Author: ROSHNA DHAKAL

Data Analysis of Concrete Strength Part II

Applied Machine Learning, BSc. Computing
Level 5 , Semester 2
2025

Table of Contents

Data after pre-processing Summary

The concrete strength training data-set consists of 722 samples with 9 columns, while the testing data-set has 308 samples. Variables include concrete strength, age, slag, fly ash, water, Superplasticizer, and aggregates. The IQR method was used to remove outliers and handles missing values and the data was scaled using z-score normalization for model performance improvement (Fan et al., 2021).

MODELLING

The main objective of using regression model is to understand factors influencing concrete strength prediction (Obianyo et al., 2020) . We will use five linear regression predictive models for data analysis.

1.1 Linear Regression

The lm() function of the caret package was used to train a linear regression model. The data set was split in half: 70% for training and 30% for testing. we evaluate the fit of the model using the summary() function as shown on fig 1 below, which provides information about the model coefficient, R squared and other relevant statistics (Tzou et al., 2023).

```
> summary(lm_model)

Call:
lm(formula = Strength ~ ., data = train_set)

Residuals:
    Min      1Q  Median      3Q     Max
-39.226  -4.170   0.170   4.628  19.426

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        72.886758  24.996707   2.916  0.00378 **
Cement              0.074895   0.008051   9.302  < 2e-16 ***
Blast.Furnace.Slag  0.046449   0.010013   4.639 4.97e-06 ***
Fly.Ash            -0.001500   0.012454  -0.120  0.90422
Water              -0.253273   0.044404  -5.704 2.50e-08 ***
Superplasticizer    0.417338   0.130095   3.208  0.00146 **
Coarse.Aggregate   -0.017685   0.009302  -1.901  0.05808 .
Fine.Aggregate     -0.020891   0.011204  -1.865  0.06307 .
Age                 0.503530   0.028138  17.895  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.915 on 349 degrees of freedom
Multiple R-squared:  0.7183,    Adjusted R-squared:  0.7119
F-statistic: 111.3 on 8 and 349 DF,  p-value: < 2.2e-16
```

Fig 1: summary of lm_model

```
> # Make predictions
> predictions <- predict(lm_model, newdata = test_set)
> # Evaluate predictions
> accuracy <- sqrt(mean((predictions - test_set$Strength)^2))
> accuracy
[1] 8.034698
> predictions_train <- predict(lm_model, newdata = train_set)
> accuracy_train <- sqrt(mean((predictions_train - train_set$Strength)^2))
> accuracy_train
[1] 7.815286
```

Fig 2: prediction and accuracy of linear regression

The linear regression model's RMSE on the testing data set is 8.03, indicating an average difference between predicted and actual concrete strength values, while the RMSE on the training data set is 7.82, indicating slightly better performance.

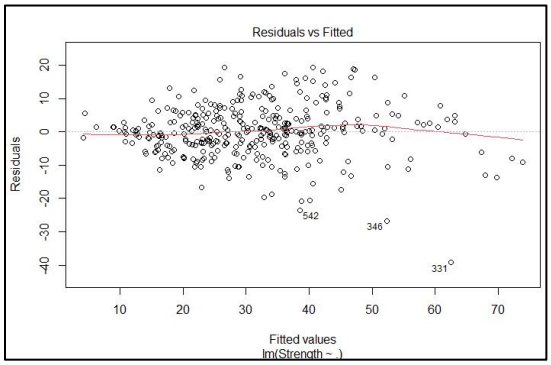

Fig 3: Residuals vs. Fitted Plot

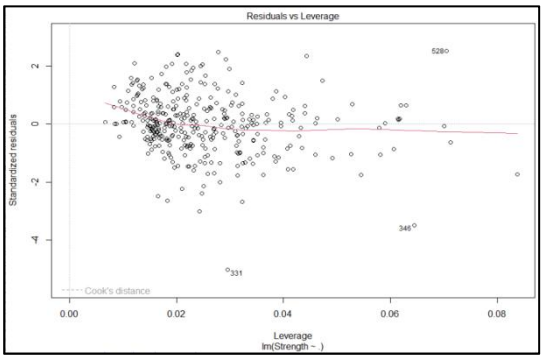

Fig 4: Residuals vs. Leverage Plot

The plot in Fig 3 reveals key data points affecting regression model coefficients, with high residuals and leverage impacting the model. The residual vs. fitted plot in fig 4 indicates heteroscedasticity.

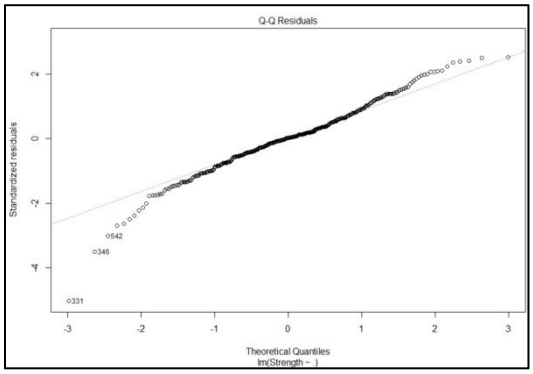

Fig 5: Q-Q Residuals

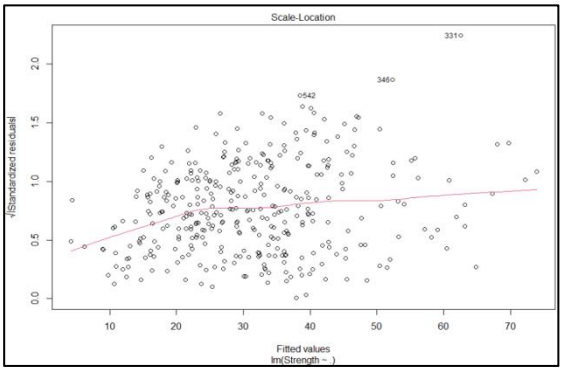

Fig 6: Scale-Locations

The Q-Q residual distribution is compared to the normal distribution using a residual plot. Deviations from the standard may indicate non-linear relationships or outliers (Feng et al., 2020). The assumption of constant variance is checked using the scale position plot, examining the residual distribution's constancy at different predictor variable levels.

