STT481 General Additive Models

Colin Copeland

December 11, 2018

Skip to line 294 for Colin's portion. Copied Jess's data transformation.

The purpose of this document is to explore tree-based methods using the Ames, Iowa housing data set. I will focus on both predictive accuracy and model interpretation.

Data

The following section will deal with data transformation. Here, I used the same methods as in the midterm.

```
train <- read.csv("train.csv", na.strings="placeholder")  # some of the categorical variables ha
ve value "NA" but it doesn't mean null
test <- read.csv("test.csv", na.strings="placeholder")
house <- rbind(train, data.frame(test, SalePrice=rep(1, nrow(test))))
```

Dealing with NA's

Because there were also strings that were "NA" as part of some scales, I noted which columns shouldn't contain the string "NA"", and I change those strings to a true `NA`.

```
## Store all columns that can have "NA" as a valid entry
na_names = c("Alley", "BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "F
ireplaceQu", "GarageType", "GarageQual", "GarageCond", "PoolQC", "Fence", "MiscFeature", "MasVnr
Type")
## Replace "NA" strings with true NA in training data
for (j in 1:ncol(house)) {
  if (sum(colnames(house)[j]==na_names)==0) {  # if the column shouldn't contain "NA"...
    for (i in 1:nrow(house)) {
      if (house[i,j]=="NA") {
        house[i,j] <- NA  # if the column shouldn't contain NA but the cell is "NA", then give i
t a null
      }
    }
  }
}
```

Checking and Changing Data Types

I will be converting scales to factors (e.g., quality) because they describe a condition, not a quantity. There are mixed opinions on how these should be handled, but I am chosing to use the "nominal categorical" method. Years will be treated as integers; months will be treated as factors.

```
## Scales / Factors
house$MSSubClass <- as.factor(house$MSSubClass)
house$MSZoning <- as.factor(house$MSZoning)
house$Street <- as.factor(house$Street)
house$Alley <- as.factor(house$Alley)
house$LotShape <- as.factor(house$LotShape)
house$LandContour <- as.factor(house$LandContour)
house$Utilities <- as.factor(house$Utilities)
house$LotConfig <- as.factor(house$LotConfig)
house$LandSlope <- as.factor(house$LandSlope)
house$Neighborhood <- as.factor(house$Neighborhood)
house$Condition1 <- as.factor(house$Condition1)
house$Condition2 <- as.factor(house$Condition2)
house$BldgType <- as.factor(house$BldgType)
house$HouseStyle <- as.factor(house$HouseStyle)
house$OverallQual <- as.factor(house$OverallQual)
house$OverallCond <- as.factor(house$OverallCond)
house$RoofStyle <- as.factor(house$RoofStyle)
house$RoofMatl <- as.factor(house$RoofMatl)
house$Exterior1st <- as.factor(house$Exterior1st)
house$Exterior2nd <- as.factor(house$Exterior2nd)
house$MasVnrType <- as.factor(house$MasVnrType)
house$ExterQual <- as.factor(house$ExterQual)
house$ExterCond <- as.factor(house$ExterCond)
house$Foundation <- as.factor(house$Foundation)
house$BsmtQual <- as.factor(house$BsmtQual)
house$BsmtCond <- as.factor(house$BsmtCond)
house$BsmtExposure <- as.factor(house$BsmtExposure)
house$BsmtFinType1 <- as.factor(house$BsmtFinType1)
house$BsmtFinType2 <- as.factor(house$BsmtFinType2)
house$Heating <- as.factor(house$Heating)
house$HeatingQC <- as.factor(house$HeatingQC)
house$CentralAir <- as.factor(house$CentralAir)
house$Electrical <- as.factor(house$Electrical)
house$KitchenQual <- as.factor(house$KitchenQual)
house$Functional <- as.factor(house$Functional)
house$FireplaceQu <- as.factor(house$FireplaceQu)
house$GarageType <- as.factor(house$GarageType)
house$GarageFinish <- as.factor(house$GarageFinish)
house$GarageQual <- as.factor(house$GarageQual)
house$GarageCond <- as.factor(house$GarageCond)
house$PavedDrive <- as.factor(house$PavedDrive)
house$PoolQC <- as.factor(house$PoolQC)
house$Fence <- as.factor(house$Fence)
house$MiscFeature <- as.factor(house$MiscFeature)
house$SaleType <- as.factor(house$SaleType)
house$SaleCondition <- as.factor(house$SaleCondition)
## Should be numeric...
house$LotFrontage <- as.integer(house$LotFrontage)
house$LotArea <- as.integer(house$LotArea)
house$MasVnrArea <- as.integer(house$MasVnrArea)
house$BsmtFinSF1 <- as.integer(house$BsmtFinSF1)
house$BsmtFinSF2 <- as.integer(house$BsmtFinSF2)
```

```
house$BsmtUnfSF <- as.integer(house$BsmtUnfSF)
house$TotalBsmtSF <- as.integer(house$TotalBsmtSF)
house$X1stFlrSF <- as.integer(house$X1stFlrSF)
house$X2ndFlrSF <- as.integer(house$X2ndFlrSF)
house$LowQualFinSF <- as.integer(house$LowQualFinSF)
house$GrLivArea <- as.integer(house$GrLivArea)
house$BsmtFullBath <- as.integer(house$BsmtFullBath)
house$BsmtHalfBath <- as.integer(house$BsmtHalfBath)
house$FullBath <- as.integer(house$FullBath)
house$HalfBath <- as.integer(house$HalfBath)
house$BedroomAbvGr <- as.integer(house$BedroomAbvGr)
house$KitchenAbvGr <- as.integer(house$KitchenAbvGr)
house$TotRmsAbvGrd <- as.integer(house$TotRmsAbvGrd)
house$Fireplaces <- as.integer(house$Fireplaces)
house$GarageCars <- as.integer(house$GarageCars)
house$GarageArea <- as.integer(house$GarageArea)
house$WoodDeckSF <- as.integer(house$WoodDeckSF)
house$OpenPorchSF <- as.integer(house$OpenPorchSF)
house$EnclosedPorch <- as.integer(house$EnclosedPorch)
house$X3SsnPorch <- as.integer(house$X3SsnPorch)
house$ScreenPorch <- as.integer(house$ScreenPorch)
house$PoolArea <- as.integer(house$PoolArea)
house$MiscVal <- as.integer(house$MiscVal)
## Dates: years as integers, months as factors
house$YearBuilt <- as.integer(house$YearBuilt)
house$YearRemodAdd <- as.integer(house$YearRemodAdd)
house$GarageYrBlt <- as.integer(house$GarageYrBlt)
house$MoSold <- as.factor(house$MoSold)
house$YrSold <- as.integer(house$YrSold)
```

