Problem and objective

My objective is to build a regression model to predict the values of the response on a test dataset of size m = 10000, given a dataset of 250 observations with 100 predictors and 1 response variable, Y.

Methods and Models

The X_{52} variable is removed due to a significant 193/250 of its observations being NA from the train-xy dataset. Consequently, we remove X_{52} from test-x dataset since we do not consider X_{52} when training our models. We have 99 Predictors now. I consider the 4 following models:

Model	Methodology
Lasso Regression	A regularisation technique that uses the shrinkage method to acquire the subset of predictors that minimises prediction error for the provided quantitative response variables with the use of an L1 penalty, using 10-fold cross-validation to train the model.
Ridge Regression	Like Lasso, a shrinkage technique is used to eliminate the multicollinearity of predictors with the L2 penalty, involving the application of the same factor on the coefficients. Unlike Lasso, no coefficient is left out. I use 10-fold cross-validation to train the model.
Forward Stepwise Selection/ Backward Stepwise Selection	**Forward stepwise selection** starts with a model with no predictors, then iteratively adds the most helpful predictor, one at a time. **Backward stepwise selection** starts with the full least squares model containing all p predictors, then iteratively removes the least helpful predictor, one at a time. **Note:** For both selections, I select the single best model using the **BIC** criteria.

Note: Principal Component Analysis is also **used but not part of my primary consideration**. More about the methodology and findings are on subsequent pages.

Findings

Model	Test MSE	No. of Predictors
Lasso Regression	24.8614	5
Ridge Regression	40.69052	99
Forward Stepwise Selection	29.91227	8
Backward Stepwise Selection	38.85134	7

Reflection

Since Lasso Regression Model consistently has the least MSE among the other 3 models, thus I will adopt the Lasso Regression Model to predict the 10000 observations test dataset. This can be further corroborated when we use different seed values other than 5 and Lasso having the smallest Test MSE majority of the time.

Further Details of training, diagnosis and validation of models

Note: For reproducibility of results, we set seed to 5 using set. seed (5) in R

Removal of X_{52} predictor variable:

```
no_of_na_in_x52 = length(which(is.na(read.csv("../data/train-xy.csv",
header = TRUE, sep = ",")[,53]))) #193
```

With the visual inspection of the data, it is clear that X_{52} needs to be removed. This is substantiated above when I realised that there is a whopping 193/250 observations with NA values, making it infeasible to use na.omit() function since large number of observations with significant influence on the models could be omitted. So, we remove the corresponding X_{52} column on both the given train_xy and test_x datasets.

```
train_xy <- read.csv("../data/train-xy.csv", header = TRUE, sep =
",")[,-53]
test_x <- read.csv("../data/test-x.csv" ,header = TRUE, sep = ",")[,-52]
```

```
# Similar to our tutorial approach, we split train_xy into training and
validation (test) sets
sampling = sample.int(2,nrow(train_xy),replace=TRUE)
# alternative split 70 % train, 30% test, commented out below:
#alt_s_train = sample(1:nrow(train_xy),0.8*nrow(train_xy),replace=FALSE)
```

I then split the train_xy dataset into the training dataset and testing dataset. Initially, I wanted to go based on what was taught in the Statistical Learning I module, split 70:30 OR 80:20 for training and validation data. Interestingly, that gave me a higher test MSE, so I settled with the roughly 50-50 style of splitting, such that each observation stands an approximately 50% chance to be grouped as 1 (train) and a 50% chance to be grouped as 2 (validation, a.k.a test)

Lasso Regression AND Ridge Regression Discussion

```
#Lasso Regression Method
lasso.cv = cv.glmnet(model.matrix(Y~.,train)[,-1],train$Y,alpha=1,nfolds
= 10)
lasso.pred = predict(lasso.cv,newx=model.matrix(Y~.,test)[,-1],s="lambda.
min",type="response")
lasso.mse = mean((lasso.pred-test$Y)^2) #24.8614 when nfolds= 10 and
25.13278 when nfolds = 5
lasso_selected_variables =
row.names(coef(lasso.cv))[as.matrix(coef(lasso.cv)!=0)]
```

