REGRESSION ANALYSIS-2 PROJECT REPORT

DATA ANALYSIS USING PISA STUDENT DATA

Lecture: Regression Analysis 2
Teacher: Gülhayat Gölbaşı Şimşek

Metin Mirza VATANSEVER - 20023055
Münir Göktan SOYKURT - 20023048
Enes COŞKUN - 21023005

ABOUT PROJECT

The main goal of this project is to analyse PISA math score of Great Britian students. Analyse will be done by using various regression models in R studio.

ABOUT DATA

The given data is considered a big one. The data contains too many variables. The first step of this process is to analyse the data and choose the variables to work on. We will walk through the variables will be needed during this analysis. To work on this complicated data we create new variables with given variables.

```
# RSN = (PV1MPRE+...+PV10MPRE)/10    - Score of math reasoning
# UNDR = (PV1MPIN+...+PV10MPIN)/10   - Score of interpreting applying and evaluating of math
# MUNDR = (PV1MPFS+...+PV10MPFS)/10 - Score of math formulation
# EMPL = (PV1MPEM+...+PV10MPEM)/10   - Score of employing math concepts
# CRT = (PV1MCUD+...+PV10MCUD)/10    - Score of uncertainty and data
# SPC = (PV1MCSS+...+PV10MCSS)/10    - Score of space and shape understanding
# MATH = (PV1MATH+...+PV10MATH)/10   - Score of math
```

0 3

SUMMARY OF THE DATA

```
> describe(dataset)
              vars     n        mean       sd   median     trimmed     mad      min      max range  skew
CNT              1 12972        1.00     0.00        1        1.00    0.00        1        1     0   NaN
CNTRYID          2 12972      826.00     0.00      826      826.00    0.00      826      826     0   NaN
CNTSCHID         3 12972 82612714.48 21624.71 82600257 82609628.39  192.74 82600001 82650120 50119  1.15
CNTSTUID         4 12972 82618948.52 19761.23 82610342 82616831.94 7652.44 82600003 82654772 54769  1.02
CYC              5 12972        1.00     0.00        1        1.00    0.00        1        1     0   NaN
NatCen           6 12972    82725.11    43.37    82700    82718.89    0.00    82700    82800   100  1.15
STRATUM          7 12972       17.95    12.96       13       17.41   14.83        1       39    38  0.35
SUBNATIO         8 12972  8260502.16   867.30  8260000  8260377.72    0.00  8260000  8262000  2000  1.15
REGION           9 12972    82613.84     3.64    82612    82613.42    1.48    82611    82620     9  1.02
OECD            10 12972        1.00     0.00        1        1.00    0.00        1        1     0   NaN
ADMINMODE       11 12972        2.00     0.00        2        2.00    0.00        2        2     0   NaN
LANGTEST_QQQ    12 11537      314.98    11.26      313      313.00    0.00      313      379    66  5.51
LANGTEST_COG    13 12972      319.61    57.08      313      313.00    0.00      313      979   666 11.08
BOOKID          14 12972       13.27     7.71       13       13.06    8.90        1       36    35  0.28
```

The data contains a lot of variables. The output is too big to fit in so we put a little piece of the output.

```
describe(dataset$MATH)
 vars     n   mean    sd median trimmed   mad    min    max  range skew kurtosis  se
    1 12972 481.82 91.52 480.74   480.9 95.88 182.77 834.49 651.72  0.1    -0.34 0.8
describe(dataset$SCIE)
 vars     n   mean    sd median trimmed    mad    min    max range skew kurtosis   se
    1 12972 492.74 96.77 492.23  492.21 103.25 193.37 837.17 643.8 0.06    -0.43 0.85
describe(dataset$READ)
 vars     n   mean    sd median trimmed    mad    min    max  range skew kurtosis   se
    1 12972 490.45 97.04 492.52   491.7 100.79 154.41 816.35 661.95 -0.1    -0.28 0.85
describe(dataset$CRT)
 vars     n   mean    sd median trimmed    mad    min   max  range skew kurtosis   se
    1 12972 490.34 99.85 488.78  489.47 103.47 168.18 889.8 721.62 0.09    -0.27 0.88
```

