Machine Learning Models for Concrete Strength Prediction

Michael Tutterrow
Computer and Mathematical Sciences
Lewis University
michaelatutterrow@lewisu.edu

***Abstract*—This analysis documents several machine learning models used to predict the strength of high-performance concrete. A dataset containing more than one thousand instances of concrete of differing formulations, each with a compressive strength determined by test, is used for this analysis. The dataset is prepared in Python, and machine learning models are built with the scikit-learn and keras libraries. Models are built based on linear regression, polynomial regression, and neural networks. Results are analyzed based on the error these models produce.**

***Keywords*— machine learning, linear regression, polynomial regression, neural network, prediction algorithms, concrete, compressive strength**

I. Introduction

Conventional concrete is composed of three basic ingredients: Portland cement, fine and coarse aggregates, and water. High-performance concrete, by contrast, incorporates several other cementitious materials, including fly ash, blast furnace slag, and superplasticizer. Current empirical equations for determining concrete's strength are based on tests of concrete without supplementary cementitious materials [1].

This report examines several supervised machine learning methods for determining the strength of high-performance concrete. The basic approach is to use the results of experiments that tested the strength of different concrete formulations to train machine learning models. These models are then used to predict the strength of other concrete formulations. Finally, these predictions are compared to actual test results for those formulations, and the accuracy of the models is determined.

II. Tools

The machine learning models are created using the Python programming language. The pandas and numpy libraries are used for data structures and analysis. The scikit-learn library is used for creating the regression models, while the keras library is used for modeling the neural networks. The matplotlib library is used for data visualization. An IPython notebook [2] is used to save and view the code and results.

III. Dataset

A. Description

The dataset was downloaded from the UCI Machine Learning Repository [3]. It was originally donated by Prof. I-Cheng Yeh of Chung-Hua University in Taiwan. The data is stored in an Excel file containing 1030

instances of concrete, with nine attributes each. The attributes include eight attributes describing the quantitative mixture of the concrete (units of kg/m^3 of mixture), with one attribute describing the quantitative compressive strength (units of megapascals, MPa), measured by test. There are no missing attributes in this dataset.

B. Data Preparation

The Excel file is opened in the Excel application, and saved as a comma-separated values (CSV) file. The CSV file is then imported into Python. The columns of data are named with appropriately chosen variable names.

To facilitate the training and testing of the machine learning models, the dataset is divided into input variables (mixture components) and a target variable (strength). These datasets are further divided into training and testing subsets. The train/test percentage split for this analysis is chosen to be 70/30, i.e. 70% of the instances are used to train the models, while the remaining 30% are used to test the accuracy of the models.

C. Sample Data

Table 1 lists the first four instances of the testing input data.

TABLE 1. TESTING INPUTS

	cement	furnace_slag	fly_ash	water	splasticizer	coarse_agg	fine_agg	age
0	491.0	26.0	123.0	201.0	3.9	822.0	699.0	28
1	277.0	0.0	0.0	191.0	0.0	968.0	856.0	14
2	192.0	288.0	0.0	192.0	0.0	929.8	716.1	90
3	236.9	91.7	71.5	246.9	6.0	852.9	695.4	28
4	251.4	0.0	118.3	188.5	6.4	1028.4	757.7	3

Table 2 lists the first four instances of the testing target data.

TABLE 2. TESTING TARGETS

	strength
0	57.92
1	21.26
2	50.73
3	28.63
4	13.12

IV. MODELS

This report examines three modeling approaches for determining the strength of concrete. The first two are variations of regression, while the third approach uses multilayered perceptrons to form a neural network.

A. Linear Regression Model

Multiple linear regression is used to model the relationship between multiple explanatory features, *x*, and a continuous target variable, *y*, as shown in Equation (1) [4]:

$$y = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m \quad (1)$$

B. Polynomial Regression Model

Polynomial regression is similar to linear regression, except we do not assume linearity between the explanatory features and the target variable. Polynomials of varying orders are included in the resulting equation. For this analysis, two orders of polynomials are considered: second order, or quadratic, and third order, or cubic.

