High Dimension Data Analysis Project

Numerai Competition Dataset

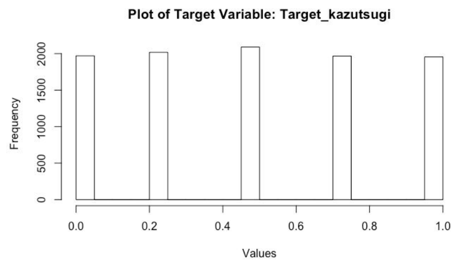
Nusrath Jahan, Vladimir Antasiuk, Luis Caicedo Torres

Prepared for Dr. Wensong Wu

In Fulfillment of STA 6636 - High Dimension Data Analysis

Introduction

Pursuant to comparing dimension reduction algorithms for the analysis of extensively large datasets, a large dataset like that of Numerai competition data proved ideal. The dataset provides encoded financial data about some particular company or portfolio with the goal of predicting a target variable. The aim of the project is to find the model that gives the best prediction accuracy of the target variable by comparing the test MSE of various models. In particular, stepwise subset selection, principal component analysis, and ridge regression will be implemented to varying degrees on this dataset. For those models that performed feature selection or dimension reduction, another aim is to understand the main predictors chosen to explain the data. Additionally, a categorical consideration of the target variable was considered and the results of a multinomial logistic regression are shown. The brief description and discussion of the models used in the project is given in the Appendix G.



Dataset

Here, this dataset provides over 501 thousand observations over 310 predictors or features as they are labeled in the dataset, with the aim of predicting a target variable: target_kazutsugi. Here, the features take encoded names such as feature_wisdom1 with no specific relation to each other. All of the features and target_kazutsugi take on values of 0,0.25,0.50,0.75, or 1, as shown to the right. Note that these are ordinal values and not categorical. The data is all scaled and cleaned so no preprocessing techniques were required or used. In order to be able to run the algorithms locally, a subset of 10,000 observations was taken, the results of which are discussed now.

Methods and Results

Mean Square Error

The mean squared error, or MSE, is calculated as the average of the squared forecast error values. The least squares regression minimizes MSE. Finance data, like Numerai data, tends to be modeled by Brownian stochastic processes which assume a bounded square variation of sample data. Therefore, we can be assured of the scalability of these models as the dataset becomes larger. Therefore, the MSE is a natural error parameter to use for this type of data.

Comparison between Training Error and Test Error

Methods	Training Error	Test Error

Least Square Regression	0.1179	0.1284
Subset Selection Methods:		
Forward Stepwise Selection	0.1179	0.1284
Backward Stepwise Selection	0.1282	0.1285
LASSO Regression (s=0.1)	0.1250	0.1243
Dimension Reduction Methods:		
Principal Component Regression (n=8)	0.1228	0.1237
Ridge Regression (lambda=5000)	0.1204	0.1247

Least Squares Regression

For the full linear model, and subsequently as well, the model was fit to 80% percent of the data, noting training MSE, and then the remaining 20 percent was used for evaluation of test MSE. Cross-validation was performed on the training set for the linear model in case of any interesting result (A visualization is provided in Appendix B).

The least squares regression model exhibited a p-value of 0.2673 with an R-squared of 0.0406 which means it does not do a very good job of estimating the training data. Note that the training error is the lowest and is our estimate of the true prediction error. However, also note that the test error is higher with this model. A probable explanation is when too many predictors are used with the least square method, the model begins finding ways to fit itself not only the underlying training set but to the noise in the training set as well which eventually will lead to the bad prediction of results. Moreover, the least square method is not also a variable selection method which may lead to the use of another model with the aim of reducing the number of predictors. [1] The residuals of this linear model can be visualized in Appendix D, further confirming that this model is not a good fit for this data.

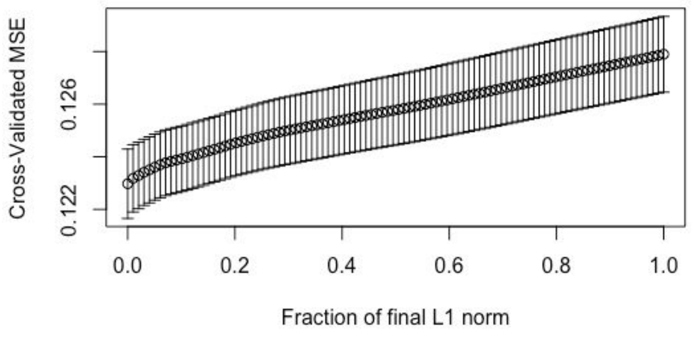
Stepwise Subset Selection

For the sake of subset selection, the forward and backward selection was used in order to try to come up with the best subset using these algorithms according to AIC criteria. The benefits here were of subset selection without evaluating every single possible subset. Test MSE is used as the comparison criterion when relevant.