NA Revisited

Change the string “NA” to “N/A” for variables that are allowed to have “NA” as a value (e.g., Alley). I don’t change the “NA” strings to `NA` here (I did it earlier) because otherwise the change of class insert interpolated values instead of `NA` s.

```
## Store all columns that can have "NA" as a valid entry
na_names = c("Alley", "BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "F
ireplaceQu", "GarageType", "GarageQual", "GarageCond", "PoolQC", "Fence", "MiscFeature", "MasVnr
Type")
## Re-level "NA" columns
for (j in 1:ncol(house)) {
  if (sum(colnames(house)[j]==na_names)!=0) {  # if the column can contain "NA" as a string in t
he training data...
    levels(house[,j])[levels(house[,j])=="NA"] <- "N/A"
  }
}
```

Remove NA Columns

Remove columns from both sets which have too many NAs in the training set, and then remove rows from the training set with NAs left.

```
## Remove NA columns
ct_na <- rep(0, length=ncol(house))
for (j in 1:ncol(house)) {
  ct_na[j] <- sum(is.na(house[,j]))
}
house <- house[-c(1:80)[ct_na>50]]
```

Interpolate NA values in the data

```
## Note which columns need to have NAs interpolated
ct_na <- rep(0, length=ncol(house))
for (j in 1:ncol(house)) {
  ct_na[j] <- sum(is.na(house[,j]))
}
ct_na
```

```
 [1]  0  0  4  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  1
[24]  1  0 23  0  0  0  0  0  0  0  1  0  1  1  1  0  0  0  1  0  0  0  0
[47]  2  2  0  0  0  0  1  0  2  0  0  0  1  1  0  0  0  0  0  0  0  0  0
[70]  0  0  0  0  0  0  1  0  0
```

```
na_boo <- ifelse(ct_na!=0, T, F)
colnames(house)[na_boo]
```

```
 [1] "MSZoning"     "Utilities"    "Exterior1st"  "Exterior2nd"
 [5] "MasVnrArea"   "BsmtFinSF1"   "BsmtFinSF2"   "BsmtUnfSF"
 [9] "TotalBsmtSF"  "Electrical"   "BsmtFullBath" "BsmtHalfBath"
[13] "KitchenQual"  "Functional"   "GarageCars"   "GarageArea"
[17] "SaleType"
```

```
## Choose most common factor
house$MSZoning[is.na(house$MSZoning)] <- "RL"
house$Utilities[is.na(house$Utilities)] <- "AllPub"
house$Exterior1st[is.na(house$Exterior1st)] <- "VinylSd"
house$Exterior2nd[is.na(house$Exterior2nd)] <- "VinylSd"
house$Electrical[is.na(house$Electrical)] <- "SBrkr"
house$KitchenQual[is.na(house$KitchenQual)] <- "TA"
house$Functional[is.na(house$Functional)] <- "Typ"
house$SaleType[is.na(house$SaleType)] <- "WD"
for (i in 1:nrow(house)) {
  ## Check logic on Masonry veneer
  if (is.na(house$MasVnrArea[i])) {
    if (house$MasVnrType[i] == "None") {
      house$MasVnrArea[i] <- 0  # if there isn't any masonry veneer, then the NA should be repla
ced with 0
    } else {
      house$MasVnrArea[i] <- mean(house$MasVnrArea, na.rm=T)  # if there is veneer, replace with
 the average
    }
  }

  ## Basement
  if (is.na(house$BsmtFinSF1[i])) {
    if (house$BsmtQual[i] != "N/A") {
      # if there is a basement, then use the average
      house$BsmtFinSF1[i] <- mean(house$BsmtFinSF1, na.rm=T)
    } else {
