protein-properties

Andrew Jensen

2/29/2020

PREAMBLE

CASP
https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure#

Preparation

- This data was pretty much quantitative so I was able to import it as is.

Column info
RMSD - Size of the residue.
F1 - Total surface area.
F2 - Non polar exposed area.
F3 - Fractional area of exposed non polar residue.
F4 - Fractional area of exposed non polar part of residue.
F5 - Molecular mass weighted exposed area.
F6 - Average deviation from standard exposed area of residue.
F7 - Euclidian distance.
F8 - Secondary structure penalty.
F9 - Spacial Distribution constraints (N,K Value).

dont forget to set working directory

```
casp <- read.csv(file = 'CASP.csv')
summary(casp)
```

```
##       RMSD              F1              F2               F3
##  Min.   : 0.000   Min.   : 2392   Min.   :  403.5   Min.   :0.0925
##  1st Qu.: 2.305   1st Qu.: 6937   1st Qu.: 1979.0   1st Qu.:0.2587
##  Median : 5.030   Median : 8899   Median : 2668.2   Median :0.3001
##  Mean   : 7.749   Mean   : 9872   Mean   : 3017.4   Mean   :0.3024
##  3rd Qu.:13.379   3rd Qu.:12126   3rd Qu.: 3786.4   3rd Qu.:0.3429
##  Max.   :20.999   Max.   :40035   Max.   :15312.0   Max.   :0.5777
##        F4               F5               F6               F7
##  Min.   : 10.31   Min.   : 319490   Min.   : 31.97   Min.   :     0
##  1st Qu.: 63.56   1st Qu.: 953591   1st Qu.: 94.76   1st Qu.:  3165
##  Median : 87.74   Median :1237219   Median :126.18   Median :  3840
##  Mean   :103.49   Mean   :1368299   Mean   :145.64   Mean   :  3990
##  3rd Qu.:133.65   3rd Qu.:1690920   3rd Qu.:181.47   3rd Qu.:  4644
##  Max.   :369.32   Max.   :5472011   Max.   :598.41   Max.   :105948
##        F8               F9
##  Min.   :  0.00   Min.   :15.23
```

```
##  1st Qu.: 31.00   1st Qu.:30.42
##  Median : 54.00   Median :35.30
##  Mean   : 69.98   Mean   :34.52
##  3rd Qu.: 91.00   3rd Qu.:38.87
##  Max.   :350.00   Max.   :55.30
```

```
# pairs(casp) takes too long

spec <- c(train = .6, test = .2, validate = .2)
casp_df <- sample(cut(seq(nrow(casp)), nrow(casp)*cumsum(c(0,spec)), labels = names(spec)), replace = TR

casp_set <- split(casp, casp_df)
```

