# Concrete Compressive Strength Analysis

Jason Park

May 2016

**Abstract**

In this paper, we find the best fit model for concrete compressive strength from given inputs: cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, and age. The concrete compressive strength is a highly nonlinear function. We run 5 different models: Linear Regression, Penalized Linear Regression, k-Nearest Neighbors, CART, and Support Vector Machines with Linear Kernel. With the best fit model, we predict a concrete compressive strength within close confidence interval.

## 1 Introduction

Concrete is one of most important material in civil engineering. However, the concrete compressive strength is a highly nonlinear function from its inputs: cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, and age. We investigate the concrete compressive strength from a data given in UCI data archive. This link is

$$https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength$$

It is a regression problem with 1030 instances, 9 attributes, and no missing values. We will skip the data wrangling part since the data is complete as it is. The detailed description of 9 attributes are as following:

- Cement (component 1) – quantitative – $kg$ in a $m^3$ mixture – Input Variable
- Blast Furnace Slag (component 2) – quantitative – $kg$ in a $m^3$ mixture – Input Variable
- Fly Ash (component 3) – quantitative – $kg$ in a $m^3$ mixture – Input Variable
- Water (component 4) – quantitative – $kg$ in a $m^3$ mixture – Input Variable
- Superplasticizer (component 5) – quantitative – $kg$ in a $m^3$ mixture – Input Variable
- Coarse Aggregate (component 6) – quantitative – $kg$ in a $m^3$ mixture – Input Variable
- Fine Aggregate (component 7) – quantitative – $kg$ in a $m^3$ mixture – Input Variable
- Age – quantitative – Day ($1\sim 365$) – Input Variable
- Concrete compressive strength – quantitative – $MPa$ – Output Variable

## 2 Preparation

We downlad the data and read data by command:

```
concrete <- read.csv("concrete.csv")
data1 <- concrete
```

We split the data into training set and testing set.

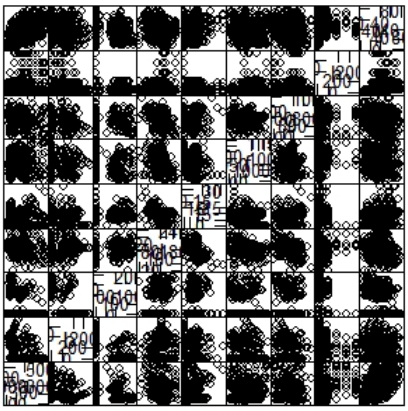
```
   inTraining <- createDataPartition(data1$X, p = .75, list =
FALSE)
   training <- data1[ inTraining,]
   testing <- data1[-inTraining,]
```

This will randomly put 75% of data into the training set and rest to testing set. We will run our analysis on training set only. Also we change the name of the columns from cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, concrete compressive strength to a, b, c, d, e, f, g, h, output. This is for just convenience purpose since each attribute names are too lentghy.

# 3 Data Visualization

We look at the comprehensive look of correlation for each variables. Library "AppliedPredictiveModeling" pacakge and plot the correlation graph.

```
    library(AppliedPredictiveModeling)
    featurePlot(x = training[, 1:8], y = training$output, plot = "pairs", auto.key
= list(columns = 3))
```