      # if there isn't a basement, use 0
      house$BsmtFinSF1[i] <- 0
    }
  }
  if (is.na(house$BsmtFinSF2[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$BsmtFinSF2[i] <- mean(house$BsmtFinSF2, na.rm=T)
    } else {
      house$BsmtFinSF2[i] <- 0
    }
  }
  if (is.na(house$BsmtUnfSF[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$BsmtUnfSF[i] <- mean(house$BsmtUnfSF, na.rm=T)
    } else {
      house$BsmtUnfSF[i] <- 0
    }
  }
  if (is.na(house$TotalBsmtSF[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$TotalBsmtSF[i] <- mean(house$TotalBsmtSF, na.rm=T)
    } else {
      house$TotalBsmtSF[i] <- 0
    }
  }
  if (is.na(house$BsmtFullBath[i])) {
```

```
    if (house$BsmtQual[i]!="N/A") {
      house$BsmtFullBath[i] <- mean(house$BsmtFullBath, na.rm=T)
    } else {
      house$BsmtFullBath[i] <- 0
    }
  }
  if (is.na(house$BsmtHalfBath[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$BsmtHalfBath[i] <- mean(house$BsmtHalfBath, na.rm=T)
    } else {
      house$BsmtHalfBath[i] <- 0
    }
  }
  ## Garage
  if (is.na(house$GarageCars[i])) {
    if (house$GarageType[i] != "N/A") {
      # if there is a garage, then use the average
      house$GarageCars[i] <- mean(house$GarageCars, na.rm=T)
    } else {
      # if there isn't a basement, use 0
      house$GarageCars[i] <- 0
    }
  }
  if (is.na(house$GarageArea[i])) {
    if (house$GarageType[i] != "N/A") {
      # if there is a garage, then use the average
      house$GarageArea[i] <- mean(house$GarageArea, na.rm=T)
    } else {
      # if there isn't a basement, use 0
      house$GarageArea[i] <- 0
    }
  }
}
## Note which columns need to have NAs interpolated
ct_na <- rep(0, length=ncol(house))
for (j in 1:ncol(house)) {
  ct_na[j] <- sum(is.na(house[,j]))
}
ct_na
```

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[71] 0 0 0 0 0 0 0 0
```

```
na_boo <- ifelse(ct_na!=0, T, F)
colnames(house)[na_boo]  # all set!
```

```
character(0)
```

Split

The following code splits the data back into the training and testing sets.

```
train <- house[1:1460,]
test <- house[1461:2919,]
```

Start of Colin’s portion

Create subsets for GAM

```
library(leaps)
library(gam)
train_forward <-regsubsets(log(SalePrice)~., data = train,nvmax = 78, method = "forward")
```

```
Reordering variables and trying again:
```

```
train_forward_summary <-summary(train_forward)
train_backward <-regsubsets(log(SalePrice)~., data = train,nvmax = 78, method = "backward")
```

```
Reordering variables and trying again:
```

```
train_backward_summary <-summary(train_backward)
train_mixed <-regsubsets(log(SalePrice)~., data = train,nvmax = 60, method = "seqrep")
```

```
Reordering variables and trying again:
```

```
train_mixed_summary <-summary(train_mixed)
min_forward <- which.min(train_forward_summary$bic)
min_backward <-which.min(train_backward_summary$bic)
min_mixed <- which.min(train_mixed_summary$bic)
min_forward
```