C. Multilayered Perceptrons (Neural Network) Model

Perceptrons were originally created to mimic how a single neuron in the brain works: it either fires, or it doesn't [4]. Multilayered perceptrons consist of at least two layers of perceptrons connected by using outputs from one layer as inputs to the subsequent layer. Each layer can contain multiple perceptrons, and there can be many layers in total. These types of multilayered perceptrons are known as neural networks.

For modeling regression problems, the final layer consists of a single node with a continuous value as its output. This analysis considers two such neural networks. The first layer of each model contains eight nodes to accommodate the eight input variables, with one node in the final layer to calculate the continuous output value. The second neural network model adds a hidden layer of five nodes between the input and output layers. Several numbers of nodes were tested for the hidden layer, and multiple numbers of hidden layers were considered. A single hidden layer with five nodes resulted in the highest accuracy.

Both models use *sigmoid* activation with *l1* regularization (also known as *LASSO*) of 0.01 to help control model complexity. *ReLU* activation was also considered, as was *l2* regularization (also known as *ElasticNet*), before settling on the final model configuration that gave the highest accuracy.

Both models use 100 epochs to converge on a solution. Higher numbers of epochs were tested, with limited benefits observed.

V. Results

A. Linear Regression Model

Using the training datasets, the linear regression model is fitted in scikit-learn with the following intercept and coefficients. The relative contributions of individual components can be seen in the coefficients; e.g. the amount of superplasticizer has the most positive correlation to the strength obtained by the formulation.

TABLE 3. LINEAR REGRESSION MODEL DEFINITION

Intercept:	
-13.529	
Coefficients:	
cement	0.116
furnace_slag	0.101
fly_ash	0.083
water	-0.160
splasticizer	0.305
coarse_agg	0.016
fine_agg	0.014
age	0.107

Equation (2) shows the first line of the inputs of Table 1 entered into the linear regression model of Table 3:

```
-13.529
+ 491 x 0.116
+  26 x 0.101
+ 123 x 0.083
+ 201 x -0.160
+   4 x 0.305
+ 822 x 0.016
+ 699 x 0.014
+  28 x 0.107
```

(2)

The resulting predicted strength for this instance of test data is 51.75 MPa. As seen in the first line of targets of Table 2, the actual target strength is 57.92 MPa. The strength for this instance is underpredicted by 6.17 MPa, or 11% of the actual.

Fig. 1 shows the plot of errors in prediction for the linear regression model. These errors are known as residuals, the difference between predicted strength values and the actual strength values of the training/testing datasets. Ideally these residuals are randomly distributed and randomly scattered around the centerline, as appears here. There are some outliers that appear slightly removed from the cluster of points around the centerline, but no extreme outliers that would be a cause for concern. RMSE model loss for the testing dataset is 9.90

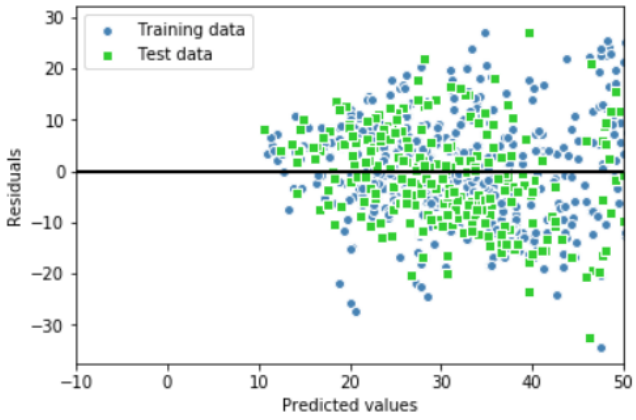

FIGURE 1. Linear Regression Model Residuals

B. Polynomial Regression Model

1) Polynomial n = 2 (quadratic)