After performing stepwise subset selection in both directions, the backward selection model proved to be more effective at subset selection than the forward selection algorithm. In fact, the forward selection did not perform subset selection at all making it virtually identical to the least squares regression, hence the identical errors.

The Backwards stepwise algorithm chose a subset of 76 predictors on which to build the model. For the sake of extremely high computation cost, the number of iterations that could be handled locally was capped at 10.

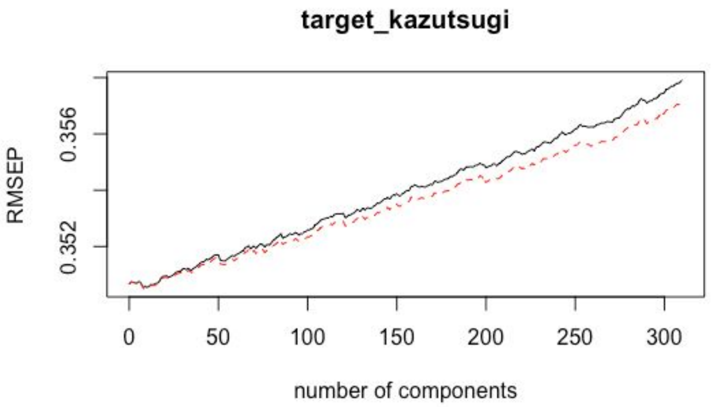
Lasso



Lasso regression is used for feature selection as an alternative to stepwise selection. Here, cross-validation is used to choose the smallest tuning parameter s that will largely explain the data. The test MSE is used to compare to other models.

The CV-MSE was used to attempt to find the optimal tuning parameter s. Here, we see that the algorithm is choosing the optimal parameter as s=0. In turn, all of the coefficients would be made zero and an intercept model would ensue. To evaluate the lasso performance, the next best model of s=0.1 was used for comparison against the least squares regression with a mean cross-validated MSE of 0.124. This model performed subset selection by choosing 102 out of the 310 predictors and provided a test error that is lower than the least squares model, thereby improving prediction accuracy. The MSE for each of the 5-fold cross validation is shown in Appendix E.

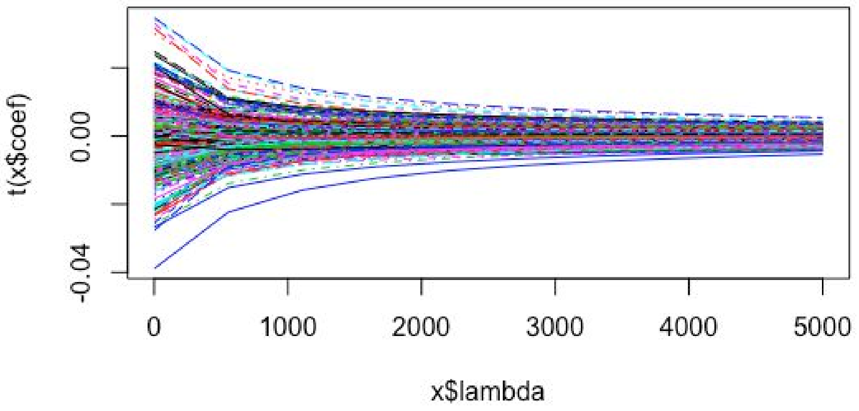
PCA/PCR



Principal component analysis and further regression are used as the main tool for dimensionality reduction. In order to test for the validity of using PCR on the data, Bartlett's sphericity test was performed on the data using the *REdaS* (see Appendix A) package for R. After determining the data is heteroskedastic, a reasonable number of principal components using CV criterion were chosen and the model predicted upon those components.

Bartlett's Sphericity Test gave a p-value of virtually 0 meaning that PCR/PCA is probably appropriate for the data. The validation plot of number of components versus CV shows that an optimal number of principal components is 8 components which explains 38.69% of the variance. In order to explain 80% of variance 60 components are needed. This comparison shows that the 8 components chosen are very significant. Note that the amount of principal components is very low compared to the number of predictors, making this model the best at dimension reduction. The testing error is also the lowest with this model compared to the other models, improving not only model simplicity but also accuracy. The top 10 predictors in the top 8 PCs chosen are printed in Appendix C.

