Artificial Neural Network Application for Reservoir Characterization

Chukwuemeka Okoli

April, 09, 2019

Contents

Introduction

As a Data Scientist in the Oil and Gas Industry, we are often faced with numerous challenges. For instance, with available Gas production data for an unconventional reservoir, can we establish a relationship between several independent variable in the production process and the dependent variable "Gas production per months" using Artificial Neural Network. In this R Markdown project, an illustration of how we can apply Artificial Neural Network to Gas production is discussed. The dataset for this project is the "Q4_data.csv" file.

The Goal

Find a relationship between the dependent variable "Gas production per months" and several independent variables. Essentially, we want to determine the Gas Production per month for the given well data based on different factors.

Preliminaries

First, the required packages are installed using the `install.packages()` function.

```r
# install knitr
install.packages("knitr")

# install the rmarkdown package
install.packages("rmarkdown")

# install ggplot for plotting
install.packages("ggplot2")

# install dplyr for data manipulation
install.packages("dplyr")

# install neuralnet for artificial neural network
```

```r
install.packages("neuralnet")
install.packages("NeuralNetTools")
```

The installed package is then loaded using the `library()` function.

```r
# load libraries
library(knitr)
library(rmarkdown)
library(ggplot2)
library(dplyr)
library(neuralnet)
library(NeuralNetTools)
library(xfun)
```

Data Import

We can call in the dataset into R using the code below. The data is stored in a `.csv` file.

```r
set.seed(440)
data <- read.csv("Q4_data.csv", header=TRUE)
```

Summaries

The data object `field_data` is complex. It contains various information about properties of the well such as the perforation interval, fracture volume, proppant, casing depth, etc. Use `summary()` to get a quick summary of the data.

```r
summary(data)
```

```
##     Perf_Int         Frac_vol          Proppant           N.frac
##  Min.   :  12.0   Min.   :   7478   Min.   :  43500   Min.   : 1.000
##  1st Qu.: 381.8   1st Qu.:  29524   1st Qu.: 164625   1st Qu.: 1.000
##  Median : 474.0   Median :  36200   Median : 296946   Median : 2.000
##  Mean   : 708.1   Mean   :  56848   Mean   : 662449   Mean   : 2.309
##  3rd Qu.: 802.5   3rd Qu.:  56876   3rd Qu.:1135250   3rd Qu.: 3.000
##  Max.   :4555.0   Max.   :3637776   Max.   :4437897   Max.   :11.000
##   Tubing.depth    Casing.depth        FTP             Choke.Size
##  Min.   : 4952   Min.   : 7325   Min.   :   0.0   Min.   :  6.00
##  1st Qu.: 7652   1st Qu.: 8176   1st Qu.: 450.0   1st Qu.: 14.00
##  Median : 8002   Median : 8745   Median : 814.1   Median : 20.00
##  Mean   : 8020   Mean   : 9044   Mean   : 905.6   Mean   : 28.18
##  3rd Qu.: 8407   3rd Qu.: 9694   3rd Qu.:1265.0   3rd Qu.: 32.00
##  Max.   :12100   Max.   :13144   Max.   :2964.0   Max.   :128.00
##      SITHP          SG_gas          Well_Type        Latitude
##  Min.   : 300   Min.   :0.5810   Min.   :1.000   Min.   :      33
##  1st Qu.:1615   1st Qu.:0.7000   1st Qu.:2.000   1st Qu.:      33
##  Median :2278   Median :0.7000   Median :3.000   Median :      33
##  Mean   :2250   Mean   :0.6922   Mean   :2.504   Mean   :    7737
##  3rd Qu.:2913   3rd Qu.:0.7000   3rd Qu.:3.000   3rd Qu.:      33
##  Max.   :3715   Max.   :1.0320   Max.   :3.000   Max.   :3436128
##      Long.           Gas_prod             Acid
##  Min.   :-97.40   Min.   :     95   Min.   :1.000
##  1st Qu.:-97.35   1st Qu.:   3840   1st Qu.:1.000
##  Median :-97.31   Median :   6392   Median :1.000
##  Mean   :-97.00   Mean   :  13281   Mean   :1.193
##  3rd Qu.:-97.26   3rd Qu.:  13155   3rd Qu.:1.000
```

```
##  Max.   : 33.04   Max.   :1189790   Max.   :2.000
```

The `set.seed()` function is useful when running simulations to ensure all results, figures, etc. are reproducible. We can then check that no data point is missing. If we have a missing data point, we need to fix the dataset.

```r
# Check that no data is missing
apply(data,2,function(x) sum(is.na(x)))
```

```
##     Perf_Int     Frac_vol     Proppant       N.frac Tubing.depth Casing.depth
##            0            0            0            0            0            0
##          FTP   Choke.Size        SITHP       SG_gas    Well_Type     Latitude
##            0            0            0            0            0            0
##        Long.     Gas_prod         Acid
##            0            0            0
```

We can see that there is no missing data. In the case where missing data exist, data cleaning will have to take place.