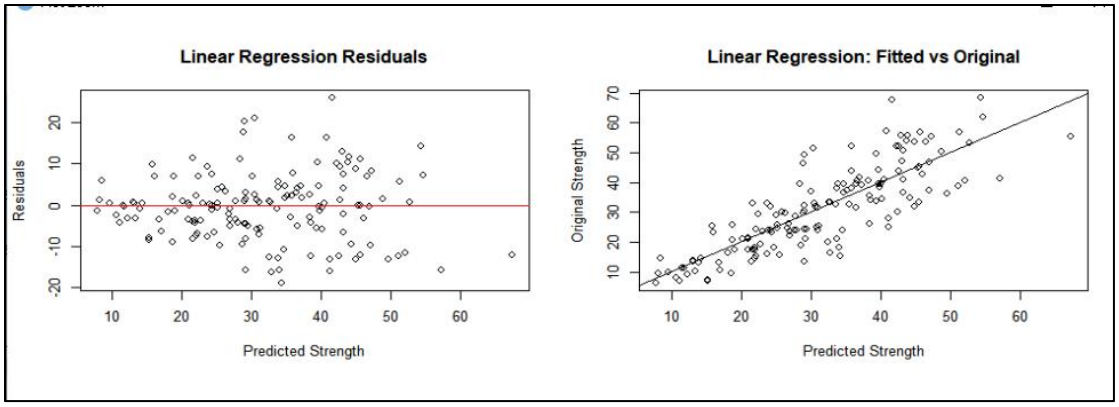

Fig 7: linear Regression residuals and fitted vs.original plot

The residuals method plots concrete strength against predicted strength, with zero residuals indicating unbiased predictions. Perfect predictions are near the diagonal line, with model accuracy increasing as points approach this line.

1.2 Decision Tree Regression Model

The decision tree regression model was trained using the rpart() function (Vigneshwari et al., 2019), and its performance was evaluated using the summary() function, obtaining information on distributions, error metrics, and tree structure.

```
> summary(cart_model)
Call:
rpart(formula = Strength ~ ., data = train_set)
  n= 358

          CP nsplit rel error    xerror       xstd
1 0.21898773      0 1.0000000 1.0060185 0.06723237
2 0.12090427      1 0.7810123 0.7906436 0.05808815
3 0.08964770      2 0.6601080 0.7674244 0.05550426
4 0.07105720      3 0.5704603 0.7207978 0.05650327
5 0.04599695      4 0.4994031 0.6441150 0.05317622
6 0.03916565      5 0.4534062 0.6042672 0.05041674
7 0.02730068      6 0.4142405 0.5324171 0.04722175
8 0.02506737      7 0.3869398 0.5232895 0.04673342
9 0.02251785      8 0.3618725 0.5055809 0.04496424
10 0.02181403     9 0.3393546 0.4934780 0.04468519
11 0.01414577    10 0.3175406 0.4682167 0.04305995
12 0.01183200    12 0.2892490 0.4513463 0.04192180
13 0.01131571    13 0.2774170 0.4461612 0.04170033
14 0.01115252    14 0.2661013 0.4500092 0.04168617
15 0.01094403    15 0.2549488 0.4510354 0.04180193
16 0.01000000    16 0.2440048 0.4484985 0.04175139

Variable importance
            Cement                Age Blast.Furnace.Slag
                23                 23                 15
  Superplasticizer    Coarse.Aggregate              Water
                13                 10                 10
           Fly.Ash     Fine.Aggregate
                 4                  2

Node number 1: 358 observations,    complexity param=0.2189877
  mean=31.36902, MSE=216.8424
  left son=2 (102 obs) right son=3 (256 obs)
```

```
Node number 2: 102 observations,    complexity param=0.0710572
  mean=20.45206, MSE=135.1421
  left son=4 (72 obs) right son=5 (30 obs)
  Primary splits:
      Cement           < 354.5    to the left,  improve=0.4001700, (0 missing)
      Superplasticizer < 10.25    to the left,  improve=0.3218109, (0 missing)
      Coarse.Aggregate < 946.9    to the right, improve=0.3178070, (0 missing)
      Water            < 168.05   to the right, improve=0.2721283, (0 missing)
      Fine.Aggregate   < 756.7    to the right, improve=0.1712717, (0 missing)
  Surrogate splits:
      Superplasticizer < 10.65    to the left,  agree=0.814, adj=0.367, (0 split)
      Coarse.Aggregate < 946.9    to the right, agree=0.794, adj=0.300, (0 split)
      Water            < 165.25   to the right, agree=0.755, adj=0.167, (0 split)
      Fly.Ash          < 122.3    to the left,  agree=0.735, adj=0.100, (0 split)

Node number 3: 256 observations,    complexity param=0.1209043
  mean=35.71875, MSE=182.9889
  left son=6 (186 obs) right son=7 (70 obs)
  Primary splits:
      Cement           < 311.45   to the left,  improve=0.20035680, (0 missing)
      Age              < 51.29161 to the left,  improve=0.14547600, (0 missing)
      Water            < 162.05   to the right, improve=0.14398630, (0 missing)
      Blast.Furnace.Slag < 17     to the left,  improve=0.11590490, (0 missing)
      Fly.Ash          < 174.8    to the right, improve=0.09484222, (0 missing)
  Surrogate splits:
      Coarse.Aggregate < 824.65   to the right, agree=0.746, adj=0.071, (0 split)
      Superplasticizer < 16.25    to the left,  agree=0.738, adj=0.043, (0 split)
      Water            < 214.3    to the left,  agree=0.730, adj=0.014, (0 split)

Node number 4: 72 observations,    complexity param=0.01131571
  mean=15.70514, MSE=55.09474
  left son=8 (36 obs) right son=9 (36 obs)
  Primary splits:
      Cement           < 250.8    to the left,  improve=0.2214452, (0 missing)
      Blast.Furnace.Slag < 187.95   to the left,  improve=0.1909759, (0 missing)
```