Ridge Regression



In order to study the benefits of coefficient shrinkage on prediction accuracy, ridge regression was used for its coefficient punishing feature. Optimal tuning parameter lambda is studied along with its effects on prediction error.

Cross-validation used as a criterion for the choosing of the optimal tuning parameter lambda always chooses the highest possible lambda. Therefore, the model tends to want to shrink all of the coefficients towards 0. For the sake of not completely shrinking every coefficient to 0, lambda of 5000 was chosen and evaluated against the other models. We note a very low testing error comparable to that of PCR and Lasso as well as improved prediction performance when compared with the least squares regression.In this case, we have the problem with underfitting: the model is maximally underfit. Normally, we don't want to choose big lambda values because the coefficients will become very small and therefore they might not be accurately reflecting what's going on.

Model Selection and Concluding Statements

- Principal Component Regression gives the most accurate models based on Test MSE (0.1237)

- In case of dimensionality reduction, the PCR model is lighter with only 8 components and therefore favorable
- Ultimately, it looks like PCR is the most favorable model.
- This data is highly heteroskedastic making it perfect for the PCR model.

Multinomial Logistic Regression

Upon further analysis, the data lends itself well to a classification type problem. The target variable takes on 1 of 5 values that may be separated and treated as different classes. To this end, a multinomial logistic regression model is fit on the data in order to study the effectiveness of treating the data as a classification problem. This model is implemented using the *nnet* library and the results are summarized in the following table:

	1	2	3	4	5
1	399	0	0	0	0
2	0	388	0	0	0
3	0	0	430	0	0
4	0	0	0	389	0
5	0	0	0	0	394

Here, we see that the multinomial logistic regression predicted the target variable perfectly on the test set. In order to ensure that the model was not biased somehow, the model was used to predict the class of 1000 more (randomly selected) points of data and the results were:

	1	2	3	4	5
1	200	0	0	0	0
2	0	192	0	0	0
3	0	0	202	0	0
4	0	0	0	226	0
5	0	0	0	0	180

Therefore, the multinomial logistic regression predicts the data perfectly. For the sake of model selection in the end, this model would be the most accurate one if one were to treat this problem as a classification problem.

Appendix A
Packages Used

[1] Data Analysis and Graphics Data and Functions (DAAG)

Source: https://cran.r-project.org/web/packages/DAAG/index.html
Published: 2020-03-10
Authors: John M. Maindonald and W. John Braun
Maintainer: W. John Braun <john.braun at ubc.ca>

[2] Companion Package to the Book 'R: Einführung durch angewandte Statistik' (REdaS)

Source: https://cran.r-project.org/web/packages/REdaS/index.html
Published: 2015-11-13
Authors: Marco Johannes Maier
Maintainer: Marco Johannes Maier <marco.maier at wu.ac.at>

[3] Partial Least Squares and Principal Component Regression (PLS)

Source: https://cran.r-project.org/web/packages/pls/index.html
Published: 2019-10-01
Authors: Bjørn-Helge Mevik, Ron Wehrens, Kristian Hovde Liland, Paul Hiemstra
Maintainer:Bjørn-Helge Mevik <b-h at mevik.net>