Residuals for the quadratic polynomial regression model are shown in Fig. 2. As with the linear regression model, the residuals appear randomly distributed, indicating a reliable model. RMSE model loss for the testing dataset is 7.31

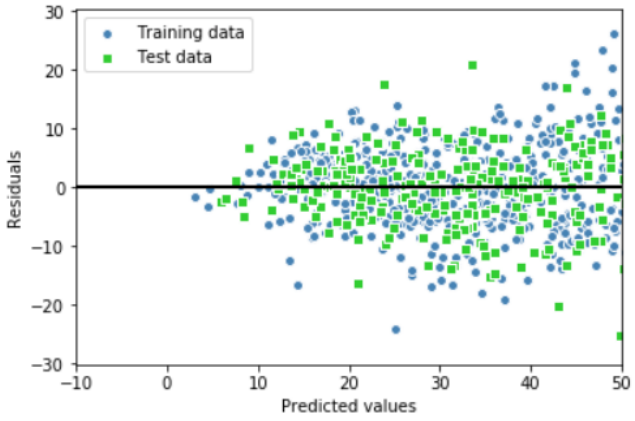

FIGURE 2. Quadratic Polynomial Regression Model Residuals

2) Polynomial n = 3 (cubic)

Residuals for the cubic polynomial regression model are shown in Fig. 3. The residuals still appear randomly distributed, while they are clustering closer to the centerline. As more points are clustering closer to the centerline, indicating higher accuracy and less error, some outliers continue to persist. RMSE model loss for the testing dataset is 6.54.

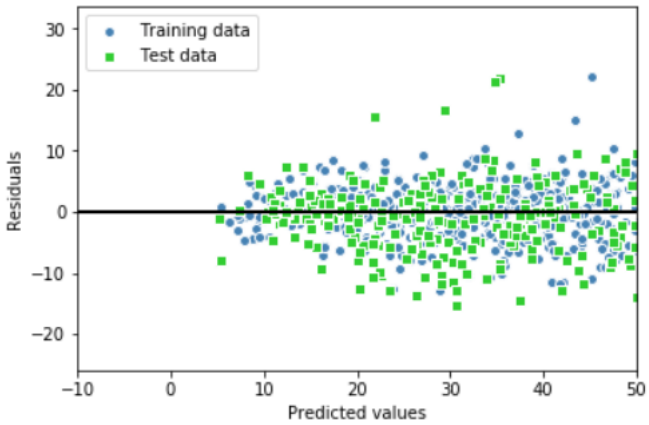

FIGURE 3. Cubic Polynomial Regression Model Residuals

C. Multilayered Perceptrons (Neural Network) Model

1) No Hidden Layers

The accuracy of the neural network models is measured with Root Mean Squared Error (RMSE) plotted for each of the model epochs. RMSE model loss after the final iteration is 6.23.

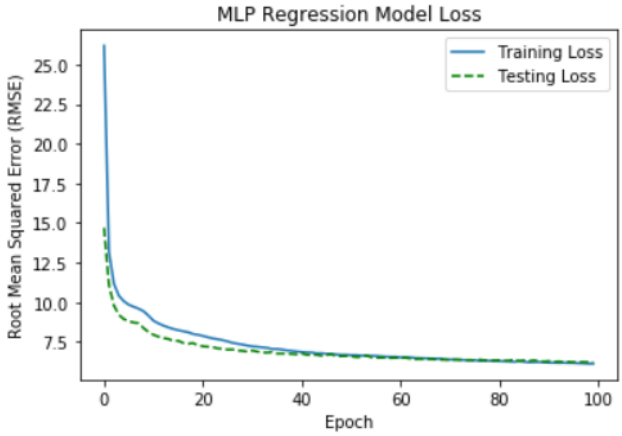

FIGURE 4. RMSE - No Hidden Layer Neural Network

2) One Hidden Layer

RMSE model loss after the final iteration is 5.90.

