Háskóli Íslands

Gervigreind

Lokaverkefni

Nemendur:
Rakel María Brynjólfsdóttir
Ævar Ingi Jóhannesson

Kennari:
Steinn Guðmundsson

20. apríl 2018

Efnisyfirlit

1 Inngangur

Í þessari skýrslu förum við yfir mismunandi aðferðir til þess að útbúa spálíkan af bótakröfum með gögnum frá Allstate tryggingafélaginu. Mismunandi aðhvarfsgreiningar líkön voru notuð til þess að fá sem bestu spánna. Til þess notuðum við tölfræði forritunarmálið R.
bForvinnsla á gögnunum fór fyrst fram. Þar skoðuðum við gögnin og völdum þær skýribreytur sem virtust skipta máli. Einnig umbreyttum við gögnunum á ákveðin hátt til þess að fá betri spá. Til þess notuðum við línulegt líkan þar sem auðvelt er að sjá hvaða breytur skipta máli og hverjar eru háðar. Það er oft gott að byrja á því að búa til einfalt líkan og bæta svo við skýribreytum til þess að sjá áhrif þeirra á líkanið. Einnig beyttum við Lasso aðhvarfsgreiningu og skoðuðum fyrir hvaða skýribreytur stuðlarnir urðu að 0. Eftir það bjuggum við til líkan úr þeim skýribreytum sem höfðu stuðul frábrugðin 0 og athuguðum hvort það hefði betri eða verri áhrif á spánna.
Að lokum beittum við Gradient Boosting á gögnin sem gaf bestu niðustöðuna. Allar þessar aðferðir eru góðar fyrir aðhvarsgreiningar verkefni og voru þess vegna notaðar. Í gögnunum eru margar strjálar skýribreytur með mörgum flokkum. Það tekur oft langan tíma að keyra svoleiðis líkön og þar af leiðandi er einnig áhugavert að skoða hvort hægt sé að einfalda líkanið en samt fá svipað góðar eða jafnvel betri niðurstöður frá reikniritunum.

2 Aðferðir

2.1 Stutt lýsing á gagnasafni

Gagnasafnið inniheldur þjálfunargögn með 188 318 mælingum á 130 inntaksbreytum. 14 þeirra eru samfelldar og 116 eru strjálar. Fyrstu 72 strjálu breyturnar hafa tvo flokka A og B á meðan hinar hafa fleiri. Við vitum ekki fyrir hvað þær standa þar sem þær eru faldnar. Úttaksbreytan er samfelld og heitir `loss` sem er mæling á bótakröfum fyrir Allstate tryggingarfyrirtækið.

Einnig höfum við prófunargagnasafn með 125 546 mælingum. Þegar við höfum valið lokalíkan spáum við fyrir um þessi gögn.

2.2 Forvinnsla á gögnum

Forvinnsla á gögnunum var ekki mikil. Engin NA gildi voru til staðar og gögnin voru mjög skýr. Við þurftum ekki að gera one-hot-kóðun þar sem það er innbyggt í föll í R. Byrjað var á því að skoða gögnin myndrænt og sást strax að betra væri að nota log-vörpun á loss.

Á mynd 2 í næsta kafla má sjá kassarit fyrir nokkar flokkabreytur. Á þeim sést að sumir flokkar hafa mjög fáar mælingar, t.a.m. hafa flokkarnir `G` og `H` í `cat89` bara eina mælingu. Þegar við skoðuðum gögnin kom í ljós að sumar flokka breytur höfðu mjög marga flokka og margir þeirra höfðu fáar mælingar. Sem dæmi er `cat116` breytan með 326 flokka og 52 þeirra höfðu aðeins 1 mælingu og 16 flokkar höfðu 2 mælingar. Við ákváðum því að fjarlægja þær mælingar sem voru í flokk með færri en 10 mælingum sem voru samtals 729. Þetta kom vel út og gaf okkur meiri nákvæmni en áður.

Einnig hentum við út tveimur samfelldum breytum sem höfðu háa fylgni við aðrar breytur. Gögnin sem við notum í framhaldinu hafa því 12 samfelldar breytur, 116 flokkabreytur og 187 589 mælingar. Skipt var svo gögnunum í þjálfunar- og prófunarsafn með 70/30 skiptingu. Fyrir Boosting notuðum við svo validation-gögn til að stilla hástikana. Þar var þjálfunargögnunum skipt í 80/20.