Fig 8: summary of cart_model

```
> # Make predictions
> predictions_tree <- predict(cart_model, newdata = test_set)
> # Evaluate predictions
> accuracy_tree <- sqrt(mean((predictions_tree - test_set$Strength)^2))
> accuracy_tree
[1] 9.598024
> predictions_tree <- predict(cart_model, newdata = train_set)
> accuracy_tree_train <- sqrt(mean((predictions_tree - train_set$Strength)^2))
> accuracy_tree_train
[1] 7.273966
```

Fig 9: Prediction and Accuracy (cart_model)

The accuracy of model was evaluated using the root mean square error after generating predictions using the predict() function which was 9.59 and 7.27 for test

and train data respectively.

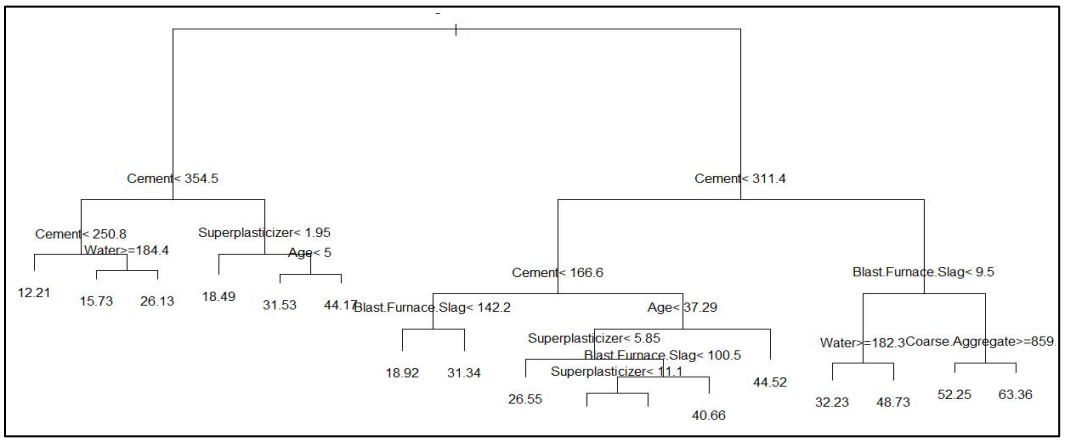

Fig 10: Decision Tree Model Prediction

The visualization in above fig 10 divides feature space into regions based on predictor variables, explaining how the model generates predictions. Decisions are represented by nodes in a tree, with possible consequences represented by branches. Predicted strength values for concrete are found at terminal nodes, known as leaves.

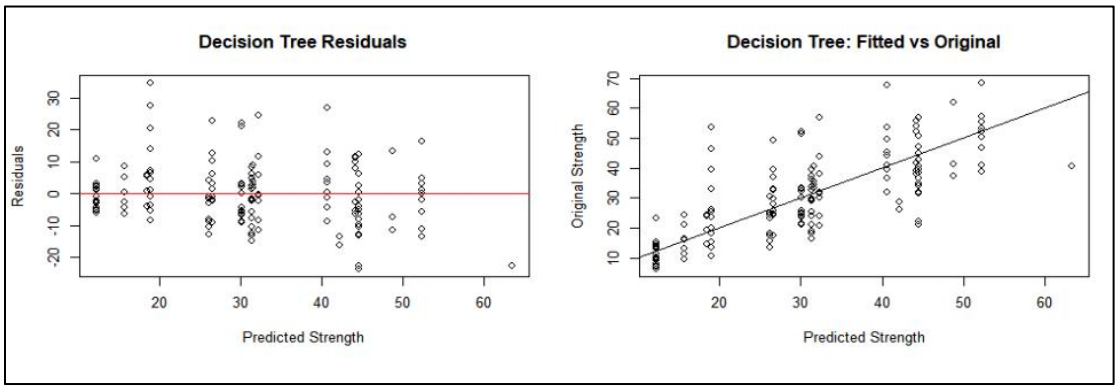

Fig 11: Decision tree residuals and fitted vs. original plots

Additionally, the diagnostic plots showing both residual values and predicted values were also created to assess model performance and identify potential errors.