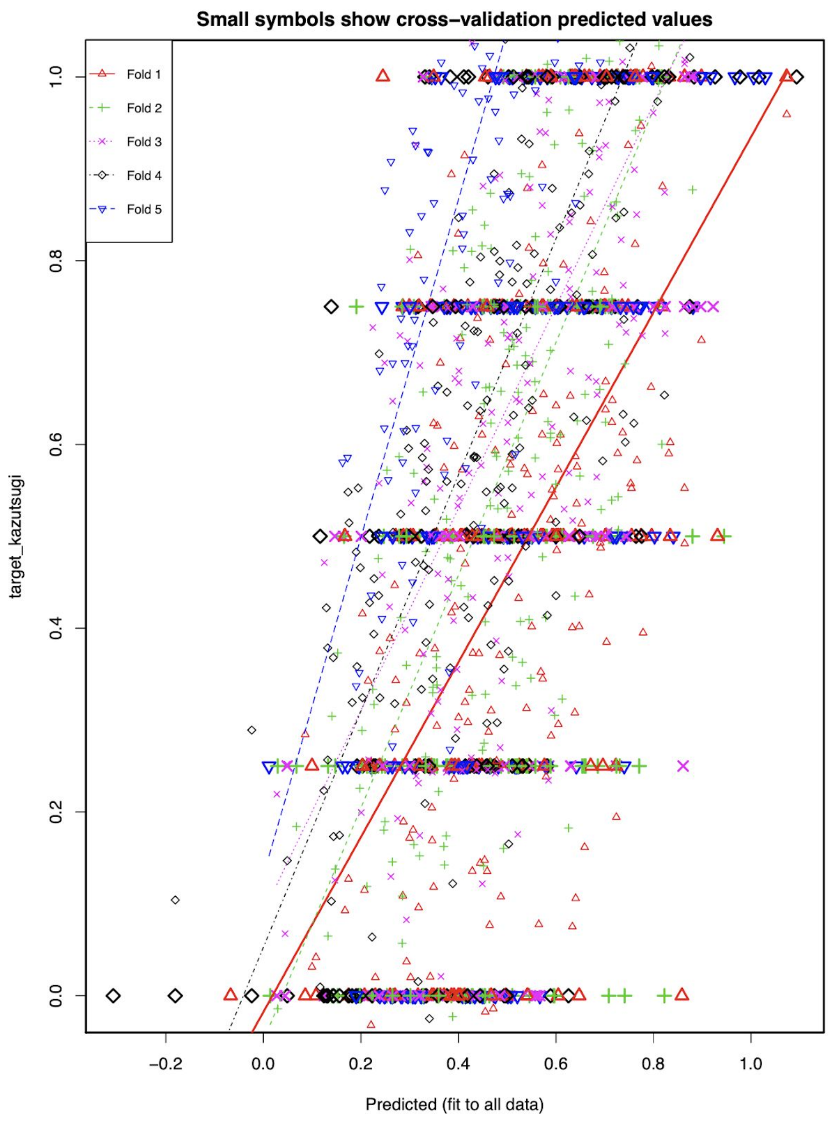
[4] Support Functions and Datasets for Venables and Ripley's MASS (MASS)

Source: https://cran.r-project.org/web/packages/MASS/index.html
Published: 2019-12-20
Authors: Brian Ripley, Bill Venables, Douglas M. Bates, Kurt Hornik, et all
Maintainer:Brian Ripley <ripley at stats.ox.ac.uk>

[5] Feed-Forward Neural Networks and Multinomial Log-Linear Models (nnet)

Source: https://cran.r-project.org/web/packages/nnet/index.html
Published: 2020-02-25
Authors: Brian Ripley, William Venables
Maintainer:Brian Ripley <ripley at stats.ox.ac.uk>