Data Analysis

To analyze the data, we first proceed by randomly splitting the data into a *train* and a *test* dataset. A linear regression model is fit to the train dataset, and tested on the test dataset.

```r
# Train-test random splitting for linear model
index <- sample(1:nrow(data), round(0.75*nrow(data)))
train <- data[index,]
test <- data[-index,]

# Fitting linear model to the train dataset
lm.fit <- lm(Gas_prod~., data=train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Gas_prod ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -47265  -10594   -1262    6029 1130490
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.242e+06  6.496e+06   0.653  0.51423
## Perf_Int      7.347e+00  9.449e+00   0.778  0.43739
## Frac_vol     -2.151e-03  1.828e-02  -0.118  0.90643
## Proppant     -2.850e-03  8.633e-03  -0.330  0.74148
## N.frac        3.468e+03  3.381e+03   1.026  0.30573
## Tubing.depth  2.724e+00  7.987e+00   0.341  0.73332
## Casing.depth -4.627e+00  7.114e+00  -0.650  0.51589
## FTP           1.334e+01  7.556e+00   1.766  0.07839 .
## Choke.Size   -1.336e+02  1.828e+02  -0.731  0.46519
## SITHP        -1.010e+01  5.776e+00  -1.749  0.08126 .
## SG_gas        2.025e+04  8.302e+04   0.244  0.80744
## Well_Type    -1.712e+04  5.909e+03  -2.897  0.00402 **
## Latitude     -1.619e+00  2.527e+00  -0.641  0.52226
## Long.         4.283e+04  6.657e+04   0.643  0.52049
## Acid         -7.482e+03  9.953e+03  -0.752  0.45275
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 64930 on 319 degrees of freedom
## Multiple R-squared:  0.05826,    Adjusted R-squared:  0.01693
## F-statistic:  1.41 on 14 and 319 DF,  p-value: 0.1465
```

```r
# Predicted data from linear model fit
pr.lm <- predict(lm.fit, test)

# Test MSE
MSE.lm <- sum((pr.lm - test$Gas_prod)^2)/nrow(test)
```

Before fitting a neural network, some preparation needs to be done. Neural networks are not that easy to train and tune. As a first step, we need to address data preprocessing. It is good practice to normalize your data before training a neural network. This step is important because, depending on your dataset, avoiding normalization may lead to useless results or to a very difficult training process (most of the time, the algorithm will not converge before the number of maximum iterations allowed). You can choose different methods to scale the data (z-normalization, min-max scale, etc.). The data were scaled using the min-max method and scaled in the interval [0,1]. We therefore scale and split the data before moving on.

```r
# Neural network fitting

# Scaling data for the Neural Network
maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)
scaled <- as.data.frame(scale(data, center = mins, scale = maxs - mins))

# Train-test split
train_ <- scaled[index,]
test_ <- scaled[-index,]
```

Since there is no fixed rule as to how many layers and neurons to use, we are going to use two (2) hidden layers with 5 and 3 neurons. To fit the network, we use the following code:

```r
# Training the neural network
n <- names(train_)
f <- as.formula(paste("Gas_prod ~", paste(n[!n %in% "Gas_prod"], collapse = " + ")))
neural_net <- neuralnet(f, data = train_, hidden = c(5,3), linear.output = T)
```

Note that the hidden argument accepts a vector with the number of neurons for each hidden layer, while the argument linear.output is used to specify whether we want to do regression **linear.output = TRUE** or **classification linear.output = FALSE**

The neuralnet package provides a nice tool to plot the model. Use the following code to plot the neural network in R:

```r
# Visual plot of the model
plot(neural_net)
```

Now we can try to predict the values for the test set and calculate the mean squared error (MSE). Remember that the net will output a normalized prediction, so we need to scale it back in order to make a meaningful comparison (or just a simple prediction). The mean squared error is one metric used to measure prediction accuracy. The MSE is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i}^2)$$

```r
# Prediction
pr.nn <- compute(neural_net, test_[,1:14])

# Results from Neural Networks are normalized (scaled)
# Descaling for comparison
pr.nn_ <- pr.nn$net.result*(max(data$Gas_prod)-min(data$Gas_prod))+min(data$Gas_prod)
test.r <- (test_$Gas_prod)*(max(data$Gas_prod)-min(data$Gas_prod))+min(data$Gas_prod)

# Calculating MSE
MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test_)
```

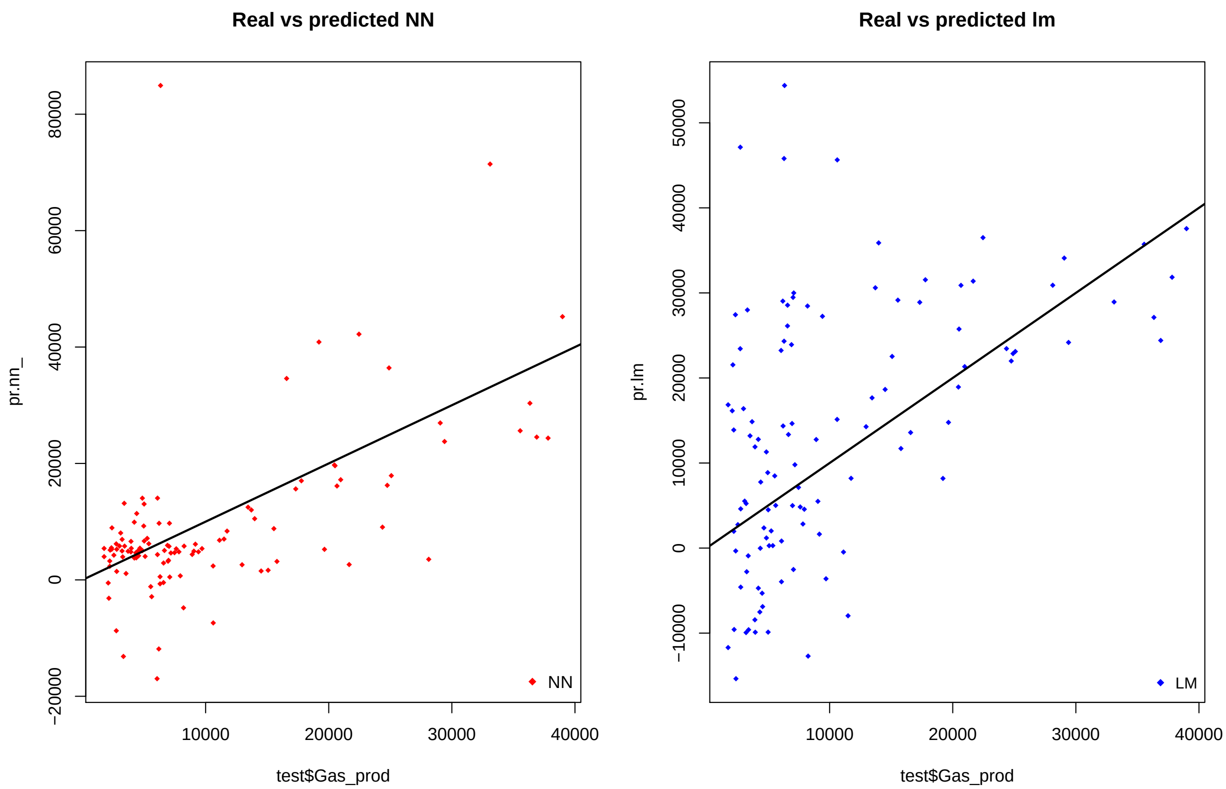
We then compare the two mean squares error obtained from the linear regression and the neural network fitting process using the code:

```r
# Compare the two Mean Squared Errors (MSEs)
print(paste(MSE.lm, MSE.nn))
```

```
## [1] "183730606.293678 132133926.405935"
```

