Novozymes

Jeremy Haakenson

2022-11-09

R Markdown

This R Markdown file describes one of my submissions to the Novozymes Enzyme Stability Prediction Kaggle competition. This competition provided training data containing mutated sequences of multiple peptides and their melting temps. The test data contained mutated sequences of a single peptide. The goal was to correctly order these mutated sequences based on predicted melting temp. (i.e. to predict relative melting temp. of all seqs in the test data).

First, the training data had to be updated because there were mistakes in the original file. I updated the train file using code from Petr1zi0 (Kaggle) in Python as follows:

import pandas as pd

df_train = pd.read_csv("train.csv", index_col="seq_id") df_train_updates = pd.read_csv("train_updates_20220929.csv", index_col="seq_id")

all_features_nan = df_train_updates.isnull().all("columns")

drop_indices = df_train_updates[all_features_nan].index df_train = df_train.drop(index=drop_indices)

swap_ph_tm_indices = df_train_updates[~all_features_nan].index df_train.loc[swap_ph_tm_indices, ["pH", "tm"]] = df_train_updates.loc[swap_ph_tm_indices, ["pH", "tm"]]

The rest of this file is the R code that I wrote.

Load packages

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stringr)
library(keras)
library(tensorflow)
library(tinytex)
```

Read in files.

```
train = read.csv('new_train.csv')
test = read.csv('test.csv')
```

Add columns for each amino acid.

```
train$A = str_count(train$protein_sequence, 'A')
train$C = str_count(train$protein_sequence, 'C')
train$D = str_count(train$protein_sequence, 'D')
train$E = str_count(train$protein_sequence, 'E')
train$F = str_count(train$protein_sequence, 'F')
train$G = str_count(train$protein_sequence, 'G')
train$H = str_count(train$protein_sequence, 'H')
train$I = str_count(train$protein_sequence, 'I')
train$K = str_count(train$protein_sequence, 'K')
train$L = str_count(train$protein_sequence, 'L')
train$M = str_count(train$protein_sequence, 'M')
train$N = str_count(train$protein_sequence, 'N')
train$P = str_count(train$protein_sequence, 'P')
train$Q = str_count(train$protein_sequence, 'Q')
train$R = str_count(train$protein_sequence, 'R')
train$S = str_count(train$protein_sequence, 'S')
train$T = str_count(train$protein_sequence, 'T')
train$V = str_count(train$protein_sequence, 'V')
train$W = str_count(train$protein_sequence, 'W')
train$Y = str_count(train$protein_sequence, 'Y')

str(train)
```

```
## 'data.frame':    28981 obs. of  25 variables:
##  $ seq_id          : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ protein_sequence: chr  "AAAAKAAALALLGEAPEVVDIWLPAGWRQPFRVFRLERKGDGVLVGMIKDAGDDPDVTHGAEIQAFVRFASEDI
##  $ pH              : num  7 7 7 7 7 7 7 7 7 7 ...
##  $ data_source     : chr  "doi.org/10.1038/s41592-020-0801-4" "doi.org/10.1038/s41592-020-0801-4" "do
##  $ tm              : num  75.7 50.5 40.5 47.2 49.5 48.4 45.7 55.9 48.1 49.7 ...
##  $ A               : int  45 28 50 20 86 33 33 15 41 14 ...
##  $ C               : int  1 0 9 5 14 4 4 16 1 2 ...
##  $ D               : int  13 10 27 19 78 16 16 18 19 7 ...
##  $ E               : int  30 52 32 29 78 19 19 17 16 7 ...
##  $ F               : int  13 6 21 12 32 16 16 3 9 7 ...
##  $ G               : int  38 18 65 16 84 33 33 28 25 20 ...
##  $ H               : int  3 4 11 7 40 9 9 10 0 4 ...
##  $ I               : int  14 13 16 10 71 16 16 8 17 6 ...
```

```
##  $ K              : int  16 19 39 17 68 27 26 22 27 9 ...
##  $ L              : int  37 23 18 28 104 33 33 27 21 7 ...
##  $ M              : int  8 2 6 2 31 11 11 6 8 1 ...
##  $ N              : int  5 6 15 9 65 13 13 18 16 7 ...
##  $ P              : int  18 8 20 16 128 19 19 21 10 12 ...
##  $ Q              : int  6 22 25 9 54 8 8 11 9 4 ...
##  $ R              : int  25 30 31 10 63 16 17 10 6 3 ...
##  $ S              : int  11 14 33 16 148 22 22 11 14 11 ...
##  $ T              : int  14 12 30 19 120 25 25 18 13 17 ...
##  $ V              : int  37 13 30 14 124 41 41 23 28 14 ...
##  $ W              : int  4 3 3 3 16 10 10 2 3 5 ...
##  $ Y              : int  3 3 16 4 47 9 9 17 4 6 ...
```

