance tradeoff, would increasing l*

* *$Increase or* ***decrease*** *the variance of the $predictor?*
* ***Increase*** *or decrease the bias of the predictor?*

*What is the relationship between a local linear regression model and a linear regression model? Is a local linear regression model "almost" linear?*

$Not similar at all not even close

*Where is a local linear regression predictor most likely to be biased?*

$Boundary

*With no interactions, what are the implications of the GAM model structure Y(x) = a + f1(x1)+ f2(x2)+ . . . + fk(xk) + e, in terms of the type of predictive relationships it can capture?*

**$**It can’t capture interaction terms

*How would you handle categorical predictors in a GAM model?*

**$**You can’t unless its ordinal categorical

*Neural networks with a linear output activation function and one hidden layer are very similar to PPR, but with the neural network logistic Hj() replaced by the completely nonparametric PPR fj() Yes*

*How can you interpret the PPR response surface and component functions?*

**$**Higher or means the function is more important. Most important coef in each v means it has the largest coef. Or just use ALE Plots

*What is the computational expense associated with stacking? Which part of the stacking algorithm is the most computationally expensive?*

**$**It’s “horrible” for the same reason n-fold cv is bad (n\*n) models

*Would the computational expense be better or worse if you used K-fold, instead of n-fold CV*

It could be somewhat better or it could still be awful. Then you’re fitting nxk models

*How does stacking relate to model selection using CV?*

**$**Can give the ones better at predicting y more weight. View it as a linear combination of betas times g(x). Viewed as not worth the effort unless you have tons of time to fit all the models. Also we have better methods such as boosting and random forests using way less computational power.

*Smaller 𝜆 generally gives better predictive performance but worse computational expense, since it requires a larger 𝑀*

**$**Higher M leads to overfitting

*The variable importance measure is simply the sum of the variable importance measures for each of the M individual trees*

**$**For interpreting the model and the effects of the predictors

* Individual marginal plots are very useful but cannot show any interactions (and can be misleading if interactions are strong)
* Pairwise marginal plots show interactions between pairs of variables
* Printing out the first few trees using print(pretty.gbm.tree(gbm1,i)) (for i = 1, 2, . . .) and attempting to interpret these may also help, but this is usually difficult

**$**Random Forest:

* M: num of random subset of features for each tree
* Ntree: number of trees that were going to use
* Nodesize: controls size, number of leafnodes of each individual tree
* OOB MSE almost = CV MSE

*Random forests can overfit if the individual trees are chosen too large, but as the # trees increases they do not increasingly overfit, unlike boosted trees (why?), so you do not have to worry about choosing ntrees too large*

**$**Because you’re just smoothing out the trees, since its just the average of all trees.

**$**Mean of squared residuals = OOB MSE

**$**% Var explained = OOD R squared

**$**%IncMSE (Permutation Based Var Importance)

**$**%IncMSE is the percentage increase in MSE if you randomly jumble the column youre looking at and try to predict the dependent variable. Randomly permuting the column breaks the dependence.

*Random forests have virtually all the same many advantages of boosted trees. They inherit almost all the advantages of trees, but, like boosted trees, usually excellent predictive power. Unlike trees, they lose interpretability. But this can be improved with the build-in partial dependence (aka marginal) plots and the variable importance measures. Overall, random forests may give a little smoother model than boosted trees, whereas boosted trees may be a little better at capturing more complex nonlinearities.*

**Time Series**

Trend: : a deterministic linear, quadratic, exponential, etc. pattern over time

Predict: Simple extrapolation (linear, quadratic, etc.)

Seasonal: : up/down variations with a regular period (yearly, weekly, daily, etc.)

Prediction: July=Average of past few Julys, March same

Cyclical: : random longer-term up/down variations with an irregular period (often tied to the economy as a whole, war, natural disasters, etc.)