```
[1] 74
```

```
min_backward
```

```
[1] 63
```

```
min_mixed
```

```
[1] 53
```

```
coef(train_forward, 12)
```

```
      (Intercept) NeighborhoodIDOTRR       OverallCond3
     11.6054774334      -0.3713319500      -0.4256520893
      OverallCond4       OverallCond7       RoofStyleHip
     -0.2446257890      -0.1263228255       0.1511919543
  RoofStyleMansard        CentralAirY     FunctionalMin2
      0.0255721843       0.3909434271      -0.0918587626
     FunctionalMod      FireplaceQuGd       GarageQualGd
      0.0182024092       0.2544843284       0.1818876077
        X3SsnPorch
      0.0003776024
```

```
coef(train_backward, 12)
```

```
       (Intercept)       RoofStyleHip   RoofStyleMansard
        11.18410673         0.13223110        -0.18246318
   RoofMatlTar&Grv    RoofMatlWdShake    RoofMatlWdShngl
         0.12316819         0.22946717         0.55280140
Exterior1stBrkComm Exterior1stBrkFace  Exterior1stCBlock
        -0.64278667         0.01160286        -0.36290181
       TotRmsAbvGrd     FunctionalMaj2       GarageQualGd
         0.12341845        -0.56221982         0.13954258
        ScreenPorch
         0.00052001
```

```
coef(train_mixed, 13)
```

```
       (Intercept)            LotArea NeighborhoodIDOTRR
      1.154315e+01       7.955635e-06      -3.572625e-01
      OverallCond3       OverallCond4       OverallCond7
     -4.241780e-01      -2.545908e-01      -1.293641e-01
      RoofStyleHip   RoofStyleMansard        CentralAirY
      1.366124e-01       2.309446e-02       3.775148e-01
    FunctionalMin2      FunctionalMod      FireplaceQuGd
     -9.893386e-02      -1.795288e-02       2.359386e-01
      GarageQualGd         X3SsnPorch
      1.804341e-01       3.464450e-04
```

for some reason, I was unable to replicate my original subset selection where I found the tw elve variables I ended up using later

