95-845 AA-MLP Neural Networks for Hospital Billing

Sammuel Hobbs

November 2018

How might a prediction for DRG cost weight be useful for billing coders?

Hypothetically, an accurate tool that estimates the payment multiplier could (a) detect instances that might be innaccurately billed or even (b) streamline the hospital's current processes used to bill insurance companies, which might lead to (i) reduced labor costs and (ii) reduced time duration for collecting payment (improved cash flow).

Preprocessing & Data Overview

Set up workspace, load libraries

```
#Change before uploading
setwd("C:/Users/Sammuel Hobbs/Desktop/Semester 3/Applied Analytics the Machine Learning Pipeline/HWK/HWK3/")
directory = "C:/Users/Sammuel Hobbs/Desktop/Semester 3/Applied Analytics the Machine Learning Pipeline/HWK/HWK3/data"

### Load helper files ###
loadlibs = function(libs) {
  for(lib in libs) {
    class(lib)
    if(!do.call(require,as.list(lib))) {install.packages(lib)}
    do.call(require,as.list(lib))
  }
}
libs = c("tidyr","magrittr","purrr","dplyr","stringr","readr","data.table", "mice", "lubridate", "imager", "naniar")
#libs = c("tidyr","magrittr","purrr","dplyr","stringr","readr","data.table", "Lubridate")
loadlibs(libs)
```

Cost Weights, Labs, and Procedure Events that occurred among all hospital admits

Loading procedure files

```
# fread strip.white=TRUE is default, so no worries on leading/trailing whitespace in d_lab
d_coditem = fread(paste0(directory, "/d_codeditems.csv")) %>% as_tibble()
str(d_coditem)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    3339 obs. of  6 variables:
##  $ itemid     : int  101351 101352 101353 101354 101355 101356 101357 101358 101359 101360 ...
##  $ code       : chr  "7975" "7976" "7978" "7979" ...
##  $ type       : chr  "PROCEDURE" "PROCEDURE" "PROCEDURE" "PROCEDURE" ...
##  $ category   : chr  NA NA NA NA ...
##  $ label      : chr  NA NA NA NA ...
##  $ description: chr  "CL REDUC DISLOC-HIP" "CL REDUC DISLOC-KNEE" "CL REDUC DISLOC-FOOT/TOE" "CL REDUC DISLOCATION NEC"
...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
drgevents = fread(paste0(directory, "/drgevents.csv")) %>% as_tibble()
str(drgevents)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    5055 obs. of  4 variables:
##  $ subject_id : int  56 37 78 67 3 26 12 61 61 31 ...
##  $ hadm_id    : int  28766 18052 15161 35878 2075 15067 12532 5712 7149 15325 ...
##  $ itemid     : int  60017 60196 60657 60004 60614 60742 60305 60596 60700 60002 ...
##  $ cost_weight: num  1.22 1.62 0.82 2.08 1.62 5.33 3.42 1.81 6.1 3.73 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
procevent = fread(paste0(directory, "/procedureevents.csv")) %>% as_tibble()
str(procevent)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    25288 obs. of  5 variables:
##  $ subject_id  : int  56 56 37 37 78 67 3 3 3 3 ...
##  $ hadm_id     : int  28766 28766 18052 18052 15161 35878 2075 2075 2075 2075 ...
##  $ itemid      : int  101834 101780 101847 100737 100109 100092 101749 101868 101783 101696 ...
##  $ sequence_num: int  2 1 1 2 1 1 1 2 4 3 ...
##  $ proc_dt     : chr  "2644-01-18 00:00:00" "2644-01-19 00:00:00" "3264-08-14 00:00:00" "3264-08-17 00:00:00" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Cost Weights Summary

```
# select relevant features
d_sumry = drgevents %>%
  select(hadm_id, itemid, cost_weight)

# count how many times each cost weight has ocurred, creating cost_count column
d_sumry = d_sumry %>%
  group_by(cost_weight) %>%
  mutate(cost_count = n())

cat("Total cost weights: ", dim(d_sumry)[1], "\n")
```

```
## Total cost weights:  5055
```

```
# Get distinct cost weights, and sort by cost count
d_sumry = d_sumry %>%
  distinct(cost_weight, .keep_all = TRUE) %>%
arrange(desc(cost_count))

cat("Unique cost wieghts: ", dim(d_sumry)[1], "\n")
```

```
## Unique cost wieghts:  417
```

```
# Basic data summary - cost_weight distributions
#Hmisc::describe(d_sumry$cost_weight)
summary(d_sumry$cost_weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.270   1.590   2.830   3.564   4.340  20.040
```

```
# Top ten
d_sumry %>%
  select(cost_weight, cost_count) %>%
  head(10)
```

```
## # A tibble: 10 x 2
## # Groups:   cost_weight [10]
##    cost_weight cost_count
##          <dbl>      <int>
##  1        1.59        228
##  2        3.6         100
##  3        1.62         97
##  4        1.67         94
##  5        1.33         94
##  6        1.01         80
##  7        1            79
##  8        1.26         77
##  9        3.61         74
## 10        1.04         65
```

```
# bottom ten
d_sumry %>%
  select(cost_weight, cost_count) %>%
  tail(10)
```