2.3 Línuleg Aðhvarfsgreining

Línuleg aðhvarfsgreining finnur þá stika θ sem lágmarka kvaðratskekkjuna

$$J(\theta) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

þar sem $\hat{y}_i$ er okkar spágildi og y_i er sanna gildið. Lausnin á þessu verkefni (þegar andhverfan er til) er gefin með

$$\theta = (X^T X)^{-1} X^T y$$

þar sem dálkarnir í X eru gildin á skýribreytunum og y er úttaksvigurinn. Fyrir þetta verkefni notuðum við `lm()` fallið í R sem notar QR-fylkjaliðun til að finna lausnina.

2.4 Lasso Aðhvarfsgreining

Eini munurinn á Lasso og Línulegri aðfhvarfsgreiningu er að kostnaðarfallið fyrir Lasso hefur einnig reglunarþátt:

$$J(\theta) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Hér erum við að refsa fyrir stór gildi á stikununum β_j. Ef $\lambda = 0$ er engin refsing og fáum við það sama og í Línulegri aðhvarfsgreinigu. Eftir því sem λ hækkar minnka β_j stuðarnir og sumir þeirra verða einfaldlega 0 sem hjálpar okkur að sjá hvaða breytur eru mikilvægar.

Við notuðum `glmnet()` fallið í R sem notar Coordinate descent aðferðina til að lágmarka J(θ). Coordinate descent er ekki ólík Aðferð mesta bratta (e. Gradient descent), en notar einungis fallgildið en ekki stigul fallsins eins og Aðferð mesta bratta gerir.

2.5 Gradient Boosting

Boosting er aðferð sem eykur spáhæfni ákvarðanatrjá. Aðferðin byrjar á því að búa til ákvarðana tré fyrir gögnin en meðhöndlar síðan leifarnar úr því líkani sem skýribreytu til að búa til næsta tré. Næsta tré er svo byggt á leifum úr fyrra tréi o.s.frv. Hástikin λ segir til um hversu mikið vægi fyrra tréið hefur á núverandi tré.

Reiknirit fyrir almennt boosting:

1. Setjum $\hat{f}(x) = 0$ og $r_i = y_i$ fyrir öll i

2. Fyrir k = 1,2,...,K gerum við eftirfarandi:

   i. Smíðum tré $\hat{f}^k$ með d skiptingum á þjálfunar gögnin.

   ii. Uppfærum $\hat{f}$ með:
   $$\hat{f}(x) = \hat{f}(x) + \lambda \hat{f}^k(x).$$

   iii. Uppfærum leifarnar með:
   $$r_i = r_i - \lambda \hat{f}^k(x_i)$$

3. Gefum frá okkur líkanið:

$$\hat{f}(x) = \sum_{k=1}^{K} \lambda \hat{f}^k(x)$$

Við notuðum `gbm()` fallið í R sem notar gradient boosting aðferðina. Hún notar Aðferð mesta bratta (e. Gradient descent) til að uppfæra f í hverri ítrun. Með því að lágmarka tap-fallið sem er kvaðrat skekkja í þessu tilfelli.

2.6 Hvernig nákvæmni spálíkana er metin

Nákvæmni spálíkana er metið með MAE eða mean absolute error. Formúlan er eftirfarandi:

$$MAE = \frac{\sum_i^n |\hat{y}_i - y_i|}{n}$$

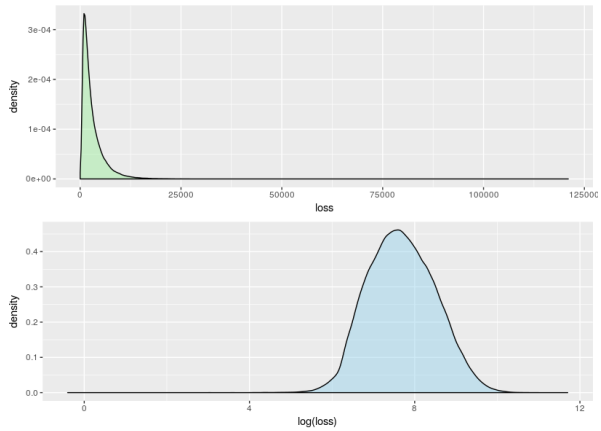
Þar sem n er fjöldi mælinga, $\hat{y}$ er spá gildið og y raunverulega gildið. Þar sem að við höfum log-vörpun á úttaksbreytunni þurfum við að taka exp-vörpun til þess að fá spágildin okkar $\hat{y}$ á rétt form.