Figure 1: Lasso Regression

```
#Ridge Regression Method
ridge.cv = cv.glmnet(model.matrix(Y~.,train)[,-1],train$Y,alpha=0, nfolds
= 10)
ridge.pred = predict(ridge.cv,newx=model.matrix(Y~.,test)[,-1],s="lambda.
min",type="response")
ridge.mse = mean((ridge.pred-test$Y)^2) #40.69052
ridge_selected_variables =
row.names(coef(ridge.cv))[as.matrix(coef(ridge.cv)!=0)]
```

Figure 2: Ridge Regression

Here, I used the glmnet library in R to perform 10-fold cross-validation on my train dataset, with the alpha parameter set as 1 to indicate lasso regression and 0 to indicate Ridge

regression. 10-fold cross-validation is adopted here to choose the optimal lambda min value considering the averaged fold errors. In my opinion, doing Lasso or ridge regression in a single fold might not be the best practice, as the lambda value might have smaller/discrepancies due to possible overfitting. Using CV will mitigate this potential setback by finding an optimal lambda min value that considers different batch sets of the training data to assess the optimality of lambda and to assess model and prediction error better. I tried using nfolds = 5, basically, 5-fold CV gave a higher Test MSE on both Ridge and Lasso so in the end nfolds set as 10. I believe this is due to the bias-variance tradeoff for K-fold CV, since a higher fold leads to more significant variance and increased flexibility. At the same time, smaller k gives larger bias for test error estimation due to decreased flexibility. Furthermore, a higher k = 10 folds mean that each of the respective models is trained on a larger training set and tested on a smaller fold, which leads to lower prediction error since the model can access more observations from the training data.

K-fold CV Algorithm

1. We randomly divide our data into k groups, or folds, of approximately equal size.
2. The first fold is treated as a validation set, and we use the remaining k-1 folds to fit the data.
3. We iterate this process over all the folds and average the validation errors.
4. For example, in the regression setting, MSE_i is the validation error from the i-th fold, and we get

After running our cv.glm models for ridge and Lasso, we then predict the Y-values using the predict function whereby s is set to lamba.min, basically the optimal lambda value for both ridge and lasso models. They are predicted on the Test dataset's response variable.

Ultimately, Lasso gave a smaller Test MSE than Ridge, as shown on the first page, with only 5 predictors selected for Lasso since Lasso can shrink some of the predictors to zero effectively unlike Ridge, as also explained on the first page of my executive summary.

Forward and Backward Stepwise Selection Discussions

As discussed above, we'll continue to utilise the training and testing datasets derived from train_xy for subsequent models. For forward stepwise and backward stepwise selection methods, we'll use the leap library to access functions like regsubsets(), to enable model selection by forward or backward stepwise in this case.

```
predict.regsubsets <- function(object, newdata, id, ...)
{
form <- as.formula(object$call[[2]])
mat <- model.matrix(form, newdata)
coefi <- coef(object, id = id)
xvars <- names(coefi)
mat[, xvars] %*% coefi
}
```