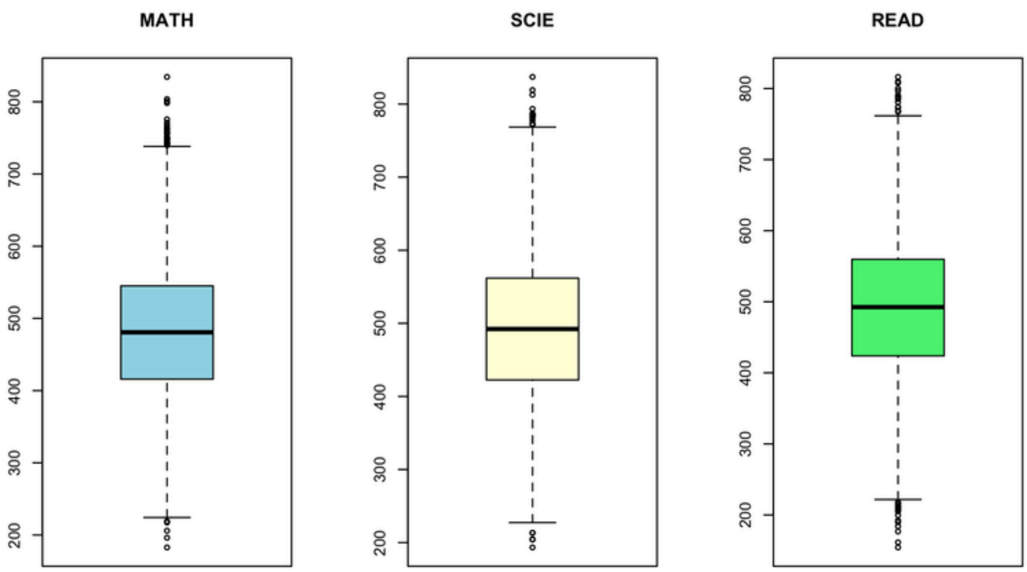
Output of descriptive statistics of variables that we will be use in regression models.

04

```
summary(dataset$MATH)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
182.8   415.9   480.7   481.8   545.1   834.5
summary(dataset$SCIE)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
193.4   422.6   492.2   492.7   561.8   837.2
summary(dataset$READ)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
154.4   423.9   492.5   490.4   559.8   816.4
```

Summary of the dependent and independent variables that we will use in regression models.

```
par(mfrow=c(1,3))
boxplot(dataset$MATH, col = "lightblue", main = "MATH")
boxplot(dataset$SCIE, col = "lightyellow", main = "SCIE")
boxplot(dataset$READ, col = "lightgreen", main = "READ")
par(mfrow=c(1,1))
```



The box plot of the variables. So we can observe the outliers

05 DATA PREPERATION

This chapter is about preperation for convinient use of data in regression models. The given rule was to choose random two students from each school.

```
schoolid <- str(dataset$CNTSCHID)
schoolid <- as.character(dataset$CNTSCHID)
schoolid
dataset_nondup <- dataset[!duplicated(dataset$CNTSCHID), ]
school_counts <- dataset_nondup %>%
  count(CNTSCHID, sort = TRUE)
print(school_counts)
```

This code gives us how many unique school ids PISA data has.

```
set.seed(123)
random_students <- dataset %>%
  group_by(CNTSCHID) %>%
  sample_n(size = 2, replace = FALSE) %>%
  ungroup()
```

This code randomly chooses two students from each unique school id. So with this code we create our data that we will work on.