```
## # A tibble: 10 x 2
## # Groups:   cost_weight [10]
##    cost_weight cost_count
##          <dbl>      <int>
##  1        6.44           1
##  2        9.3            1
##  3        3.37           1
##  4       10.6            1
##  5        3.51           1
##  6        1.5            1
##  7        7.7            1
##  8        4.5            1
##  9        2.18           1
## 10        4.23           1
```

Procedure Events Summary

```
# left join procedures and d_codeditems, keeping hadmin_id, description, itemid
procevent = procevent %>%
  left_join(d_coditem, by="itemid") %>%
  select(hadm_id, description, itemid)

# inner join drgevents and procedures, keeping cost_weight (y), hadmin_id, and description (to be Xs)
### drg_proc is full data set to be included in model
drg_proc = drgevents %>%
  inner_join(procevent, by="hadm_id") %>%
  select(cost_weight, hadm_id, description)
cat("Total procedures: ", dim(drg_proc)[1], "\n")
```

```
## Total procedures:  25252
```

```
# count how many times each procedure has ocurred, creating proc_count column
drg_proc = drg_proc %>%
  group_by(description) %>%
  mutate(proc_count = n())

# Get distinct descriptions, and sort by procedure count
proc_top = drg_proc %>%
  distinct(description, .keep_all = TRUE) %>%
arrange(desc(proc_count))
cat("Total unique procedure descriptions: ", dim(proc_top)[1], "\n")
```

```
## Total unique procedure descriptions:  932
```

```
# Basic data summary - event count distribution
summary(proc_top$proc_count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    3.00   27.09   11.00 1992.00
```

```
# Top ten
proc_top %>%
  select(description, proc_count) %>%
  head(10)
```

```
## # A tibble: 10 x 2
## # Groups:   description [10]
##    description              proc_count
##    <chr>                         <int>
##  1 CONTINUOUS INVASIVE MECH       1992
##  2 VENOUS CATHETER NEC            1923
##  3 INSERT ENDOTRACHEAL TUBE       1484
##  4 PACKED CELL TRANSFUSION        1267
##  5 EXT INFUS CONC NUTRITION       1114
##  6 ARTERIAL CATHETERIZATION        632
##  7 PARENTERAL INFUS CONC NU        524
##  8 CORONAR ARTERIOGR-2 CATH        511
##  9 HEMODIALYSIS                    491
## 10 SERUM TRANSFUSION NEC           413
```

```
# bottom ten
proc_top %>%
  select(description, proc_count) %>%
  tail(10)
```

```
## # A tibble: 10 x 2
## # Groups:   description [10]
##    description              proc_count
##    <chr>                         <int>
##  1 LOC EXC LES RADIUS/ULNA           1
##  2 GUM BIOPSY                        1
##  3 ABDOMINAL ENDARTERECTOMY          1
##  4 OTHER PART LARYNGECTOMY           1
##  5 ENDOVASCULAR REMOVAL OBS          1
##  6 UNIL FEMOR HERN REP NEC           1
##  7 THORACOSCOPIC EXCISION O          1
##  8 BILIARY TRACT OP NEC              1
##  9 THORACOSCOPIC DRAINAGE            1
## 10 REMOV SMALL BOWEL TUBE            1
```

Lab Events Summary