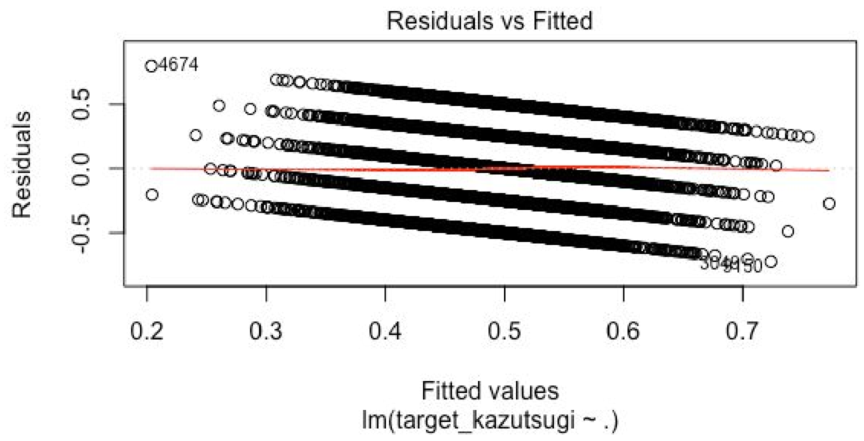
Appendix B

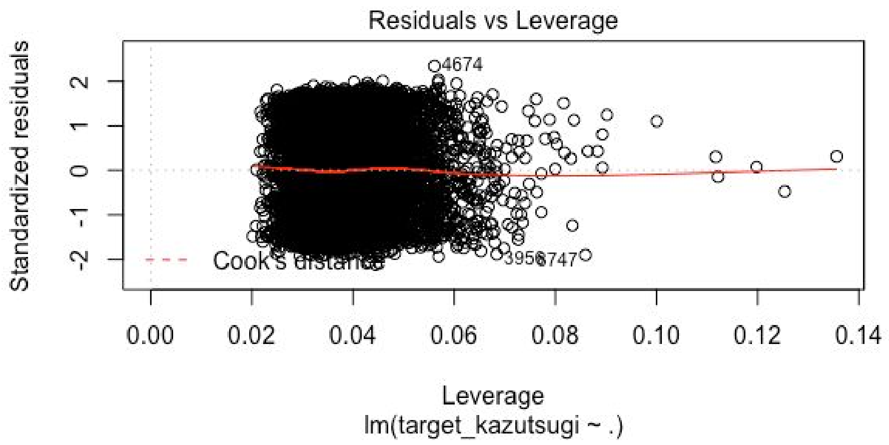

Small symbols show cross-validation predicted values
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
target_kazutsugi
Predicted (fit to all data)
0.0
0.2
0.4
0.6
0.8
1.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

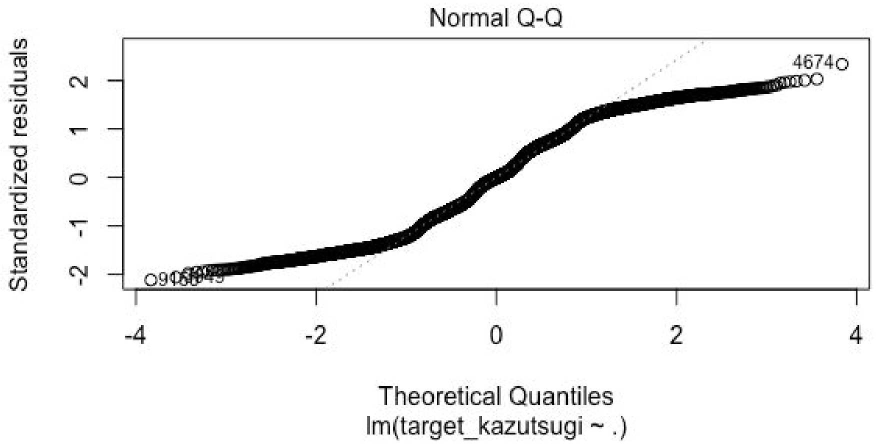
Appendix C
Top 10 Predictors Chosen by 8 Principal Components

TOP10	PC1	PC2	PC3	PC4
1	feature_charisma31	feature_wisdom46	feature_wisdom43	feature_constitution106
2	feature_strength23	feature_wisdom12	feature_wisdom33	feature_constitution75
3	feature_charisma51	feature_wisdom2	feature_wisdom3	feature_constitution25
4	feature_strength16	feature_wisdom7	feature_wisdom24	feature_constitution22
5	feature_intelligence10	feature_wisdom28	feature_constitution105	feature_constitution32
6	feature_constitution35	feature_wisdom15	feature_constitution49	feature_constitution114
7	feature_intelligence9	feature_wisdom21	feature_constitution35	feature_constitution38
8	feature_strength2	feature_wisdom27	feature_constitution4	feature_constitution11
9	feature_dexterity2	feature_wisdom38	feature_constitution50	feature_dexterity11
10	feature_dexterity10	feature_wisdom39	feature_constitution12	feature_constitution77
TOP10	PC5	PC6	PC7	PC8
1	feature_constitution65	feature_constitution85	feature_constitution59	feature_constitution102
2	feature_constitution63	feature_constitution84	feature_constitution81	feature_constitution110
3	feature_constitution2	feature_constitution80	feature_constitution51	feature_constitution91
4	feature_constitution73	feature_constitution62	feature_constitution14	feature_constitution56
5	feature_constitution24	feature_constitution43	feature_constitution30	feature_constitution9
6	feature_constitution82	feature_constitution26	feature_constitution42	feature_constitution33
7	feature_wisdom14	feature_constitution1	feature_constitution61	feature_constitution60
8	feature_wisdom1	feature_constitution20	feature_constitution96	feature_constitution87
9	feature_wisdom31	feature_constitution90	feature_constitution62	feature_constitution112
10	feature_wisdom38	feature_constitution88	feature_constitution20	feature_constitution46

