Supervised Project

Federico Matteucci - 955753

September 2020

Contents

1 Abstract

I shall present a supervised analysis of the concrete dataset[1], in which I confront the performance of linear, lasso, polynomial and tree regressors to predict the compressive strength of concrete from its constituents.
My goal here was to find which predictor could outperform the others, my guess was the tree predictor would.
Then I present a brief case study, supposing to have a limited quantity of some ingredients.
In this paper I did not standardize the variables (as it didn't affect the quality of the models and I wanted to retain the real values for the sake of interpretability).

2 Dataset

The dataset consists of 9 variables, with no missing value:

1. cement [kg/m^3]
2. blast furnace slag [kg/m^3]
3. fly ash [kg/m^3]
4. water [kg/m^3]
5. superplasticizer [kg/m^3]
6. coarse aggregate [kg/m^3]
7. fine aggregate [kg/m^3]
8. age [$days$]
9. **compressive strength** [MPa]

From the boxplot I see that there are no outliers in the data.
More, the dataset does not show any multicollinearity.

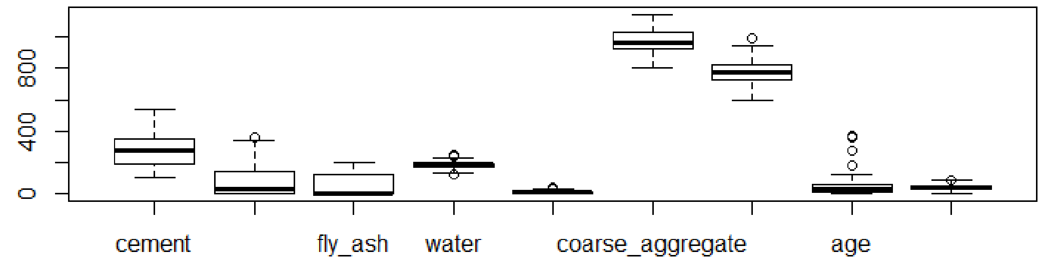

Figure 1: Boxplot of concrete.

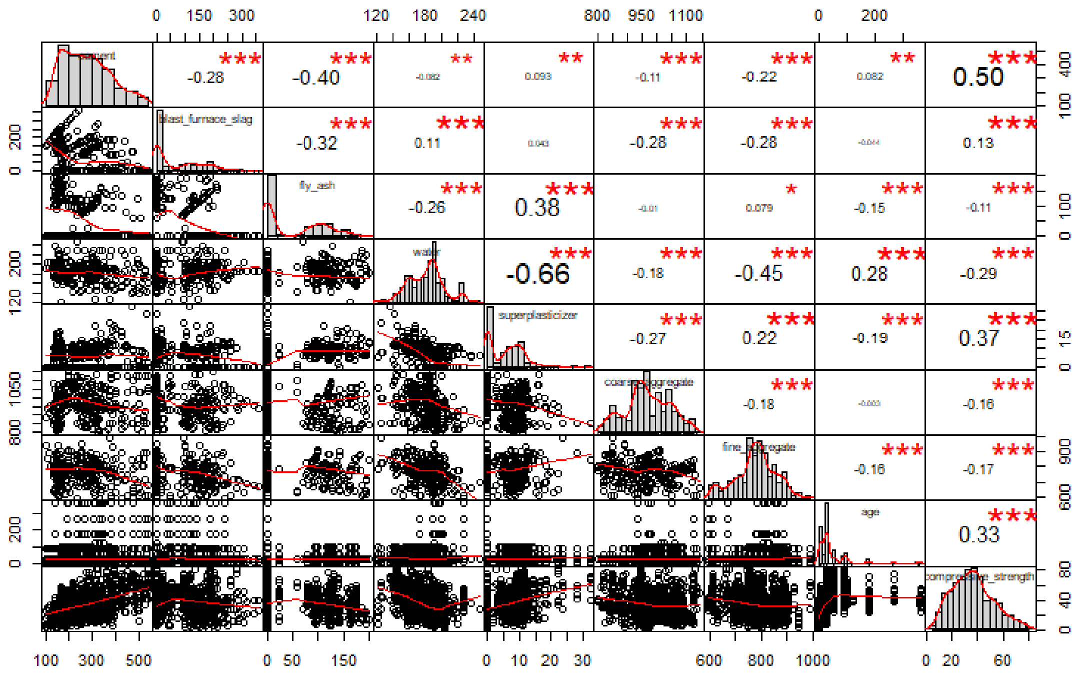

Figure 2: Correlation chart: no attributes are seriously correlated.