```
d_labitem = fread(paste0(directory, "/d_labitems.csv")) %>% as_tibble()
str(d_labitem)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    713 obs. of  6 variables:
##  $ itemid           : int  50225 50226 50227 50228 50229 50230 50231 50232 50234 50235 ...
##  $ test_name        : chr  "POTASSIUM" "SODIUM" "SURF/ALBU" "TOT BILI" ...
##  $ fluid            : chr  "OTHER BODY FLUID" "OTHER BODY FLUID" "OTHER BODY FLUID" "OTHER BODY FLUID" ...
##  $ category         : chr  "CHEMISTRY" "CHEMISTRY" "CHEMISTRY" "CHEMISTRY" ...
##  $ loinc_code       : chr  "2821-7" "2950-4" NA "1974-5" ...
##  $ loinc_description: chr  "Potassium [Moles/volume] in Body fluid" "Sodium [Moles/volume] in Body fluid" NA "Bilirubin
[Mass/volume] in Body fluid" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
labevents = fread(paste0(directory, "/labevents.csv")) %>% as_tibble()
str(labevents)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    3740682 obs. of  9 variables:
##  $ subject_id: int  56 56 56 56 56 56 56 56 56 56 ...
##  $ hadm_id   : int  28766 28766 28766 28766 28766 28766 28766 28766 28766 28766 ...
##  $ icustay_id: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ itemid    : int  50124 50112 50177 50090 50159 50149 50083 50172 50068 50468 ...
##  $ charttime : chr  "2644-01-17 00:30:00" "2644-01-17 00:30:00" "2644-01-17 00:30:00" "2644-01-17 00:30:00" ...
##  $ value     : chr  "HOLD" "155" "21" "0.7" ...
##  $ valuenum  : num  NA 155 21 0.7 128 4 91 27 14 10 ...
##  $ flag      : chr  NA "abnormal" "abnormal" NA ...
##  $ valueuom  : chr  NA "mg/dL" "mg/dL" "mg/dL" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Join Lab Data sets, pre-process missing, feature engineering

```
# filter to only include hadm_id present in DRG Events...
# itemid.x is itemid from labevents
labev_drg = labevents %>%
  inner_join(drgevents, by="hadm_id") %>%
      select(hadm_id, itemid.x, flag)
cat("Total lab events: ", dim(labev_drg)[1], "\n")
```

```
## Total lab events:  2867084
```

```
### Expert knowledge Assumption: only abnormal events are recorded as abnormal, so assume NA is normal
# reset column name, then encode flag NA to normal
labev_drg = rename(labev_drg, itemid = itemid.x)
labev_drg = labev_drg %>%
  mutate(flag  = ifelse(is.na(flag), "normal", flag))

#Join for descriptions, d_labitem to get descriptions
### labs_labels to be included in Model
labs_labels = labev_drg %>%
  inner_join(d_labitem, by="itemid") %>%
  select(hadm_id, itemid, test_name, fluid, category, loinc_description, flag)

### Where loinc_description is missing, create a compound description which is combination of test_name, fluid, category
### Description encoded as "miss_desc_test_name_fluid_category"
labs_labels = labs_labels %>%
  mutate(loinc_description  = ifelse(is.na(loinc_description), paste("miss_desc", test_name, fluid, category, sep="_"), loin
c_description))

# Create 'Tuple' (event description, flag) encoded as "event-flag"
labs_labels = labs_labels %>%
  mutate(event_flag = paste(loinc_description, flag, sep="-"))

# count how many times each lab event_flag has ocurred, creating proc_count column
labs_labels = labs_labels %>%
  group_by(event_flag) %>%
  mutate(event_flag_qty = n())

# Get distinct event_flag, and sort by count
lab_top = labs_labels %>%
  distinct(event_flag, .keep_all = TRUE) %>%
arrange(desc(event_flag_qty)) %>%
  select(hadm_id, itemid, event_flag, event_flag_qty)

cat("Total unique lab event-flag combinations: ", dim(lab_top)[1], "\n")
```

```
## Total unique lab event-flag combinations:  852
```

```
# Basic data summary - event count distribution
summary(lab_top$event_flag_qty)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     6.0    48.0  3365.1   564.2 78040.0
```

```
# Top ten
lab_top %>%
  select(event_flag, event_flag_qty) %>%
  head(10)
```

```
## # A tibble: 10 x 2
## # Groups:   event_flag [10]
##    event_flag                                              event_flag_qty
##    <chr>                                                            <int>
##  1 miss_desc_TYPE_BLOOD_BLOOD GAS-normal                            78040
##  2 Hematocrit [Volume Fraction] of Blood-abnormal                   76919
##  3 Hemoglobin [Mass/volume] in Blood-abnormal                       72550
##  4 Base excess in Blood-normal                                      72426
##  5 Potassium [Moles/volume] in Serum or Plasma-normal               69597
##  6 Anion gap in Blood-normal                                        65188
##  7 Erythrocytes [#/volume] in Blood-abnormal                        63701
##  8 Magnesium [Mass/volume] in Serum or Plasma-normal                61713
##  9 Sodium [Moles/volume] in Serum or Plasma-normal                  58546
## 10 Erythrocyte mean corpuscular volume [Entitic volume]-no~         56410
```

```
# bottom ten
lab_top %>%
  select(event_flag, event_flag_qty) %>%
  tail(10)
```

```
## # A tibble: 10 x 2
## # Groups:   event_flag [10]
##    event_flag                                              event_flag_qty
##    <chr>                                                            <int>
##  1 Glucose-6-Phosphate dehydrogenase [Enzymatic activity/m~             1
##  2 Hemoglobin S/Hemoglobin.total in Blood-abnormal                      1
##  3 Metamyelocytes/100 leukocytes in Cerebral spinal fluid-~             1
##  4 Protein [Mass/volume] in Synovial fluid-normal                       1
##  5 Lymphocytes Variant/100 leukocytes in Synovial fluid-ab~             1
##  6 Glucose [Mass/volume] in Synovial fluid-normal                       1
##  7 Thrombin time in Platelet poor plasma by Coagulation as~             1
##  8 CD16 cells/100 cells in Blood-normal                                 1
##  9 miss_desc_MacrPRL_BLOOD_CHEMISTRY-normal                             1
## 10 RBC casts [#/area] in Urine sediment by Microscopy low ~             1
```

Intuitively, there is are more lab events than procedures. Interesting to see the blend of abnormal to normal flags inculded with the lab event, and to see that trends like “while there is high count of ‘Hemoglobin-abnormal’ instances, ‘Hemoglobin-normal’ is not part of the top ten”. Understanding the dynamics of the data we’re dealing with supports the need to (a) log transform the data and (b) regularize during training.