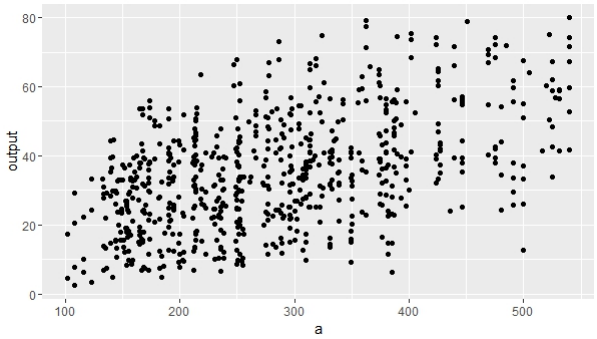


Scatter Plot Matrix

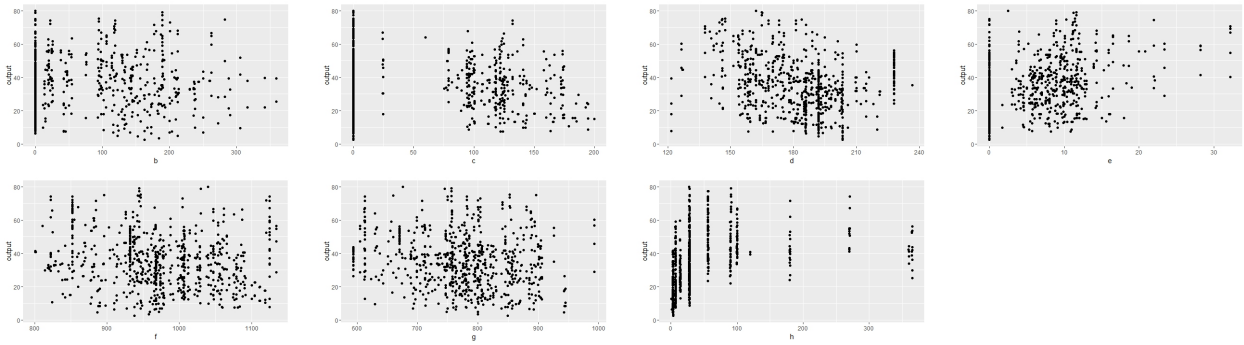
It is not clear to see much correlations. We will look at correlation between cement and concrete compressive strength in the package "ggplot2"

```
   ggplot(aes(x = a, y = output), data = training) + geom_jitter(alpha = 1, color
= 'black')
```

Above command creates the graph.



We can see clearly correlation between cement and concrete compressive. However, there is no obvious linear correlation for other attributes and the output.



# 4 Analysis

We use the 'caret' package for our analysis. It is one of the most complete packages in R. We can run analysis in relatively concise and simple commands. Library the 'caret' packages first.

## 4.1 Analysis

We run 5 different models to fit the best model. They are Linear Regression, Penalized Linear Regression, k-Nearest Neighbors, CART, and Support Vector Machines with Linear Kernel.

```
formula <- output   a + b + c + d + e + f + g + h
fit.lm <- train(formula, data=training ,method='lm')
fit.glm <- train(formula, data=training ,method='glmnet')
fit.knn <- train(formula, data=training ,method='knn')
fit.rpart <- train(formula, data=training ,method='rpart')
fit.svmLinear <- train(formula, data=training ,
method='svmLinear')
```

Following command gives the the result comparison for each model performance.

```
resamps <- resamples(list(LM = fit.lm, GLM = fit.glm, KNN =
fit.knn RPART = fit.rpart,SVM =fit.svmLinear))
summary(resamps)
```

We have following results.

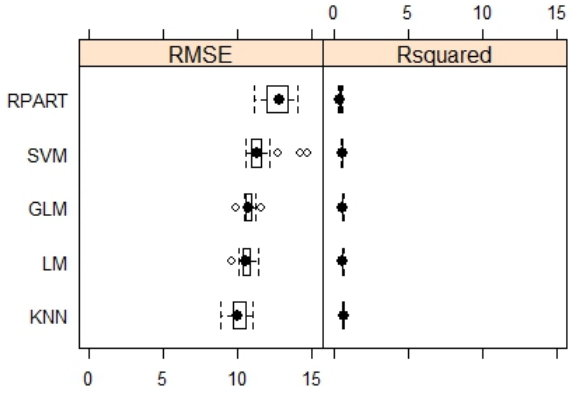
*RMSE*

| | *Min.* | 1*stQu.* | *Median* | *Mean* | 3*rdQu.* | *Max.* | *NA's* |
|---|---|---|---|---|---|---|---|
| *LM* | 9.623 | 10.360 | 10.50 | 10.56 | 10.78 | 11.33 | 0 |
| *GLM* | 9.874 | 10.530 | 10.73 | 10.72 | 10.90 | 11.54 | 0 |
| *KNN* | 8.817 | 9.678 | 10.01 | 10.06 | 10.52 | 11.04 | 0 |
| *RPART* | 11.140 | 11.930 | 12.80 | 12.64 | 13.35 | 14.01 | 0 |
| *SVM* | 10.540 | 10.880 | 11.25 | 11.50 | 11.60 | 14.71 | 0 |