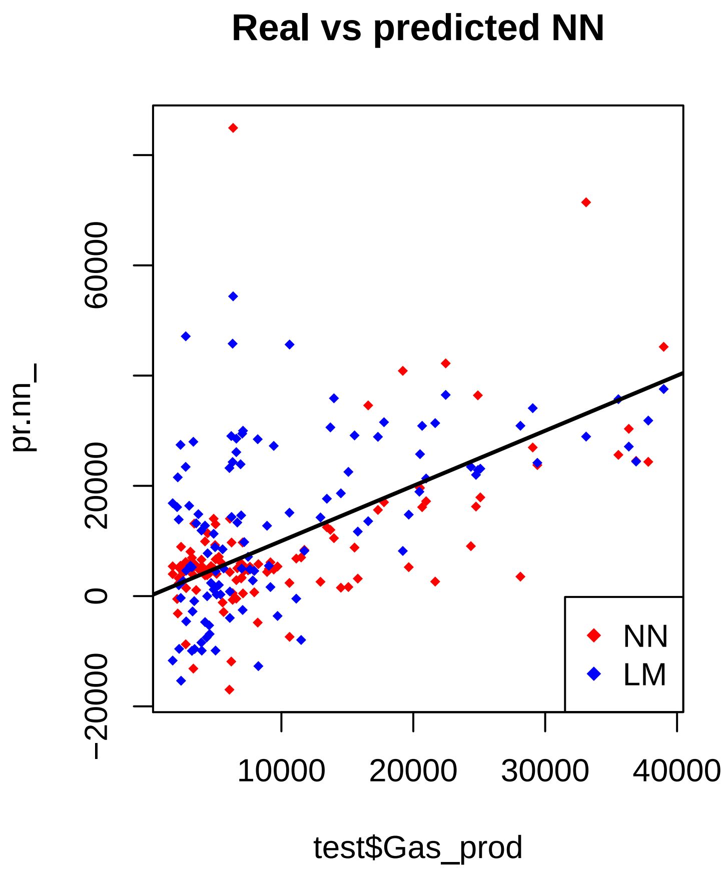
This shows that the network is doing a better job at predicting Gas_prod than the linear model. The prediction from the neural network is **129413554.366771** which is better than the **229551018.949411** obtained from the linear model. We can perform a fast cross visualization in order to be more confident of the result. A visual approach to the performance of the network and the linear model is plotted below.

```r
# Plot predictions
par(mfrow=c(1,2))

plot(test$Gas_prod, pr.nn_, col='red', main='Real vs predicted NN', pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright', legend='NN', pch=18, col='red', bty='n')

plot(test$Gas_prod, pr.lm, col='blue', main='Real vs predicted lm', pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright', legend='LM', pch=18, col='blue', bty='n', cex=.95)
```



By visually inspecting the plot, we can see that the predictions made by the neural network are more concentrated around the line than those made by the linear model. We can obtain a more useful visual comparison using the code below

```
# Compare predictions on the same plot
plot(test$Gas_prod, pr.nn_, col='red', main='Real vs predicted NN', pch=18, cex=0.7)
points(test$Gas_prod, pr.lm, col='blue', pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright', legend=c('NN','LM'), pch=18, col=c('red','blue'))
```



The comparison of prediction on the same plot is shown above.In analyzing the model above, we allowed the network select the training, validation and testing dataset itself by randomly splitting the data into a train and a test set, then a linear regression model is fit and then tested on the test set.

