Subset Selection

The following code chunk imports the pre-processed data. The test data set is given a dummy variable so that future functions can work on this data (there needs to be a dummy response so that the predict.regsubsets function can apply a formula to the test set for prediction).

```
load("train.JV.RData"); load("test.JV.RData")

test$SalePrice <- rep(1, nrow(test))  # make a "dummy" column for the test SalePrice in order fo
r the prediction to work
```

This code is used to check whether the model matrices for the training set and the test set match up. Any columns in the training model matrix that aren't present in the test model matrix have their variable removed because that variable won't be used to predict in the test set.

```
train_mat <- model.matrix(SalePrice ~.-Id, data=train); test_mat <- model.matrix(SalePrice~.-Id,
data=test)
for (i in 1:length(colnames(train_mat))) {
  if (sum(colnames(train_mat)[i] == colnames(test_mat))!=1) {
    print(colnames(train_mat)[i])
  }
}
```

```
bad_i <- c(1:length(colnames(test)))[colnames(test) %in% c("Condition2", "HouseStyle", "RoofMat
l", "Exterior1st", "Exterior2nd", "Heating", "Electrical", "GarageQual", "PoolQC", "MiscFeature"
)]
train <- train[,-bad_i]
test <- test[,-bad_i]
```

The `leaps` library has a `regsubsets` function that can be used to fit stepwise models and best subset models. The `Metrics` library has a `rmsle` function that can be used to evaluate error, because this is the same metric that Kaggle uses.

```
library(leaps)
library(Metrics)
```

The `regsubsets` function used in the `leaps` library doesn't have a prediction attribute, so this function will serve that purpose.

```
predict.regsubsets <- function (object, newdata , id, ...) {
  form <- as.formula(object$call[[2]])  # formula of full model
  test.mat <- model.matrix(form, newdata)  # building an X matrix from newdata
  coefi <- coef(object, id = id) # coefficient estimates associated with the object model
  xvars <- names(coefi) # names of the non-zero coefficient estimates

  for (i in 1:length(xvars)) {
    if (sum(xvars[i]==colnames(test.mat))!=1) {
      print(xvars[i])
    }
  }


  return(test.mat[,xvars] %*% coefi) # X[,non-zero variables] * Coefficients[non-zero variables]
}
```

Best Subsets Selection

Best subsets selection tests all the possible models from 0- to p-dimensional models, selecting the best m-dimensional model using the model RSS. Then all of the $p+1$ models are compared to each other using an estimation of the test error (e.g., adjusted R^2), and then the best model is selected.
Best subsets selection examines 2^p different models; in this case, the number of predictors results in more than 10^{20} models, which is computationally very expensive, so I will not perform this method.

Forward Stepwise Selection

Forward stepwise selection starts with a null model ($Y = \beta_0 + \epsilon$), and then tests all models of m dimension and selects the best using the model RSS. Each subsequent model includes all the previously selected predictors plus one more predictor that most improves the model. Once there are $p+1$ models (one for each possible dimension), the best model is selected using some sort of test error estimation criteria (e.g., adjusted R^2).

The following code fits a forward stepwise model to the training data, using a maximum of 200 variables (which would be almost a full model using all qualitative and categorical-dummy variables); a higher number of variables seems excessive.

```
fwd.reg <- regsubsets(SalePrice~.-Id, data=train, nvmax=200, method="forward")  # perform forwar
d stepwise method
```

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

```
fwd.smry <- summary(fwd.reg)  # store summary
```

Adjusted R^2

The adjusted R^2 metric is used to select “good” models; it is an estimate of a test error. It is an adjustment to the training error that accounts for the fact that the R^2 of a model always increases when there are more predictors.

