HW4 Q3

Code

The goal of this problem is to predict the performance decay over time of the Gas Turbine (GT) compressor. The dataset contains 11934 observations. The range of decay of compressor has been sampled with a uniform grid of precision 0.001 so to have a good granularity of representation. For the compressor decay state discretization the kMc coefficient has been investigated in the domain [0.95,1]. Ship speed has been investigated sampling the range of feasible speed from 3 knots to 27 knots with a granularity of representation equal to tree knots. A series of measures (13 features) which indirectly represents of the state of the system subject to performance decay has been acquired and stored in the dataset over the parameter's space. For each record it is provided: - A 13-feature vector containing the GT measures at steady state of the physical asset: Lever position (lp) Ship speed (v) [knots] Gas Turbine (GT) shaft torque (GTT) [kN m] GT rate of revolutions (GTn) [rpm] Gas Generator rate of revolutions (GGn) [rpm] Port Propeller Torque (Tp) [kN] Hight Pressure (HP) Turbine exit temperature (T48) [C] GT Compressor outlet air temperature (T2) [C] HP Turbine exit pressure (P48) [bar] GT Compressor outlet air pressure (P2) [bar] GT exhaust gas pressure (Pexh) [bar] Turbine Injecton Control (TIC) [%] Fuel flow (mf) [kg/s] - GT Compressor decay state coefficient The dataset is provided as "Shiptrain.csv" and "Shiptest.csv". The last column of the datasets correspond to the output we want to predict. We have split the dataset into training (80%) "Shiptrain.csv" and test (20%) "Shiptest.csv" sets. In order to predict the performance decay over time of the GT compressor and turbines, we are going to use the following models: 1. Ridge Regression 2. Lasso Regression 3. Adaptive Lasso Regression 4. Elastic Net Regression For each of the models please do the following:

- Fit the model on the training dataset.
- Report optimal tuning parameters obtained using cross-validation. Note: You must tune the lambda parameter for all models and the alpha parameter for the elastic net regression model.
- Report the coefficients obtained with the optimal parameters.
- Report the Mean Square Prediction Error for the test set. Note that you should standardized the data. Conclusion: Which model will you select to predict the performance decay over time of the GT compressor and turbines? Why?

Hide

```
library(caret)
```

```
Warning message:
In readChar(file, size, TRUE) : truncating string with embedded nuls
```

Hide

```
library(glmnet)
library(Metrics)

set.seed(42)

train <- read.csv('C:/Users/simskel/Downloads/Shiptrain-1.csv', header=FALSE)
test <- read.csv('C:/Users/simskel/Downloads/Shiptest-1.csv', header=FALSE)

x.train <- as.matrix(train[1:13])
y.train <- as.matrix(train[14])

x.test <- as.matrix(test[1:13])
y.test <- as.matrix(test[14])
```

Standardized data

Hide

```
scaler <- preProcess(x.train)

x.train.scaled <- as.matrix(predict(scaler, x.train))
x.test.scaled <- as.matrix(predict(scaler, x.test))
```

Ridge Regression

Hide

```
cv.ridge <- cv.glmnet(x.train.scaled, y.train, alpha=0, standardize=FALSE, intercept=TRUE,  fami
ly='gaussian')
```

Best Lambda

Hide

```
lambda <- cv.ridge$lambda.min
lambda
```

```
[1] 6.622278e-05
```

Train model on best lambda and report coefficients & MSE on test set

Hide

```
ridge <- glmnet(x.train.scaled, y.train, alpha=0, standardize=FALSE, intercept=TRUE, lambda=lamb
da, family='gaussian')

coef(ridge)
```

```
14 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  0.9750687199
V1           0.0076967963
V2           0.0066634405
V3           0.0174111687
V4          -0.0001287048
V5           0.0024110668
V6          -0.0081026917
V7          -0.0038114709
V8          -0.0664173096
V9           0.0279600682
V10          0.0354834603
V11         -0.0053850494
V12         -0.0080991792
V13         -0.0058837913
```

Hide

```
ridge.test.pred <- predict(ridge, x.test.scaled)

mse(ridge.test.pred, y.test)
```

```
[1] 0.0001222675
```

LASSO Regression

Hide

```
cv.lasso = cv.glmnet(x.train.scaled, y.train, family = "gaussian", alpha = 1, intercept = TRUE)
```