```
# for the record, I had cubic splines and smoothing splines mixed up in variable naming, so belo
w, cubic_splines means smoothing splines, and smoothing_splines means cubic splines.
# smoothing splines
gam_cubic_splines <- gam(log(SalePrice) ~ s(LotArea) + OverallQual + OverallCond + s(YearBuilt)
 + s(YearRemodAdd) + s(X1stFlrSF) + X2ndFlrSF + s(BsmtFullBath) + s(KitchenAbvGr) + s(TotRmsAbvG
rd) + Fireplaces + GarageCars, data = train)
gam_cubic_splines_10 <- gam(log(SalePrice) ~ s(LotArea, df = 10) + OverallQual + OverallCond + s
(YearBuilt, df = 10) + s(YearRemodAdd, df = 10) + s(X1stFlrSF, df = 10) + X2ndFlrSF + s(BsmtFull
Bath, df = 10) + s(KitchenAbvGr, df = 10) + s(TotRmsAbvGrd, df = 10) + Fireplaces + GarageCars,
 data = train)
gam_cubic_splines_20 <- gam(log(SalePrice) ~ s(LotArea, df = 20) + OverallQual + OverallCond + s
(YearBuilt, df = 20) + s(YearRemodAdd, df = 20) + s(X1stFlrSF, df = 20) + X2ndFlrSF + s(BsmtFull
Bath, df = 20) + s(KitchenAbvGr, df = 20) + s(TotRmsAbvGrd, df = 20) + Fireplaces + GarageCars,
 data = train)
gam_cubic_splines_2 <- gam(log(SalePrice) ~ s(LotArea, df = 2) + OverallQual + OverallCond + s(Y
earBuilt, df = 2) + s(YearRemodAdd, df = 2) + s(X1stFlrSF, df = 2) + X2ndFlrSF + s(BsmtFullBath,
 df = 2) + s(KitchenAbvGr, df = 2) + s(TotRmsAbvGrd, df = 2) + Fireplaces + GarageCars, data = t
rain)
gam_cubic_splines_15 <- gam(log(SalePrice) ~ s(LotArea, df = 15) + OverallQual + OverallCond + s
(YearBuilt, df = 15) + s(YearRemodAdd, df = 15) + s(X1stFlrSF, df = 15) + X2ndFlrSF + s(BsmtFull
Bath, df = 15) + s(KitchenAbvGr, df = 15) + s(TotRmsAbvGrd, df = 15) + Fireplaces + GarageCars,
 data = train)
# natural splines
gam_natural_splines_10 <- gam(log(SalePrice) ~ ns(LotArea, df = 10) + OverallQual + OverallCond
 + ns(YearBuilt, df = 10) + ns(YearRemodAdd, df = 10) + ns(X1stFlrSF, df = 10) + X2ndFlrSF + ns
(BsmtFullBath, df = 10) + ns(KitchenAbvGr, df = 10) + ns(TotRmsAbvGrd, df = 10) + Fireplaces + G
arageCars, data = train)
gam_natural_splines_2 <- gam(log(SalePrice) ~ ns(LotArea, df = 2) + OverallQual + OverallCond +
 ns(YearBuilt, df = 2) + ns(YearRemodAdd, df = 2) + ns(X1stFlrSF, df = 2) + X2ndFlrSF + ns(BsmtF
ullBath, df = 2) + ns(KitchenAbvGr, df = 2) + ns(TotRmsAbvGrd, df = 2) + Fireplaces + GarageCar
s, data = train)
gam_natural_splines_20 <- gam(log(SalePrice) ~ ns(LotArea, df = 20) + OverallQual + OverallCond
 + ns(YearBuilt, df = 20) + ns(YearRemodAdd, df = 20) + ns(X1stFlrSF, df = 20) + X2ndFlrSF + ns
(BsmtFullBath, df = 20) + ns(KitchenAbvGr, df = 20) + ns(TotRmsAbvGrd, df = 20) + Fireplaces + G
arageCars, data = train)
# local regression
gam_local_regression <- gam(log(SalePrice) ~ lo(LotArea) + OverallQual + OverallCond + lo(YearBu
ilt) + lo(YearRemodAdd) + lo(X1stFlrSF) + X2ndFlrSF + lo(BsmtFullBath) + lo(KitchenAbvGr) + lo(T
otRmsAbvGrd) + Fireplaces + GarageCars, data = train)
gam_local_regression_20 <- gam(log(SalePrice) ~ lo(LotArea, span = .2) + OverallQual + OverallCo
nd + lo(YearBuilt, span = .2) + lo(YearRemodAdd, span = .2) + lo(X1stFlrSF, span = .2) + X2ndFlr
SF + lo(BsmtFullBath, span = .2) + lo(KitchenAbvGr, span = .2) + lo(TotRmsAbvGrd, span = .2) + F
ireplaces + GarageCars, data = train)
gam_local_regression_10 <- gam(log(SalePrice) ~ lo(LotArea, span = .1) + OverallQual + OverallCo
nd + lo(YearBuilt, span = .1) + lo(YearRemodAdd, span = .2) + lo(X1stFlrSF, span = .1) + X2ndFlr
SF + lo(BsmtFullBath, span = .2) + lo(KitchenAbvGr, span = .2) + lo(TotRmsAbvGrd, span = .2) + F
ireplaces + GarageCars, data = train)
#cubic splines
gam_smoothing_splines_20 <- gam(log(SalePrice) ~ bs(LotArea, df = 20) + OverallQual + OverallCon
d + bs(YearBuilt, df = 20) + bs(YearRemodAdd, df = 20) + bs(X1stFlrSF, df = 20) + X2ndFlrSF + bs
(BsmtFullBath, df = 20) + bs(KitchenAbvGr, df = 20) + bs(TotRmsAbvGrd, df = 20) + Fireplaces + G
arageCars, data = train)
gam_smoothing_splines_10 <- gam(log(SalePrice) ~ bs(LotArea, df = 10) + OverallQual + OverallCon
```

```
d + bs(YearBuilt, df = 10) + bs(YearRemodAdd, df = 10) + bs(X1stFlrSF, df = 10) + X2ndFlrSF + bs
(BsmtFullBath, df = 10) + bs(KitchenAbvGr, df = 10) + bs(TotRmsAbvGrd, df = 10) + Fireplaces + G
arageCars, data = train)
gam_smoothing_splines <- gam(log(SalePrice) ~ bs(LotArea) + OverallQual + OverallCond + bs(YearB
uilt) + bs(YearRemodAdd) + bs(X1stFlrSF) + X2ndFlrSF + bs(BsmtFullBath) + bs(KitchenAbvGr) + bs
(TotRmsAbvGrd) + Fireplaces + GarageCars, data = train)
```