The following code extracts the dimension of the model that has the best (maximum) adjusted R^2 metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
fwd.id.adjr2 <- c(0:length(fwd.smry$adjr2)-1)[(fwd.smry$adjr2 == max(fwd.smry$adjr2))==T]  # whi
ch dimension of the forward model is best in terms of adj.R^2

fwd.pred.adjr2 <- predict.regsubsets(object=fwd.reg, newdata=test, id=fwd.id.adjr2)  # predictio
n using forward stepwise adj.R^2  model

fwd.pred.adjr2.df <- data.frame(Id=test$Id, SalePrice=fwd.pred.adjr2); write.csv(fwd.pred.adjr2.
df, "fwd.pred.adjr2.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

fwd.rmslek.adjr2 <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    fwd.kpred.adjr2 <- predict.regsubsets(fwd.reg, newdata=test.k, id=fwd.id.adjr2)  # make pred
iction for this test fold
    fwd.rmslek.adjr2 <- c(fwd.rmslek.adjr2, rmsle(actual=true.y, predicted=fwd.kpred.adjr2))  #
 store the RMSLE metric for this test fold
}

fwd.rmsle.adjr2 <- mean(fwd.rmslek.adjr2)  # calculate the average RMSLE
```

Mallow's C_p

The Mallow's C_p metric is used to select "good" models; it is an estimate of a test error. It is an adjustment to the training error that penalizes high-dimensional models.

The following code extracts the dimension of the model that has the best (minimum) Mallow's C_p metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
fwd.id.cp <- c(0:length(fwd.smry$cp)-1)[(fwd.smry$cp == min(fwd.smry$cp))==T]  # which dimension
 of the forward model is best in terms of Cp

fwd.pred.cp <- predict.regsubsets(object=fwd.reg, newdata=test, id=fwd.id.cp)  # prediction usin
g forward stepwise Cp model

fwd.pred.cp.df <- data.frame(Id=test$Id, SalePrice=fwd.pred.cp); write.csv(fwd.pred.cp.df, "fwd.
pred.cp.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

fwd.rmslek.cp <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    fwd.kpred.cp <- predict.regsubsets(fwd.reg, newdata=test.k, id=fwd.id.cp)  # make prediction
 for this test fold
    fwd.rmslek.cp <- c(fwd.rmslek.cp, rmsle(actual=true.y, predicted=fwd.kpred.cp))  # store the
 RMSLE metric for this test fold
}

fwd.rmsle.cp <- mean(fwd.rmslek.cp)  # calculate the average RMSLE
```

Bayes's Information Criterion

The Bayes's Information Criterion ("BIC") is used to select "good" models; it is an estimate of a test error. It is an adjustment to the training error that penalizes high-dimensional models based on the number of points used to fit the model. This method tends to prefer lower-dimensional models than the adjusted R^2 or Mallow's C_p.

The following code extracts the dimension of the model that has the best (minimum) BIC metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
fwd.id.bic <- c(0:length(fwd.smry$bic)-1)[(fwd.smry$bic == min(fwd.smry$bic))==T]  # which dimen
sion of the forward model is best in terms of bic

fwd.pred.bic <- predict.regsubsets(object=fwd.reg, newdata=test, id=fwd.id.bic)  # prediction us
ing forward stepwise bic model

fwd.pred.bic.df <- data.frame(Id=test$Id, SalePrice=fwd.pred.bic); write.csv(fwd.pred.bic.df, "f
wd.pred.bic.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

fwd.rmslek.bic <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    fwd.kpred.bic <- predict.regsubsets(fwd.reg, newdata=test.k, id=fwd.id.bic)  # make predicti
on for this test fold
    fwd.rmslek.bic <- c(fwd.rmslek.bic, rmsle(actual=true.y, predicted=fwd.kpred.bic))  # store
 the RMSLE metric for this test fold
}

fwd.rmsle.bic <- mean(fwd.rmslek.bic)  # calculate the average RMSLE
```

Cross Validation

The cross-validation approach to model selection is used to select tuning parameters; it estimates the test error, and then selects the tuning parameter with the lowest estimated test error.

The following code uses 5-fold cross-validation to fit a forward stepwise model, finds the “best” dimension ,and finds the estimated RMSLE for that dimension. It then makes a prediction using the full training data set and prints the results to a .csv file for submission.