*RSquared*

| | *Min.* | 1*stQu.* | *Median* | *Mean* | 3*rdQu.* | *Max.* | *NA's* |
|---|---|---|---|---|---|---|---|
| *LM* | 0.5534 | 0.5711 | 0.5880 | 0.5915 | 0.6095 | 0.6323 | 0 |
| *GLM* | 0.5050 | 0.5634 | 0.5824 | 0.5796 | 0.5976 | 0.6482 | 0 |
| *KNN* | 0.5738 | 0.6207 | 0.6326 | 0.6357 | 0.6554 | 0.7339 | 0 |
| *RPART* | 0.2680 | 0.3631 | 0.4240 | 0.4200 | 0.4614 | 0.5692 | 0 |
| *SVM* | 0.4484 | 0.5475 | 0.5676 | 0.5639 | 0.5914 | 0.6199 | 0 |

We see that KNN performance is the best with RMSE mean 10.06 and RSquared 0.6357. The graph shows it more clearly.




## 4.2 Analysis with Cross Validation

We run the 5 models with cross validation by dividing the training set into 5 partitions.

```
ctrl<-trainControl(method = 'cv' ,number = 5)
fit.cv.lm<-train(output ., data = training, method = 'lm' ,
trControl = ctrl, metric='RMSE')
fit.cv.glm <-train(output ., data = training, method =
'glmnet' , trControl = ctrl, metric= 'RMSE')
fit.cv.knn <-train(output ., data = training, method ='knn'
, trControl = ctrl, metric= 'RMSE')
fit.cv.rpart <-train(output ., data = training, method =
'rpart' , trControl = ctrl, metric= 'RMSE')
fit.cv.svmLinear <-train(output ., data = training, method =
'svmLinear' , trControl = ctrl, metric='RMSE')
```

We have following results.

*RMSE*

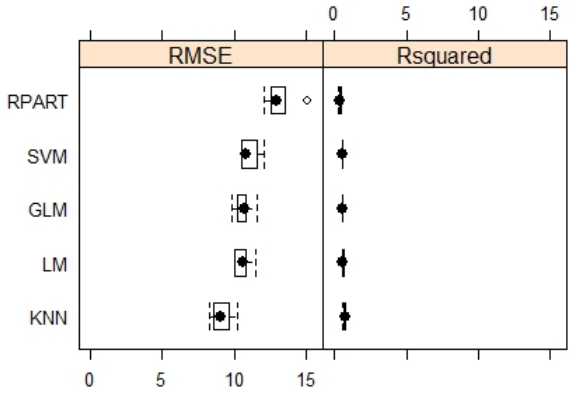
| | *Min.* | 1*stQu.* | *Median* | *Mean* | 3*rdQu.* | *Max.* | *NA's* |
|---|---|---|---|---|---|---|---|
| *LM* | 10.040 | 10.060 | 10.620 | 10.600 | 10.810 | 11.46 | 0 |
| *GLM* | 9.809 | 10.210 | 10.760 | 10.640 | 10.810 | 11.59 | 0 |
| *KNN* | 8.328 | 8.619 | 9.049 | 9.186 | 9.681 | 10.25 | 0 |
| *RPART* | 12.110 | 12.550 | 12.950 | 13.250 | 13.540 | 15.13 | 0 |
| *SVM* | 10.480 | 10.490 | 10.810 | 11.090 | 11.620 | 12.06 | 0 |

*Rsquared*

| | *Min.* | 1*stQu.* | *Median* | *Mean* | 3*rdQu.* | *Max.* | *NA's* |
|---|---|---|---|---|---|---|---|
| *LM* | 0.5651 | 0.5684 | 0.6047 | 0.5975 | 0.6205 | 0.6288 | 0 |
| *GLM* | 0.5608 | 0.5633 | 0.5735 | 0.5906 | 0.5768 | 0.6787 | 0 |
| *KNN* | 0.6073 | 0.6479 | 0.7211 | 0.6976 | 0.7359 | 0.7758 | 0 |
| *RPART* | 0.2504 | 0.3349 | 0.3509 | 0.3688 | 0.4333 | 0.4746 | 0 |
| *SVM* | 0.5353 | 0.5478 | 0.6041 | 0.5816 | 0.6073 | 0.6135 | 0 |

KNN's performance is the best with mean RMSE 9.186 and mean RSquared 0.7359. The graph shows it

clearly.