Appendix D
Visualization of Residuals







Appendix E

Cross Validation Results for LASSO (for a 10,000 subset of data)

S	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Fold 1	0.123	0.124	0.125	0.125	0.125	0.126	0.126	0.126	0.127	0.127	0.128
Fold 2	0.124	0.125	0.126	0.126	0.127	0.128	0.129	0.129	0.130	0.131	0.131
Fold 3	0.121	0.122	0.123	0.123	0.124	0.124	0.125	0.125	0.126	0.126	0.127
Fold 4	0.123	0.125	0.125	0.126	0.126	0.127	0.128	0.128	0.129	0.129	0.130
Fold 5	0.124	0.125	0.126	0.126	0.127	0.127	0.128	0.129	0.129	0.130	0.131
Mean	0.123	0.124	0.125	0.125	0.126	0.126	0.127	0.127	0.128	0.129	0.129
SD	0.001	0.001	0.001	0.001	0.001	0.002	0.002	0.002	0.002	0.002	0.002

Appendix F
Code in R

```
setwd("/Users/Lsrex15/Desktop/Project Things")
#### import data/data split ####
data = read.csv("10000sample.csv")
data = data[,5:315] #getting rid of qualitative data

#Visualization of the Target variable
dim(data)
table(data$target_kazutsugi) #about the same number of response for each
hist(data$target_kazutsugi,main = "Plot of Target Variable:
Target_kazutsugi", xlab = "Values")

#next we decided that to split data in training and test set we will use
proportion 80/20
#we chose set of 1000 observations for the simplicity of calculations
set.seed(123)
N=sample(nrow(data),nrow(data)*0.8)
train_set=data[N,]#N -rows
test_set=data[-N,]# Not N rows

#### Least Squares Regression Model ####

#We ran CV for least squares regression for educational purposes
#install.packages("DAAG")
library(DAAG) #We use the cv.lm function of the DAAG library to run the k-
fold validation (reference added in final report)
linear_model=cv.lm(data=train_set,form.lm=formula(target_kazutsugi~.),m=5)
attributes(linear_model)
#as we se in the result of output MSE for CV is 0.275
# Also in the output we can see that only a few variables are significant.

# Now, we run a least squares regression on the simple train split

leastsquares_train=lm(target_kazutsugi~.,data=train_set)
leastsquares_train_predict=predict(leastsquares_train)
leastsquares_predict=predict(leastsquares_train,newdata=test_set)
summary(leastsquares_train)
plot(leastsquares_train)
MSE_Train_lsr=mean((train_set$target_kazutsugi-
leastsquares_train_predict)^2)
MSE_Test_lsr=mean((test_set$target_kazutsugi-leastsquares_predict)^2)
anovatable = anova(leastsquares_train)
bar <- subset(anovatable, `Pr(>F)`<0.025) #Stores variables that are most
statistically significant with very low threshhold.

#### Forward Stepwise Selection ####
library(MASS)

#Suppress all the step outputs
sink("/dev/null")
forward_train = stepAIC(leastsquares_train,direction ='forward')
sink()
summary(forward_train)
#F statistic - " 1.05 on 310" means it chose all 310 variables
```

Appendix F (Continued)

```
#No subset selection was performed by the Forward Stepwise Regression
#Therefore, the results for this model will be the same as the least
squares regression

#### Backward Stepwise Regression ####
library(MASS)

#Suppress all the step outputs - This computation was particularly
expensive - ran for over 5 minutes
#sink("/dev/null")
backward_train = stepAIC(leastsquares_train,direction ='backward',steps =
10)
#sink()
summary(backward_train)
pred_back = predict(backward_train, newdata = train_set)
MSE_train_back = mean((pred_back - data$target_kazutsugi)^2)
backward_train_predict= predict(backward_train,newdata =test_set)
MSE_Test_Backward = mean((backward_train_predict -
data$target_kazutsugi)^2) #output is 0.0985822




#### Lasso Regression ####
#install.packages("lars")
library(lars)
lasso.tr=lars(x=as.matrix(train_set[,
1:310]),y=train_set$target_kazutsugi,type="lasso") # note this function
takes x and y separately, unlike other functions that take formula y~x
#plot(lasso.tr, plottype="coefficients") #beware the computation expense
coef(lasso.tr)[109,coef(lasso.tr)[109,]!=0] # Print of the first ten
nonzero predictors

plot(lasso.tr)

# select tuning parameter s by CV, the lower the better but one standard
deviation rule applies
cvlasso.tr=cv.lars(x=as.matrix(train_set[,
1:310]),y=train_set$target_kazutsugi,type="lasso")
bestfrac = cvlasso.tr$index[which.min(cvlasso.tr$cv)]
bestfrac


##### Now, for the CV-Lasso
x=as.matrix(train_set[,1:310])
y=as.matrix(train_set$target_kazutsugi)
K=5
n=nrow(x)
sam=sample(1:n,n) #create a random list of subject indices
S=list(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1)
for(i in 1:K)
{
  ind = sam[round((i-1)*n/K + 1):round(i*n/K)] # partition the random list
of indices into K equal parts and pick the ith part to be validation
```