```
set.seed(1)  # set seed for consistency of fold breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

nv <- 200

fwd.rmsleik.cv <- matrix(NA, ncol=nv, nrow=5)

for (k in 1:5) {
  train.k <- train[fold.index != k,]  # k training data
  test.k <- train[fold.index == k,]  # k test data
  true.y <- test.k[,"SalePrice"]  # k test response

  fwd.regk <- regsubsets(SalePrice ~ .-Id, data = train.k, nvmax = nv, method="forward")  # fit
 model using training fold

  for (i in 1:nv) {
    fwd.predk <- predict(fwd.regk, test.k, id = i)  # make predictions using all possible values
 of model dimension
    fwd.rmsleik.cv[k,i] <- rmsle(actual=true.y, predicted=fwd.predk)  # store the prediction err
ors for each possible dimension
  }
}
```

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

```
fwd.rmslei.cv <- colMeans(fwd.rmsleik.cv, na.rm=T)  # calculate the estimated prediction error f
or each possible dimension

fwd.id.cv <- which(fwd.rmslei.cv==min(fwd.rmslei.cv))-1  # which dimension is the best dimensio
n?

fwd.pred.cv <- predict.regsubsets(object=fwd.reg, newdata=test, id=fwd.id.cv)  # prediction usin
g forward stepwise CV model

fwd.pred.cv.df <- data.frame(Id=test$Id, SalePrice=fwd.pred.cv); write.csv(fwd.pred.cv.df, "fwd.
pred.cv.df.csv", row.names=F)  # write to CSV
```

Summary of Results

Method	Model	Dimension	Estimated RMSLE	Actual RMSLE
Forward Stepwise	Adjusted R^2	125	0.14826	0.29415
Forward Stepwise	Mallow's Cp	71	0.18468	0.22928
Forward Stepwise	Bayes' IC	39	0.24152	0.22727
Forward Stepwise	Cross-Validation	199	0.14963	0.22282

Backward Stepwise Selection

Backward stepwise selection starts with a full model ($Y = \beta_0 + +\beta_1 X_1 + \ldots + \beta_p X_p + \epsilon$), and then tests all models of m dimension and selects the best using the model RSS. Each subsequent model includes all the previously selected predictors except one less predictor whose removal most improves the model. Once there are $p + 1$ models (one for each possible dimension), the best model is selected using some sort of test error estimation criteria.

The following code fits a backward stepwise model to the training data, using a maximum of 200 variables (which would be almost a full model using all qualitative and categorical-dummy variables); a higher number of variables seems excessive.

```
bwd.reg <- regsubsets(SalePrice~.-Id, data=train, nvmax=200, method="backward")  # perform backw
ard stepwise method
```

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

```
bwd.smry <- summary(bwd.reg)  # store summary
```

Adjusted R^2

The adjusted R^2 metric is used to select “good” models; it is an estimate of a test error. It is an adjustment to the training error that accounts for the fact that the R^2 of a model always increases when there are more predictors.

The following code extracts the dimension of the model that has the best (maximum) adjusted R^2 metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
bwd.id.adjr2 <- c(0:length(bwd.smry$adjr2)-1)[(bwd.smry$adjr2 == max(bwd.smry$adjr2))==T]  # whi
ch dimension of the backward model is best in terms of adj.R^2

bwd.pred.adjr2 <- predict.regsubsets(object=bwd.reg, newdata=test, id=bwd.id.adjr2)  # predictio
n using backward stepwise adj.R^2  model

bwd.pred.adjr2.df <- data.frame(Id=test$Id, SalePrice=bwd.pred.adjr2); write.csv(bwd.pred.adjr2.
df, "bwd.pred.adjr2.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

bwd.rmslek.adjr2 <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    bwd.kpred.adjr2 <- predict.regsubsets(bwd.reg, newdata=test.k, id=bwd.id.adjr2)  # make pred
iction for this test fold
    bwd.rmslek.adjr2 <- c(bwd.rmslek.adjr2, rmsle(actual=true.y, predicted=bwd.kpred.adjr2))  #
 store the RMSLE metric for this test fold
}

bwd.rmsle.adjr2 <- mean(bwd.rmslek.adjr2)  # calculate the average RMSLE
```

Mallow’s C_p

The Mallow’s C_p metric is used to select “good” models; it is an estimate of a test error. It is an adjustment to the training error that penalizes high-dimensional models.