If we choose to select the training, validation and testing datasets ourselves, we normalize our data and split into training and test data as before.

```
# Neural net fitting
set.seed(440)
mydata <- read.csv("Q4_data.csv", header=TRUE)

#Normalizing the dataset
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

maxmindf <- as.data.frame(lapply(mydata, normalize))
```

```
# Display head of data
head(mydata)
```

```
##   Perf_Int Frac_vol Proppant N.frac Tubing.depth Casing.depth   FTP Choke.Size
## 1      280 19454.76   408550      2         7754         8067 185.0         42
## 2      757 58477.00   407000      1         8843         9081 490.0         40
## 3      532 27262.00   129400      1         7963         8515 565.0         22
```

```
## 4      345 31622.74   321300      1          7832          8039  615.0          20
## 5      856 18596.76   390532      2          8602          9065  715.0          24
## 6      470 34785.71   190000      1          7470          7810 1235.2          10
##   SITHP SG_gas Well_Type Latitude     Long. Gas_prod Acid
## 1  2815 0.6790         1 33.05307 -97.19147  7135.18    1
## 2  1915 0.5890         1 33.16222 -97.38986  3720.74    1
## 3  3015 0.7000         1 33.01159 -97.27166  6200.33    1
## 4  2350 0.8000         1 33.14988 -97.14882  4096.05    1
## 5  3250 0.6621         1 33.04807 -97.27381  4687.49    1
## 6  2250 0.7000         1 33.23879 -97.30187  3551.04    1
```

```r
# Display tail of data
tail(mydata)
```

```
##     Perf_Int Frac_vol Proppant N.frac Tubing.depth Casing.depth  FTP Choke.Size
## 441      718 52929.00   701000      1         8851         8869  700         32
## 442      575 28945.00   465800      2         8115         8532  680         24
## 443      786 52379.00   352572      3         8426         9151 1365         20
## 444      740 23752.00   203500      2         8149         9200  439         18
## 445      702 24676.00   294765      2         8484         9245 1365         14
## 446      372 28928.57   115250      1         7370         7644 1500          9
##     SITHP SG_gas Well_Type Latitude     Long.  Gas_prod Acid
## 441  2200   0.60         3 33.14809 -97.16659 5882.4167    2
## 442  2450   0.64         3 33.19194 -97.25327 5013.8594    1
## 443  2910   0.70         3 33.25706 -97.21546 1226.4308    2
## 444  2714   0.72         3 33.24925 -97.20252  887.2319    1
## 445  3070   0.70         3 33.25470 -97.20097 4863.2192    1
## 446  2865   0.70         3 33.01804 -97.31887 2798.9756    1
```

Since we know the size of the data, we then proceed to test and train the network. We have chosen to split the training and testing dataset into a 70/30 ratio .i.e. 70% (for training) and 30% (for testing) instead of allowing the network do the splitting for us.

```r
# Training and Test Data
trainset <- maxmindf[1:312, ]
testset <- maxmindf[313:446, ]
```

We fit the network and arbitrarily decide on the number of hidden neurons. Deciding on the number of hidden layers in a neural network is not an exact science. In fact, there are instances where accuracy will likely be higher without any hidden layers. Therefore, trial and error plays a significant role in this process. One possibility is to compare how the accuracy of the predictions change as we modify the number of hidden layers. If we use 3 and 2 hidden layers, we obtain:

```r
# Neural Network Training
library(neuralnet)
nn <- neuralnet(Gas_prod ~ Perf_Int + Frac_vol + Proppant + N.frac +
                Tubing.depth + Casing.depth + FTP + Choke.Size + SITHP +
                SG_gas + Well_Type + Latitude + Long. + Acid, data=trainset,
                hidden=c(3,2), linear.output=TRUE, threshold=0.01)
nn$result.matrix

# Visual plot of the model
plot(nn)
```

The output from the neural network training is shown above. Our neural network has been created using the training data. We then compare this to the test data to gauge the accuracy of the neural network forecast.

```r
#Test the resulting output
temp_test <- subset(testset, select = c("Gas_prod", "Perf_Int", "Frac_vol",
                                        "Proppant", "N.frac", "Tubing.depth",
                                        "Casing.depth",  "FTP", "Choke.Size",
                                        "SITHP", "SG_gas", "Well_Type", "Latitude",
                                        "Long.", "Acid"))
head(temp_test)
```

```
##         Gas_prod   Perf_Int    Frac_vol   Proppant N.frac Tubing.depth
## 313 0.0115296075 0.14902047 0.006530593 0.08508562    0.0    0.4555120
## 314 0.0045662511 0.05414924 0.005322704 0.04983619    0.2    0.4542529
## 315 0.0041483181 0.14880035 0.005769774 0.05718191    0.0    0.3940963
## 316 0.0029264959 0.08056350 0.004923563 0.01567906    0.0    0.3567431
## 317 0.0009303225 0.11182038 0.007402423 0.07036233    0.3    0.4847510
## 318 0.0042217903 0.15100154 0.003430199 0.03379303    0.1    0.4956631
##     Casing.depth        FTP Choke.Size     SITHP     SG_gas Well_Type
## 313    0.2342327 0.84480432 0.03278689 0.9411420 0.04212860         1
## 314    0.2495274 0.29159919 0.08196721 0.8533821 0.23725055         1
## 315    0.2488400 0.17948718 0.11475410 0.8752562 0.04212860         1
## 316    0.1031105 0.43657220 0.06557377 0.8096633 0.26385809         1
## 317    0.3101908 0.08434548 0.34426230 0.1317716 0.01419069         1
## 318    0.2758206 0.36099865 0.08196721 0.8067350 0.26385809         1
##         Latitude        Long. Acid
## 313 5.820619e-08 0.0009938915    0
## 314 6.377984e-08 0.0007750097    1
## 315 6.267615e-08 0.0011130577    0
## 316 2.029004e-08 0.0009770522    0
## 317 5.168088e-08 0.0017725913    0
## 318 7.107848e-08 0.0010241715    0
```

```r
nn.results <- compute(nn, temp_test)
```