## 4.3 Tuning in KNN

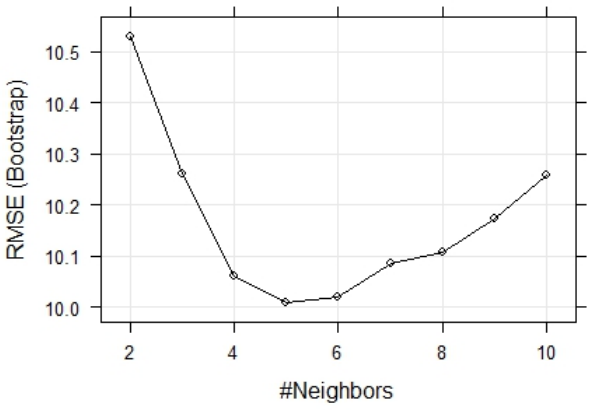
We conclude KNN model is the best fitting model for our analysis. However, we can improve accuracy a bit further since 'caret' package uses the default value k kor KNN analysis. We run KNN model from $k = 2$ to $k = 10$.

```
knnGrid <- expand.grid(.k=c(2:10))
fit.knn.tune <- train(formula, data=training ,method='knn',
tuneLength = 10, tuneGrid = knnGrid )
```

Then we have the following results.

| $k$ | $RMSE$ | $Rsquared$ |
|---|---|---|
| 2 | 10.53107 | 0.6103871 |
| 3 | 10.26003 | 0.6211551 |
| 4 | 10.06226 | 0.6318049 |
| 5 | 10.00809 | 0.6343718 |
| 6 | 10.01974 | 0.6335351 |
| 7 | 10.08588 | 0.6281979 |
| 8 | 10.10716 | 0.6266619 |
| 9 | 10.17382 | 0.6220246 |
| 10 | 10.25962 | 0.6158488 |

When k=5, we have the optimal model, which outperforms other k-valued models. We can obersve it clearly from the graph.



## 4.4 Test with Final Model

KNN model with k=5 is the final model. We define as following.

```
knnGrid.final <- expand.grid(.k= 5)
fit.knn.final <- train(formula, data=training ,method='knn', tuneLength = 10, tuneGrid
= knnGrid.final )
```

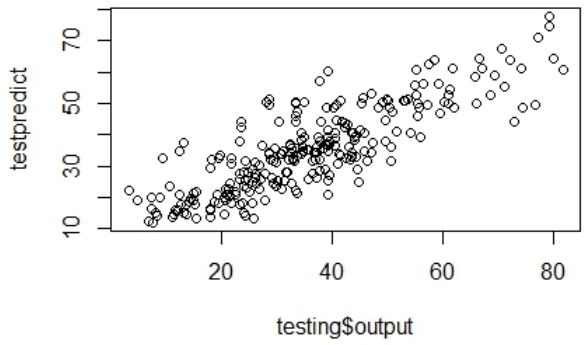
We use our unused testing set to see the performance of our final model.

```
testpredict <- predict(fit.knn.final, testing)
```

We obtain the final result.

| $RMSE$ | $Rsquared$ |
|---|---|
| 9.2424244 | 0.7033052 |

Rsquared value was even smaller than any previous analysis run.




The predicted values and actual values show very strong linear relation in the graph.

## 4.5 PreProcessing

Since we are using the KNN model, preProcessing might our model a bit better. We scale and center the data set. For each data value $x$, we have standard score

$$Z_x = \frac{x - \mu}{\sigma}$$

where $\mu$ and $\sigma$ are the mean and standard deviation of the attribute that $x$ belongs.