1.3 Random Forest Regression Model

The randomForest() method of the randomForest package was used to train the random forest regression model (Nguyen-Sy et al., 2020). As with the other models, the data set was divided into training and testing data sets in a 70-30 ratio.

```
> predictions_random <- predict(rf_model, newdata = test_data_imputed)
> # Evaluate predictions
> accuracy_random <- sqrt(mean((predictions_random - test_data_imputed$Strength)^2))
> accuracy_random
[1] 7.711118
> predictions_random_train <- predict(rf_model, newdata = train_data_imputed)
> accuracy_random_train <- sqrt(mean((predictions_random_train - train_data_imputed$Strength)^
2))
> accuracy_random_train
[1] 3.673824
```

Fig 12: Prediction and Accuracy (random forest)

The predict() function was used to make the predictions and the root mean square error (RMSE) was used to calculate mode accuracy.

```
> summary(rf_model)
                Length Class  Mode
call              3    -none- call
type              1    -none- character
predicted       510    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times       510    -none- numeric
importance        8    -none- numeric
importanceSD      0    -none- NULL
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y               510    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
>
```

Fig 13: Summary of rf_model

The summary() function information about the overall performance of the random forest, including number of trees, mean squared error (MSE), and variable significance.

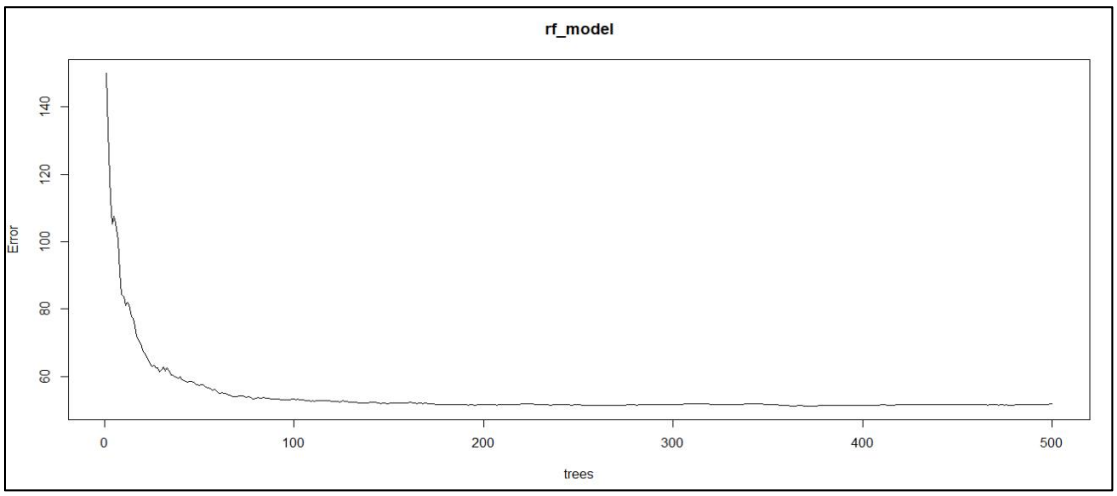

Fig 14: Random forest regression model plot

The plot(rf_model) function generates diagnostic plots for the random forest model, providing insight into the significance of predictor variables and non-linear relationships between them and the target variable, identifying influential variables and revealing tree structure.

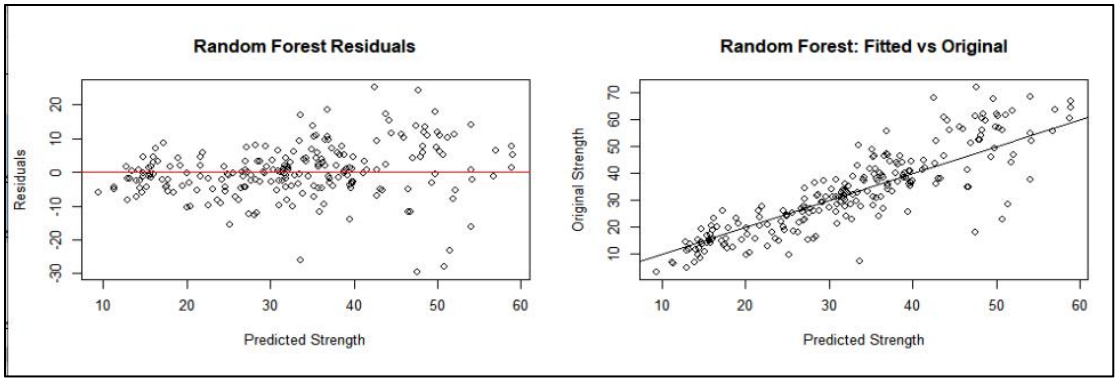

Fig 15: Random Forest residuals and fitted vs. original plots

The first plot shows residuals against predicted concrete strength values, with zero residuals in the ideal scenario. The second plot shows the relationship between predicted and original concrete strength values, with a diagonal line indicating perfect predictions.