```
test$A = str_count(test$protein_sequence, 'A')
test$C = str_count(test$protein_sequence, 'C')
test$D = str_count(test$protein_sequence, 'D')
test$E = str_count(test$protein_sequence, 'E')
test$F = str_count(test$protein_sequence, 'F')
test$G = str_count(test$protein_sequence, 'G')
test$H = str_count(test$protein_sequence, 'H')
test$I = str_count(test$protein_sequence, 'I')
test$K = str_count(test$protein_sequence, 'K')
test$L = str_count(test$protein_sequence, 'L')
test$M = str_count(test$protein_sequence, 'M')
test$N = str_count(test$protein_sequence, 'N')
test$P = str_count(test$protein_sequence, 'P')
test$Q = str_count(test$protein_sequence, 'Q')
test$R = str_count(test$protein_sequence, 'R')
test$S = str_count(test$protein_sequence, 'S')
test$T = str_count(test$protein_sequence, 'T')
test$V = str_count(test$protein_sequence, 'V')
test$W = str_count(test$protein_sequence, 'W')
test$Y = str_count(test$protein_sequence, 'Y')

str(test)
```

```
## 'data.frame':    2413 obs. of  24 variables:
##  $ seq_id         : int  31390 31391 31392 31393 31394 31395 31396 31397 31398 31399 ...
##  $ protein_sequence: chr  "VPVNPEPDATSVENVAEKTGSGDSQSDPIKADLEVKGQSALPFDVDCWAILCKGAPNVLQRVNEKTKNSNRDR
##  $ pH             : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ data_source    : chr  "Novozymes" "Novozymes" "Novozymes" "Novozymes" ...
##  $ A              : int  22 22 22 22 22 22 22 22 23 22 ...
##  $ C              : int  4 4 4 5 4 4 4 4 4 5 ...
##  $ D              : int  15 15 15 15 15 15 15 15 15 15 ...
##  $ E              : int  8 7 7 7 7 7 7 7 7 7 ...
##  $ F              : int  10 10 10 10 11 10 10 10 10 10 ...
##  $ G              : int  19 19 19 19 19 20 19 19 19 19 ...
##  $ H              : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ I              : int  6 6 6 6 6 6 6 7 6 6 ...
##  $ K              : int  24 25 24 23 23 23 23 23 24 24 ...
##  $ L              : int  10 10 10 11 11 11 11 11 11 11 ...
##  $ M              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ N              : int  19 19 19 19 19 19 19 19 19 19 ...
##  $ P              : int  17 17 17 17 17 17 17 17 17 17 ...
```

```
##  $ Q              : int  13 13 13 13 13 13 13 13 13 13 ...
##  $ R              : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ S              : int  18 18 18 18 18 18 18 18 18 18 ...
##  $ T              : int  8 8 8 8 8 8 8 8 7 7 ...
##  $ V              : int  13 13 13 13 13 13 13 13 13 13 ...
##  $ W              : int  6 6 6 6 6 6 6 6 6 6 ...
##  $ Y              : int  6 6 6 6 6 6 6 6 6 6 ...
```

Look for correlation between variables.