```
# for making csv's of predictions
#gam_cubic_pred <- predict(gam_cubic_splines, test)
#write.csv(gam_cubic_pred, "gam_cubic_pred.csv")
#gam_cubic_pred_10 <- predict(gam_cubic_splines_10, test)
#write.csv(gam_cubic_pred_10, "gam_cubic_pred_10.csv")
#gam_cubic_pred_20 <- predict(gam_cubic_splines_20, test)
#write.csv(gam_cubic_pred_20, "gam_cubic_pred_20.csv")
#gam_cubic_pred_2 <- predict(gam_cubic_splines_2, test)
#write.csv(gam_cubic_pred_2, "gam_cubic_pred_2.csv")
##gam_cubic_pred_15 <- predict(gam_cubic_splines_15, test)
#write.csv(gam_cubic_pred_15, "gam_cubic_pred_15.csv")
#gam_natural_pred_10 <- predict(gam_natural_splines_10, test)
#write.csv(gam_natural_pred_10, "gam_natural_splines_10.csv")
#gam_natural_pred_2 <- predict(gam_natural_splines_2, test)
#write.csv(gam_natural_pred_2, "gam_natural_splines_2.csv")
#gam_natural_pred_20 <- predict(gam_natural_splines_20, test)
#write.csv(gam_natural_pred_20, "gam_natural_splines_20.csv")
#gam_local_regression_pred <- predict(gam_local_regression, test)
#write.csv(gam_local_regression_pred, "gam_local_regression.csv")
#gam_local_regression_20_pred <- predict(gam_local_regression_20, test)
#write.csv(gam_local_regression_20_pred, "gam_local_regression_20.csv")
#gam_local_regression_10_pred <- predict(gam_local_regression_10, test)
#write.csv(gam_local_regression_10_pred, "gam_local_regression_10.csv")
#gam_smoothing_splines_20_pred <- predict(gam_smoothing_splines_20, test)
#write.csv(gam_smoothing_splines_20_pred, "gam_smoothing_splines_20.csv")
#gam_smoothing_splines_10_pred <- predict(gam_smoothing_splines_10, test)
#write.csv(gam_smoothing_splines_10_pred, "gam_smoothing_splines_10.csv")
#gam_smoothing_splines_pred <- predict(gam_smoothing_splines, test)
#write.csv(gam_smoothing_splines_pred, "gam_smoothing_splines.csv")
```

Cross Validation

```
# preserve train for the future but also remove the two rows where OverallQual == 1 and makes it
 impossible to use cross-validation because of the factor newlevels error
future_train <- train
train <- subset(train, OverallQual != "1")
```

```
set.seed(505)
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE)
# create error vectors for each prediction
error_gam_cubic_10 <- rep(0, 10)
error_gam_cubic_20 <- rep(0, 10)
error_gam_cubic_2 <- rep(0, 10)
error_gam_cubic_15 <- rep(0, 10)
error_gam_cubic <- rep(0, 10)
error_gam_natural_10 <- rep(0, 10)
error_gam_natural_20 <- rep(0, 10)
error_gam_natural_2 <- rep(0, 10)
error_gam_local <- rep(0, 10)
error_gam_local_20 <- rep(0, 10)
error_gam_local_10 <- rep(0, 10)
error_gam_smooth_20 <- rep(0, 10)
error_gam_smooth_10 <- rep(0, 10)
error_gam_smooth <- rep(0, 10)
### NOTE
# lack of smoothing splines and local regression cross validation due to lack of >= 4 unique val
ues; smoothing by smoothing splines and local regression requires that.
for (i in 1:10){
  #gam_cubic_splines <- gam(log(SalePrice) ~ s(LotArea) + OverallQual + OverallCond + s(YearBuil
t) + s(YearRemodAdd) + s(X1stFlrSF) + X2ndFlrSF + s(BsmtFullBath) + KitchenAbvGr + s(TotRmsAbvGr
d) + Fireplaces + GarageCars, data = train[!(fold.index == i),])
  #gam_cubic_splines_10 <- gam(log(SalePrice) ~ s(LotArea, df = 10) + OverallQual + OverallCond
 + s(YearBuilt, df = 10) + s(YearRemodAdd, df = 10) + s(X1stFlrSF, df = 10) + X2ndFlrSF + s(Bsmt
FullBath, df = 10) + KitchenAbvGr + s(TotRmsAbvGrd, df = 10) + Fireplaces + GarageCars, data = t
rain[!(fold.index == i),])
  #gam_cubic_splines_20 <- gam(log(SalePrice) ~ s(LotArea, df = 20) + OverallQual + OverallCond
 + s(YearBuilt, df = 20) + s(YearRemodAdd, df = 20) + s(X1stFlrSF, df = 20) + X2ndFlrSF + s(Bsmt
FullBath, df = 20) + KitchenAbvGr + s(TotRmsAbvGrd, df = 20) + Fireplaces + GarageCars, data = t
rain[!(fold.index == i),])
  #gam_cubic_splines_2 <- gam(log(SalePrice) ~ s(LotArea, df = 2) + OverallQual + OverallCond +
 s(YearBuilt, df = 2) + s(YearRemodAdd, df = 2) + s(X1stFlrSF, df = 2) + X2ndFlrSF + s(BsmtFullB
ath, df = 2) + KitchenAbvGr + s(TotRmsAbvGrd, df = 2) + Fireplaces + GarageCars, data = train[!
(fold.index == i),])
  #gam_cubic_splines_15 <- gam(log(SalePrice) ~ s(LotArea, df = 15) + OverallQual + OverallCond
 + s(YearBuilt, df = 15) + s(YearRemodAdd, df = 15) + s(X1stFlrSF, df = 15) + X2ndFlrSF + s(Bsmt
FullBath, df = 15) + KitchenAbvGr + s(TotRmsAbvGrd, df = 15) + Fireplaces + GarageCars, data = t
rain[!(fold.index == i),])