The predicted results are compared to the actual results. The code to do this and a snippet of the result is shown below:

```r
#Comparison of Predicted to Actual
results <- data.frame(actual = testset$Gas_prod, prediction = nn.results$net.result)
results
```

```
##           actual    prediction
## 313 0.0115296075  0.0019106450
## 314 0.0045662511  0.0024498630
## 315 0.0041483181  0.0053135253
## 316 0.0029264959  0.0061688705
## 317 0.0009303225  0.0071988651
## 318 0.0042217903  0.0023350951
## 319 0.0052641792  0.0032005803
## 320 0.0065977434  0.0061051029
## 321 0.0033872906  0.0021572123
## 322 0.0051492189 -0.0008556349
## 323 0.0024511084  0.0028816770
## 324 0.0074102049  0.0031570602
## 325 0.0128688700  0.0066883438
## 326 0.0028917568  0.0025738101
## 327 0.0076796542  0.0070670009
```

```
## 328 0.0063919576  0.0019990207
## 329 0.0025868395  0.0033576773
## 330 0.0122347524  0.0011338745
## 331 0.0080792988  0.0022152307
## 332 0.0026730553  0.0028496391
## 333 0.0051960545  0.0020899506
## 334 0.0025265874  0.0030840613
## 335 0.0055463795  0.0101144939
## 336 0.0032587987  0.0028024665
## 337 0.0031702311  0.0033898908
## 338 0.0022246117  0.0018411395
## 339 0.0057922529  0.0034355646
## 340 0.0037300610  0.0020282848
## 341 0.0059741519  0.0056426277
## 342 0.0029149908  0.0029275215
## 343 0.0028963985  0.0023299764
## 344 0.0026221237  0.0044959766
## 345 0.0022295377  0.0025276060
## 346 0.0055147633  0.0014203938
## 347 0.0052257485  0.0012533214
## 348 0.0055185430 -0.0025373695
## 349 0.0048508995  0.0083208787
## 350 0.0202480711  0.0027120368
## 351 0.0027945065  0.0036926835
## 352 0.0042450023  0.0020766388
## 353 0.0026071469  0.0019366854
## 354 0.0028924986  0.0035374400
## 355 0.0020379466  0.0027016946
## 356 0.0032831344  0.0060492186
## 357 0.0067473376  0.0132027709
## 358 0.0063339190  0.0043717599
## 359 0.0025950512  0.0056797039
## 360 0.0043399464  0.0058699440
## 361 0.0016026216  0.0031946049
## 362 0.0020620621  0.0133968439
## 363 0.0050385813  0.0041586095
## 364 0.0039734073  0.0104244432
## 365 0.0050441210  0.0142747948
## 366 0.0066381921  0.0007017620
## 367 0.0075260401  0.0069337214
## 368 0.0050424689  0.0031700968
## 369 0.0014968031 -0.0008751567
## 370 0.0089576112  0.0029930621
## 371 0.0033743265  0.0026784136
## 372 0.0092120054 -0.0006881117
## 373 0.0061213079  0.0113017858
## 374 0.0038408854  0.0028286028
## 375 0.0051341073  0.0102053763
## 376 0.0048609221  0.0087810308
## 377 0.0032527518  0.0029781614
## 378 0.0048795788  0.0057323195
## 379 0.0018798613  0.0023678317
## 380 0.0082982237  0.0029974922
## 381 0.0064806760  0.0124950185
```

```
## 382 0.0076059858  0.0073598299
## 383 0.0013962169  0.0140567454
## 384 0.0063839266 -0.0001841956
## 385 0.0095841260  0.0054693880
## 386 0.0031402867  0.0020134217
## 387 0.0108157871  0.0056730923
## 388 0.0028230359  0.0040575689
## 389 0.0024468152  0.0020102769
## 390 0.0019184353  0.0028645098
## 391 0.0043033654  0.0047982593
## 392 0.0070424613  0.0010465482
## 393 0.0028381297  0.0034696260
## 394 0.0026265987  0.0036621758
## 395 0.0054502705  0.0033907977
## 396 0.0040325006  0.0044313539
## 397 0.0034569259  0.0153328127
## 398 0.0032441908  0.0055820766
## 399 0.0011016794  0.0106780563
## 400 0.0036218667  0.0013556403
## 401 0.0082787771  0.0058920605
## 402 0.0011750911  0.0117593425
## 403 0.0029935590  0.0042161652
## 404 0.0027938822  0.0035519214
## 405 0.0131948458  0.0072520477
## 406 0.0036116002  0.0030407259
## 407 0.0073100298  0.0081023595
## 408 0.0071470353  0.0017243764
## 409 0.0086737917  0.0003841065
## 410 0.0010922081  0.0109864318
## 411 0.0031952354  0.0008215876
## 412 0.0058242991  0.0031394397
## 413 0.0030387570  0.0025490632
## 414 0.0047543021  0.0032786154
## 415 0.0037518257  0.0047466254
## 416 0.0097896740  0.0030648271
## 417 0.0015991316  0.0022256113
## 418 0.0076269617  0.0067306381
## 419 0.0033755965  0.0069786044
## 420 0.0041155273  0.0040409716
## 421 0.0093742924  0.0084525858
## 422 0.0124304849 -0.0012568872
## 423 0.0045273122  0.0029354787
## 424 0.0073678437  0.0036896410
## 425 0.0067944282  0.0028812478
## 426 0.0056610335 -0.0001937247
## 427 0.0008516125  0.0069017708
## 428 0.0027208112  0.0017862551
## 429 0.0098302408  0.0003964174
## 430 0.0018593832  0.0078471476
## 431 0.0010102950  0.0030705402
## 432 0.0121165880  0.0022883773
## 433 0.0041011084  0.0132032088
## 434 0.0029064556  0.0103449242
## 435 0.0016188727  0.0016224234
```

```
## 436 0.0000000000  0.0085056374
## 437 0.0115888526  0.0058139393
## 438 0.0030315893  0.0029723253
## 439 0.0044476249  0.0049675545
## 440 0.0019702894  0.0034315170
## 441 0.0048646222  0.0042154551
## 442 0.0041345550  0.0027763874
## 443 0.0009510259  0.0061772055
## 444 0.0006659118  0.0028755114
## 445 0.0040079341  0.0030563107
## 446 0.0022728309  0.0064055386
```

We then test the accuracy of the model

```
#Testing The Accuracy Of The Model
Gas_prod <- trainset$Gas_prod
predicted = results$prediction * abs(diff(range(Gas_prod))) + min(Gas_prod)
actual = results$actual * abs(diff(range(Gas_prod))) + min(Gas_prod)
comparison = data.frame(predicted, actual)

deviation = ((actual-predicted)/actual)
comparison = data.frame(predicted, actual, deviation)
```

In the above code, we are converting the data back to its original format. Note that we are also converting our data back into standard values given that they were previously scaled using the max-min normalization technique.