```
cor(train[,-c(1, 2, 4)])
```

```
##    pH          tm           A          C          D           E          F
## pH  1          NA          NA         NA         NA          NA         NA
## tm NA  1.00000000 -0.06368781 -0.1507635 -0.1660776 -0.09826074 -0.1575996
## A  NA -0.06368781  1.00000000  0.5597421  0.8339315  0.84821786  0.7588805
## C  NA -0.15076355  0.55974206  1.0000000  0.6431774  0.55520117  0.6316747
## D  NA -0.16607762  0.83393149  0.6431774  1.0000000  0.89103337  0.8324413
## E  NA -0.09826074  0.84821786  0.5552012  0.8910334  1.00000000  0.7634343
## F  NA -0.15759956  0.75888054  0.6316747  0.8324413  0.76343432  1.0000000
## G  NA -0.05955940  0.82916573  0.6254079  0.7795064  0.72170366  0.7629963
## H  NA -0.14344716  0.80269498  0.6661098  0.8289686  0.81429703  0.7968257
## I  NA -0.16497260  0.77031493  0.5813342  0.8644737  0.79132371  0.8570663
## K  NA -0.17751958  0.78443868  0.5548939  0.8891254  0.90168110  0.7731555
## L  NA -0.08288258  0.88209727  0.6146057  0.8561091  0.88165530  0.8429380
## M  NA -0.17061164  0.78715069  0.5756794  0.8031898  0.78371978  0.7988002
## N  NA -0.18564332  0.74722598  0.6291125  0.8655134  0.76297505  0.8091492
## P  NA -0.07728468  0.78758872  0.5810022  0.7384382  0.71804362  0.7044033
## Q  NA -0.15944041  0.83022883  0.5735716  0.7951807  0.83855755  0.7060269
## R  NA -0.06235717  0.84824348  0.6034747  0.8529106  0.88372189  0.7670429
## S  NA -0.17446445  0.82574285  0.6396229  0.8584832  0.82737562  0.7865843
## T  NA -0.16405951  0.85735693  0.6525436  0.8793945  0.83568674  0.8231055
## V  NA -0.10483402  0.88544881  0.6371723  0.8786097  0.84663947  0.8505043
## W  NA -0.09482738  0.64137894  0.5575501  0.6817176  0.62351650  0.7262911
## Y  NA -0.10255378  0.69996607  0.6019171  0.7969435  0.70705685  0.8448077
##             G          H          I          K          L          M
## pH         NA         NA         NA         NA         NA         NA
## tm -0.0595594 -0.1434472 -0.1649726 -0.1775196 -0.08288258 -0.1706116
## A   0.8291657  0.8026950  0.7703149  0.7844387  0.88209727  0.7871507
## C   0.6254079  0.6661098  0.5813342  0.5548939  0.61460571  0.5756794
## D   0.7795064  0.8289686  0.8644737  0.8891254  0.85610909  0.8031898
## E   0.7217037  0.8142970  0.7913237  0.9016811  0.88165530  0.7837198
## F   0.7629963  0.7968257  0.8570663  0.7731555  0.84293804  0.7988002
## G   1.0000000  0.7682254  0.7333530  0.6916946  0.75961665  0.7343501
## H   0.7682254  1.0000000  0.7729814  0.7635739  0.84189398  0.7916016
## I   0.7333530  0.7729814  1.0000000  0.8261954  0.82590947  0.8239909
## K   0.6916946  0.7635739  0.8261954  1.0000000  0.81891526  0.7808101
## L   0.7596166  0.8418940  0.8259095  0.8189153  1.00000000  0.8203655
## M   0.7343501  0.7916016  0.8239909  0.7808101  0.82036551  1.0000000
## N   0.7366308  0.7741797  0.8537618  0.8231789  0.77493578  0.7799707
## P   0.8282473  0.7734054  0.6498044  0.6870047  0.73564425  0.7054048
## Q   0.7380101  0.8177946  0.7147059  0.7854907  0.84039591  0.7811064
## R   0.7779188  0.8215846  0.7516959  0.7939329  0.87618386  0.7814367
```

```
## S   0.7903421  0.8381284  0.7874734  0.8293743  0.84163663  0.7887276
## T   0.8203594  0.8346257  0.8435905  0.8342451  0.84648177  0.8006576
## V   0.8393953  0.8308473  0.8627909  0.8202688  0.88885438  0.8165936
## W   0.6567920  0.6849189  0.6334529  0.5936254  0.70343496  0.6215103
## Y   0.7411886  0.7599398  0.8138472  0.7310199  0.75204172  0.7456781
##             N           P          Q           R          S          T
## pH         NA          NA         NA          NA         NA         NA
## tm -0.1856433 -0.07728468 -0.1594404 -0.06235717 -0.1744645 -0.1640595
## A   0.7472260  0.78758872  0.8302288  0.84824348  0.8257428  0.8573569
## C   0.6291125  0.58100223  0.5735716  0.60347472  0.6396229  0.6525436
## D   0.8655134  0.73843821  0.7951807  0.85291056  0.8584832  0.8793945
## E   0.7629750  0.71804362  0.8385576  0.88372189  0.8273756  0.8356867
## F   0.8091492  0.70440327  0.7060269  0.76704286  0.7865843  0.8231055
## G   0.7366308  0.82824735  0.7380101  0.77791879  0.7903421  0.8203594
## H   0.7741797  0.77340540  0.8177946  0.82158463  0.8381284  0.8346257
## I   0.8537618  0.64980443  0.7147059  0.75169585  0.7874734  0.8435905
## K   0.8231789  0.68700471  0.7854907  0.79393287  0.8293743  0.8342451
## L   0.7749358  0.73564425  0.8403959  0.87618386  0.8416366  0.8464818
## M   0.7799707  0.70540481  0.7811064  0.78143668  0.7887276  0.8006576
## N   1.0000000  0.69971613  0.7735300  0.73153308  0.8451433  0.8574810
## P   0.6997161  1.00000000  0.7677346  0.78325866  0.8427404  0.8165790
## Q   0.7735300  0.76773460  1.0000000  0.80634004  0.8429678  0.8239716
## R   0.7315331  0.78325866  0.8063400  1.00000000  0.8297578  0.8171987
## S   0.8451433  0.84274042  0.8429678  0.82975780  1.0000000  0.9083172
## T   0.8574810  0.81657900  0.8239716  0.81719868  0.9083172  1.0000000
## V   0.7991295  0.78656646  0.7831897  0.84512302  0.8479689  0.9026142
## W   0.6274534  0.57831379  0.6119439  0.66070130  0.6227268  0.6596205
## Y   0.8042488  0.67693342  0.6598900  0.72379267  0.7329416  0.7855531
##             V           W          Y
## pH         NA          NA         NA
## tm -0.1048340 -0.09482738 -0.1025538
## A   0.8854488  0.64137894  0.6999661
## C   0.6371723  0.55755009  0.6019171
## D   0.8786097  0.68171761  0.7969435
## E   0.8466395  0.62351650  0.7070568
## F   0.8505043  0.72629114  0.8448077
## G   0.8393953  0.65679200  0.7411886
## H   0.8308473  0.68491889  0.7599398
## I   0.8627909  0.63345290  0.8138472
## K   0.8202688  0.59362538  0.7310199
## L   0.8888544  0.70343496  0.7520417
## M   0.8165936  0.62151029  0.7456781
## N   0.7991295  0.62745340  0.8042488
## P   0.7865665  0.57831379  0.6769334
## Q   0.7831897  0.61194393  0.6598900
## R   0.8451230  0.66070130  0.7237927
## S   0.8479689  0.62272683  0.7329416
## T   0.9026142  0.65962052  0.7855531
## V   1.0000000  0.67971505  0.7878602
## W   0.6797150  1.00000000  0.6927781
## Y   0.7878602  0.69277810  1.0000000
```

Most amino acids are correlated with many other amino acids. There are three groups: everything except C or W; C; and W.

Add charged variable.