Create Data Frame for Deep Learning Model

Here we build the dataset that will be fed into the NN model later on. Instructed to keep the top 2,000 potential features from each table, we will keep all features since each table had less than 1,000 each. It’s evident that the frequency of events per hospital admin is quite sparse. We will log transform the data, both features and target following f(x) = log(1+x).

```
# from our dataframe with all events, get the those that are part of top-procedure events
proc_events =  drg_proc %>%
  filter(description %in% proc_top$description) %>%
  select(hadm_id, description)

# rename column for consistency
proc_events = rename(proc_events, event = description)

# count where hospital admission incurred a given event (hadm_id, event)
proc_events = proc_events %>%
  group_by(hadm_id, event)%>%
  mutate(event_count = n())

# from our dataframe with all lab events, get the those that are part of top-lab events
lab_events = labs_labels %>%
  filter(itemid %in% lab_top$itemid) %>%
  select(hadm_id, event_flag)

# rename column for consistency
lab_events = rename(lab_events, event = event_flag)

# count where hospital admission incurred a given event (hadm_id, event)
lab_events = lab_events %>%
  group_by(hadm_id, event)%>%
  mutate(event_count = n())

# combine the two dfs into a single dataframe
combined = bind_rows(proc_events,lab_events)
cat("combined dimension\n rows/instances: ", dim(combined)[1], "\n columns/variables: ", dim(combined)[2], "\n")
```

```
## combined dimension
##  rows/instances:  2876212
##  columns/variables:  3
```

```
# Transpose/go from Long to Wide - Spread
dataset = combined %>%
  distinct(.keep_all = TRUE) %>%
  spread(event, event_count, fill=0)
cat("dataset dimension\n examples/instances: ", dim(dataset)[1], "\n features/variables: ", dim(dataset)[2], "\n")
```

```
## dataset dimension
##  examples/instances:  5050
##  features/variables:  1785
```

```
# add cost weight to dataframe
dataset = dataset %>%
  inner_join(select(drgevents, hadm_id, cost_weight), by="hadm_id")

cat("dataset dimension w/ cost_weight\n examples/instances: ", dim(dataset)[1], "\n features/variables: ", dim(dataset)[2],
"\n")
```

```
## dataset dimension w/ cost_weight
##  examples/instances:  5050
##  features/variables:  1786
```

```
# log transform the dataset - hadm_id is furthest left column 1 and cost_weight is furthest right 1786
dataset[,2:1786] = log(dataset[,2:1786]+1)
```

```
# simply cleaning up workspace...memory management as some variables contain millions of records
rm(combined, d_coditem, d_sumry, procevent, drg_proc, drgevents, lab_events, lab_top, labs_labels, proc_events, proc_top, d_
labitem, labevents, labev_drg)
```

Create Training / Test sets

```
# ungroup dataset so we can delete hadm_id from data that will be loaded into model
dataset = dataset %>%
  ungroup()

# get random 70/30 split, based on hadm_id - the unique examples
library(caTools)
set.seed(123)
split = sample.split(dataset$hadm_id, SplitRatio = 0.7)
train = subset(dataset, split == TRUE)
test = subset(dataset, split == FALSE)
# delete the hadm_id
train = subset(train, select=-hadm_id)
test = subset(test, select=-hadm_id)
# Test set
xtest = test %>% select(-cost_weight) %>% as.matrix()
ytest = test %>% select(cost_weight) %>% as.matrix()
# Train set
xtrain = train %>% select(-cost_weight) %>% as.matrix()
ytrain = train %>% select(cost_weight) %>% as.matrix()

cat("Train Set dimensions \n examples/instances: ", dim(xtrain)[1], "\n features/variables: ", dim(xtrain)[2], "\n")
```

```
## Train Set dimensions
##  examples/instances:  3535
##  features/variables:  1784
```

```
cat("Test Set dimensions \n examples/instances: ", dim(xtest)[1], "\n features/variables: ", dim(xtest)[2], "\n")
```

```
## Test Set dimensions
##  examples/instances:  1515
##  features/variables:  1784
```

Create NN models - Instructed to make 3 hidden layers of size 32

Baseline

```
library(ggplot2)
library(keras)
### Specify the architecture
baseline = keras_model_sequential()
baseline %>%
  layer_dense(units = 32, activation = 'relu',
              input_shape = c(ncol(xtrain))) %>%
  #layer_dropout(rate = 0.5) %>%
  layer_dense(units = 32, activation = 'relu') %>%
  layer_dense(units = 32, activation = 'tanh') %>%
  layer_dense(units = 1, activation = 'linear')

summary(baseline)
```

```
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## dense_1 (Dense)              (None, 32)                57120
## _________________________________________________________________________
## dense_2 (Dense)              (None, 32)                1056
## _________________________________________________________________________
## dense_3 (Dense)              (None, 32)                1056
## _________________________________________________________________________
## dense_4 (Dense)              (None, 1)                 33
## =========================================================================
## Total params: 59,265
## Trainable params: 59,265
## Non-trainable params: 0
## _________________________________________________________________________
```

```
### Specify loss, batch size, optimizer, extra performance measures
baseline %>% compile(
  loss = c('mse'),
  optimizer = 'adam',
  metrics = c('mse')
)
```

Train and Report Models

I create 3 NN Regression models: baseline, L1 Regularized, and L1+Modifications. In previous iterations, L1 seemed to perform better than L2, so I've excluded L2 from this report. I was instructed to build a NN with 3 hidden layers of size 32 with any choice of activation functions. I've chosen 'Relu' as activation function on these hidden layers. Since the data was (a) log transformed and (b) this is a regression problem - so I use Mean Square Error as the loss function - I've included the last layer to have a linear function. The L1 model uses Lasso on two layers, and the L1+Mod. includes a dropout on one layer, and clipnorm in the loss function.

```
### Run baseline model
base_history =
  baseline %>% fit(xtrain, ytrain,
              epochs = 50,
              batch_size = 100,
              validation_split = 0.2,
              shuffle=T,
              verbose = F
  )
```