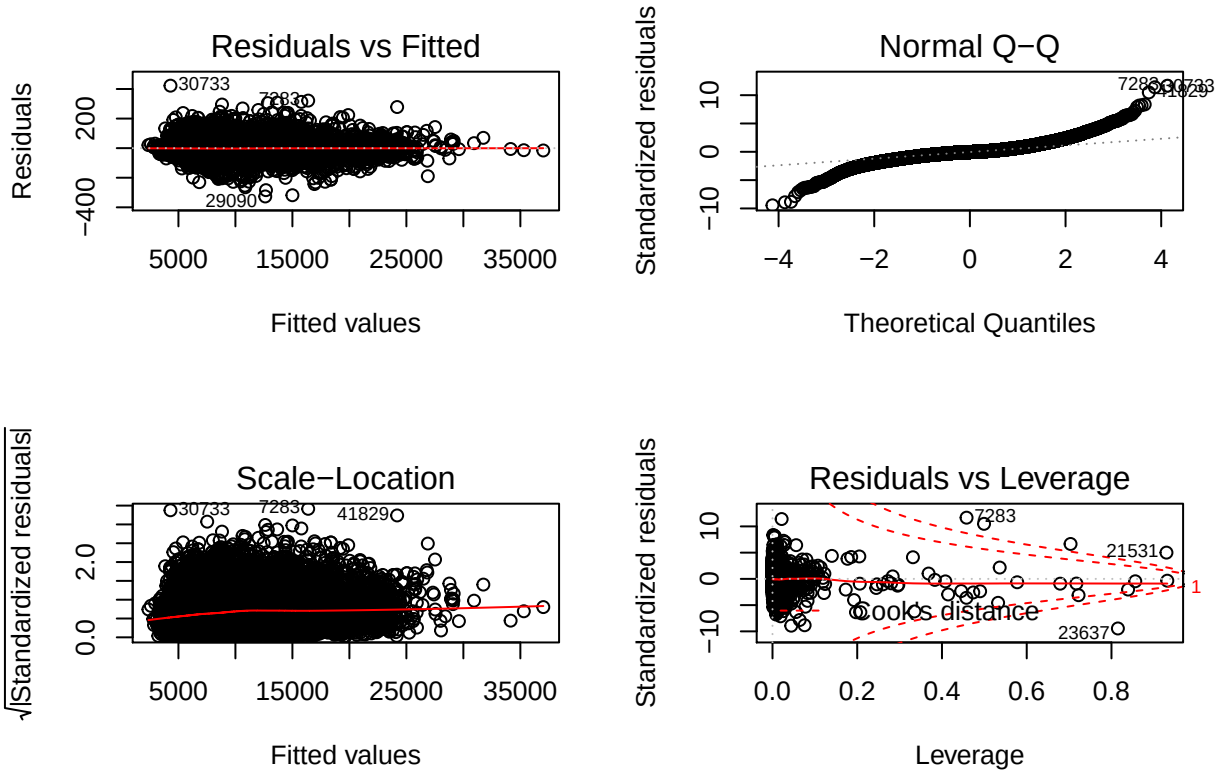
Linear Regression

```
lm0 <- lm(F1~F2*F3*F5*F6*F7*F8, data = casp_set$train)
summary(lm0)
```

```
##
## Call:
## lm(formula = F1 ~ F2 * F3 * F5 * F6 * F7 * F8, data = casp_set$train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -326.58  -16.21   -1.78   13.35  422.50
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.346e+02  4.315e+01   3.119  0.00181 **
## F2                  3.614e+00  4.501e-02  80.309  < 2e-16 ***
## F3                 -8.030e+02  1.392e+02  -5.767 8.14e-09 ***
## F5                  6.498e-03  8.639e-05  75.214  < 2e-16 ***
## F6                  1.647e+01  1.021e+00  16.128  < 2e-16 ***
## F7                 -2.267e-02  1.232e-02  -1.840  0.06571 .
## F8                 -6.151e+00  5.948e-01 -10.342  < 2e-16 ***
## F2:F3               1.474e+00  1.030e-01  14.318  < 2e-16 ***
## F2:F5              -2.020e-08  3.165e-08  -0.638  0.52336
## F3:F5              -2.580e-02  3.556e-04 -72.543  < 2e-16 ***
## F2:F6              -4.456e-03  3.237e-04 -13.765  < 2e-16 ***
## F3:F6              -5.943e+01  3.420e+00 -17.376  < 2e-16 ***
## F5:F6              -6.066e-06  3.836e-07 -15.812  < 2e-16 ***
## F2:F7               2.253e-06  1.078e-05   0.209  0.83451
## F3:F7               1.975e-01  4.133e-02   4.780 1.76e-06 ***
## F5:F7               7.894e-08  2.504e-08   3.152  0.00162 **
## F6:F7              -1.654e-03  2.431e-04  -6.804 1.04e-11 ***
## F2:F8              -2.857e-03  6.237e-04  -4.580 4.66e-06 ***
## F3:F8               2.171e+01  1.967e+00  11.040  < 2e-16 ***
## F5:F8               4.815e-06  1.051e-06   4.583 4.61e-06 ***
## F6:F8              -9.670e-02  1.188e-02  -8.138 4.20e-16 ***
## F7:F8               9.271e-04  1.542e-04   6.013 1.84e-09 ***
## F2:F3:F5           -2.934e-07  9.460e-08  -3.102  0.00193 **
## F2:F3:F6           -4.214e-03  7.532e-04  -5.595 2.22e-08 ***
## F2:F5:F6            7.739e-11  7.828e-11   0.989  0.32281
```

```
## F3:F5:F6           6.602e-05  1.977e-06  33.389  < 2e-16 ***
## F2:F3:F7          -1.703e-04  2.751e-05  -6.189 6.13e-10 ***
## F2:F5:F7          -1.402e-11  7.380e-12  -1.900  0.05742 .