2.7 Hvernig gildi á hástikum (e. hyperparameter) er valin

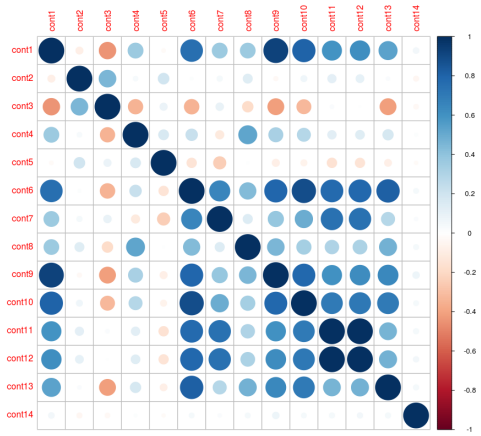
Við notum „Grid search“ á λ, d (dýpt trjáa) og n (fjölda trjáa) fyrir boosting og höldum svo úti staðfestingar (e. validation) gögnum til að prófa. Fyrir Lasso notum við 10-fold-cross-validation til þess að velja gildið á reglunarþættinum λ. Sjá betur í niðurstöðu kafla.

3 Niðurstöður

3.1 Töflur, myndir og lýsingar



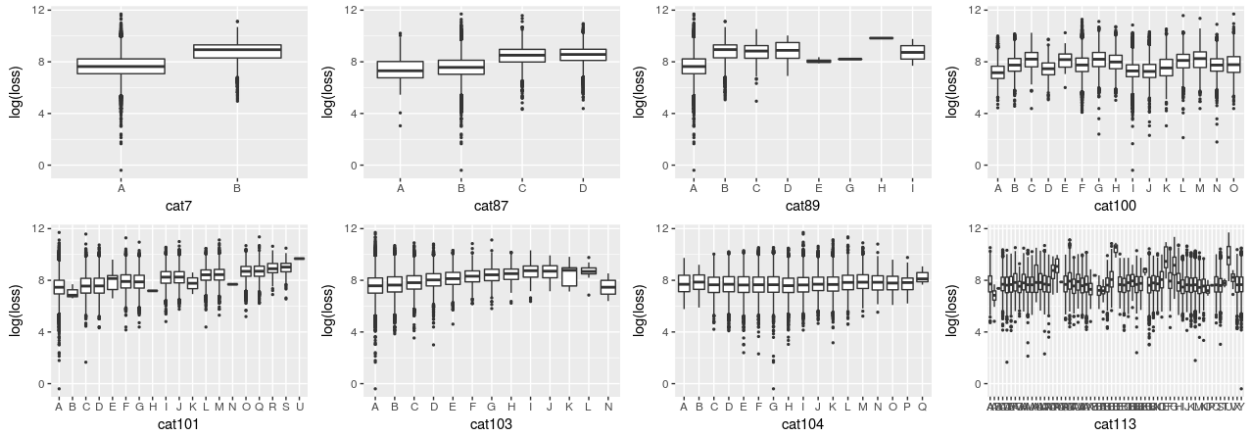
Mynd 1: Dreifing `loss` breytunnar



Mynd 2: Fylgnifylki samfelldu breytanna

Eins og sést á mynd 1 er dreifingin á `loss` breytunni mun líkari normal dreifingu eftir log ummyndun. Við prófuðum að spá fyrir um bæði `loss` og log(`loss`) og fengum meiri nákvæmni með log(`loss`). Við skoðuðum einnig dreifinguna á samfelldu breytunum en það hentaði ekki að beyta log ummyndun á þær.

Fylgnin milli breytanna `cont9` og `cont1` var 0.9299 og á milli `cont11` og `cont12` var hún 0.9944, þannig að við fjarlægðum `cont9` og `cont12`.



Mynd 3: Kassarit af nokkrum flokkabreytum

Á mynd 3 má sjá kassarit af nokkrum flokkabreytum áður en forvinnsla fór fram, fyrir utan að y-ásin er log(`loss`).

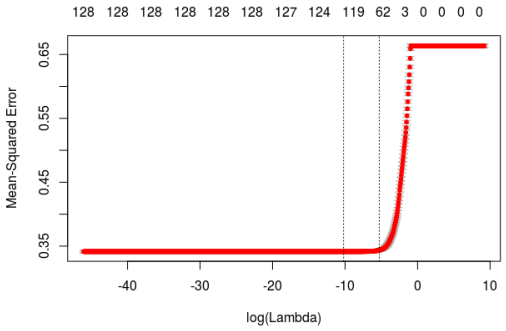
Fyrsta aðferðin sem við prófuðum var Línuleg aðhvarfsgreining, þá kom í ljós að sumar flokkarbreyturnar eru línulega háðar sem veldur því að metorð (e. rank) X fylkisins sé minna en fjöldi dálka. Í okkar tilfelli var $\text{rank}(X) = 740$ og fjöldi dálka 796, þar af leiðandi er X ekki andhverfanlegt. R reynir þó að leysa þetta en sumir stuðlarnir verða NA þannig ekki er hægt að spá fyrir um prófunargögnin. Við tókum því út þær breytur sem fengu NA stuðla sem voru:

cat74	cat81	cat85	cat87	cat89	cat90	cat91	cat92
cat98	cat99	cat100	cat101	cat102	cat103	cat104	cat106
cat107	cat108	cat111	cat113	cat114	cat116		