In the k-nearest neighbor model, all datas are assigend to one of the 5 points. In Euclidean metric, one attribute might have relatively larger number than the other attribute. For instance, superplasticizer has the domain $[0, 32.2]$ but blast furnace slag has the domain $[0, 359.4]$. Blast furnace slag would be valued more importantly for measuring the distance in Euclidean sense because it is simply a larger number. Changing the raw data into the standard scores $Z$ would reduce this type of problem. We use the following code:

```
preProc <- c("center", "scale")
fit.knn2 <- train(output   ., data = training, method = 'knn' , trControl = ctrl,
metric= 'RMSE', preProc=preProc)
```

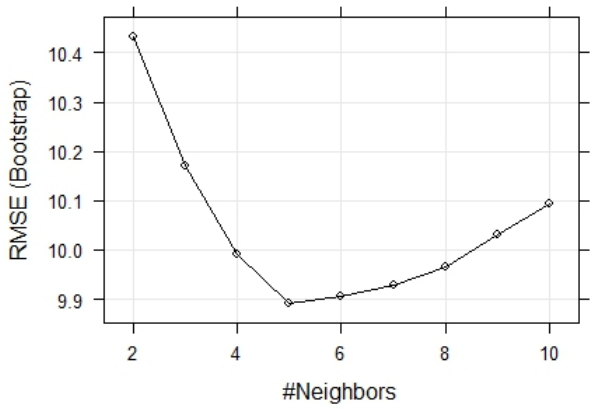
We find the optimal $k$ for our model.

```
knnGrid <- expand.grid(.k=c(2:10))
it.knn2.tune <- train(formula, data=training ,method='knn', tuneLength = 10,
tuneGrid = knnGrid, preProc=preProc )
```

We obtain the following results:

| $k$ | $RMSE$ | $Rsquared$ |
|---|---|---|
| 2 | 10.433967 | 0.6185847 |
| 3 | 10.170432 | 0.6288326 |
| 4 | 9.990404 | 0.6383930 |
| 5 | 9.890306 | 0.6433327 |
| 6 | 9.906930 | 0.6415088 |
| 7 | 9.929218 | 0.6390837 |
| 8 | 9.966993 | 0.6361390 |
| 9 | 10.030968 | 0.6314831 |
| 10 | 10.093267 | 0.6274357 |

Observe that both RMSE and Rsquared values are improved slightly. We see that we have the best fit model when we have $k = 5$. The graph shows it more clearly.




## 4.6 Final Result

The preProcessed KNN model with $k = 5$ is the best fitting model. We run the final analysis with the testing set.

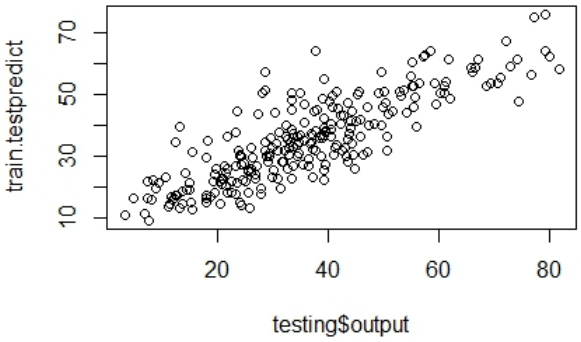
```
knnGrid.final <- expand.grid(.k= 5)
fit.knn2.final <- train(formula, data=training ,method='knn', tuneLength = 10, tuneGrid
= knnGrid.final, preProc = preProc )
train.testpredict <- predict(fit.knn2.final, testing)
```

We obtain the final result.

| $RMSE$ | $Rsquared$ |
|---|---|
| 9.1991733 | 0.7059835 |

The graph between actual value in testing set and the predicted value from our best model shows clear

linear relation.




# 5 Conclusion

This research conclude the preProcessed KNN model with k=5 is the best fitting model for the concrete compressive strength from the given variables: cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, and age. It predicts the predicted value is within 9.1991733 RMSE with 95% confidence, which is a pretty good model.

Any economical factor or duration of making concrete was not main focus in this research. The only concern was how to predict the concrete compressive strength from the given data. It would be interesting to find optimal compressive strength with most economical price. If the age is the major factor, for instance, construction company might want to have a certain strength of concrete in the short period of time, then the question could be 'what would be the opitmal way to make desired compressive strength within the time frame?' This would be also an interesting research.

The final model in this research can be used to predict the concrete compressive strength with given input data. With the limited amount of resources, it is essential to create optimal concrete compressive strength in civil engineering. The data analysis plays a key role to estimate the optimal strength.