3 Regression

All regressors were trained on the entire dataset, their R^2 calculated there.

3.1 Linear regression

	coefficient	**std error**	**t value**	**Pr($>|t|$)**
(Intercept)	-23.163756	26.588421	-0.871	0.383851
cement	0.119785	0.008489	14.110	<2e-16 ***
blast_furnace_slag	0.103847	0.010136	10.245	<2e-16 ***
fly_ash	0.087943	0.012585	6.988	5.03e-12 ***
water	-0.150298	0.040179	-3.741	0.000194 ***
superplasticizer	0.290687	0.093460	3.110	0.001921 **
coarse_aggregate	0.018030	0.009394	1.919	0.055227 .
fine_aggregate	0.020154	0.010703	1.883	0.059968 .
age	0.114226	0.005427	21.046	<2e-16 ***

Table 1: Linear regression summary. The most relevant component is the superplasticizer.
$adjR^2$: 0.612
cross validated MSE: 109.3

3.2 Lasso regression

Lasso algorithms have two main benefits: they stabilize the model and reduce the dimensionality of the dataset. I preferred lasso over PCA for interpretability.
The regressor is:

$$w_{lasso} = \min_{w} \|Xw - y\|^2 + \lambda |w|$$

where, if p is the number of explanatory variables, $|w| = \sum_i^p |w_i|$.
λ is an hyerparameter and needs to be tuned minimizing the MSE estimated with cross validation and then selecting the greatest λ that achieves a MSE one standard deviation above the minimal one, as shown in Figure 4. As λ grows, more coefficients are set to 0, there is a trade off between mse of the model and number of variables; this method provides a good balance.
The optimal λ is 0.5597827, with 7 surviving attributes.

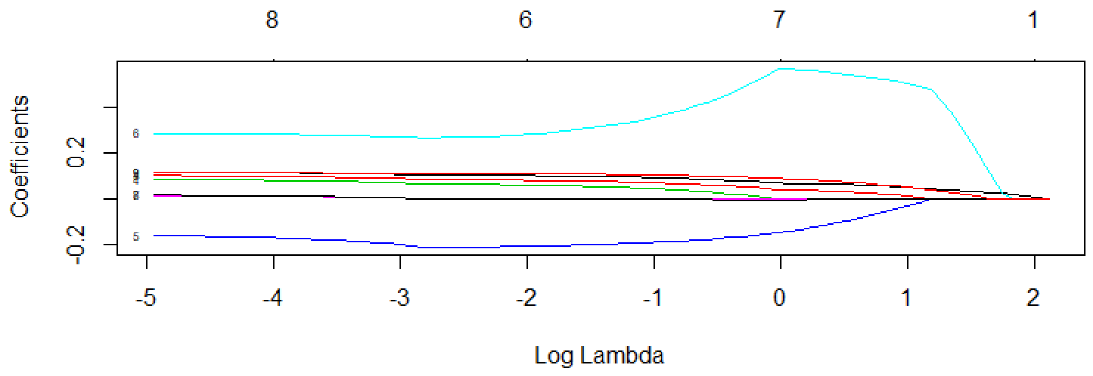

Figure 3: Coefficients against $log(\lambda)$. As λ grows, more coefficients are set to 0.