Prediction: Average of the past few observations (best prediction of the future value is just the current level

Random: : the completely random, totally unpredictable part that is left over

Prediction: Don’t even try

**MA:**

####MA filtering/prediction for chem.csv

chem<-read.csv("chem.csv",header=F)

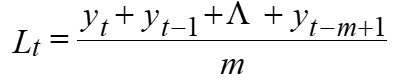
y<-ts(chem[[1]], frequency=1)

m=20;k=20;n=length(y) #m = MA window length, k = prediction horizon

plot(y,type="b",xlim=c(0,n+k))

MAchem<-filter(y, filter=rep(1/m,m), method = "convolution", sides = 1) (sides=1: non-centered moving average, 2:centered moving average)

yhat=c(NA, MAchem, rep(MAchem[n],k-1)) #One-step-ahead forecasts. The output of MAchem is L\_t. The leading NA in yhat gives yhat\_t = L\_(t-1)



**EWMA:**

####Manual EWMA filtering/prediction for chem.csv

y<-ts(chem[[1]], frequency=1)

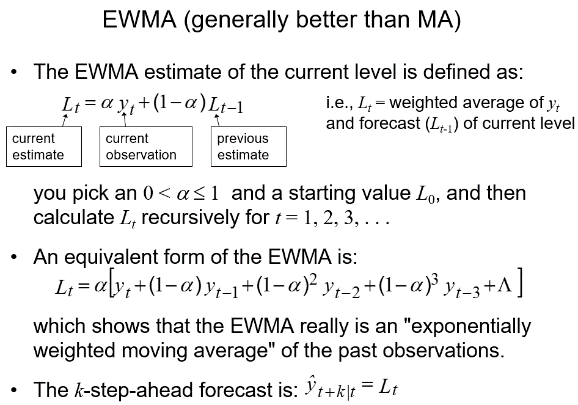
alpha=0.2;k=20;n=length(y) #alpha = EWMA parameter, k = prediction horizon

plot(y,type="b",xlim=c(0,n+k))

EWMAchem<-filter(alpha\*y, filter=1-alpha, method = "recursive", sides = 1, init=y[1])

yhat=c(NA,EWMAchem,rep(EWMAchem[n],k-1))

lines(yhat,col="red")



**Holt Winters:**

y<-ts(chem[[1]], frequency=1)

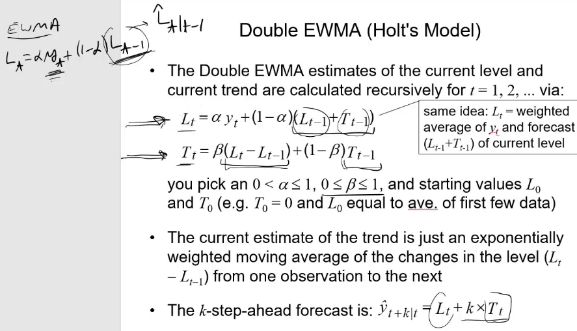
k=20;n=length(y) #k = prediction horizon

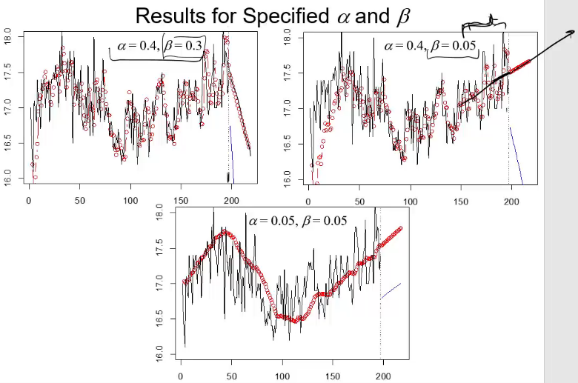
EWMAchem<-HoltWinters(y, seasonal = "additive", beta = FALSE, gamma = FALSE)

EWMAchemPred<-predict(EWMAchem, n.ahead=k, prediction.interval = TRUE, level = 0.95)