  gam_natural_splines_10 <- gam(log(SalePrice) ~ ns(LotArea, df = 10) + OverallQual + OverallCon
d + ns(YearBuilt, df = 10) + ns(YearRemodAdd, df = 10) + ns(X1stFlrSF, df = 10) + X2ndFlrSF + ns
(BsmtFullBath, df = 10) + KitchenAbvGr + ns(TotRmsAbvGrd, df = 10) + Fireplaces + GarageCars, da
ta = train[!(fold.index == i),])
  gam_natural_splines_2 <- gam(log(SalePrice) ~ ns(LotArea, df = 2) + OverallQual + OverallCond
 + ns(YearBuilt, df = 2) + ns(YearRemodAdd, df = 2) + ns(X1stFlrSF, df = 2) + X2ndFlrSF + ns(Bsm
tFullBath, df = 2) + KitchenAbvGr + ns(TotRmsAbvGrd, df = 2) + Fireplaces + GarageCars, data = t
rain[!(fold.index == i),])
  gam_natural_splines_20 <- gam(log(SalePrice) ~ ns(LotArea, df = 20) + OverallQual + OverallCon
d + ns(YearBuilt, df = 20) + ns(YearRemodAdd, df = 20) + ns(X1stFlrSF, df = 20) + X2ndFlrSF + ns
(BsmtFullBath, df = 20) + KitchenAbvGr + ns(TotRmsAbvGrd, df = 20) + Fireplaces + GarageCars, da
```

```
ta = train[!(fold.index == i),])


  #gam_local_regression <- gam(log(SalePrice) ~ lo(LotArea) + OverallQual + OverallCond + lo(Yea
rBuilt) + lo(YearRemodAdd) + lo(X1stFlrSF) + X2ndFlrSF + lo(BsmtFullBath) + KitchenAbvGr + lo(To
tRmsAbvGrd) + Fireplaces + GarageCars, data = train[!(fold.index == i),])
  #gam_local_regression_20 <- gam(log(SalePrice) ~ lo(LotArea, span = .2) + OverallQual + Overal
lCond + lo(YearBuilt, span = .2) + lo(YearRemodAdd, span = .2) + lo(X1stFlrSF, span = .2) + X2nd
FlrSF + lo(BsmtFullBath, span = .2) + KitchenAbvGr + lo(TotRmsAbvGrd, span = .2) + Fireplaces +
 GarageCars, data = train[!(fold.index == i),])
  #gam_local_regression_10 <- gam(log(SalePrice) ~ lo(LotArea, span = .1) + OverallQual + Overal
lCond + lo(YearBuilt, span = .1) + lo(YearRemodAdd, span = .2) + lo(X1stFlrSF, span = .1) + X2nd
FlrSF + lo(BsmtFullBath, span = .2) + KitchenAbvGr + lo(TotRmsAbvGrd, span = .2) + Fireplaces +
 GarageCars, data = train[!(fold.index == i),])


  gam_smoothing_splines_20 <- gam(log(SalePrice) ~ bs(LotArea, df = 20) + OverallQual + OverallC
ond + bs(YearBuilt, df = 20) + bs(YearRemodAdd, df = 20) + bs(X1stFlrSF, df = 20) + X2ndFlrSF +
 bs(BsmtFullBath, df = 20) + KitchenAbvGr + bs(TotRmsAbvGrd, df = 20) + Fireplaces + GarageCars,
 data = train[!(fold.index == i),])
  gam_smoothing_splines_10 <- gam(log(SalePrice) ~ bs(LotArea, df = 10) + OverallQual + OverallC
ond + bs(YearBuilt, df = 10) + bs(YearRemodAdd, df = 10) + bs(X1stFlrSF, df = 10) + X2ndFlrSF +
 bs(BsmtFullBath, df = 10) + KitchenAbvGr + bs(TotRmsAbvGrd, df = 10) + Fireplaces + GarageCars,
 data = train[!(fold.index == i),])
  gam_smoothing_splines <- gam(log(SalePrice) ~ bs(LotArea) + OverallQual + OverallCond + bs(Yea
rBuilt) + bs(YearRemodAdd) + bs(X1stFlrSF) + X2ndFlrSF + bs(BsmtFullBath) + KitchenAbvGr + bs(To
tRmsAbvGrd) + Fireplaces + GarageCars, data = train[!(fold.index == i),])
#pred_gam_cubic_10 <- predict(gam_cubic_splines_10, train[(fold.index == i),])
#pred_gam_cubic_20 <- predict(gam_cubic_splines_20, train[(fold.index == i),])
#pred_gam_cubic_2 <- predict(gam_cubic_splines_2, train[(fold.index == i),])
#pred_gam_cubic_15 <- predict(gam_cubic_splines_15, train[(fold.index == i),])
#pred_gam_cubic <- predict(gam_cubic_splines, train[(fold.index == i),])