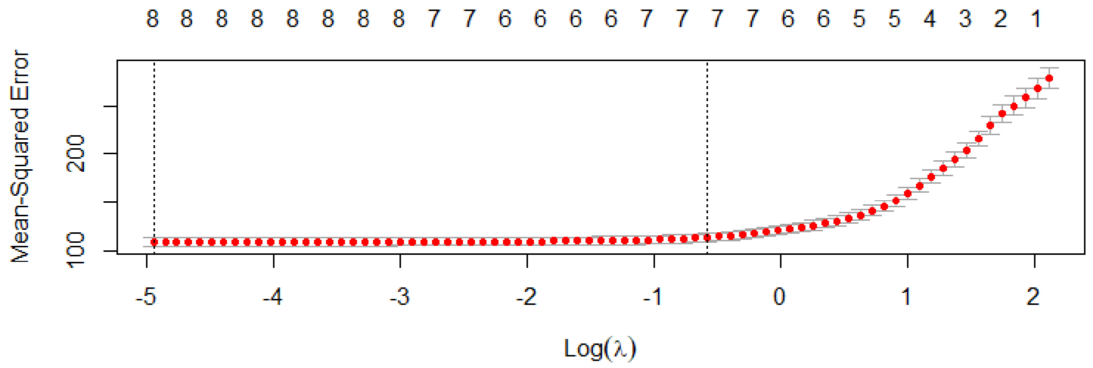

Figure 4: MSE against $log(\lambda)$. The λ of choice (rightmost vertical line) is the one that achieves a MSE one standard deviation above the minimum.

	coefficient
(Intercept)	32.732898486
cement	0.085134262
blast_furnace_slag	0.061363150
fly_ash	0.030949497
water	-0.177244053
superplasticizer	0.421164433
coarse_aggregate	.
fine_aggregate	-0.002599309
age	0.098908705

Table 2: Lasso regression summary. The superplasticizer is even more relevant than in linear regression.
$adjR^2$: 0.598
cross validated MSE: 109.7

3.3 Polynomial regression

A polynomial expansion (of degree 2) allows for the study of interactions between the variables (the mixed second degree terms terms) and for the study of convexity of the attributes: a negative coefficient for C^2 means that the marginal improvement in mixing C lowers the more C is in the mixture.
No interaction is preponderant on the others, but most have at least a significant coefficient.
Now, the coefficient of Xpoly0.0.0.0.2.0.0.0 (superplasticizer^2) is -0.04548, thus the benefit of adding superplasticizer to the compound vanishes the more its added.
This gives hints to the simple fact that a concrete made only of superplasticizer wouldn't be the best solution to maximize the compressive strength (and it wouldn't even be a concrete).