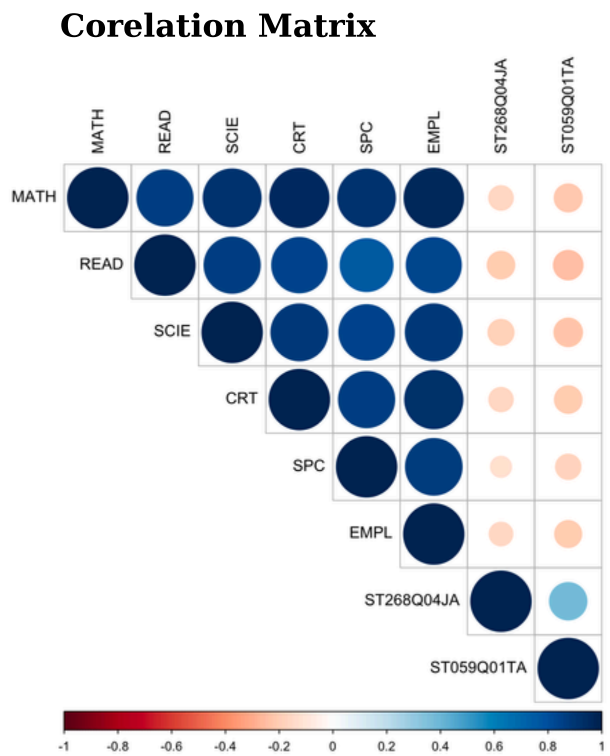
```
> random_students
# A tibble: 902 × 657
   CNT   CNTRYID CNTSCHID  NTSTUID CYC   NatCen STRATUM SUBNATIO REGION  OECD ADMINMODE LANGTEST_QQQ LANGTEST_COG
   <chr>   <dbl>    <dbl>    <dbl> <chr>  <dbl> <chr>      <dbl>  <dbl> <dbl>     <dbl>        <dbl>        <dbl>
 1 GBR       826 82600001  2609582 08MS   82700 QUK02    8260000  82611     1         2          313          313
 2 GBR       826 82600001  2612396 08MS   82700 QUK02    8260000  82611     1         2          313          313
 3 GBR       826 82600002  2604924 08MS   82700 QUK03    8260000  82611     1         2          313          313
 4 GBR       826 82600002  2601162 08MS   82700 QUK03    8260000  82611     1         2          313          313
 5 GBR       826 82600003  2604422 08MS   82700 QUK03    8260000  82611     1         2          313          313
 6 GBR       826 82600003  2610261 08MS   82700 QUK03    8260000  82611     1         2          313          313
 7 GBR       826 82600004  2609871 08MS   82700 QUK25    8260000  82612     1         2          313          313
 8 GBR       826 82600004  2606038 08MS   82700 QUK25    8260000  82612     1         2          313          313
 9 GBR       826 82600005  2601525 08MS   82700 QUK02    8260000  82611     1         2          313          313
10 GBR       826 82600005  2612599 08MS   82700 QUK02    8260000  82611     1         2          313          313
# ℹ 892 more rows
# ℹ 644 more variables: BOOKID <dbl>, ST001D01T <dbl>, ST003D02T <dbl>, ST003D03T <dbl>, ST250D06JA <dbl>,
#   ST250D07JA <dbl>, ST251Q01JA <dbl>, ST251Q02JA <dbl>, ST251Q03JA <dbl>, ST251Q04JA <dbl>, ST251Q06JA <dbl>,
#   ST251Q07JA <dbl>, ST251D08JA <dbl>, ST251D09JA <dbl>, ST253Q01JA <dbl>, ST254Q01JA <dbl>, ST254Q02JA <dbl>,
#   ST254Q03JA <dbl>, ST254Q04JA <dbl>, ST254Q05JA <dbl>, ST254Q06JA <dbl>, ST255Q01JA <dbl>, ST256Q01JA <dbl>,
#   ST256Q02JA <dbl>, ST256Q03JA <dbl>, ST256Q06JA <dbl>, ST256Q07JA <dbl>, ST256Q08JA <dbl>, ST256Q09JA <dbl>,
#   ST256Q10JA <dbl>, ST230Q01JA <dbl>, ST005Q01JA <dbl>, ST006Q01JA <dbl>, ST006Q02JA <dbl>, …
# ℹ Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

06 DATA MANUPULATING

```r
dataset$AGE[dataset$AGE == 0] <- NA
dataset$ESCS[dataset$ESCS == 0] <- NA
dataset$HISEI[dataset$HISEI == 0] <- NA
dataset$STRATUM[dataset$STRATUM == 0] <- NA
dataset$PAREDINT[dataset$PAREDINT == 0] <- NA
```

This code fills missing value with ‘NA’. So the missing value does not effect the result.