L1 Regularization and L1+Modifications

```
### Lasso  - L1 Regularization
l1_reg = keras_model_sequential()
l1_reg %>%
  layer_dense(units = 32, activation = 'relu',
                input_shape = c(ncol(xtrain)),
              kernel_regularizer = regularizer_l1(l = 0.005)) %>%
  layer_dense(units = 32, activation = 'relu') %>%
  layer_dense(units = 32, activation = 'relu',
              kernel_regularizer = regularizer_l1(l = 0.001)) %>%
  layer_dense(units = 1, activation = 'linear')

# Mean Square error as loss and metric to evalutate
l1_reg %>% compile(
  loss = c('mse'),
  optimizer = 'adam',
  metrics = c('mse')
)

### L1 + adding dropout and clipping
l1_mod = keras_model_sequential()
l1_mod %>%
  layer_dense(units = 32, activation = 'relu',
                input_shape = c(ncol(xtrain)),
              kernel_regularizer = regularizer_l2(l = 0.005)) %>%
  layer_dense(units = 32, activation = 'relu') %>%
                      layer_dropout(rate = 0.3) %>%
  layer_dense(units = 32, activation = 'relu',
              kernel_regularizer = regularizer_l2(l = 0.001)) %>%
  layer_dense(units = 1, activation = 'linear')


# Mean Square error as loss and metric to evalutate
l1_mod %>% compile(
  loss = c('mse'),
  optimizer = optimizer_nadam(clipnorm = 10),
  metrics = c('mse')
)
```

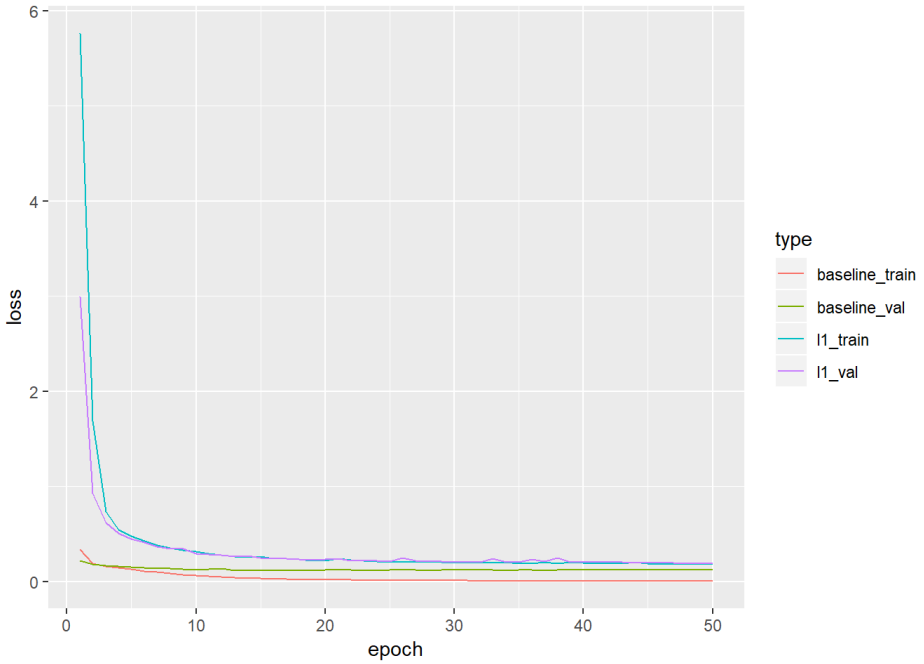
L1 Run and Plot

```
### Run L1 Model
l1_history =
  l1_reg %>% fit(xtrain, ytrain,
              epochs = 50,
              batch_size = 100,
              validation_split = 0.2,
              shuffle=T,
              verbose = F
  )
```

Plot L1 to baseline Comparisons

```
library(tibble)
compare_l1 <- data.frame(
  baseline_train = base_history$metrics$loss,
  baseline_val = base_history$metrics$val_loss,
  l1_train = l1_history$metrics$loss,
  l1_val = l1_history$metrics$val_loss
) %>%
  rownames_to_column() %>%
  mutate(rowname = as.integer(rowname)) %>%
  gather(key = "type", value = "value", -rowname)

ggplot(compare_l1, aes(x = rowname, y = value, color = type)) +
  geom_line() +
  xlab("epoch") +
  ylab("loss")
```