	coefficient	**std error**	**t value**	**Pr(> $\|t\|$)**
Xpoly1.0.0.0.0.0.0.0	2.610e+00	7.366e-01	3.543	0.000414 ***
Xpoly2.0.0.0.0.0.0.0	-3.536e-04	1.253e-04	-2.821	0.004882 **
Xpoly0.1.0.0.0.0.0.0	2.192e+00	8.635e-01	2.539	0.011278 *
Xpoly1.1.0.0.0.0.0.0	-5.059e-04	2.942e-04	-1.719	0.085847 .
Xpoly0.2.0.0.0.0.0.0	-2.663e-04	1.823e-04	-1.461	0.144262
Xpoly0.0.1.0.0.0.0.0	1.668e+00	1.034e+00	1.614	0.106841
Xpoly1.0.1.0.0.0.0.0	-1.910e-04	3.513e-04	-0.544	0.586777
Xpoly0.1.1.0.0.0.0.0	1.034e-04	4.282e-04	0.241	0.809237
Xpoly0.0.2.0.0.0.0.0	1.644e-04	3.046e-04	0.540	0.589615
Xpoly0.0.0.1.0.0.0.0	1.442e+01	2.712e+00	5.318	1.30e-07 ***
Xpoly1.0.0.1.0.0.0.0	-4.668e-03	8.029e-04	-5.815	8.21e-09 ***
Xpoly0.1.0.1.0.0.0.0	-3.975e-03	1.041e-03	-3.817	0.000144 ***
Xpoly0.0.1.1.0.0.0.0	-4.497e-03	1.215e-03	-3.701	0.000226 ***
Xpoly0.0.0.2.0.0.0.0	-9.035e-03	1.785e-03	-5.061	4.97e-07 ***
Xpoly0.0.0.0.1.0.0.0	3.118e+01	7.603e+00	4.101	4.46e-05 ***
Xpoly1.0.0.0.1.0.0.0	-1.097e-02	2.172e-03	-5.052	5.20e-07 ***
Xpoly0.1.0.0.1.0.0.0	-1.046e-02	2.616e-03	-3.999	6.83e-05 ***
Xpoly0.0.1.0.1.0.0.0	-1.749e-02	3.396e-03	-5.151	3.13e-07 ***
Xpoly0.0.0.1.1.0.0.0	-3.385e-02	1.111e-02	-3.048	0.002363 **
Xpoly0.0.0.0.2.0.0.0	-4.548e-02	1.564e-02	-2.909	0.003711 **
Xpoly0.0.0.0.0.1.0.0	2.689e+00	8.830e-01	3.046	0.002382 **
Xpoly1.0.0.0.0.1.0.0	-6.330e-04	2.792e-04	-2.267	0.023595 *
Xpoly0.1.0.0.0.1.0.0	-6.144e-04	3.147e-04	-1.952	0.051207 .
Xpoly0.0.1.0.0.1.0.0	-3.315e-04	3.871e-04	-0.857	0.391911
Xpoly0.0.0.1.0.1.0.0	-4.987e-03	1.003e-03	-4.970	7.91e-07 ***
Xpoly0.0.0.0.1.1.0.0	-1.014e-02	2.846e-03	-3.561	0.000386 ***
Xpoly0.0.0.0.0.2.0.0	-3.609e-04	1.719e-04	-2.100	0.036000 *
Xpoly0.0.0.0.0.0.1.0	3.716e+00	9.854e-01	3.771	0.000172 ***
Xpoly1.0.0.0.0.0.1.0	-9.344e-04	3.093e-04	-3.022	0.002580 **
Xpoly0.1.0.0.0.0.1.0	-7.130e-04	3.616e-04	-1.972	0.048923 *
Xpoly0.0.1.0.0.0.1.0	-4.686e-04	4.287e-04	-1.093	0.274625
Xpoly0.0.0.1.0.0.1.0	-5.715e-03	1.135e-03	-5.033	5.73e-07 ***
Xpoly0.0.0.0.1.0.1.0	-1.224e-02	2.862e-03	-4.277	2.08e-05 ***
Xpoly0.0.0.0.0.1.1.0	-1.006e-03	3.592e-04	-2.800	0.005217 **
Xpoly0.0.0.0.0.0.2.0	-8.334e-04	2.118e-04	-3.934	8.94e-05 ***
Xpoly0.0.0.0.0.0.0.1	7.891e-02	4.340e-01	0.182	0.855759
Xpoly1.0.0.0.0.0.0.1	1.986e-04	1.625e-04	1.222	0.222052
Xpoly0.1.0.0.0.0.0.1	4.363e-04	1.652e-04	2.642	0.008378 **
Xpoly0.0.1.0.0.0.0.1	7.003e-04	2.766e-04	2.532	0.011510 *
Xpoly0.0.0.1.0.0.0.1	-7.814e-05	7.374e-04	-0.106	0.915626
Xpoly0.0.0.0.1.0.0.1	3.380e-03	2.083e-03	1.622	0.105023
Xpoly0.0.0.0.0.1.0.1	-5.541e-05	1.336e-04	-0.415	0.678418
Xpoly0.0.0.0.0.0.1.1	2.459e-04	1.862e-04	1.321	0.186911
Xpoly0.0.0.0.0.0.0.2	-6.149e-04	4.139e-05	-14.856	¡ 2e-16 ***

Table 3: Polynomial regression.$adjR^2$: 0.802,MSE: 52.8 (train = whole set)

4 Tree

I trained a tree regressor on the whole dataset, here is the result:

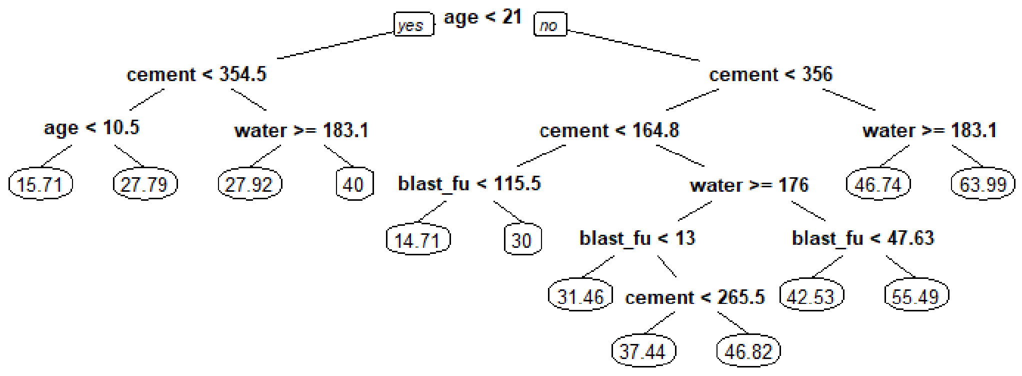

Figure 5: Tree predictor.$adjR^2$: 0.738,MSE: 72.4 (train = whole set)

5 Comparison

Using adjusted R^2 as scoring, on the entire dataset the polynomial model (R^2: 0.802) outperformed both the tree (R^2: 0.738), linear (R^2: 0.612) and lasso (R^2: 0.598). The model of choice would therefore be the polynomial one, against my expectations.

6 Case Study

Let's suppose we have to make one cubic meter of concrete, but using at most 200 kg of cement and having it rest less than 60 days. How can one find a "region" of mixtures with high average strength?
To answer this question I used a tree predictor on a proper subset of the dataset.

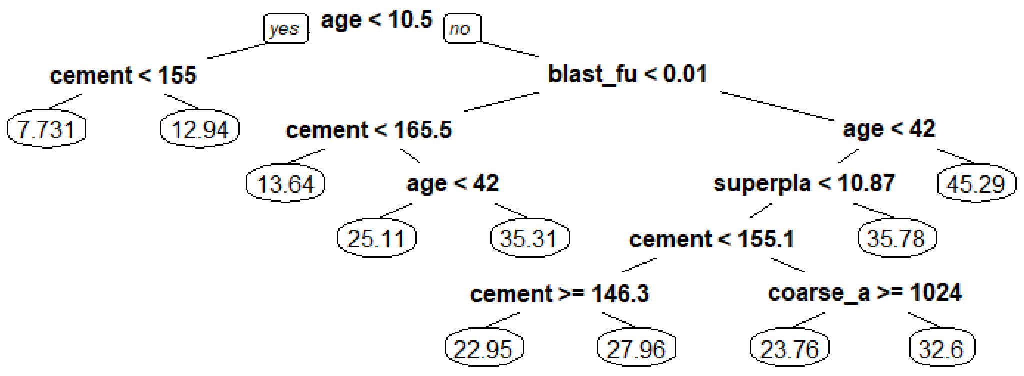

Figure 6: Tree predictor for case data. $adjR^2$: 0.807. The greatest compressive strength is 45.29.

Figure 6 shows that the best solution in the scenario is to use more than 0.01 blast furnace slag and let it rest more than 42 days.

6.1 Comment

45.29 is not the greatest strength achievable in the case study (that being 53.77), but it is the greatest strength one can *consistently* expect from a mixture of concrete, the average strength of a mixture with less than 200 kg of cement, more than 0.01 kg of slag and stiffened for 42 to 60 days. The tree model on this subset achieved an $adjR^2$ of 0.807, versus the benchmark linear model ($adjR^2$: 0.752).

7 Conclusion

- Superplasticizer is the best component in the concrete, but its utility lowers with its quantity.
- A long enough aging process is pivotal in making the concrete hard.
- A polynomial predictor is far better than a linear predictor and slightly better than a tree one.