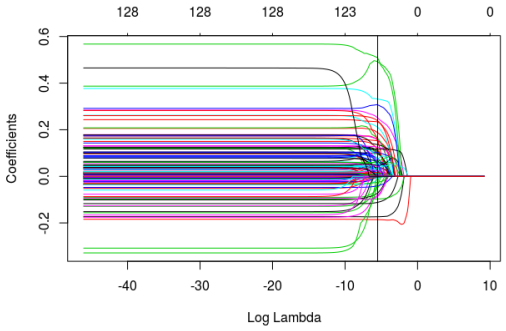
Tafla 1: Breytur fjarlægðar fyrir Línulega aðhvarfsgreiningu

Þá eru 12 samfelldar og 94 flokkabreytur eftir. Línulega líkanið á þessi gögn gaf okkur MAE = 1266.915 fyrir prófunargögnin.

Næst prófuðum við Lasso aðhvarfgreiningu á allt gagnasettið, þ.e.a.s. einnig með breytunum í töflu 1.
Við notuðum fallið `cv.glmnet()` til að framkvæma 10-fold-cross-validation á 1000 misumandi gildi á λ frá 1×10^{-20} uppí 1×10^{4}. Besta gildið var $\lambda = 0.004070142$. Í töflu 2 má sjá þær breytur sem höfðu stuðla frábrugðna 0 í Lasso og á mynd 4 sést hvernig MSE breytist með λ, efsta línan sýnir hversu margar breytur eru í líkaninu fyrir gefið λ. Mynd 5 sýnir svo gildin fyrir stuðlana í líkaninu sem fall af λ. Lóðrétta strikið sýnir svo það gildi á λ sem gaf bestu niðurstöðuna úr cross-validation. Athugið að λ er á log-skala á mynd 5 og mynd 4. Fyrir þetta líkan fengum við MAE = 1275.478 á prófunargögnin okkar.



Mynd 4: Áhrif λ á MSE



Mynd 5: Áhrif λ á stuðlana

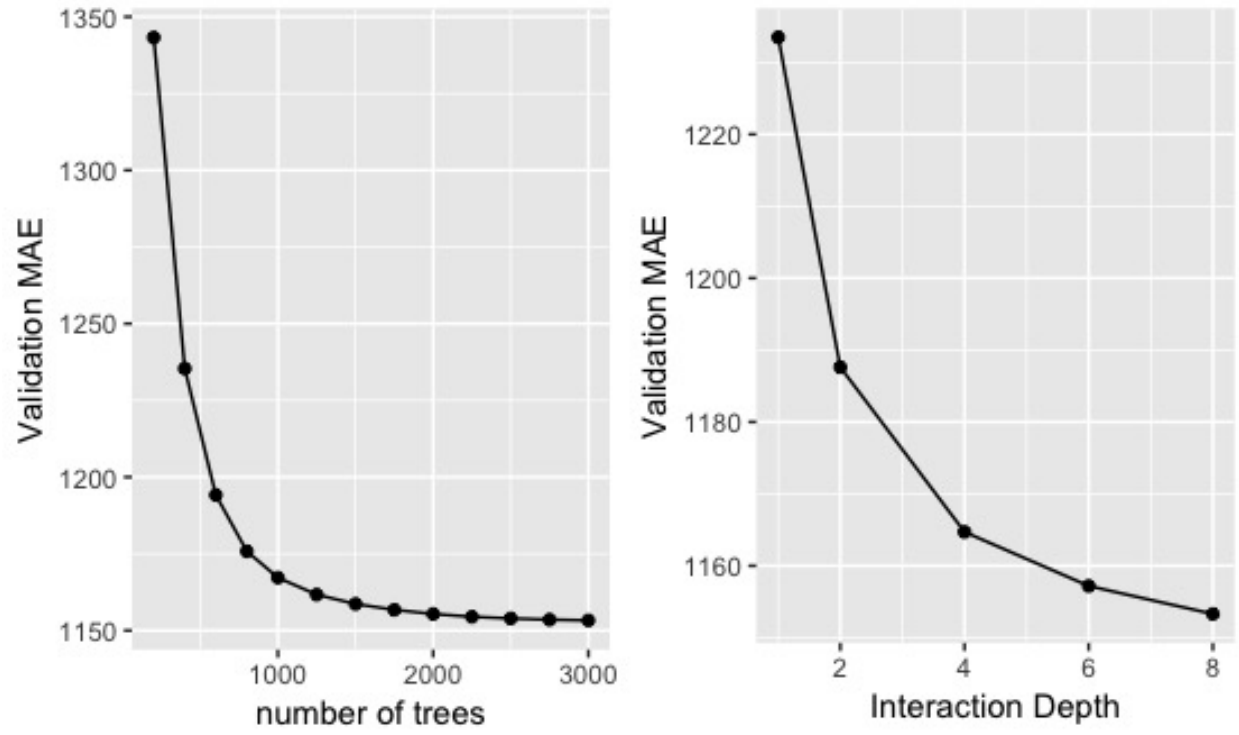
cat1	cat32	cat57	cat93
cat2	cat34	cat59	cat97
cat3	cat35	cat65	cat98
cat4	cat36	cat67	cat99
cat5	cat37	cat71	cat100
cat6	cat38	cat72	cat101
cat7	cat39	cat73	cat102
cat8	cat41	cat75	cat103
cat10	cat42	cat77	cat105
cat11	cat44	cat78	cat111
cat12	cat46	cat79	cat112
cat13	cat47	cat80	cat114
cat19	cat48	cat81	cont1
cat21	cat49	cat82	cont2
cat23	cat51	cat83	cont3
cat24	cat52	cat85	cont8
cat26	cat53	cat87	cont4
cat27	cat54	cat88	cont7
cat29	cat56	cat92	cont14
cat31			