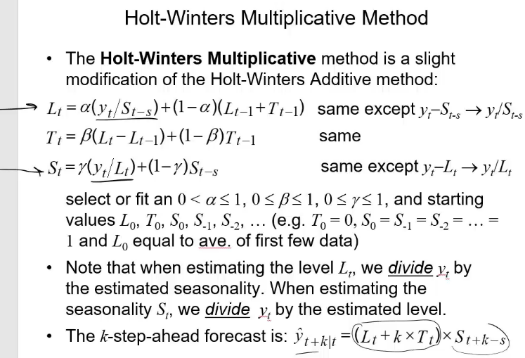
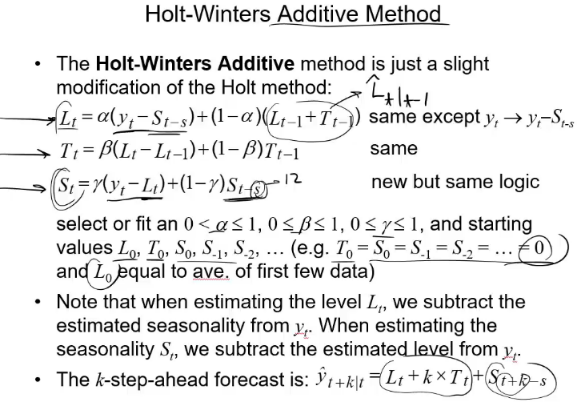
plot(EWMAchem,EWMAchemPred,type="b")

y<-ts(trade[[1]], deltat=1/12) #sampling interval corresponds to 1/12 the seasonality period

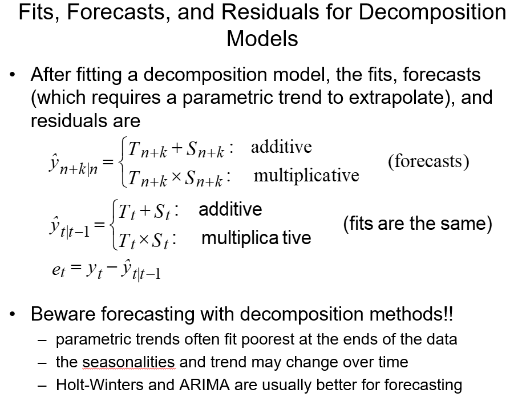




**Full Holt Winters:**



**Decomposition:**



**Usually the fit is worse at the ends of the data**

***In the decomposition model results, why are there no trend and residuals estimated for the first six months and last six months?***