The following code extracts the dimension of the model that has the best (minimum) Mallow’s C_p metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
bwd.id.cp <- c(0:length(bwd.smry$cp)-1)[(bwd.smry$cp == min(bwd.smry$cp))==T]  # which dimension
 of the backward model is best in terms of Cp

bwd.pred.cp <- predict.regsubsets(object=bwd.reg, newdata=test, id=bwd.id.cp)  # prediction usin
g backward stepwise Cp model

bwd.pred.cp.df <- data.frame(Id=test$Id, SalePrice=bwd.pred.cp); write.csv(bwd.pred.cp.df, "bwd.
pred.cp.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

bwd.rmslek.cp <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    bwd.kpred.cp <- predict.regsubsets(bwd.reg, newdata=test.k, id=bwd.id.cp)  # make prediction
 for this test fold
    bwd.rmslek.cp <- c(bwd.rmslek.cp, rmsle(actual=true.y, predicted=bwd.kpred.cp))  # store the
 RMSLE metric for this test fold
}

bwd.rmsle.cp <- mean(bwd.rmslek.cp)  # calculate the average RMSLE
```

Bayes's Information Criterion

The Bayes's Information Criterion ("BIC") is used to select "good" models; it is an estimate of a test error. It is an adjustment to the training error that penalizes high-dimensional models based on the number of points used to fit the model. This method tends to prefer lower-dimensional models than the adjusted R^2 or Mallow's C_p.

The following code extracts the dimension of the model that has the best (minimum) BIC metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
bwd.id.bic <- c(0:length(bwd.smry$bic)-1)[(bwd.smry$bic == min(bwd.smry$bic))==T]  # which dimen
sion of the backward model is best in terms of bic

bwd.pred.bic <- predict.regsubsets(object=bwd.reg, newdata=test, id=bwd.id.bic)  # prediction us
ing backward stepwise bic model

bwd.pred.bic.df <- data.frame(Id=test$Id, SalePrice=bwd.pred.bic); write.csv(bwd.pred.bic.df, "b
wd.pred.bic.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

bwd.rmslek.bic <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    bwd.kpred.bic <- predict.regsubsets(bwd.reg, newdata=test.k, id=bwd.id.bic)  # make predicti
on for this test fold
    bwd.rmslek.bic <- c(bwd.rmslek.bic, rmsle(actual=true.y, predicted=bwd.kpred.bic))  # store
 the RMSLE metric for this test fold
}

bwd.rmsle.bic <- mean(bwd.rmslek.bic)  # calculate the average RMSLE
```

Cross Validation

The cross-validation approach to model selection is used to select tuning parameters; it estimates the test error, and then selects the tuning parameter with the lowest estimated test error.

The following code uses 5-fold cross-validation to fit a backward stepwise model, finds the “best” dimension ,and finds the estimated RMSLE for that dimension. It then makes a prediction using the full training data set and prints the results to a .csv file for submission.

```
set.seed(1)  # set seed for consistency of fold breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

nv <- 200

bwd.rmsleik.cv <- matrix(NA, ncol=nv, nrow=5)

for (k in 1:5) {
  train.k <- train[fold.index != k,]  # k training data
  test.k <- train[fold.index == k,]  # k test data
  true.y <- test.k[,"SalePrice"]  # k test response

  bwd.regk <- regsubsets(SalePrice ~ .-Id, data = train.k, nvmax = nv, method="backward")  # fit
 model using training fold

  for (i in 1:nv) {
    bwd.predk <- predict(bwd.regk, test.k, id = i)  # make predictions using all possible values
 of model dimension
    bwd.rmsleik.cv[k,i] <- rmsle(actual=true.y, predicted=bwd.predk)  # store the prediction err
ors for each possible dimension
  }
}
```

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

```
bwd.rmslei.cv <- colMeans(bwd.rmsleik.cv, na.rm=T)  # calculate the estimated prediction error f
or each possible dimension

bwd.id.cv <- which(bwd.rmslei.cv==min(bwd.rmslei.cv))-1  # which dimension is the best dimensio
n?

bwd.pred.cv <- predict.regsubsets(object=bwd.reg, newdata=test, id=bwd.id.cv)  # prediction usin
g forward stepwise CV model

bwd.pred.cv.df <- data.frame(Id=test$Id, SalePrice=bwd.pred.cv); write.csv(bwd.pred.cv.df, "bwd.
pred.cv.df.csv", row.names=F)  # write to CSV
```