Best lambda

Hide

```
lambda = cv.lasso$lambda.min
lambda
```

```
[1] 6.622625e-08
```

Train model on best lambda and report coefficients & MSE on test set

Hide

```
lasso <- glmnet(x.train.scaled, y.train, alpha=1, standardize=FALSE, intercept=TRUE, family="gau
ssian", lambda=lambda)

coef(lasso)
```

```
14 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept)  0.97506872
V1           0.12272143
V2          -0.07448618
V3          -0.10515645
V4          -0.02591671
V5           0.04410520
V6          -0.08046635
V7           0.05323488
V8          -0.28074465
V9           0.28269152
V10          0.07409583
V11         -0.02932109
V12         -0.01386269
V13          0.03478208
```

Hide

```
lasso.test.predict <- predict(lasso, x.test.scaled)

mse(lasso.test.predict, y.test)
```

```
[1] 3.108613e-05
```

Adaptive Lasso Regression

Hide

```
gamma = 2
b.ols = solve(t(x.train.scaled)%*%x.train.scaled)%*%t(x.train.scaled)%*%y.train
cv.ridge = cv.glmnet(x.train.scaled, y.train, family = "gaussian", alpha = 0, intercept = TRUE,
 standardize=FALSE)
l.ridge = cv.ridge$lambda.min
b.ridge = matrix(coef(ridge, s = l.ridge))
w1 = 1/abs(b.ols)^gamma
w2 = 1/abs(b.ridge)^gamma
alasso1.cv = cv.glmnet(x.train.scaled, y.train, family = "gaussian", alpha = 1, intercept = TRUE
, penalty.factor = w1, standardize=FALSE, maxit=200000)
alasso2.cv = cv.glmnet(x.train.scaled, y.train, family = "gaussian", alpha = 1, intercept = TRUE
, penalty.factor = w2, standardize=FALSE, maxit=200000)
```

Best Lambda

Hide

```
lambda1 = alasso1.cv$lambda.min
lambda2 = alasso2.cv$lambda.min
print(paste("Best OLS adaptive lambda: ", lambda1))
```

```
[1] "Best OLS adaptive lambda:  1.29852370104309e-05"
```

Hide

```
print(paste("Best Ridge adaptive lambda: ", lambda2))
```

```
[1] "Best Ridge adaptive lambda:  0.0250229750283985"
```

Train model on best lambda and report coefficients & MSE on test set

Hide

```
alasso1 = glmnet(x.train.scaled, y.train, family = "gaussian", alpha = 1, lambda=lambda1, interc
ept = TRUE, penalty.factor = w1, standardize=FALSE, maxit=200000)
alasso2 = glmnet(x.train.scaled, y.train, family = "gaussian", alpha = 1, lambda=lambda2, interc
ept = TRUE, penalty.factor = w2, standardize=FALSE, maxit=200000)
coef.ols = matrix(coef(alasso1))
coef.ridge = matrix(coef(alasso2))
cbind.data.frame(coef.ols, coef.ridge)
```

coef.ols <dbl>	coef.ridge <dbl>
0.97506872	0.975068720
0.14391033	-0.051582680
-0.08934431	0.000000000
-0.05257588	0.000000000
0.00000000	-0.122507151
0.04701474	0.000000000
-0.09935561	0.000000000
0.04633247	-0.020637297
-0.28823746	-0.012880745
0.20500107	0.177280349

1-10 of 14 rows Previous **1** 2 Next

Hide

```
alasso.ols.predict <- predict(alasso1, x.test.scaled)
alasso.ridge.predict <- predict(alasso2, x.test.scaled)

alasso.ols.mse <- mse(alasso.ols.predict, y.test)
alasso.ridge.mse <- mse(alasso.ridge.predict, y.test)

cbind.data.frame(alasso.ols.mse, alasso.ridge.mse)
```