Tafla 2: Mikilvægustu breytur skv. Lasso

Að lokum notuðum við Gradient boosting. Notað var Grid search til að finna bestu samsetningu á hástikana. Hlaupið var í gegnum eftirfarandi gildi:

$\lambda = \{0.001, 0.004, 0.007, 0.01, 0.04, 0.12, 0.2\}$
$n = \{200, 400, 600, 800, 1000, 1250, 1500, 1750, 2000, 2250, 2500, 2750, 3000\}$
$d = \{1, 2, 4, 6, 8\}$.

Besta lausnin var $\lambda = 0.01, n = 3000, d = 8$ með MAE $= 1153.3$ á staðfestingar gögnin.



Mynd 6: Staðfestingar MAE gegn d og n með λ=0.01

Á mynd 6 getum við séð hvernig staðfestingar MAE er sem fall af tveimur hástikum n og d þegar við höldum $\lambda = 0.01$.

Breyta	Relative influance
cat80	34.966263
cat100	7.245774
cat116	7.216493
cat101	5.859022
cat79	5.101299
cat112	4.120626
cat12	3.431625
cat81	3.248094
cat114	2.906671
cont2	2.722017

Tafla 3: þær 10 inntaksbreytur sem skipta mestu máli

Tafla 3 gefur okkur þær 10 inntaksbreytum sem virðast skipta mestu máli fyrir boosting líkanið. Fyrir þetta líkan fengum við MAE = 1154.547 á prófunargögnin okkar.

Aðferð:	MAE
Línulegt	1266.915
Lasso	1275.478
Boosting	1154.547

Tafla 4: Niðurstöður á þjálfunargögn

Svo keyrðum við Boosting líkanið á prófunargagnasafnið sem á að senda inn á Kaggle með öllum þjálfunargögnunum og fengum MAE = 1149.54121.

3.2 Boosting á minna gagnasafn

Áhugavert er að skoða hvort að það myndi hafa einhver áhrif á spáhæfni líkansins að nota færri inntaksbreytur. Við tókum þær breytur sem voru ekki með 0 stuðul í Lasso líkaninu okkar. Þær voru 77 og má sjá þær í töflu 2. Hástikarnir voru fundnir á nákvæmlega sama hátt og fyrir stóra líkanið og fengust sömu niðurstöður. Það er $\lambda = 0.01, n = 3000, d = 8$ nema með MAE = 1152.02 á staðfestingar gögnin sem er lægra heldur en fyrir stóra líkanið.
Að lokum keyrðum við Boosting líkanið á prófunargagnasafnið sem á að senda inn á Kaggle með öllum þjálfunargögnunum og fengum MAE = 1145.317 sem er mun betra en fyrir stóra líkanið.