1.4 KNN Regression Model

The train() function of the Caret package was used to build a K-Nearest Neighbors (KNN) regression model. This model predicts the target variable by calculating the distance between the data points.

```
> predictions_KNN <- predictions_KNN[1:nrow(test_set)]
> accuracy_knn <- sqrt(mean((predictions_KNN - test_set$Strength)^2))
> accuracy_knn
[1] 18.46102
> predictions_KNN_train <- predictions_KNN_train[1:nrow(train_set)]
> accuracy_knn_train <- sqrt(mean((predictions_KNN_train - train_set$Strength)^2))
> accuracy_knn_train
[1] 18.36707
```

Fig 16: Prediction and accuracy (KNN)

The model's predictions are approximately 18.46 units away from the actual concrete strength values in the testing data set and 18.36 units away in train, respectively.The summary of model is shown in fig below:

```
> summary(model)
           Length Class      Mode
learn      2      -none-     list
k          1      -none-     numeric
theDots    0      -none-     list
xNames     8      -none-     character
problemType 1     -none-     character
tuneValue  1      data.frame list
obsLevels  1      -none-     logical
param      0      -none-     list
>
```

Fig 17: Summary of KNN model

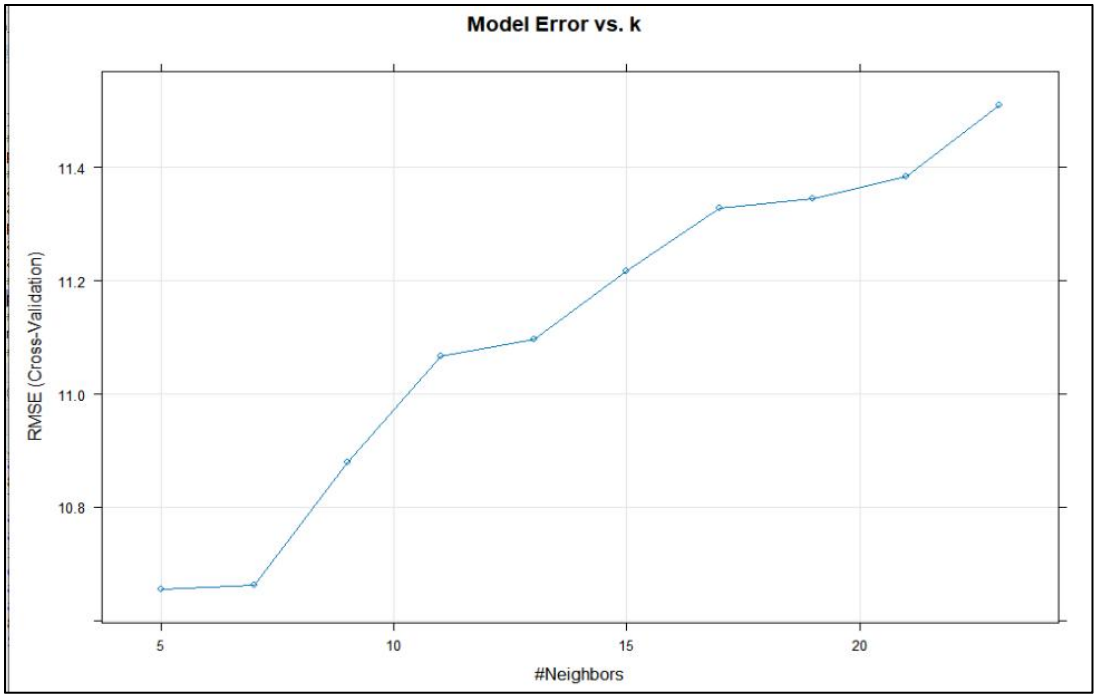

Fig 18: KNN model error vs. K plot

Tenfold cross-validation was employed to adjust model parameters and reduce errors. The root mean square error (RMSE) was used to assess forecast accuracy (Vakharia & Gujar, 2019). The "Model Error vs. k" visualization illustrates the relationship between 'k' and RMSE, aiding in determining the ideal value of 'k' to minimize errors.

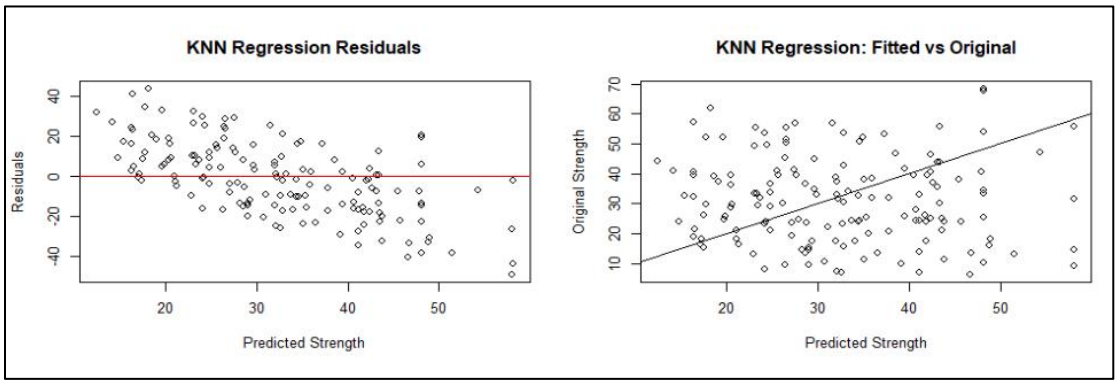

Fig 19: KNN regression residuals and fitted vs. Original plot

The "KNN Regression Residuals Plot" displays the difference between actual and predicted concrete strength values. Perfect predictions would be indicated by the red line along the diagonal (y = x) , while deviations from this line indicate model predictions errors.