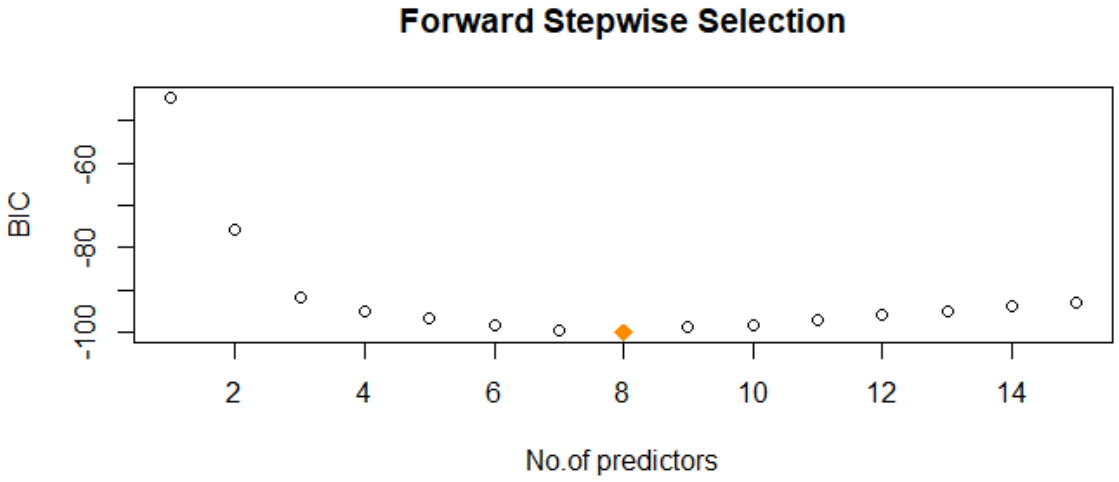
We'll also formulate a predict.regsubsets function to enable prediction of our selection models.

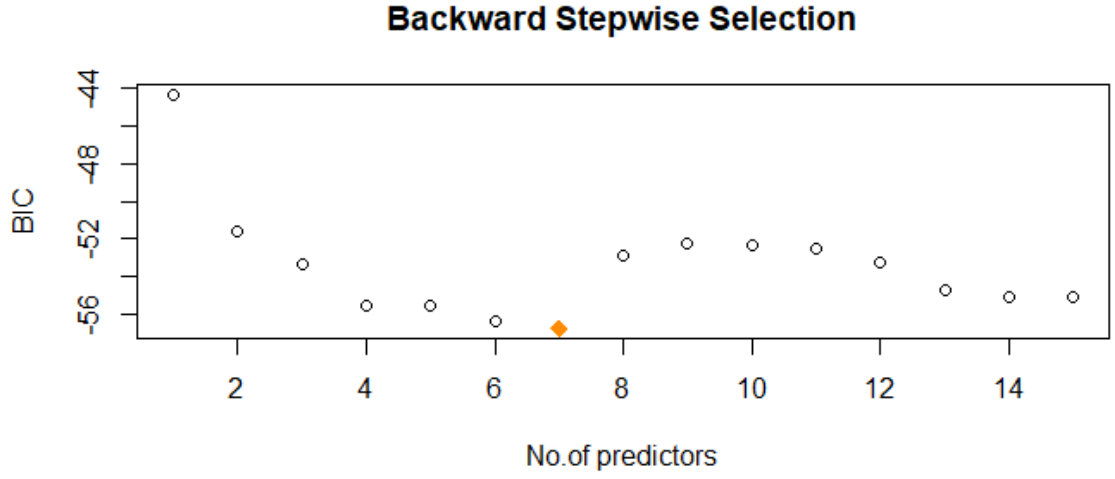
```
regfit.fwd.pred = predict(regfit.fwd,test,k)
fwd_step_mse = mean((regfit.fwd.pred-test$Y)^2) #29.91227
```

Forward stepwise resulted in a smaller test MSE of 29.91227 as compared to Backward stepwise selection method which resulted in a higher 38.85134 test MSE.

```
#backward stepwise
regfit.bwd = regsubsets(Y~.,train,method="backward",nvmax = 15)
k <- which.min(summary(regfit.bwd)$bic)
regfit.bwd.pred = predict(regfit.bwd,test,k)
bwd_step_mse = mean((regfit.bwd.pred-test$Y)^2) #38.85134
```

Stepwise Selection Plots Diagnosis and Findings:





Interestingly, Forward Stepwise Selection gave a smaller test MSE than Backward Stepwise Selection. So I diagnose my findings with the BIC vs. Number of Predictors selected plot for both of the stepwise selection methods.