We compute the accuracy of network with (3,2) hidden layer. An accuracy of 61.8% on a mean absolute deviation basis (i.e. the average deviation between estimated and actual Gas production per month) was obtained.

```
accuracy = 1-abs(mean(deviation))
accuracy
```

```
## [1] 0.6187621
```

You can see that we obtain a high accuracy of 61.8 % accuracy using a (3,2) hidden configuration. This is quite good, especially considering that our dependent variable is in the interval format. However, let's see if we can get it higher!

What happens if we now use a (5,2) hidden configuration in our neural network? Here is the generated output:

```
nn <- neuralnet(Gas_prod ~ Perf_Int + Frac_vol + Proppant + N.frac + Tubing.depth
                + Casing.depth + FTP + Choke.Size + SITHP + SG_gas + Well_Type
                + Latitude + Long. + Acid,data=trainset, hidden=c(5,2),
                linear.output=TRUE, threshold=0.01)
nn$result.matrix
```

```
##                                   [,1]
## error                     3.653272e-03
## reached.threshold         8.863541e-03
## steps                     1.902000e+03
## Intercept.to.1layhid1     2.412039e+00
## Perf_Int.to.1layhid1     -3.022457e+00
## Frac_vol.to.1layhid1     -2.706479e+00
## Proppant.to.1layhid1      2.138009e+00
## N.frac.to.1layhid1        6.900633e+00
```

```
## Tubing.depth.to.1layhid1  3.020498e+00
## Casing.depth.to.1layhid1 -1.606061e-01
## FTP.to.1layhid1          -3.427725e-01
## Choke.Size.to.1layhid1    2.963539e+00
## SITHP.to.1layhid1        -7.742261e-02
## SG_gas.to.1layhid1       -3.814676e-01
## Well_Type.to.1layhid1     6.916603e-01
## Latitude.to.1layhid1     -8.057662e-01
## Long..to.1layhid1        -1.101212e+00
## Acid.to.1layhid1          1.165234e+01
## Intercept.to.1layhid2    -2.018957e+00
## Perf_Int.to.1layhid2      4.331287e+00
## Frac_vol.to.1layhid2      5.084424e-01
## Proppant.to.1layhid2     -7.344221e+00
## N.frac.to.1layhid2       -3.569281e+00
## Tubing.depth.to.1layhid2 -5.241477e-01
## Casing.depth.to.1layhid2 -1.099082e+00
## FTP.to.1layhid2          -1.129401e+01
## Choke.Size.to.1layhid2    3.903377e+00
## SITHP.to.1layhid2         4.751302e+00
## SG_gas.to.1layhid2        2.653925e+00
## Well_Type.to.1layhid2     7.466284e+01
## Latitude.to.1layhid2     -1.541364e+00
## Long..to.1layhid2         9.906799e+00
## Acid.to.1layhid2          1.626440e+00
## Intercept.to.1layhid3     3.169506e+00
## Perf_Int.to.1layhid3     -8.899430e+00
## Frac_vol.to.1layhid3     -1.330377e+00
## Proppant.to.1layhid3      8.086839e+01
## N.frac.to.1layhid3       -6.402368e+00
## Tubing.depth.to.1layhid3 -4.970220e-01
## Casing.depth.to.1layhid3 -3.024804e+01
## FTP.to.1layhid3          -1.423828e+01
## Choke.Size.to.1layhid3    4.380193e+01
## SITHP.to.1layhid3         1.046698e+01
## SG_gas.to.1layhid3        9.484196e+00
## Well_Type.to.1layhid3     5.987929e+01
## Latitude.to.1layhid3      1.540507e-01
## Long..to.1layhid3        -7.249328e+01
## Acid.to.1layhid3         -8.130646e+00
## Intercept.to.1layhid4    -3.390121e+00
## Perf_Int.to.1layhid4      7.855518e+00
## Frac_vol.to.1layhid4      7.786203e-01
## Proppant.to.1layhid4     -2.010102e+01
## N.frac.to.1layhid4        6.809802e+00
## Tubing.depth.to.1layhid4  4.636598e-01
## Casing.depth.to.1layhid4  1.147943e+01
## FTP.to.1layhid4           1.591844e+01
## Choke.Size.to.1layhid4   -7.406063e+01
## SITHP.to.1layhid4        -9.272816e+00
## SG_gas.to.1layhid4       -1.074510e+01
## Well_Type.to.1layhid4    -1.392136e+01
## Latitude.to.1layhid4      2.439652e-01
## Long..to.1layhid4         1.728355e+01
```

```
## Acid.to.1layhid4         -1.436555e+02
## Intercept.to.1layhid5    -3.292692e-01
## Perf_Int.to.1layhid5      1.132030e+00
## Frac_vol.to.1layhid5     -1.340956e+01
## Proppant.to.1layhid5      3.907111e+00
## N.frac.to.1layhid5        3.751044e+00
## Tubing.depth.to.1layhid5 -2.037910e-01
## Casing.depth.to.1layhid5  9.653307e-01
## FTP.to.1layhid5           2.657104e+00
## Choke.Size.to.1layhid5   -9.004901e-01
## SITHP.to.1layhid5        -1.096745e+00
## SG_gas.to.1layhid5       -2.615459e-01
## Well_Type.to.1layhid5    -1.459861e+00
## Latitude.to.1layhid5      1.317501e+00
## Long..to.1layhid5        -2.295573e+00
## Acid.to.1layhid5         -4.290123e-01
## Intercept.to.2layhid1     4.378251e-02
## 1layhid1.to.2layhid1     -6.734787e+00
## 1layhid2.to.2layhid1     -9.150114e-01
## 1layhid3.to.2layhid1      1.619814e+01
## 1layhid4.to.2layhid1     -5.075946e+01
## 1layhid5.to.2layhid1     -5.584953e+00
## Intercept.to.2layhid2    -9.768579e-01
## 1layhid1.to.2layhid2      3.356849e+00
## 1layhid2.to.2layhid2     -1.576312e-01
## 1layhid3.to.2layhid2      4.974827e-01
## 1layhid4.to.2layhid2     -5.260432e+01
## 1layhid5.to.2layhid2     -2.530036e-01
## Intercept.to.Gas_prod     9.835952e-01
## 2layhid1.to.Gas_prod     -5.681484e-01
## 2layhid2.to.Gas_prod     -4.410568e-01
```

```
plot(nn)
```

```
#Test the resulting output
temp_test <- subset(testset, select = c("Gas_prod", "Perf_Int", "Frac_vol",
                                        "Proppant", "N.frac", "Tubing.depth",
                                        "Casing.depth",  "FTP", "Choke.Size",
                                        "SITHP", "SG_gas", "Well_Type", "Latitude",
                                        "Long.", "Acid"))
head(temp_test)
```

```