Appendix F (Continued)

```
  Xin = x[-ind,]; Yin = y[-ind]
  Xout = x[ind,]; Yout = y[ind]
  for(s in S)
  {
    lasso.in=lars(x=as.matrix(Xin),y=Yin,type="lasso")
    pred.out=predict(lasso.in,newx=Xout,type="fit",mode="fraction",s=s)$fit
    err.te=mean((Yout-pred.out)^2)
    print(i)
    print(s)
    print(err.te)
  }
}

las.coef = predict(lasso.tr, type="coefficients", mode="fraction", s=0)
las.coef
#well actually it turns out that the CV tells us that the best s is 0
#However in this case all the coefficients are shrinked to 0
#In Our case it actually turns out that The less parameter s is the smaller
is MSE.
#So after 0 the best parameter is 0.1 with mean CV MSE = 0,14548894.
# Appying one standart deviation rule we do obtain new MSE 0,158621133 -
wich is new trashhold.
#starting from the smallest s=0.1 we ca see it is our trashhold, so optimal
S=0.1

las.coef = predict(lasso.tr, type="coefficients", mode="fraction", s=0.1)
las.coef
sum(las.coef$coefficients!=0)
#102 variables sellected
pred.te=predict(lasso.tr,newx=test_set[,
1:310],type="fit",mode="fraction",s=0.1)$fit
MSE_Test_Lasso=mean((test_set$target_kazutsugi-pred.te)^2)


#### Principal Component Regression ####

#Implementing Bartlett's Sphericity Test
#install.packages("REdaS")
library(REdaS)
B = bart_spher(data[,1:310])
B$p.value #yield 0 which ensures heteroskedacity in data

#implementing PCR
library(pls)
pcr.tr = pcr(target_kazutsugi~.,data=train_set, validation="CV")
summary(pcr.tr)
#well actually i did run it several times and it seems like intercept is
the smallest
#after the intercept 6 components are the smallest
validationplot(pcr.tr)
pred.tr = predict(pcr.tr,data=train_set , ncomp=6)
MSE_TR_PCR=mean((train_set$target_kazutsugi-pred.tr)^2)
pred.test=predict(pcr.tr,newdata=test_set, ncomp=6)
MSE_TEST_PCR=mean((test_set$target_kazutsugi-pred.test)^2)
```

```
#60 comps for 80 percent variance explained
pred.tr.60 = predict(pcr.tr,data=train_set , ncomp=60)
MSE_TR_PCR.60=mean((train_set$target_kazutsugi-pred.tr.60)^2)
pred.test.60=predict(pcr.tr,newdata=test_set, ncomp=60)
MSE_TEST_PCR.60=mean((test_set$target_kazutsugi-pred.test)^2)


#### Ridge Regression ####
#install.packages("MASS")
library(MASS)
ridreg.tr=lm.ridge(target_kazutsugi~.,
data=train_set,lambda=seq(0,5000,len=10))
ridreg.tr$coef
plot(ridreg.tr)
select(ridreg.tr)
# i run ridreg.tr=lm.ridge(target_kazutsugi~.,
data=train_set,lambda=seq(0,5000,len=10)) for different values
# of len for 50, 1000, and 5000 it tends to choose the highest lamda
possible
#prediction of ridge for lambda=500
ridreg.tr=lm.ridge(target_kazutsugi~., data=train_set,lambda=5000)
X.tr=as.matrix(cbind(rep(1,nrow(train_set)),train_set[,1:310]))#original X
with vector of 1's for intercept on training data
pred.tr=X.tr%*%coef(ridreg.tr)
MSE_Train_Ridge=mean((train_set$target_kazutsugi-pred.tr)^2)
X.te=as.matrix(cbind(rep(1,nrow(test_set)),test_set[,1:310])) #original X
with vector of 1's for intercept on test data
pred.te=X.te%*%coef(ridreg.tr)
MSE_Test_Ridge=mean((test_set$target_kazutsugi-pred.te)^2) #test error


#### Logistic Regression ####
install.packages("nnet")
library(nnet)

g = train_set$target_kazutsugi*4 + 1

# Create Indicator Matrix Y #
Y_data = matrix(0,nrow=dim(train_set)[1],ncol =5)
for (i in 1:5){
  Y_data[g == i , i ]=1
}

newset = cbind(train_set[,1:310],Y_data)

multinomModel <- multinom(Y_data~ ., data=newset,maxit=1000,MaxNWts=1585) #
multinom Model
summary (multinomModel) # model summary

g_t = test_set$target_kazutsugi*4 + 1

# Create Indicator Matrix Y #
Y_data_t = matrix(0,nrow=dim(test_set)[1],ncol =5)
```

```
for (i in 1:5){
  Y_data_t[g_t == i , i ]=1
}

newset_t = cbind(test_set[,1:310],Y_data_t)

predicted_scores <- predict (multinomModel, newset_t, "probs") # predict on
new data
predicted_scores

predicted_class <- predict (multinomModel, newset_t)
predicted_class

table(predicted_class, g_t)

mean(as.character(predicted_class) != as.character(g_t))

oldset = read.csv("1000sample.csv")
oldset = oldset[,5:315]
g_0 = oldset$target_kazutsugi*4 + 1

# Create Indicator Matrix Y #
Y_data_0 = matrix(0,nrow=dim(oldset)[1],ncol =5)
for (i in 1:5){
  Y_data_0[g_0 == i , i ]=1
}

oldset_t = cbind(oldset[,1:310],Y_data_0)

predicted_class <- predict (multinomModel, oldset_t)
predicted_class

table(predicted_class, g_0)

mean(as.character(predicted_class) != as.character(g_0))

xc = train_set[,1:310]
sv = svd(xc)
U = sv$u # normalized scores
U[,1] # PC1 normalized scores
V = sv$v # pc directions
pc1 = V[,1] #PC1 direction
pc2 = V[,2]
plot = (pc1)
D = sv$d # d_j's
U%*%diag(D) # unnormalized score
Z=xc%*%V # unnormalized score, same as U*D
Z[,1] #PC1 unnormalized scores
D^2/nrow(xc)

ord = order(abs(V[,5]),decreasing=TRUE)
x = as.matrix(xc[,ord[1:10]])
colnames(x)
```