Summary of Results

Method	Model	Dimension	Estimated RMSLE	Actual RMSLE
Backward Stepwise	Adjusted R^2	133	0.14081	0.21879
Backward Stepwise	Mallow's Cp	87	0.16272	0.21734
Backward Stepwise	Bayes' IC	43	0.20075	0.23554
Backward Stepwise	Cross-Validation	184	0.14970	0.22234

Mixed Stepwise Selection

Mixed stepwise selection starts with a null model ($Y = \beta_0 + \epsilon$), and then selects the best 1-dimensional method. For all subsequent models, it tries to add or remove variables, and then selects the best $m - 1$ or $m + 1$ model that improves the existing model.

The following code fits a mixed stepwise model to the training data, using a maximum of 100 variables (this is smaller than the other two methods used above, but is less computationally expensive).

```
mwd.reg <- regsubsets(SalePrice~.-Id, data=train, nvmax=100, method="seqrep")  # perform mixed s
tepwise method
```

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

```
mwd.smry <- summary(mwd.reg)  # store summary
```

Adjusted R^2

The adjusted R^2 metric is used to select "good" models; it is an estimate of a test error. It is an adjustment to the training error that accounts for the fact that the R^2 of a model always increases when there are more predictors.

The following code extracts the dimension of the model that has the best (maximum) adjusted R^2 metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
mwd.id.adjr2 <- c(0:length(mwd.smry$adjr2)-1)[(mwd.smry$adjr2 == max(mwd.smry$adjr2))==T]  # whi
ch dimension of the mixed model is best in terms of adj.R^2

mwd.pred.adjr2 <- predict.regsubsets(object=mwd.reg, newdata=test, id=mwd.id.adjr2)  # predictio
n using mixed stepwise adj.R^2  model

mwd.pred.adjr2.df <- data.frame(Id=test$Id, SalePrice=mwd.pred.adjr2); write.csv(mwd.pred.adjr2.
df, "mwd.pred.adjr2.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

mwd.rmslek.adjr2 <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    mwd.kpred.adjr2 <- predict.regsubsets(mwd.reg, newdata=test.k, id=mwd.id.adjr2)  # make pred
iction for this test fold
    mwd.rmslek.adjr2 <- c(mwd.rmslek.adjr2, rmsle(actual=true.y, predicted=mwd.kpred.adjr2))  #
 store the RMSLE metric for this test fold
}

mwd.rmsle.adjr2 <- mean(mwd.rmslek.adjr2)  # calculate the average RMSLE
```

Mallow's C_p

The Mallow's C_p metric is used to select "good" models; it is an estimate of a test error. It is an adjustment to the training error that penalizes high-dimensional models.

The following code extracts the dimension of the model that has the best (minimum) Mallow's C_p metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
mwd.id.cp <- c(0:length(mwd.smry$cp)-1)[(mwd.smry$cp == min(mwd.smry$cp))==T]  # which dimension
 of the mixed model is best in terms of Cp

mwd.pred.cp <- predict.regsubsets(object=mwd.reg, newdata=test, id=mwd.id.cp)  # prediction usin
g mixed stepwise Cp model

mwd.pred.cp.df <- data.frame(Id=test$Id, SalePrice=mwd.pred.cp); write.csv(mwd.pred.cp.df, "mwd.
pred.cp.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

mwd.rmslek.cp <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    mwd.kpred.cp <- predict.regsubsets(mwd.reg, newdata=test.k, id=mwd.id.cp)  # make prediction
 for this test fold
    mwd.rmslek.cp <- c(mwd.rmslek.cp, rmsle(actual=true.y, predicted=mwd.kpred.cp))  # store the
 RMSLE metric for this test fold
}

mwd.rmsle.cp <- mean(mwd.rmslek.cp)  # calculate the average RMSLE
```

Bayes's Information Criterion

The Bayes's Information Criterion ("BIC") is used to select "good" models; it is an estimate of a test error. It is an adjustment to the training error that penalizes high-dimensional models based on the number of points used to fit the model. This method tends to prefer lower-dimensional models than the adjusted R^2 or Mallow's C_p.