References

[1] I-Cheng Yeh. Concrete Compressive Strength Data Set. https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

8 Appendix

```
library(readr)
library(party)
library(rpart)
library(rpart.plot)
library(magrittr)
library(glmnet)
library(boot)
library(PerformanceAnalytics)

# ADJUSTED rsq
rsq <- function(predictor, model, X, y){
        n <- nrow(X)
        p <- ncol(X)

        pred <- predictor(model, X)

        RSS <- sum((y-pred)^2)
        TSS <- sum((y-mean(y))^2)

        out <- 1 - RSS/(n-p-1) * (n-1)/TSS

        return(out)
}

#---- Dataset, Multicollinearity ----

concrete_data <- read.csv("C:/Users/Io/Desktop/SML_Supervised/Concrete_Data.csv", sep=';') #%>% standardize

X <- model.matrix(compressive_strength ~.-compressive_strength, data=concrete_data) %>% subset(select=c(-1))
y <- concrete_data$compressive_strength
Xpoly <- poly(X, 2, raw=TRUE)

chart.Correlation(concrete_data, labels=names(concrete_data))
boxplot(concrete_data)
#---- Model Predictivity ----

# Linear
linear_model <- glm(compressive_strength ~ .-compressive_strength, data=concrete_data)
summary(linear_model)
cv.glm(concrete_data, linear_model, K=10)$delta[1]
# 109.3
rsq(predict, linear_model, data.frame(X), y)
# 0.612

# Lasso (has some stochastic component in the design)
test_lasso <- glmnet(X, y)
plot(test_lasso, xvar='lambda', label=TRUE)

cv_lasso <- cv.glmnet(X, y)
plot(cv_lasso)
min(cv_lasso$cvm)
# 109.7

cv_lasso$lambda.1se
# 0.560

lasso_model <- glmnet(X, y, lambda=cv_lasso$lambda.1se)
coef(lasso_model)

rsq(predict, lasso_model, X, y)
# 0.598


# naive error for poly and tree - NOT USED
train <- sample(nrow(concrete_data),nrow(concrete_data)*0.75,replace=FALSE)
poly_train <- glm(concrete_data$compressive_strength[train] ~ Xpoly[train,])

sum((y[-train]-predict(poly_train, Xpoly[-train,] %>% data.frame))^2)/length(y[-train])
#...

# Polynomial
poly_model <- glm(compressive_strength ~ Xpoly, data=concrete_data)
summary(poly_model)
cv.glm(data.frame(Xpoly, y), poly_model, K=10)$delta[1]
#524???
sum((y-predict(poly_model, Xpoly))^2)/length(y)
#52.8

rsq(predict.glm, poly_model, Xpoly%>%data.frame, y)
# 0.802

# Tree
complete_tree <- rpart(compressive_strength ~ .-compressive_strength, data=concrete_data)
plotcp(complete_tree)
prp(complete_tree, roundint=FALSE, digits=4)

rsq(rpart.predict, complete_tree, data.frame(X), y)
```

```
# 0.738
sum((y-rpart.predict(complete_tree, data.frame(X)))^2)/length(y)
# 72.4

#---- Case Study ----
# Tree
# Let's assume we have at most 200 kg of cement per m^3 AND only 60 days to make it stiff
# What is the best combination of other materials?

target <- 200
time <- 60
case_data <- concrete_data %>% subset(cement < target) %>% subset(age < time)

boxplot(case_data)

Xc <- model.matrix(compressive_strength ~.-compressive_strength, data=case_data)
yc <- case_data$compressive_strength

# does anything change in the coefficients? YES it does, a lot
linear_model <- glm(compressive_strength ~ .-compressive_strength, data=case_data)

rsq(predict, linear_model, data.frame(Xc), yc)
# 0.752

#---- CASE study tree ----
max(case_data$compressive_strength)

case_tree <- rpart(compressive_strength ~ .-compressive_strength, data=case_data)
plotcp(case_tree)
prp(case_tree, roundint=FALSE, digits=4)

rsq(rpart.predict, case_tree, data.frame(Xc), yc)
# 0.807

# best value
case_data[case_data$compressive_strength == max(case_data$compressive_strength), ]
```