Forward Stepwise Selection selected 8 variables as opposed to Backward Stepwise Selection method which selected only 7. Furthermore, the BIC when k = 8 predictors for forward stepwise selection is much lower than that of the backward stepwise selection method.

The BIC is also much lower for forward stepwise as compared with backward stepwise as shown in the above 2 plots. My rationale in using BIC as the penalty criteria is because of the high number of predictors and possible overfitting that can be mitigated with BIC. BIC also has a larger penalty term than AIC for sample observations greater than 7, thus I felt with 99 predictor variables BIC would be the better option to account for the curse of dimensionality seen in our training and test datasets.

Limitations

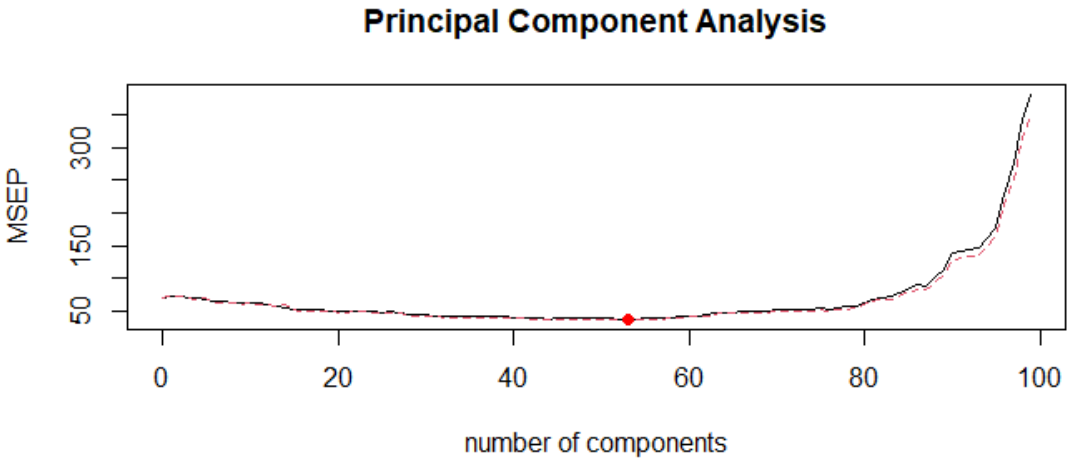
However, I acknowledge that a lower BIC might not totally imply one model is superior over the other since it revolves around approximations.

Unlike best susbset selection method, both forward and backward also does not consider all possible combination of potential predictors, thus, it is not guaranteed to select the best possible combination of predictors for both of my stepwise selection methods, that may have resulted in Lasso still having the best and lowest MSE among all the other methods.

Whilst I tried to use the best subset selection method, it was computationally too expensive as I realised it could not be loaded and there's a long waiting time. Out of practicality reasons which I believe in the real world, it will be rational and pragmatic not to use the best subset selection method, because take our case of this project, with 99 predictors, we need to find the best among the 2^{99} predictor combinations, which is computationally expensive and inefficient. In the outside world, there may be even more predictors and observations.

Another alternative method that I considered but did not include:

Brief summary of Principal Component Analysis



Method and Findings

Principal Component Analysis (PCA) is a method that resolves and transforms high dimensions data into lower dimensions in the form of components whilst maintaining as much crucial information as possible by taking into account the variance of each attribute and converting the observations of correlated features into a set of linearly uncorrelated features by leveraging orthogonal transformation. From the plot above, we see that the mean squared error of prediction (MSEP) is the lowest when 53 components are predicted, with a test MSE of 39.80535, which is not as superior as the Lasso or Forward Stepwise Selection methods.

Furthermore, the low interpretability of principal components should be considered since that makes it hard for us to tell which are the key features in the dataset after computing principal components since principal components comprise the linear combinations of features. There's furthermore loss of information, which is integral in PCA, thus resulting in inevitable discrepancies.

Conclusively, I decided not to consider this method and its findings in my executive summary.

-End of Report-