**$Because it’s using a centered moving average filter**

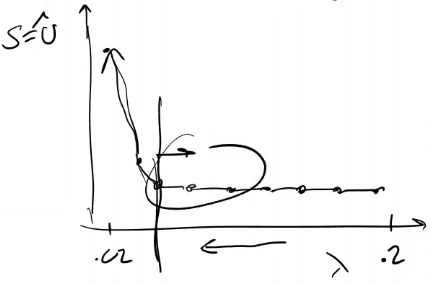
**Extras:**

given large n

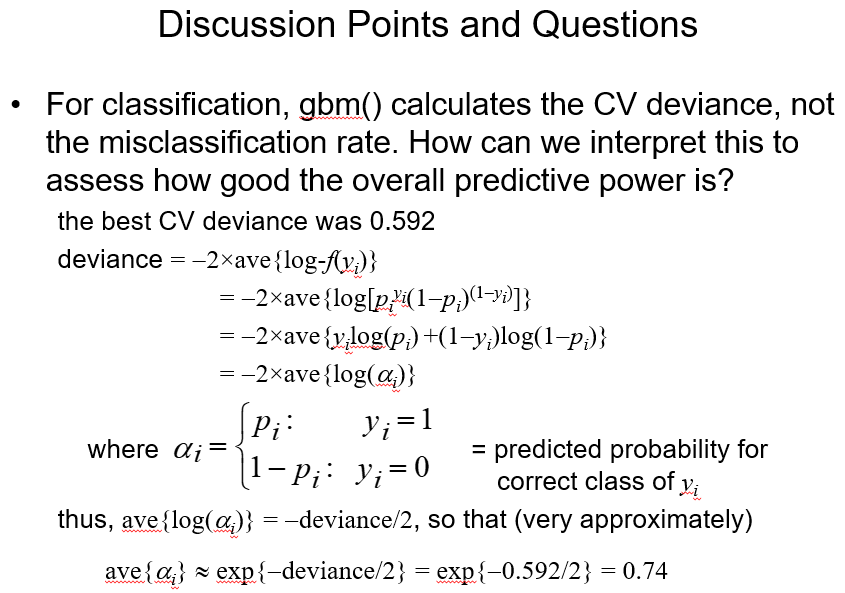
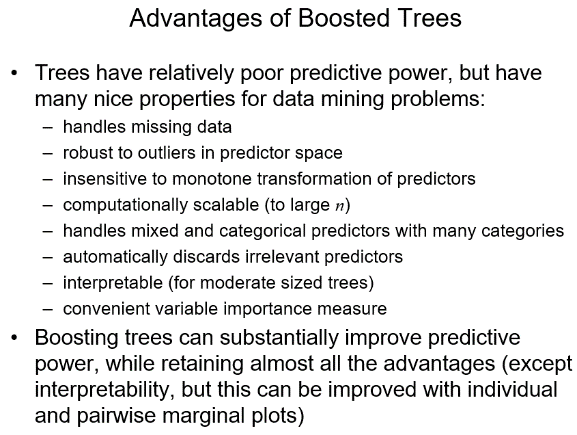
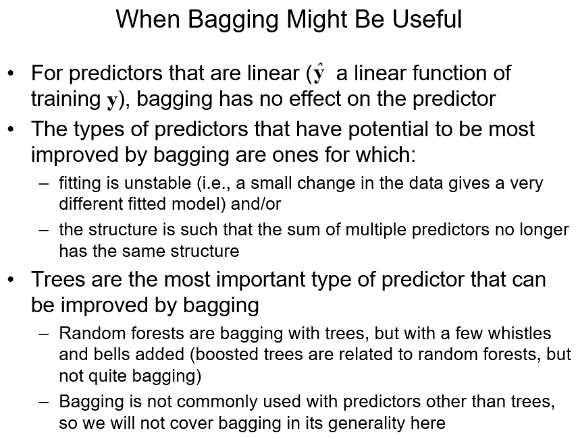
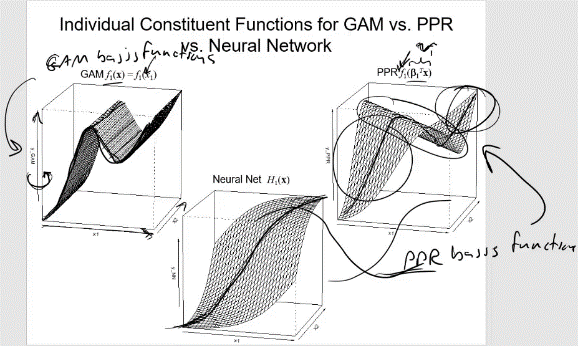
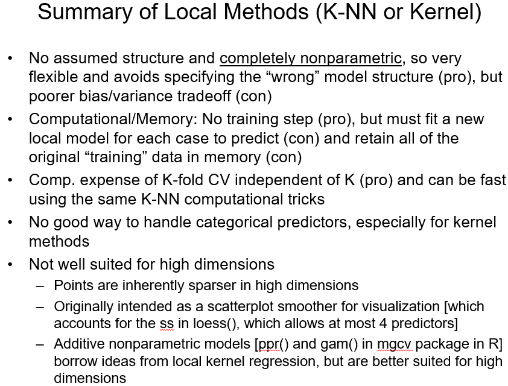
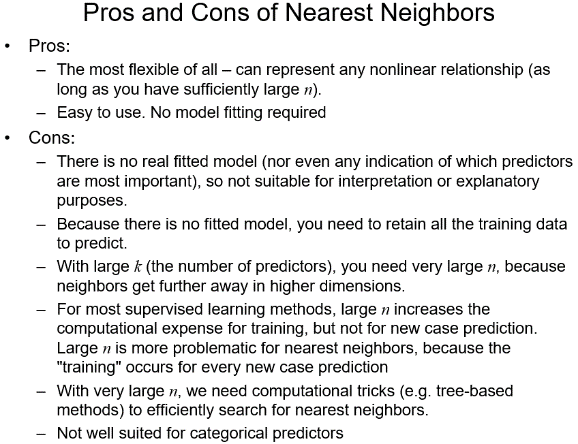
;