## F3:F5:F7          -9.088e-08  9.927e-08  -0.915  0.35995
## F2:F6:F7           3.387e-07  6.622e-08   5.115 3.15e-07 ***
## F3:F6:F7           5.408e-03  8.181e-04   6.611 3.90e-11 ***
## F5:F6:F7           5.900e-10  8.073e-11   7.308 2.78e-13 ***
## F2:F3:F8          -2.411e-02  1.344e-03 -17.940  < 2e-16 ***
## F2:F5:F8          -1.902e-09  3.094e-10  -6.147 8.01e-10 ***
## F3:F5:F8           5.183e-05  3.829e-06  13.535  < 2e-16 ***
## F2:F6:F8           2.872e-05  3.169e-06   9.062  < 2e-16 ***
## F3:F6:F8           3.751e-01  3.832e-02   9.789  < 2e-16 ***
## F5:F6:F8           3.615e-08  4.149e-09   8.713  < 2e-16 ***
## F2:F7:F8           1.007e-07  1.190e-07   0.846  0.39736
## F3:F7:F8          -3.969e-03  5.187e-04  -7.652 2.04e-14 ***
## F5:F7:F8          -9.963e-10  2.500e-10  -3.986 6.75e-05 ***
## F6:F7:F8           1.661e-05  2.565e-06   6.474 9.69e-11 ***
## F2:F3:F5:F6       -1.226e-10  2.188e-10  -0.561  0.57509
## F2:F3:F5:F7        9.128e-11  2.231e-11   4.092 4.30e-05 ***
## F2:F3:F6:F7        5.290e-08  1.609e-07   0.329  0.74237
## F2:F5:F6:F7       -9.411e-15  1.455e-14  -0.647  0.51762
## F3:F5:F6:F7       -4.590e-09  3.972e-10 -11.556  < 2e-16 ***
## F2:F3:F5:F8        8.253e-09  9.387e-10   8.792  < 2e-16 ***
## F2:F3:F6:F8        5.139e-05  7.000e-06   7.342 2.16e-13 ***
## F2:F5:F6:F8        4.371e-12  6.528e-13   6.695 2.19e-11 ***
## F3:F5:F6:F8       -4.877e-07  1.835e-08 -26.580  < 2e-16 ***
## F2:F3:F7:F8        3.714e-06  3.011e-07  12.334  < 2e-16 ***
## F2:F5:F7:F8        4.047e-13  6.358e-14   6.366 1.97e-10 ***
## F3:F5:F7:F8       -4.439e-09  8.678e-10  -5.115 3.16e-07 ***
## F2:F6:F7:F8       -3.602e-09  5.975e-10  -6.029 1.67e-09 ***
## F3:F6:F7:F8       -5.694e-05  8.170e-06  -6.969 3.26e-12 ***
## F5:F6:F7:F8       -4.677e-12  8.018e-13  -5.833 5.51e-09 ***
## F2:F3:F5:F6:F7    -1.783e-14  4.068e-14  -0.438  0.66123
## F2:F3:F5:F6:F8    -1.324e-11  1.852e-12  -7.147 9.07e-13 ***
## F2:F3:F5:F7:F8    -1.622e-12  1.942e-13  -8.350  < 2e-16 ***
## F2:F3:F6:F7:F8    -4.419e-09  1.391e-09  -3.177  0.00149 **
## F2:F5:F6:F7:F8    -6.053e-16  1.128e-16  -5.366 8.12e-08 ***
## F3:F5:F6:F7:F8     5.436e-11  3.292e-12  16.511  < 2e-16 ***
## F2:F3:F5:F6:F7:F8  2.071e-15  3.137e-16   6.601 4.16e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.46 on 27215 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 5.067e+06 on 63 and 27215 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2,2))
plot(lm0)
```




```
pred <- predict(lm0, newdata=casp_set$test)
mean((pred - casp_set$test$F1)^2)
```

```
## [1] 1523.905
```

```
cor(pred, casp_set$test$F1)
```

```
## [1] 0.9999546
```