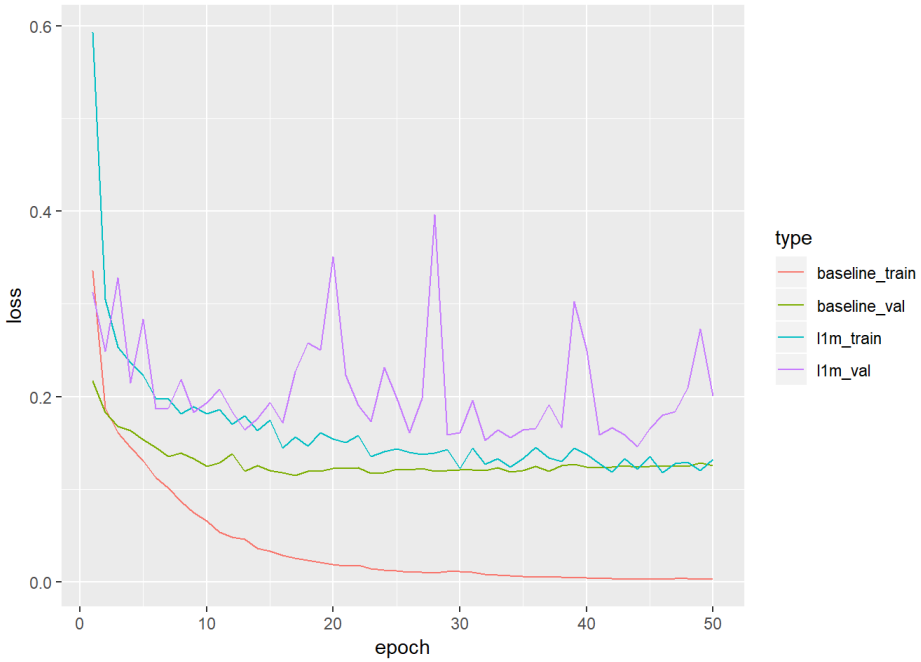


L1+modified Run and Plot

```
### Run L1 modified model
l1mod_hist =
  l1_mod %>% fit(xtrain, ytrain,
              epochs = 50,
              batch_size = 100,
              validation_split = 0.2,
              shuffle=T,
              verbose = F
  )
```

```
# Plotting L1 modified
compare_l1m <- data.frame(
  baseline_train = base_history$metrics$loss,
  baseline_val = base_history$metrics$val_loss,
  l1m_train = l1mod_hist$metrics$loss,
  l1m_val = l1mod_hist$metrics$val_loss
) %>%
  rownames_to_column() %>%
  mutate(rowname = as.integer(rowname)) %>%
  gather(key = "type", value = "value", -rowname)

plotL2 = ggplot(compare_l1m, aes(x = rowname, y = value, color = type)) +
  geom_line() +
  xlab("epoch") +
  ylab("loss")
plotL2
```

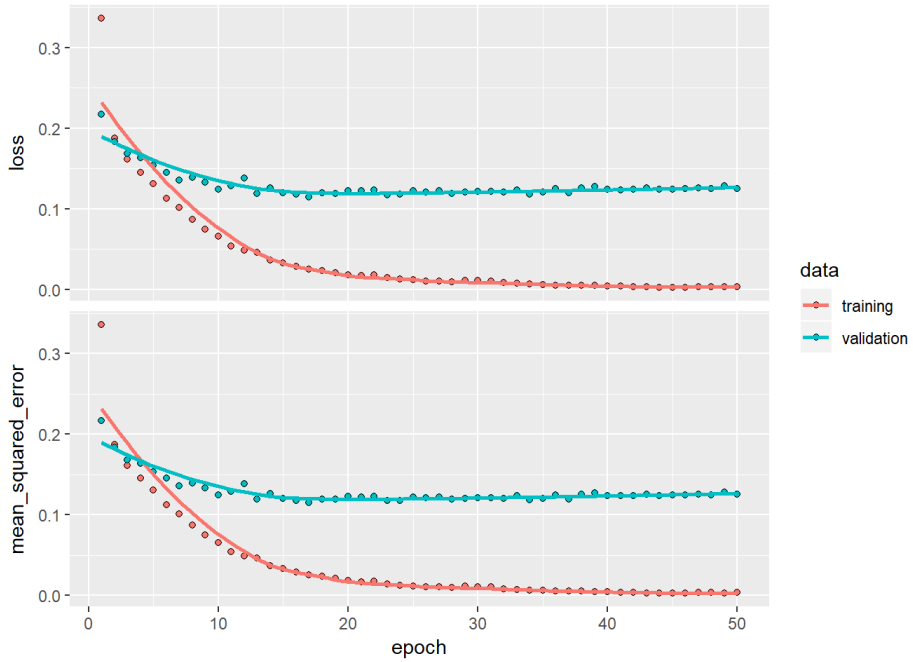


Side-by-side comparisons: Baseline, L1, L1+Modifications

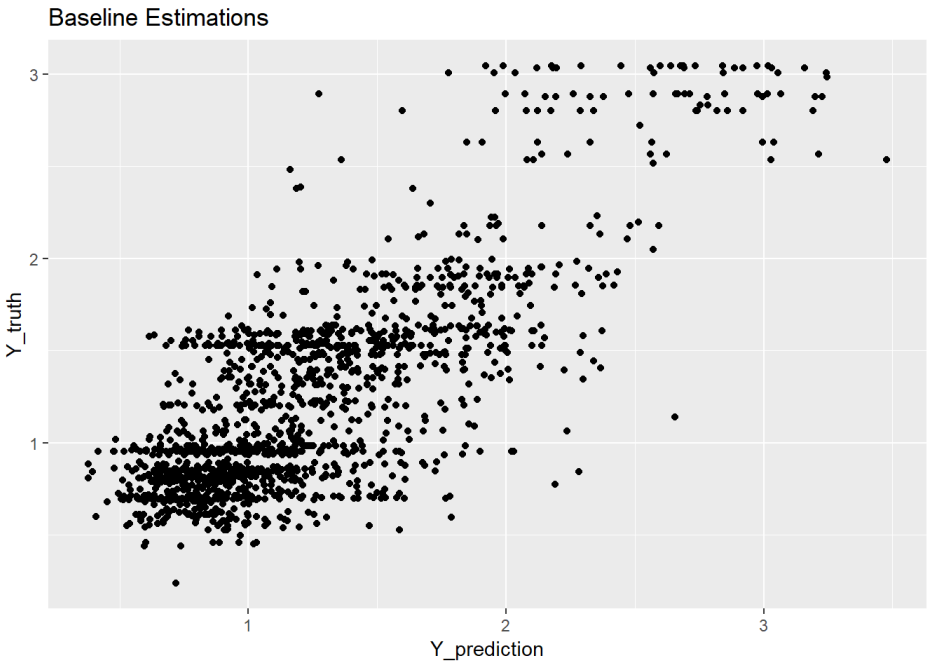
```
# baseline Evaluating and Plotting
baseline %>% evaluate(xtest, ytest)
```

```
## $loss
## [1] 0.130378
##
## $mean_squared_error
## [1] 0.130378
```

```
plot(base_history, main="base_history")
```