##         Gas_prod   Perf_Int    Frac_vol   Proppant N.frac Tubing.depth
## 313 0.0115296075 0.14902047 0.006530593 0.08508562    0.0    0.4555120
## 314 0.0045662511 0.05414924 0.005322704 0.04983619    0.2    0.4542529
## 315 0.0041483181 0.14880035 0.005769774 0.05718191    0.0    0.3940963
## 316 0.0029264959 0.08056350 0.004923563 0.01567906    0.0    0.3567431
## 317 0.0009303225 0.11182038 0.007402423 0.07036233    0.3    0.4847510
## 318 0.0042217903 0.15100154 0.003430199 0.03379303    0.1    0.4956631
##     Casing.depth        FTP Choke.Size     SITHP     SG_gas Well_Type
## 313    0.2342327 0.84480432 0.03278689 0.9411420 0.04212860         1
## 314    0.2495274 0.29159919 0.08196721 0.8533821 0.23725055         1
## 315    0.2488400 0.17948718 0.11475410 0.8752562 0.04212860         1
## 316    0.1031105 0.43657220 0.06557377 0.8096633 0.26385809         1
## 317    0.3101908 0.08434548 0.34426230 0.1317716 0.01419069         1
```

```
## 318    0.2758206 0.36099865 0.08196721 0.8067350 0.26385809         1
##         Latitude        Long. Acid
## 313 5.820619e-08 0.0009938915    0
## 314 6.377984e-08 0.0007750097    1
## 315 6.267615e-08 0.0011130577    0
## 316 2.029004e-08 0.0009770522    0
## 317 5.168088e-08 0.0017725913    0
## 318 7.107848e-08 0.0010241715    0
```

```r
nn.results <- compute(nn, temp_test)

results <- data.frame(actual = testset$Gas_prod, prediction = nn.results$net.result)
results
```

```
##           actual  prediction
## 313 0.0115296075 0.008300917
## 314 0.0045662511 0.003234412
## 315 0.0041483181 0.004158014
## 316 0.0029264959 0.005043247
## 317 0.0009303225 0.005416683
## 318 0.0042217903 0.004526156
## 319 0.0052641792 0.006067041
## 320 0.0065977434 0.004446578
## 321 0.0033872906 0.003426161
## 322 0.0051492189 0.006288651
## 323 0.0024511084 0.003069765
## 324 0.0074102049 0.006491783
## 325 0.0128688700 0.004747918
## 326 0.0028917568 0.003669933
## 327 0.0076796542 0.004852330
## 328 0.0063919576 0.004888599
## 329 0.0025868395 0.003139192
## 330 0.0122347524 0.007153281
## 331 0.0080792988 0.003405139
## 332 0.0026730553 0.003052315
## 333 0.0051960545 0.003472151
## 334 0.0025265874 0.004468474
## 335 0.0055463795 0.005473454
## 336 0.0032587987 0.004468865
## 337 0.0031702311 0.003823624
## 338 0.0022246117 0.006047512
## 339 0.0057922529 0.003990564
## 340 0.0037300610 0.003230375
## 341 0.0059741519 0.005444103
## 342 0.0029149908 0.003086980
## 343 0.0028963985 0.003280721
## 344 0.0026221237 0.003584792
## 345 0.0022295377 0.005236458
## 346 0.0055147633 0.005324590
## 347 0.0052257485 0.004506027
## 348 0.0055185430 0.007202385
## 349 0.0048508995 0.005307475
## 350 0.0202480711 0.009255940
## 351 0.0027945065 0.003404355
## 352 0.0042450023 0.004465126
```

```
## 353 0.0026071469 0.006823074
## 354 0.0028924986 0.004499221
## 355 0.0020379466 0.006769294
## 356 0.0032831344 0.003416402
## 357 0.0067473376 0.004588043
## 358 0.0063339190 0.003569944
## 359 0.0025950512 0.003234358
## 360 0.0043399464 0.003710581
## 361 0.0016026216 0.003513286
## 362 0.0020620621 0.003623415
## 363 0.0050385813 0.003436350
## 364 0.0039734073 0.004271991
## 365 0.0050441210 0.004083065
## 366 0.0066381921 0.005796130
## 367 0.0075260401 0.005085557
## 368 0.0050424689 0.004880218
## 369 0.0014968031 0.005440637
## 370 0.0089576112 0.005165527
## 371 0.0033743265 0.004758854
## 372 0.0092120054 0.006631951
## 373 0.0061213079 0.003768386
## 374 0.0038408854 0.003698180
## 375 0.0051341073 0.003505616
## 376 0.0048609221 0.005627254
## 377 0.0032527518 0.005303995
## 378 0.0048795788 0.004800600
## 379 0.0018798613 0.003175926
## 380 0.0082982237 0.003584443
## 381 0.0064806760 0.002314679
## 382 0.0076059858 0.004361622
## 383 0.0013962169 0.003467843
## 384 0.0063839266 0.005785117
## 385 0.0095841260 0.004192045
## 386 0.0031402867 0.003340642
## 387 0.0108157871 0.006436505
## 388 0.0028230359 0.003959738
## 389 0.0024468152 0.005709419
## 390 0.0019184353 0.005534736
## 391 0.0043033654 0.004617391
## 392 0.0070424613 0.004870793
## 393 0.0028381297 0.002915599
## 394 0.0026265987 0.004618701
## 395 0.0054502705 0.005617472
## 396 0.0040325006 0.003585161
## 397 0.0034569259 0.006376342
## 398 0.0032441908 0.004414020
## 399 0.0011016794 0.003308433
## 400 0.0036218667 0.005170827
## 401 0.0082787771 0.004202852
## 402 0.0011750911 0.003319521
## 403 0.0029935590 0.004279368
## 404 0.0027938822 0.003705281
## 405 0.0131948458 0.005850676
## 406 0.0036116002 0.005026201
```

```
## 407 0.0073100298 0.006137818
## 408 0.0071470353 0.003366669
## 409 0.0086737917 0.005552846
## 410 0.0010922081 0.003658184
## 411 0.0031952354 0.005105643
## 412 0.0058242991 0.005246135
## 413 0.0030387570 0.004443858
## 414 0.0047543021 0.003627044
## 415 0.0037518257 0.002821291
## 416 0.0097896740 0.005605115
## 417 0.0015991316 0.004736523
## 418 0.0076269617 0.005048061
## 419 0.0033755965 0.004853462
## 420 0.0041155273 0.003288909
## 421 0.0093742924 0.004309765
## 422 0.0124304849 0.007663523
## 423 0.0045273122 0.003313905
## 424 0.0073678437 0.005077185
## 425 0.0067944282 0.005346648
## 426 0.0056610335 0.006804813
## 427 0.0008516125 0.003503882
## 428 0.0027208112 0.004164567
## 429 0.0098302408 0.006292320
## 430 0.0018593832 0.003830886
## 431 0.0010102950 0.002918345
## 432 0.0121165880 0.005388643
## 433 0.0041011084 0.005148416
## 434 0.0029064556 0.004514085
## 435 0.0016188727 0.004069269
## 436 0.0000000000 0.002383447
## 437 0.0115888526 0.005132637
## 438 0.0030315893 0.003801824
## 439 0.0044476249 0.003059378
## 440 0.0019702894 0.002449279
## 441 0.0048646222 0.002951381
## 442 0.0041345550 0.004209439
## 443 0.0009510259 0.004430913
## 444 0.0006659118 0.003935497
## 445 0.0040079341 0.005405100
## 446 0.0022728309 0.005761867
```