K - nearest neighbor

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
kfit1 <- knnreg(casp_set$train[,-2], casp_set$train[,2], k=3, prob = TRUE)
res1 <- predict(kfit1, casp_set$test[,-2])
cor_knn1 <- cor(res1, casp_set$test$F1)
mse_knn1 <- mean((res1 - casp_set$test$F1)^2)

tr_scaled <- casp_set$train[,-2]
means <- sapply(tr_scaled, mean)
stdvs <- sapply(tr_scaled, sd)
tr_scaled <- scale(casp_set$train[,-2], center = means, scale = stdvs)
ts_scaled <- scale(casp_set$test[,-2], center = means, scale = stdvs)
kfit2 <- knnreg(tr_scaled, casp_set$train[,2], k=5, prob = TRUE)
res2 <- predict(kfit2, ts_scaled)
cor_knn2 <- cor(res2, casp_set$test[,2])
```

```
mse_knn2 <- mean((res2 - casp_set$test[,2])^2)

cor_knn1
```

```
## [1] 0.9978489
```

```
mse_knn1
```

```
## [1] 72126.92
```

```
cor_knn2
```

```
## [1] 0.9963542
```

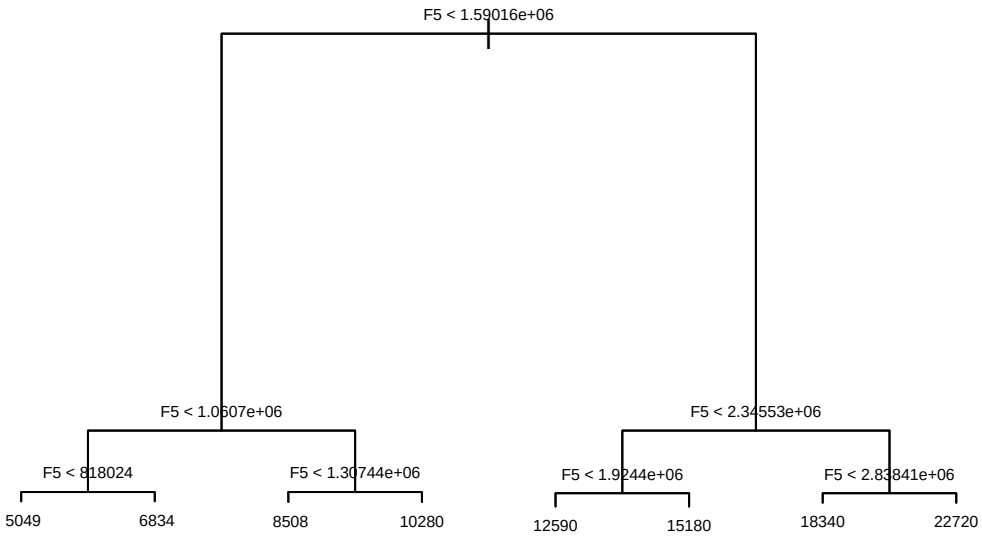
```
mse_knn2
```

```
## [1] 124236.3
```

Scaling the data actually produced a worse model and close to doubled the mean standard error.

Decision Tree

```
library(tree)
tree_casp <- tree(F1~., data=casp)
plot(tree_casp)
text(tree_casp, cex=0.5, pretty=1)
```




```
tree_pred1 <- predict(tree_casp, newdata=casp_set$test)
cor(tree_pred1, casp_set$test$F1)
```