3.3 Tilraunir sem skiluðu litlu

Við reyndum við aðferðina Huber regression (Robust regression). Það gekk mjög illa þar sem að við náðum aldrei að láta það spá fyrir prófunargögnin okkar. Við náðum að búa til líkan en fengum alltaf villu í R þegar við reyndum að spá fyrir um gögnin okkar. Mögulegar ástæður fyrir því gæti verið að það voru skýribreytur með 0 mælingum í ákveðnum flokkum og þá verða til 0 dálkar í one-hot-encoding fylkinu sem R býr til. Eða þá að fylkið hefur dálka sem eru línulega háðir og á sér því ekki andhverfu. Svipað og gerðist fyrir línulega líkanið okkar þegar við höfðum allar breyturnar með.

4 Samantekt

4.1 Helstu ályktanir

Það sem hægt er að ályktа frá þessari skýrslu er að línuleg aðhvarfsgreiningar líkön eru með verri spáhæfni heldur en Boosting aðferðir fyrir þessi gögn. Línuleg líkön gera ráð fyrir að gögnin fylgi normal-dreifingu og það skemmir fyrir spáhæfni. Hinsvegar er oft þægilegt að byrja á einföldu línulegu líkani til þess að átta sig á gögnunum og hjálpa til við forvinnslu á þeim. Það er áhugavert að sjá að minna líkanið með aðeins 77 skýribreytum hefur betri spáhæfni heldur en líkan með 129 skýribreytum. Oft er því gott að nota aðferðir eins og Lasso eða forward stepwise selection til þess að minnka líkanið og fjarlægja suð. Það getur bætt spáhæfni líkansins og einnig minnkað tíman sem það tekur að stilla líkanið. Það kom í ljós að boosting aðferðir eru mjög tímafrekar, það er að segja að finna réttu gildin á hástikunum. Að finna hástikana á stóra líkaninu tók rúmlega 18 klst í keyrslu. Það tók undir 10 klst að finna hástikana fyrir minna líkanið.

4.2 Næstu skref

Næstu skref gæti verið að keyra í gegnum stærra safn af hyperparametrum með því að hækka t.d d stikan í 10 eða 12 fyrir Boosting. Einnig væri hægt að prófa aðrar aðferðir eins og t.d að keyra fjöllaga tauganet á gögnin og athuga hvort það myndi lækka MAE enn frekar. Einnig er XGboosting (Extreme gradient boosting) aðferð sem er vinsæl núna og hægt væri að athuga hvort hún gæti gert betur heldur en Gradient boosting.

5 Heimildaskrá og lýsing á framlagi

5.1 Heimildaskrá

Hastie, T., James, G., Tibshirani, R. og Witten, D. (2015). *An introduction to statistical learning(6. útgáfa).* doi:10.1007/978-1-4614-7138-7

5.2 Lýsing á framlagi

Framlag hópmeðlima var nokkuð jafnt. Við tókum bæði þátt í forvinnslu gagna og að prófa mismunandi hluti hvað það varðar. Unnið var saman að aðferðum og var Github notað til þess að deila kóða og uppfæra. Ævar sá meira um að keyrslu á kóðanum og að setja upp grid search og Rakel teiknaði upp flestar myndir. Einnig var unnið sameiginlega að skýrslugerðinni inná Overleaf.

6 Viðauki

6.1 Kóði

```
#load packages
library(plyr); library(dplyr);
library(ggplot2); library(grid); library(gridExtra);
library(glmnet); library(e1071); library(knitr)
library(gbm); library(corrplot); library(caret)
library(mlbench); library(gdata)

#load data
data = read.csv("train.csv")
data$id = NULL
data.submit = read.csv("test.csv")
id = data.submit$id

#Check for NA values
dim(na.omit(data)) == dim(data)

#Check distribution of response
density1 <- ggplot(data,aes(loss))+
  geom_density(fill = "palegreen2", alpha = 0.4)

density2 <- ggplot(data,aes(log(loss)))+
  geom_density(fill = "skyblue", alpha = 0.4)

grid.arrange(density1,density2,nrow=2)

skewness(data$loss) #3.794898
skewness(log(data$loss)) #0.09297306
data$loss = log(data$loss)

#plots of continuous variables
p1 = ggplot(data=data, aes(x=cont2,y=loss))+
      geom_point(size=0.4)+ylab("log(loss)")

p2 = ggplot(data=data, aes(x=cont3,y=loss))+
      geom_point(size=0.4)+ylab("log(loss)")

p3 = ggplot(data=data, aes(x=cont13,y=loss))+
      geom_point(size=0.4)+ylab("log(loss)")

p4 = ggplot(data=data, aes(x=cont14,y=loss))+
      geom_point(size=0.4)+ylab("log(loss)")

grid.arrange(p1,p2,p3,p4,nrow = 2)
grid.arrange(p1,p2,p3,p4,nrow = 2)