pred_gam_natural_10 <- predict(gam_natural_splines_10, train[(fold.index == i),])
pred_gam_natural_20 <- predict(gam_natural_splines_20, train[(fold.index == i),])
pred_gam_natural_2 <- predict(gam_natural_splines_2, train[(fold.index == i),])
#pred_gam_local <- predict(gam_local_regression, train[(fold.index == i),])
#pred_gam_local_20 <- predict(gam_local_regression_20, train[(fold.index == i),])
#pred_gam_local_10 <- predict(gam_local_regression_10, train[(fold.index == i),])
pred_gam_smooth_20 <- predict(gam_smoothing_splines_20, train[(fold.index == i),])
pred_gam_smooth_10 <- predict(gam_smoothing_splines_10, train[(fold.index == i),])
pred_gam_smooth <- predict(gam_smoothing_splines, train[(fold.index == i),])
cv_test <- log(train[fold.index==i,]$SalePrice)
#diff_gam_cubic_10 <- sqrt(mean((cv_test-pred_gam_cubic_10)^2))
#diff_gam_cubic_20 <- sqrt(mean((cv_test-pred_gam_cubic_20)^2))
#diff_gam_cubic_2 <- sqrt(mean((cv_test-pred_gam_cubic_2)^2))
#diff_gam_cubic_15 <- sqrt(mean((cv_test-pred_gam_cubic_15)^2))
#diff_gam_cubic <- sqrt(mean((cv_test-pred_gam_cubic)^2))
diff_gam_natural_10 <- sqrt(mean((cv_test-pred_gam_natural_10)^2))
diff_gam_natural_20 <- sqrt(mean((cv_test-pred_gam_natural_20)^2))
diff_gam_natural_2 <- sqrt(mean((cv_test-pred_gam_natural_2)^2))
#diff_gam_local <- sqrt(mean((cv_test-pred_gam_local)^2))
#diff_gam_local_20 <- sqrt(mean((cv_test-pred_gam_local_20)^2))
#diff_gam_local_10 <- sqrt(mean((cv_test-pred_gam_local_10)^2))
```

```
diff_gam_smooth_20 <- sqrt(mean((cv_test-pred_gam_smooth_20)^2))
diff_gam_smooth_10 <- sqrt(mean((cv_test-pred_gam_smooth_10)^2))
diff_gam_smooth <- sqrt(mean((cv_test-pred_gam_smooth)^2))
#error_gam_cubic_10[i] <- diff_gam_cubic_10
#error_gam_cubic_20[i] <- diff_gam_cubic_20
#error_gam_cubic_2[i] <- diff_gam_cubic_2
#error_gam_cubic_15[i] <- diff_gam_cubic_15
#error_gam_cubic[i] <- diff_gam_cubic
error_gam_natural_10[i] <- diff_gam_natural_10
error_gam_natural_20[i] <- diff_gam_natural_20
error_gam_natural_2[i] <- diff_gam_natural_2
#error_gam_local[i] <- diff_gam_local
#error_gam_local_20[i] <- diff_gam_local_20
#error_gam_local_10[i] <- diff_gam_local_10
error_gam_smooth_20[i] <- diff_gam_smooth_20
error_gam_smooth_10[i] <- diff_gam_smooth_10
error_gam_smooth[i] <- diff_gam_smooth
result_vector <- c(#"smoothing_10" = mean(error_gam_cubic_10), "smoothing_20" = mean(error_gam_c
ubic_20),
#"smoothing_2" = mean(error_gam_cubic_2),
#"smoothing_15" = mean(error_gam_cubic_15),
#"smoothing" = mean(error_gam_cubic),
"natural_10" = mean(error_gam_natural_10),
"natural_20" = mean(error_gam_natural_20),
"natural_2" = mean(error_gam_natural_2),
#"local" = mean(error_gam_local),
#"local_20" = mean(error_gam_local_20),
#"local_10" = mean(error_gam_local_10),
"cubic_20" = mean(error_gam_smooth_20),
"cubic_10" = mean(error_gam_smooth_10),
"cubic" = mean(error_gam_smooth))
result_vector
}
result_vector
```

```
natural_10 natural_20  natural_2   cubic_20   cubic_10      cubic
 0.1361267  0.1375086  0.1434455  0.1794352  0.1598580  0.1355766
```