alasso.ols.mse <dbl>	alasso.ridge.mse <dbl>
3.06907e-05	0.0001044206

1 row

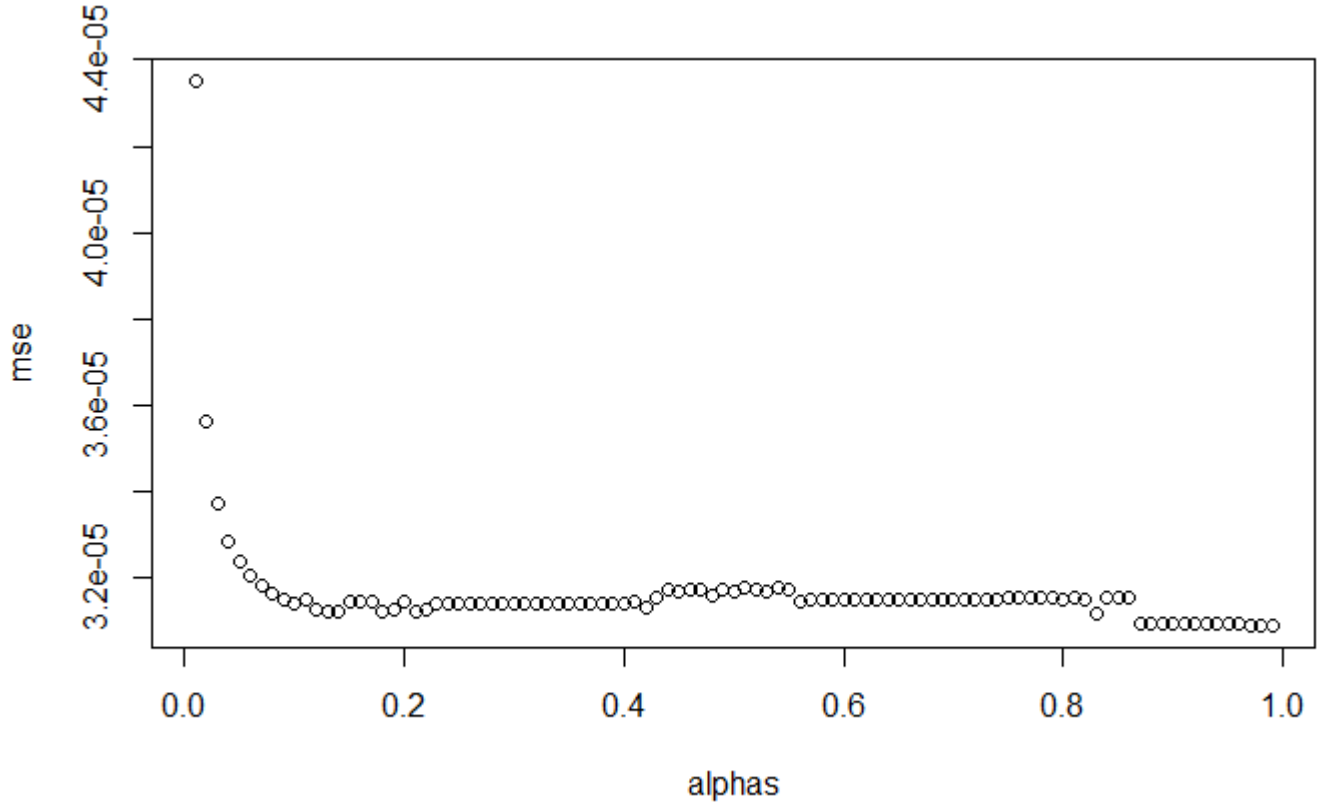
Elastic Net

Hide

```
alphas <- seq(0.01,0.99,by=0.01)

mses <- array(numeric())
as <- array(numeric())
lams <- array(numeric())

for (a in alphas){
  cv.enet <- cv.glmnet(x.train.scaled, y.train, alpha=a, standardize=FALSE, intercept=TRUE,  fam
ily='gaussian')

  lambda <- cv.enet$lambda.min

  enet <- glmnet(x.train.scaled, y.train, alpha=a, standardize=FALSE, intercept=TRUE,  family='g
aussian', lambda=lambda)

  enet.pred <- predict(enet, x.train.scaled)

  enet.mse <- mse(enet.pred, y.train)

  mses <- append(mses, enet.mse)
  as <- append(as, a)
  lams <- append(lams, lambda)
}
```

Hide

```
plot(as, mses, xlab='alphas', ylab = 'mse')
```



Best Lambda & Alpha

Hide

```
idx.lowest.mse <- which.min(mses)

best.alpha <- as[idx.lowest.mse]
best.lambda <- lams[idx.lowest.mse]

cbind.data.frame(best.lambda, best.alpha)
```

best.lambda <dbl>	best.alpha <dbl>
6.68917e-08	0.99

1 row

Train model on best lambda & alpha and report coefficients & MSE on test set

Hide

```
enet <- glmnet(x.train.scaled, y.train, alpha=best.alpha, standardize=FALSE, intercept=TRUE,  fa
mily='gaussian', lambda=best.lambda)

coef(enet)
```

```
14 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept)  0.97506872
V1           0.12228555
V2          -0.07410431
V3          -0.10512256
V4          -0.02595878
V5           0.04409073
V6          -0.08039724
V7           0.05323765
V8          -0.28068974
V9           0.28266696
V10          0.07417509
V11         -0.02932655
V12         -0.01385538
V13          0.03467319
```

Hide

```
enet.test.pred <- predict(enet, x.test.scaled)

mse(enet.test.pred, y.test)
```

```
[1] 3.10895e-05
```

Results

Either Lasso or Elastic Net would be the model of choice. Elastic Net however produced a “best alpha” of 0.99 which is more or less a Lasso model. This is why the proivded nearly simliary MSE on the test set of 3.1e-5