```

```r
#check correlation
data.cont = data[,117:131]
cormat = cor(data.cont[,-15])
corrplot(cormat, method = "circle")

#drop highly correlated variables
data$cont9 = NULL
data$cont12 = NULL

data.try = data

#Remove those categories with observations under 10
for(i in 1:116){
  var = eval(parse(text = paste("data.try$cat",as.character(i),sep="")))
  data.try = data.try[var %in% names(which(table(var) > 10)), ]
}
data.try = droplevels(data.try)

#Remove variables with NA coefficients in Lin.Reg
cat74 = data.try$cat74; data.try$cat74 = NULL
cat81 = data.try$cat81; data.try$cat81 = NULL
cat85 = data.try$cat85; data.try$cat85 = NULL
cat87 = data.try$cat87; data.try$cat87 = NULL
cat89 = data.try$cat89; data.try$cat89 = NULL
cat90 = data.try$cat90; data.try$cat90 = NULL
cat91 = data.try$cat91; data.try$cat91 = NULL
cat92 = data.try$cat92; data.try$cat92 = NULL
cat98 = data.try$cat98; data.try$cat98 = NULL
cat99 = data.try$cat99; data.try$cat99 = NULL
cat100 = data.try$cat100; data.try$cat100 = NULL
cat101 = data.try$cat101; data.try$cat101 = NULL
cat102 = data.try$cat102; data.try$cat102 = NULL
cat103 = data.try$cat103; data.try$cat103 = NULL
cat104 = data.try$cat104; data.try$cat104 = NULL
cat106 = data.try$cat106; data.try$cat106 = NULL
cat107 = data.try$cat107; data.try$cat107 = NULL
cat108 = data.try$cat108; data.try$cat108 = NULL
cat111 = data.try$cat111; data.try$cat111 = NULL
cat113 = data.try$cat113; data.try$cat113 = NULL
cat114 = data.try$cat114; data.try$cat114 = NULL
cat116 = data.try$cat116; data.try$cat116 = NULL


#Split to train and test.
set.seed(5)
n = dim(data.try)[1]
train = sample(n,floor(n*0.7))
data.train = data.try[train,]
data.test = data.try[-train,]
```

```
95
96  #Fit a linear model and predict.
97  lm.try = lm(loss~.,data.train)
98  pred.try = predict(lm.try,data.test)
99  MAE.try = mean(abs(exp(data.test$loss) - exp(pred.try)))
100 MAE.try
101
102
103 #Lasso
104 set.seed(1000)
105 ydata.train = data.matrix(data.train[,"loss"])
106 Xdata.train = data.matrix(data.train[,!(colnames(data.train) %in% c("loss"))])
107 Xdata.test = data.matrix(data.test[,!(colnames(data.train) %in% c("loss"))])
108
109 #Cross-validation
110 grid = 10^seq(4, -20, length=1000)
111 fit.lasso = cv.glmnet(Xdata.train,
112                       ydata.train,
113                       alpha=1,
114                       lambda=grid,
115                       thresh=1e-12)
116
117 #Get the best lambda and predict
118 best.lambda=fit.lasso$lambda.min
119 pred.lasso = predict(fit.lasso,Xdata.test,s=best.lambda)
120 MAELasso = mean(abs(exp(data.test$loss) - exp(pred.lasso)))
121 MAELasso
122
123 #Boosting
124
125 #Create a validation set for grid search
126 n3 = dim(data.train)[1]
127 val = sample(n3,floor(0.2*n3))
128 data.val = data.train[val,]
129 data.train = data.train[-val,]
130
131 #Grid Search.
132 set.seed(1000)
133 lambd <-  seq(0.001, 0.01, by=0.003)
134 lambd <- c(lambd,seq(0.04,0.2,by=0.08))
135 ntree <- seq(200,800,by=200)
136 ntree <- c(ntree,seq(1000,3000,by=250))
137 depth <- c(1,2,4,6,8)
138 m <- length(lambd)
139 l <- length(ntree)
140 t <- length(depth)
141 testErr <- array(dim=c(m,l,t))
142 for (i in 1:m){
143   for(d in 1:t){
```

```
      boostCol = gbm(loss ~., data = data.train,
                     distribution = "gaussian",
                     n.trees = 3000,
                     shrinkage = lambd[i],
                     interaction.depth = depth[d])
      for(k in 1:l){
      testPred = predict(boostCol,
                         data.val,
                         n.trees = ntree[k])
      testErr[i,k,d] = mean(abs(exp(data.val$loss) - exp(testPred)))
      }
  print(i)
  }
}

which(testErr == min(testErr),arr.ind = T)
bestlam = lambd[4]
bestntree = ntree[13]
bestdepth = depth[5]

#Create plots of hyperparameters.
MAEplotNtree = testErr[4,,5]