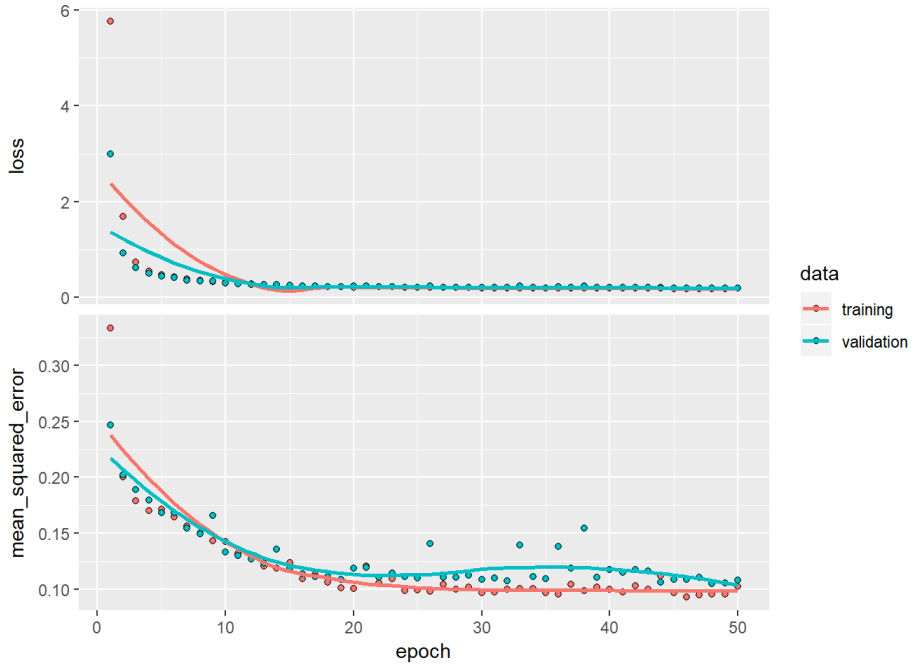

```
data.frame(Y_truth = ytest[,1], Y_prediction = baseline %>% predict(xtest)) %>%
  ggplot(., aes(x=Y_prediction, y=Y_truth)) +
  ggtitle("Baseline Estimations")+
  geom_point()
```



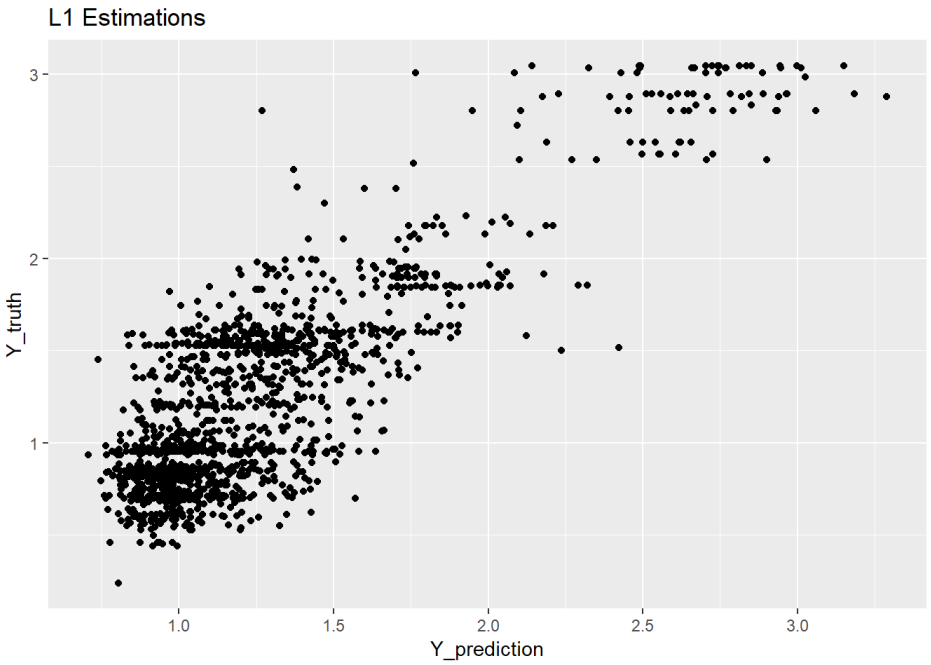

```
# L1 Evaluating and Plotting
l1_reg %>% evaluate(xtest, ytest)
```

```
## $loss
## [1] 0.1819627
##
## $mean_squared_error
## [1] 0.09400506
```

```
plot(l1_history, main="l1_history")
```