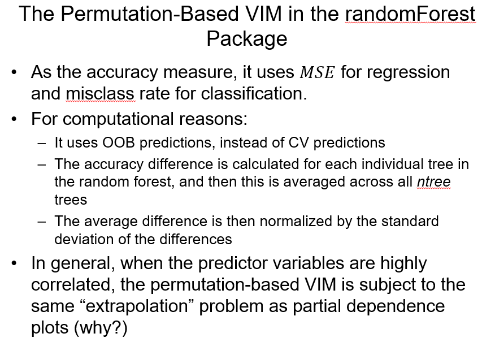
**where s can not depend on y, only x’s. S is an n x n matrix**

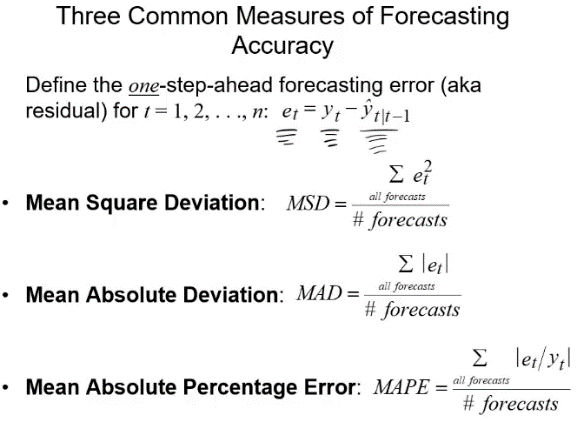
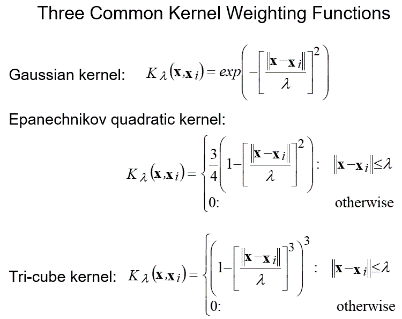
**= residual standard error**



**Summary(gbm1,n.trees=best.iter)=feature importance**







**Sqrt(MSEAve) = CV estimate of the prediction error**

**> test<-c(59, 0, 10, 0, 3, 0, 4, 300)**

**> xbar<-apply(as.matrix(IHD[,2:9]),2,mean)**

**> std<-apply(as.matrix(IHD[,2:9]),2,sd)**

**> test<-(test-xbar)/std #must standardize**

**> test<-matrix(test,1,8)**

**> K<-8**

**> out<-ann(as.matrix(IHD1[,2:9]),test,K,verbose=F)**

**> ind<-matrix(out$knnIndexDist[,1:K],nrow(test),K1)**

**> yhat<-mean(log10(IHD1$cost[ind]))**

**> ind #indices of 8 nearest neighbors**

**[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]**

**[1,] 278 717 778 517 37 659 114 456**

**> yhat**

**[1] 3.379489**

**PPR: Most significant terms have low p-value**

**> xbar<-apply(as.matrix(IHD[,2:9]),2,mean)**

**> std<-apply(as.matrix(IHD[,2:9]),2,sd)**

**> test<-(test-xbar)/std #must standardize**

**> best.iter <- gbm.perf(gbm1,method="cv");best.iter**

**> SSECV<-gbm1$cv.error[best.iter]\*nrow(IHD1);SSECV**

**MISCLASS[j,]=sum(y != yhat1)/length(y)**

**> k=24;n=length(y) #k = prediction horizon**

**> Decair<-decompose(y, type = "additive") (multiplicative for multiplicative model)**

**> plot(Decair,type="b")**

**> y\_hat<-Decair$trend+Decair$seasonal**

**> plot(y,type="b")**

**> lines(y\_hat,col="red")**