1.5 KNN Regression Model with New Parameter

A K-Nearest Neighbors (KNN) regression model was trained using the train() method of the caret package to maximize the choice of the tuning parameter "k". The tune Grid parameter was set to try various values of k (3, 5, and 7) to minimize the RMSE.

```
> predictions_KNN_new <- predict(model_new, newdata = test_data_imputed)
> # Evaluate predictions
> accuracy_knn_new <- sqrt(mean((predictions_KNN_new - test_data_imputed$Strength)^2))
> accuracy_knn_new
[1] 10.36748
> #for trainset
> predictions_KNN_new_train <- predict(model_new, newdata = train_data_imputed)
> accuracy_knn_new_train <- sqrt(mean((predictions_KNN_new_train - train_data_imputed$Strength)
^2))
> accuracy_knn_new_train
[1] 8.784008
```

Fig 20: predictions and accuracy (KNN_New model)

In the testing data set, the model's predictions were approximately 10.37 units off from the actual concrete strength values and 8.78 units off in train data set.

```
> summary(model_new)
            Length Class      Mode
learn       2      -none-     list
k           1      -none-     numeric
theDots     0      -none-     list
xNames      8      -none-     character
problemType 1      -none-     character
tuneValue   1      data.frame list
obsLevels   1      -none-     logical
param       0      -none-     list
>
```

Fig 21: Summary of knn_new model

A summary of the new KNN regression model (model_new) contains details about its parameters and performance, including the ideal tuning value (k).

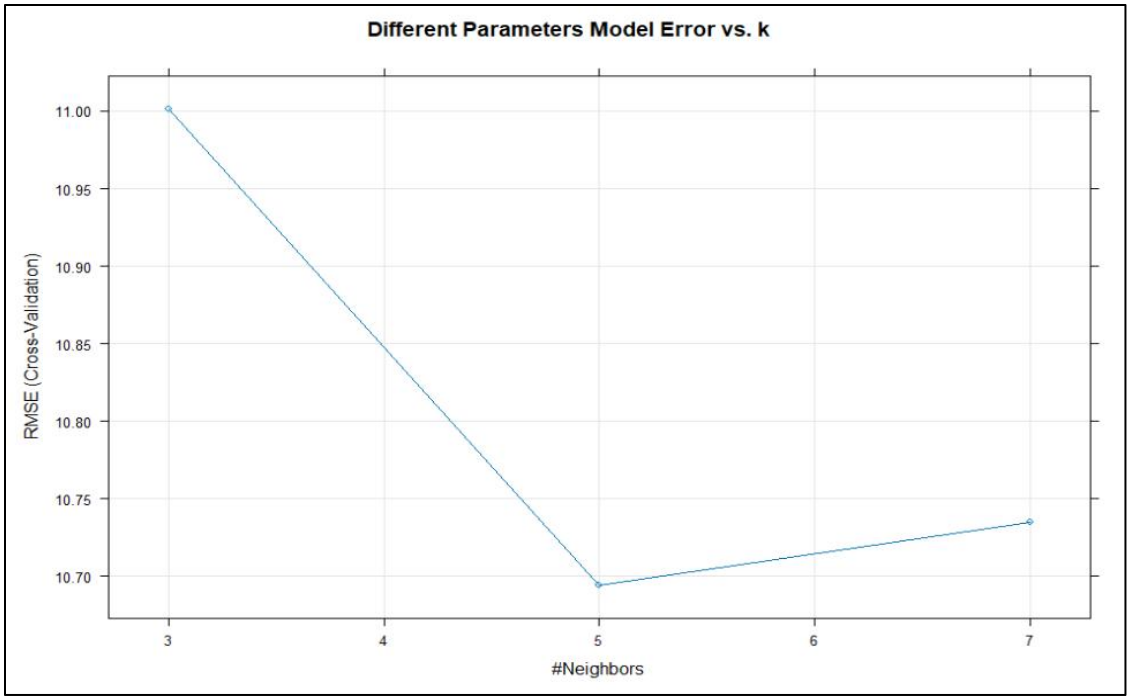

Fig 22: Different Parameters KNN model error vs. K plot

The model performance was calculated utilizing cross-validation with tenfold error. The "Different Parameters Model Error vs. k" figure shows the model error (RMSE) for different tuning parameter values, to find the value that minimizes root mean square error and indicates the ideal number of nearest neighbors for prediction. The optimal model is represented as the lowest RMSE, k =5.

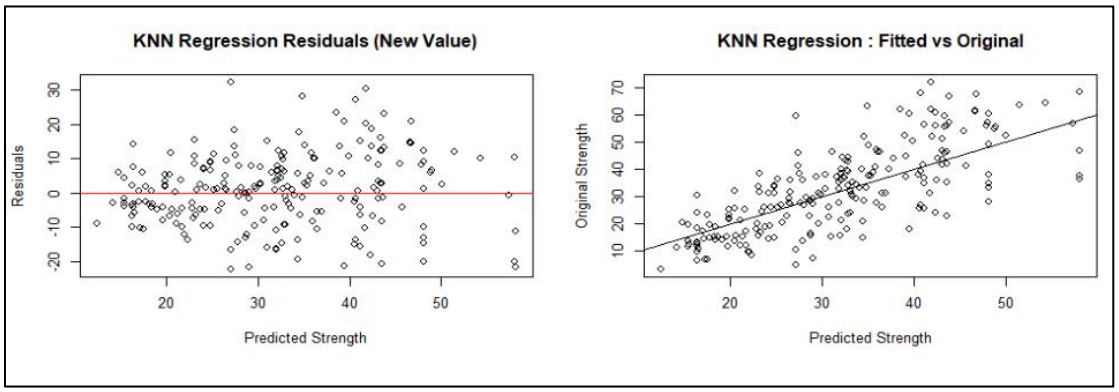

Fig 23: KNN Regression residuals and fitted vs. Original plot for new value

Evaluation of model data fit and error distribution between predicted and actual values of concrete strength is simplified with these residual visualizations and fitted versus original plots.