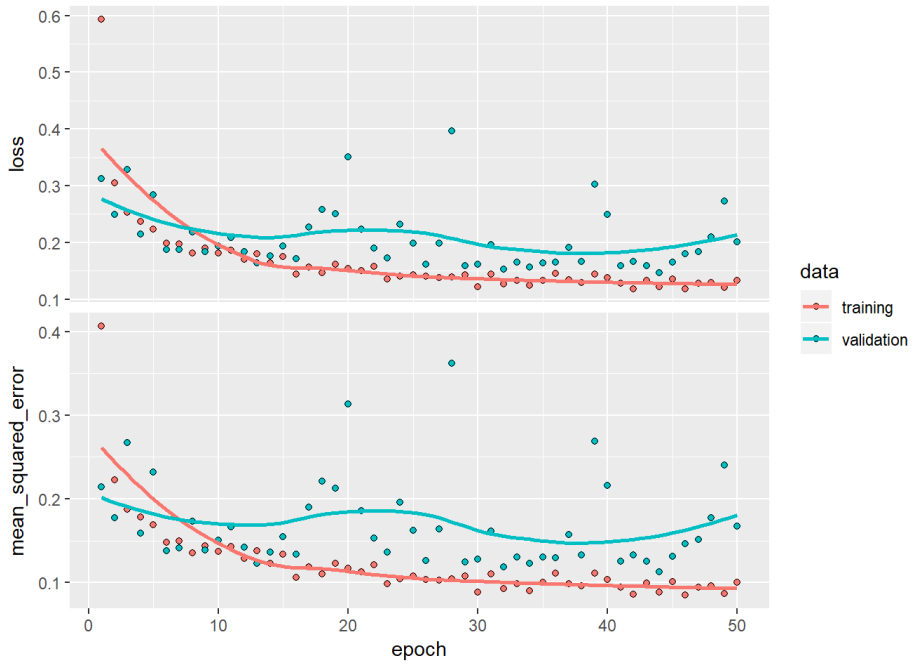

```
data.frame(Y_truth = ytest[,1], Y_prediction = l1_reg %>% predict(xtest)) %>%
  ggplot(., aes(x=Y_prediction, y=Y_truth)) +
  ggtitle("L1 Estimations")+
  geom_point()
```



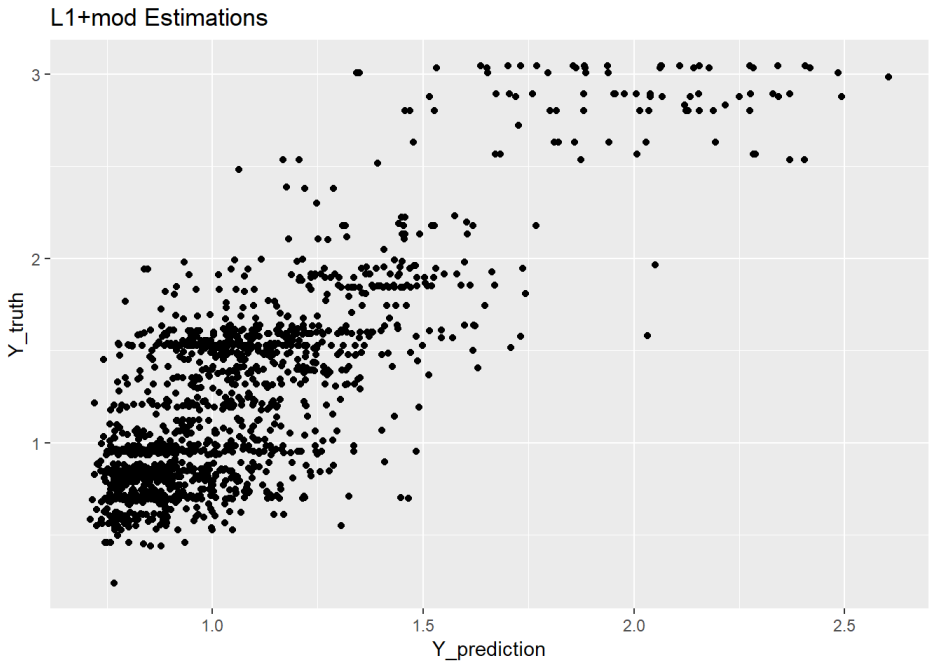

```
# L1+mod Evaluating and Plotting
l1_mod %>% evaluate(xtest, ytest)
```

```
## $loss
## [1] 0.1978052
##
## $mean_squared_error
## [1] 0.1648385
```

```
plot(l1mod_hist, main="l1mod_hist")
```




```
data.frame(Y_truth = ytest[,1], Y_prediction = l1_mod %>% predict(xtest)) %>%
  ggplot(., aes(x=Y_prediction, y=Y_truth)) +
  ggtitle("L1+mod Estimations")+
  geom_point()
```



In most iterations, L1 regularization model keeps a tighter, linear estimate distribution. The L1+mod seems to take longer to converge, and possibly needs more training

Given this excercise, adjusting hyperparameters might yield improved results. In my experience, this consumes a lot of time and often yields smaller incremental gainz in performance.

MIMIC II has a few other tables that might have relevance in estimating costs, particular tables on medication consumption during length of stay. Considering, we would need to determine if it makes sense to add the addtional features to the already-present 1700 features, or maintain a cap limit of how many features to include from each category (procedures, labs, medication).