MAEplotInt = testErr[4,13,]
boostPlot =ggplot(data.frame(x=ntree,y=MAEplotNtree), aes(x=x, y=y)) +
            xlab("number of trees") +
            ylab("Validation MAE") +
            geom_point()+
            geom_line()
boostPlot

boostPlot2 =ggplot(data.frame(x=depth,y=MAEplotInt), aes(x=x, y=y)) +
            xlab("Interaction Depth") +
            ylab("Validation MAE") +
            geom_point()+
            geom_line()
boostPlot2

grid.arrange(boostPlot,
             boostPlot2,ncol=2)

#Predict for Boosting
set.seed(1000)
boost.fit <- gbm(loss ~ ., data = data.train,
                 distribution = "gaussian",
                 n.trees = bestntree,
                 shrinkage =bestlam,
                 interaction.depth = bestdepth)
boost.pred <- predict(boost.fit,
                      data.test,
```

```
                        n.trees = bestntree)
MAEBoost = mean(abs(exp(data.test$loss) - exp(boost.pred)))
MAEBoost

#Get 10 most important variables from Boosting.
head(summary(boost.fit),10)

#Create a data set from non-zero coefs from Lasso
coefs_temp <- predict(fit.lasso,
                          s = fit.lasso$lambda.1se,
                          type = "coefficients")
coefs_temp2 <- data.frame(name = coefs_temp@Dimnames[[1]][coefs_temp@i + 1],
                              coefficient = coefs_temp@x)
names <- levels(coefs_temp2[,1])
names <- names[2:length(names)]
names <- c(names,"loss")
data.lasso = data.try[,names]
data.lasso.train = data.lasso[train,]
data.lasso.test = data.lasso[-train,]


# Boosting for smaller model.

#Create a validation set.
set.seed(1000)
n3 = dim(data.lasso.train)[1]
val = sample(n3,floor(0.2*n3))
data.lasso.val = data.lasso.train[val,]
data.lasso.train = data.lasso.train[-val,]


#Grid search to find hyperparameters.
set.seed(1000)
lambd <-  seq(0.001, 0.01, by=0.003)
lambd <- c(lambd,seq(0.04,0.2,by=0.08))
ntree <- seq(200,800,by=200)
ntree <- c(ntree,seq(1000,3000,by=250))
depth <- c(1,2,4,6,8)
m <- length(lambd)
l <- length(ntree)
t <- length(depth)
testErr2 <- array(dim=c(m,l,t))
for (i in 1:m){
  for(d in 1:t){
      boostCol = gbm(loss ~., data = data.lasso.train,
                         distribution = "gaussian",
                         n.trees = 3000,
                         shrinkage = lambd[i],
                         interaction.depth = depth[d])
```

```
      for(k in 1:l){
      testPred = predict(boostCol,
                         data.lasso.val,
                         n.trees = ntree[k])
      testErr2[i,k,d] = mean(abs(exp(data.lasso.val$loss) - exp(testPred)))
      }
  print(i)
  }
}
which(testErr2 == min(testErr2),arr.ind = T)
#Same results as before.


set.seed(1000)
boost.fit.lasso <- gbm(loss ~ ., data = data.lasso,
                  distribution = "gaussian",
                  n.trees = bestntree,
                  shrinkage =bestlam,
                  interaction.depth = bestdepth)
boost.pred.small <- predict(boost.fit.lasso,
                      data.submit,
                      n.trees = bestntree)
MAEBoost.small = mean(abs(exp(data.lasso.test$loss) - exp(boost.pred.small)))
MAEBoost.small


#Submit to Kaggle

#Use all the training data set for full model
boost.fit <- gbm(loss ~ ., data = data.train,
                  distribution = "gaussian",
                  n.trees = bestntree,
                  shrinkage =bestlam,
                  interaction.depth = bestdepth)

pred.submit = predict(boost.fit,
                        data.submit,
                        n.trees=bestntree)
Submit = data.frame(id,loss=exp(pred.submit))
write.csv(Submit,file="Submit2.csv",row.names=FALSE)


#Smaller model
pred.submit2 = predict(boost.fit.lasso,
                        data.submit,
                        n.trees=bestntree)
Submit = data.frame(id,loss=exp(pred.submit2))
write.csv(Submit,file="Submit3.csv",row.names=FALSE)
```