The predicted results are compared to the actual results. We then test the accuracy of the model

```r
#Testing The Accuracy Of The Model
predicted = results$prediction * abs(diff(range(Gas_prod))) + min(Gas_prod)
actual = results$actual * abs(diff(range(Gas_prod))) + min(Gas_prod)
comparison = data.frame(predicted,actual)
deviation = ((actual-predicted)/actual)
comparison = data.frame(predicted,actual,deviation)
```

And compute the accuracy of network with (5,2) hidden layer.

```r
accuracy = 1-abs(mean(deviation))
accuracy
```

```
## [1] 0.7784113
```

You can see that we obtain 77.84 % network accuracy using a (5,2) hidden configuration. We see that our accuracy rate has now increased to nearly 78 %, indicating that modifying the number of hidden nodes has enhanced our model! This shows that we can increase the accuracy of the network to predict to a higher value by increasing the number of hidden layers.

Model Interpretability

From the solution above, we can observe that neural networks resemble black boxes a lot: explaining their outcome is much more difficult than explaining the outcome of simpler model such as a linear model. Therefore, depending on the kind of application you need, you might want to take into account this factor too. Furthermore, as you have seen above, extra care will be needed to fit a neural network and small changes can lead to different results.

In addition, we showed that neural network is better at predicting "Gas_prod" than the linear regression model. A better mean squared error (MSE) value was obtained using neural network than linear model. Finally, it is possible for neural networks to be more accurate at prediction than regression; however, it will take trial and error of many different hidden layer configurations to get this better prediction.

Conclusion

This project was developed to illustrate the relationship between a dependent variable and several independent variables using **Artificial Neural Network**. We have been able to develop a way such that we were able to select the training, validation and testing datasets. We also investigated how varying number of hidden layers and neurons affect the artificial neural network results. The result using Artificial Neural Network was compared to result from using linear regression. We can conclude that the Artificial Neural Network is better at prediction than the linear regression.