Appendix G
Description of Models and Rationale for Their Use

Least Squares Regression	One of the most common and popular techniques used for prediction purposes. Often is used as a benchmark to compare with other more sophisticated models Assumes linear relationship between Y and X. May be incorrect, but it is the most we can extract robustly from the data. Usually good approximation.
Stepwise Subset Selection	If many predictors are available we can have the following problems:prediction accuracy (unrelated inputs induce noises), interpretation. A set of selected predictors may yield better prediction accuracy. Goal: more bias but less variance so that the prediction error is reduced. Only the most significant variables are used for the prediction
Lasso Regression	Lasso is a type of linear regression that uses shrinkage. The coefficients are shrunk towards a central point. Some coefficients may be shrunk to exactly zero. Thus, it may work as a variable selection method. A tuning parameter(s) controls the penalty term in lasso regression. It is basically the amount of shrinkage.
Principal Component Regression	Aim is to reduce the number of dimensions by performing a regression on the top principal components RCR helps to save the structure of original data but helps to make a dimensionality reduction. In PCR, features are transformed to a smaller number of linear combinations of features, and regressYon them.
Ridge Regression	May help reduce overfitting to the training set and bias in the full linear model, Similar to Lasso, but shrinks coefficients towards 0, not exactly to 0.
Logistic Regression	Approximate posterior probability by a nonlinear function of x, guaranteed to fall between 0 and 1and to sum up to 1. Attempts to estimate posterior probabilities logistic curves between 0 and 1. Perfect tool for classification, also logistic regression is mostly used as a data analysis and inference tool, where the goal is to understand the role of input variables in explaining the outcome.

References

[1] Hastie, T., Friedman, J., & Tisbshirani, R. (2017). The Elements of statistical learning: data mining, inference, and prediction. New York: Springer.