The following code extracts the dimension of the model that has the best (minimum) BIC metric. It then uses that dimension to make a prediction for the `SalePrice` of the test data using the model developed in the beginning of this section. Then the results are exported to a .csv file for submission to Kaggle.

```
mwd.id.bic <- c(0:length(mwd.smry$bic)-1)[(mwd.smry$bic == min(mwd.smry$bic))==T]  # which dimen
sion of the mixed model is best in terms of bic

mwd.pred.bic <- predict.regsubsets(object=mwd.reg, newdata=test, id=mwd.id.bic)  # prediction us
ing mixed stepwise bic model

mwd.pred.bic.df <- data.frame(Id=test$Id, SalePrice=mwd.pred.bic); write.csv(mwd.pred.bic.df, "m
wd.pred.bic.df.csv", row.names=F)  # write to CSV
```

The following code estimates the test error using 5-fold cross-validation from the results above.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

mwd.rmslek.bic <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.k <- train[fold.index != k,]  # fold training set
    test.k <- train[fold.index == k,]  # fold test set
    true.y <- test.k[,"SalePrice"]  # fold test response

    mwd.kpred.bic <- predict.regsubsets(mwd.reg, newdata=test.k, id=mwd.id.bic)  # make predicti
on for this test fold
    mwd.rmslek.bic <- c(mwd.rmslek.bic, rmsle(actual=true.y, predicted=mwd.kpred.bic))  # store
 the RMSLE metric for this test fold
}

mwd.rmsle.bic <- mean(mwd.rmslek.bic, na.rm=T)  # calculate the average RMSLE
```

Cross Validation

The cross-validation approach to model selection is used to select tuning parameters; it estimates the test error, and then selects the tuning parameter with the lowest estimated test error.

The following code uses 5-fold cross-validation to fit a mixed stepwise model, finds the “best” dimension ,and finds the estimated RMSLE for that dimension. It then makes a prediction using the full training data set and prints the results to a .csv file for submission. I will use a maximum dimension of 75 for computational efficiency.

```
set.seed(1)  # set seed for consistency of fold breaks
fold.index <- cut(sample(1:nrow(train)), breaks=5, labels=FALSE)  # split data into 5 folds

nv <- 75

mwd.rmsleik.cv <- matrix(NA, ncol=nv, nrow=5)

for (k in 1:5) {
  train.k <- train[fold.index != k,]  # k training data
  test.k <- train[fold.index == k,]  # k test data
  true.y <- test.k[,"SalePrice"]  # k test response

  mwd.regk <- regsubsets(SalePrice ~ .-Id, data = train.k, nvmax = nv, method="seqrep")  # fit m
odel using training fold

  for (i in 1:nv) {
    mwd.predk <- predict(mwd.regk, test.k, id = i)  # make predictions using all possible values
 of model dimension
    mwd.rmsleik.cv[k,i] <- rmsle(actual=true.y, predicted=mwd.predk)  # store the prediction err
ors for each possible dimension
  }
}
```

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

```
mwd.rmslei.cv <- colMeans(mwd.rmsleik.cv, na.rm=T)  # calculate the estimated prediction error f
or each possible dimension

mwd.id.cv <- which(mwd.rmslei.cv==min(mwd.rmslei.cv))-1  # which dimension is the best dimensio
n?

mwd.pred.cv <- predict.regsubsets(object=mwd.reg, newdata=test, id=mwd.id.cv)  # prediction usin
g mixed stepwise CV model

mwd.pred.cv.df <- data.frame(Id=test$Id, SalePrice=mwd.pred.cv); write.csv(mwd.pred.cv.df, "mwd.
pred.cv.df.csv", row.names=F)  # write to CSV
```

Summary of Results

Method	Model	Dimension	Estimated RMSLE	Actual RMSLE
Mixed Stepwise	Adjusted R^2	99	0.15639	0.27081
Mixed Stepwise	Mallow’s Cp	41	0.17950	0.20876
Mixed Stepwise	Bayes’ IC	79	0.20814	0.23318
Mixed Stepwise	Cross-Validation	71	0.20407	0.22880