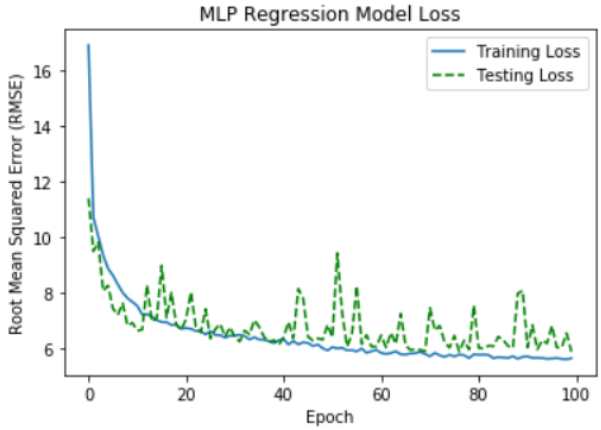

FIGURE 5. RMSE - One Hidden Layer Neural Network

D. Results Summary

Table 4 summarizes the root mean squared errors for the five models considered. RMSE is progressively reduced as each model is considered in order. The smallest RMSE value for the regression models is derived for the cubic polynomial regression model for the training dataset (RMSE = 4.30). This RMSE value increases by more than 50% when the testing dataset is used, indicating that this model overfits to the training dataset.

The smallest RMSE overall of ~4 is achieved with the hidden layer neural network and the training dataset. As with the cubic polynomial model, the training RMSE value appears to be significantly better than the testing value of 5.90, indicating overfitting. Due to the relatively low error, with consistency of results between training and testing, the neural network with no hidden layer is considered best. The quadratic polynomial regression model is considered a close runner-up, with slightly higher error, but highly consistent results between training and testing.

The issue of overfitting for the cubic polynomial model and the hidden layer neural network should be addressed before these models are considered valid.

TABLE 4. ROOT MEAN SQUARED ERRORS

Model	Training	Testing	Comments	Consistent	Error	Overall Rank
Linear Regression	10.56	9.90	relatively consistent between training and testing	yes	high	3
Polynomial Regression, n=2	7.37	7.31	highly consistent between training and testing	yes	medium	2
Polynomial Regression, n=3	4.30	6.54	overfitting indicated by higher test error	no	low	5
Neural Network, No Hidden Layer	~6	6.22	highly consistent between training and testing	yes	low	1
Neural Network, One Hidden Layer	~4	5.90	overfitting indicated by higher test error	no	low	4

VI. CONCLUSION

Several different machine learning models are compared in this analysis, with progressively increasing accuracy obtained. While increasingly complex models do appear to increase the overall accuracy, the error rates between training and testing diverge. The best balance of low error with consistent results comes from the simpler models.

These types of models may be useful in the civil engineering industry, given that current empirical equations for determining concrete's strength do not consider the supplementary cementitious materials contained in high-performance concrete. The usefulness of these models, and the required accuracy, will depend on the application.

Beyond predicting strength of existing formulations, the linear regression model provides intuition regarding the contributions to a formulation's strength by the individual components. This type of knowledge could allow higher strength formulations to be developed in the future.

REFERENCES

[1] I-Cheng Yeh, "Modeling of strength of high performance concrete using artificial neural networks," *Cement and Concrete Research*, Vol. 28, No. 12, pp. 1797-1808 (1998).

[2] M. Tutterrow. CPSC-55000. Assignment Submission: "Week 7 Assignment - Programming Assignment (Final Project 2 of 3)", Department of Computer and Mathematical Sciences, Lewis University, Romeoville, IL, July 3, 2018.

[3] UCI Machine Learning Repository, "Concrete Compressive Strength Data Set". [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength [Accessed July 3, 2018].

[4] S. Raschka, V. Mirjalili, *Python Machine Learning Second Edition*. Birmingham, UK: Packt, 2017.