```
## [1] 0.9833599
```

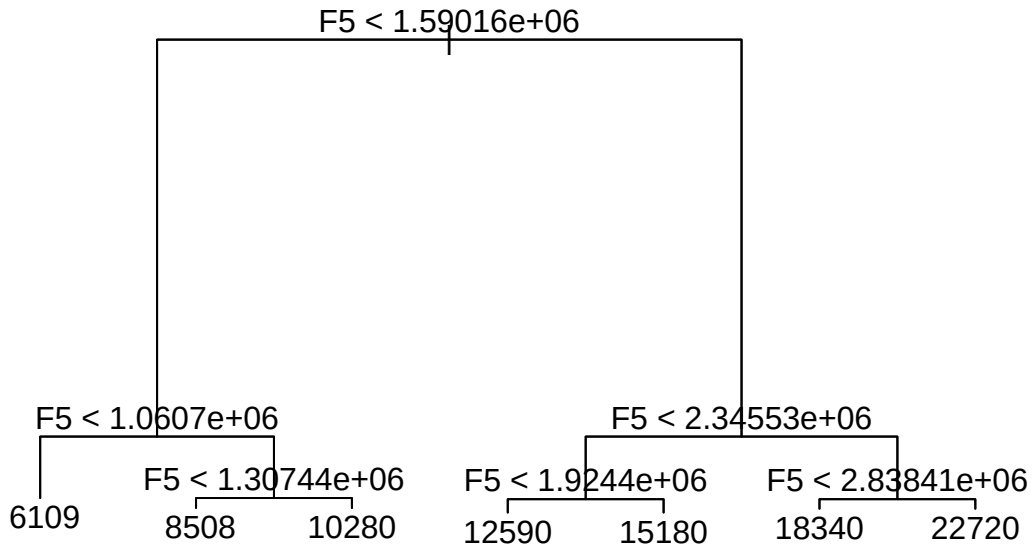
```
sqrt(mean((tree_pred1 - casp_set$test$F1)^2))
```

```
## [1] 744.0433
```

```
mean((tree_pred1 - casp_set$test$F1)^2)
```

```
## [1] 553600.4
```

```
tree_pruned <- prune.tree(tree_casp, best=7)
plot(tree_pruned)
text(tree_pruned, pretty=0)
```




```
pred_tree_pruned <- predict(tree_pruned, newdata=casp_set$test)
cor(pred_tree_pruned, casp_set$test$F1)
```

```
## [1] 0.9754188
```

```
sqrt(mean((pred_tree_pruned - casp_set$test$F1)^2))
```

```
## [1] 902.4871
```

```
mean((pred_tree_pruned - casp_set$test$F1)^2)
```

```
## [1] 814482.9
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1234)
rf <- randomForest(F1~F2+F4+F5+F6+F8, data=casp_set$train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = F1 ~ F2 + F4 + F5 + F6 + F8, data = casp_set$train,      importance = TRUE)
```

```
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 60203.57
##                     % Var explained: 99.63
```

```
pred_rf <- predict(rf, newdata=casp_set$test)
cor_rf <- cor(pred_rf, casp_set$test$F1)
print(paste('corr:', cor_rf))
```

```
## [1] "corr: 0.997936625288098"
```

```
rmse_rf <- sqrt(mean((pred_rf-casp_set$test$F1)^2))
print(paste('rmse:', rmse_rf))
```

```
## [1] "rmse: 265.595654162798"
```

RESULTS

Algorithms ranked

1. **Linear Regression** - Corr:0.9999492 MSE:1705.029

- Produced great correlations for many predictor combinations, especially the simpler the set.
- Quickest implementation without costing too much runtime.
- Reasonably low MSE given the data.

2. **KNN** - Corr:0.9978489 MSE:72126.92

- Gave relatively good results and easier implementation using the caret version of this algorithm.
- Like clarified earlier, every predictor is significant, so scaling was not productive.
- Very high MSE, closer to twice as worse after pruning.
- Still seems ok overall.

3. **Decision Tree** - Corr:0.9833599 MSE:553600.4

- Correlation and MSE are the worst of the three.
- Like clarified earlier, every predictor is significant, so pruning was not productive.
- Very high MSE, closer to twice as worse after pruning.

ENSAMBLE METHOD

Random forest - Corr:0.999028410740686 MSE:26218 - Easier to implement than xgboost - Extremely long execution time (Ive tried a few groups of predictors) - Mean standard error is in the middle but not spectacular compared to LinReg - Better correlation than most others. It would be second to linear regressions

ANALYSIS

Information was very sparse on this data set. However, its obvious how each predictor is predictive of itself and all others, showing some and statistical ubiquity of protein despite physical variation. Data outside the predictive capabilities could be seen as malformed anomolies or extreme case data.