Result of Regression Models

The performance of each regression model was demonstrated by the results given below:

S. N.	Model	R2	AdjustR2	MSE	RMSE	MAE
1	LR	0.718326727	0.71187003	64.55637	8.034698	6.097285
2	Decision Tree	0.555209804	0.52701888	418.94180	9.598024	16.293280
3	Random Forest	0.767041969	0.75729024	59.46134	7.711118	5.550879
4	KNN	0.004867616	-0.05080413	199.75838	18.461020	10.969422
4	KNN(different value)	0.567699378	0.55168824	107.48469	10.367482	8.182018

```
> print(results)
                        Model          R2   Adjust_R2       MSE      RMSE       MAE
1          Linear Regression 0.718326727  0.71187003  64.55637  8.034698  6.097285
2              Decision Tree 0.555209804  0.52701888 418.94180  9.598024 16.293280
3              Random Forest 0.767041969  0.75729024  59.46134  7.711118  5.550879
4                        KNN 0.004867616 -0.05080413 199.75838 18.461020 10.969422
5 KNN (Different Parameters) 0.567699378  0.55168824 107.48469 10.367482  8.182018
> )
```

Fig 24: results five regression models

Model Interpretation

 # Random Forest

The best selected model for concrete strength prediction is Random forest. The model's significance metrics demonstrate the relative importance of predictor variables in the overall predictive accuracy of a model and outperforms other models in comparison..

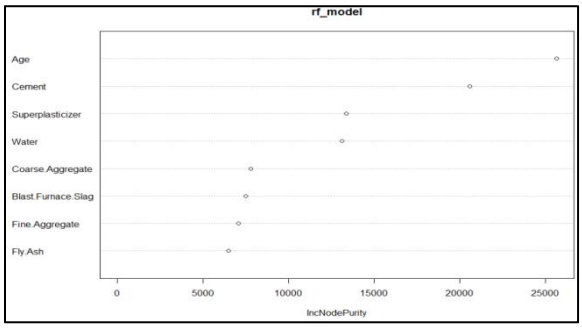

Fig 25: feature importance plot(rf_model)

```
> importance(rf_model)
                  IncNodePurity
Cement                20599.352
Blast.Furnace.Slag     7483.433
Fly.Ash                6497.986
Water                 13115.184
Superplasticizer      13372.964
Coarse.Aggregate       7805.029
Fine.Aggregate         7046.389
Age                   25656.385
> )
```

Fig 26: important variables (rf_model)

As shown in fig 25 and 26, the random forest Model Interpretation is shown below:

• Cement: Highest predictor of concrete strength.
• Age: Influential in predicting concrete strength.
• Superplasticizer: Enhances model accuracy.
• Blast.furnace.Slag, Water, Fine.Aggregate, Coarse.Aggregate: Moderate significance values support prediction abilities.
• Fly.Ash: Adds to model accuracy, not as significant as other components.

 # Linear regression

Each predictor variable's contribution to the predicted results (concrete strength) can be observed by the coefficients derived from the linear regression model.

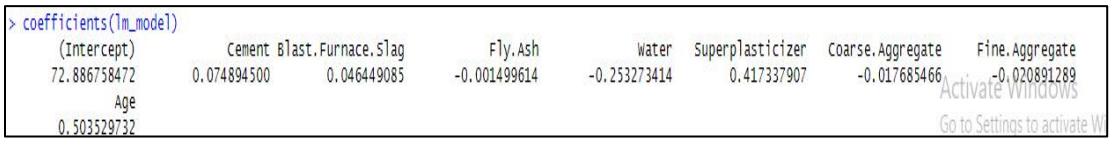

Fig 27: coefficients if linear regression model

The interpretation based on fig25 above is shown below:
Concrete Strength Analysis
• Intercept: Represents strength when all predictors are zero.
• Cement: Strength improves with every unit increase in cement content.
• Blast furnace slag slightly increases strength.
• Fly ash: Higher content correlates with decreased strength.
• Water: Significantly reduces strength.
• Superplasticizer: Increases strength with coefficient 0.417.
• Aggregate: Minimal impact on strength.
• Age: Significantly increases strength with coefficient 0.504.

#Decision tree

The decision tree calculates the compressive strength of concrete based on its age, water content, and blast furnace slag amount. A water content of less than five years old results in an average strength of 354.5 MPa. If the water content is greater than 184.4 but less than 26.13 MPa, the strength is 311.4 MPa.hence, this moel is less suitable.