```
train$charged = str_count(train$protein_sequence, 'D') +
  str_count(train$protein_sequence, 'E') +
  str_count(train$protein_sequence, 'H') +
  str_count(train$protein_sequence, 'K') +
  str_count(train$protein_sequence, 'R')

test$charged = str_count(test$protein_sequence, 'D') +
  str_count(test$protein_sequence, 'E') +
  str_count(test$protein_sequence, 'H') +
  str_count(test$protein_sequence, 'K') +
  str_count(test$protein_sequence, 'R')
```

Add hydrophobic variable.

```
train$hydrophobic = str_count(train$protein_sequence, 'G') +
  str_count(train$protein_sequence, 'A') +
  str_count(train$protein_sequence, 'V') +
  str_count(train$protein_sequence, 'L') +
  str_count(train$protein_sequence, 'I') +
  str_count(train$protein_sequence, 'P')

test$hydrophobic = str_count(test$protein_sequence, 'G') +
  str_count(test$protein_sequence, 'A') +
  str_count(test$protein_sequence, 'V') +
  str_count(test$protein_sequence, 'L') +
  str_count(test$protein_sequence, 'I') +
  str_count(test$protein_sequence, 'P')
```

Scale data.

```
train.scale = cbind.data.frame(train$seq_id, scale(train[,5:27]))
colnames(train.scale)[1] = 'seq_id'
test.scale = cbind.data.frame(test$seq_id, scale(test[, 5:26]))
colnames(test.scale)[1] = 'seq_id'
```

Make a single layer neural network model.

```
x = as.matrix(train.scale[, c(3:24)])
y = train.scale$tm

modnn3 = keras_model_sequential() %>%
  layer_dense(units = 20, activation = 'relu',
              input_shape = 22) %>%
  layer_dropout(rate = .4) %>%
  layer_dense(units = 1)
```

```
## Loaded Tensorflow version 2.5.0
```

```
modnn3 %>% compile(loss = 'mse', optimizer = optimizer_rmsprop(),
                   metrics = list('mean_absolute_error'))

history3 = modnn3 %>%
  fit(x, y, epochs = 5, validation_split = .2)
```

Validation error = 0.56.

Predict.

```
testnn = as.matrix(test.scale[, 2:23])
tm = predict(modnn3, testnn)
range(tm)
```

```
## [1] -1.366112  4.469519
```

```
modnn3.test = cbind.data.frame(test$seq_id, tm)
write.csv(modnn3.test, 'modnn3.csv')

summary(modnn3)
```

```
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## dense_1 (Dense)              (None, 20)                460
## _________________________________________________________________________
## dropout (Dropout)            (None, 20)                0
## _________________________________________________________________________
## dense (Dense)                (None, 1)                 21
## =========================================================================
## Total params: 481
## Trainable params: 481
## Non-trainable params: 0
## _________________________________________________________________________
```

Kaggle score = 0.15. As of November 9th, 2022, this puts me in 662nd place out of 1227 teams.