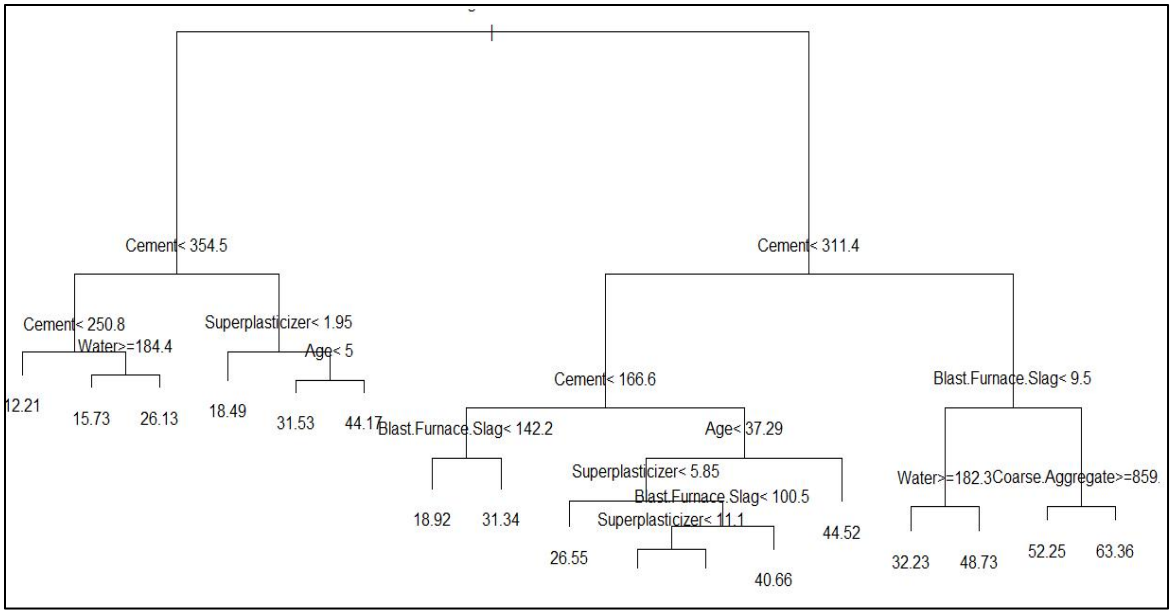

Fig 28: Decision Tree Model

#KNN Model and KNN model with different value

The k-Nearest Neighbors (KNN) model is a non-parametric regression approach that uses cross-validation to determine the optimal parameter setting (Ali et al., 2020). The optimal tuning parameter is 7, based on the seven closest neighbors for KNN model, and the best tuning parameter is 5, compared to 3, 5, and 7, with various values.The KNN model has the highest MSE and negative r2 value which means over fitting model. Hence, it is not a suitable model to select.

```
> model_new$bestTune
  k
2 5
> model$bestTune
  k
2 7
>
```

Fig 29: Best tune parameters for KNN models

Conclusions

The study analyzes concrete properties and compares various regression models to determine compressive strength. The Random Forest model outperforms other models with the lowest measurement error and the best R2 score of 0.7670. The most important predictors variables are cement, age, and Superplasticizer, which show strong correlations with the target variable. Linear regression also shows encouraging results with an R2 of 0.7183. Decision tree and KNN models show slightly lower prediction accuracy. The KNN model improves with parameter adjustment but is not compatible compare with other models.

Bibliography

Nguyen-Sy, T., Wakim, J., To, Q.-D., Vu, M.-N., Nguyen, T.-D. & Nguyen, T.-T. (2020) Predicting the Compressive Strength of Concrete from Its Compositions and Age Using the Extreme Gradient Boosting Method. Construction and Building Materials [Online], 260 November, p. 119757. Available from: <http://dx.doi.org/10.1016/j.conbuildmat.2020.119757>.

Obianyo, I. I., Anosike-Francis, E. N., Ihekweme, G. O., Geng, Y., Jin, R., Onwualu, A. P. & Soboyejo, A. B. O. (2020) Multivariate Regression Models for Predicting the Compressive Strength of Bone Ash Stabilized Lateritic Soil for Sustainable Building.

Fan, C., Chen, M., Wang, X., Wang, J. & Huang, B. (2021) A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data. Frontiers in Energy Research [Online], 9 March. Available from: <http://dx.doi.org/10.3389/fenrg.2021.652801>.

Tzou, S.-J., Peng, C.-H., Huang, L.-Y., Chen, F.-Y., Kuo, C.-H., Wu, C.-Z. & Chu, T.-W. (2023) Comparison between Linear Regression and Four Different Machine Learning Methods in Selecting Risk Factors for Osteoporosis in a Chinese Female Aged Cohort. Journal of the Chinese Medical Association [Online], 86 (11) September, pp. 1028–1036. Available from: <http://dx.doi.org/10.1097/jcma.0000000000000999>.

Feng, C., Li, L. & Sadeghpour, A. (2020) A Comparison of Residual Diagnosis Tools for Diagnosing Regression Models for Count Data. BMC Medical Research Methodology [Online], 20 (1) July. Available from: <http://dx.doi.org/10.1186/s12874-020-01055-2>.

Vakharia, V. & Gujar, R. (2019) Prediction of Compressive Strength and Portland Cement Composition Using Cross-Validation and Feature Ranking Techniques. Construction and Building Materials [Online], 225 November, pp. 292–301. Available from: <http://dx.doi.org/10.1016/j.conbuildmat.2019.07.224>.

Vigneshwari, S., Bharathi, B., Sasikala, T. & Mukkamala, S. (2019) A Study on the Application of Machine Learning Algorithms Using R. Journal of Computational and Theoretical Nanoscience [Online], 16 (8) August, pp. 3466–3472. Available from: <http://dx.doi.org/10.1166/jctn.2019.8309>.

Ali, A., Hamraz, M., Kumam, P., Khan, D. M., Khalil, U., Sulaiman, M. & Khan, Z. (2020) A K-Nearest Neighbours Based Ensemble via Optimal Model Selection for Regression. *IEEE Access* [Online], 8, pp. 132095–132105. Available from: <http://dx.doi.org/10.1109/access.2020.3010099>.