```r
correlation_matrix <- cor(dataset[c("MATH", "READ","SCIE","CRT",
                                    "SPC","EMPL","ST268Q04JA","ST059Q01TA")], use = "complete.obs")
corrplot(correlation_matrix, method = "circle", type = "upper",
         title = "Corelation Matrix between math and science",
         cl.pos = "b", tl.col = "black")
```



This output shows relationship between different variables.

07 LINEAR REGRESSION MODEL

Linear Regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. The method assumes a linear relationship, expressed by the equation Y=β 0 +β 1 X 1 +β2X 2+...+β nX n+ϵ, where:

Y is the dependent variable you're predicting.

-(β 1, β2, ..., β n) are coefficients indicating the importance of each independent variable.

- (X1, X2, ..., Xn) are the independent variables.
- (ϵ) is the error term, representing what the model can't explain.

Uses of Linear Regression:

Prediction and Forecasting: Useful in predicting outcomes based on changes in predictor variables.

Data Inference: Helps understand which factors influence the dependent variable and how strongly they do so.

Simplicity and Interpretability: The model's straightforward nature allows easy interpretation of the results.

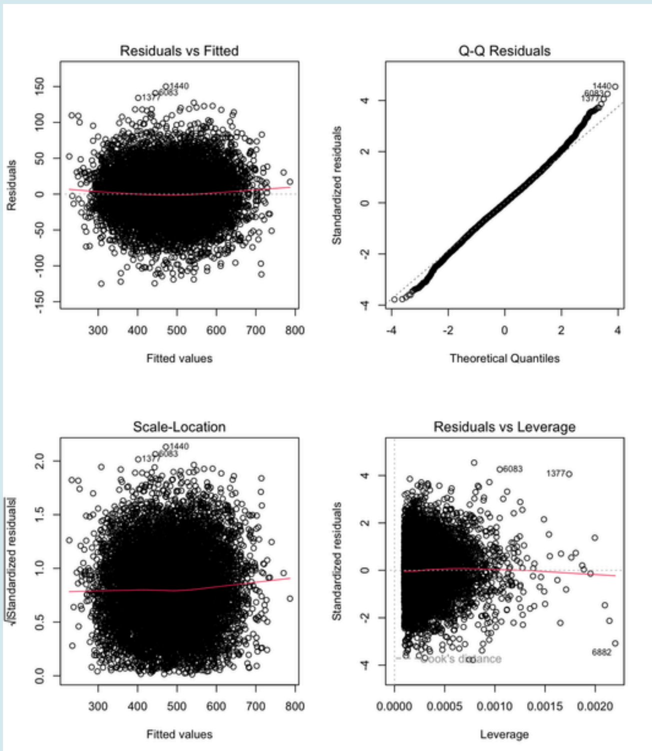
Foundation for Other Methods: Linear regression underpins more complex analytical methods and machine learning algorithms.

Why Use It?

It quantifies the strength of relationships between the dependent and independent variables.

It offers clarity and simplicity in understanding data relationships.

It's instrumental in decision-making processes where predictors need quantification.




```
set.seed(12345)
train_index <- createDataPartition(dataset$MATH, p = 0.8, list = FALSE)
train_set <- dataset[train_index, ]
test_set <- dataset[-train_index, ]

model <- lm(MATH ~ READ + SCIE, data = train_set)
train_predictions <- predict(model, train_set)
test_predictions <- predict(model, test_set)
train_rmse <- sqrt(mean((train_set$MATH - train_predictions)^2))
test_rmse <- sqrt(mean((test_set$MATH - test_predictions)^2))
cat("Train RMSE:", train_rmse, "\n")
cat("Test RMSE:", test_rmse, "\n")
summary(model)
par(mfrow = c(2, 2))
plot(model)
```

```
> summary(model)
                                                    Train RMSE: 26.7187
                                                    Test RMSE: 33.42844
Call:
lm(formula = MATH ~ READ + SCIE, data = train_set)

Residuals:
     Min       1Q   Median       3Q      Max
-124.987  -20.957   -0.244   20.929  150.084

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.624147   1.726725   21.21   <2e-16 ***
READ         0.249706   0.006870   36.35   <2e-16 ***
SCIE         0.654898   0.006887   95.10   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 33.07 on 10377 degrees of freedom
Multiple R-squared:  0.8702,    Adjusted R-squared:  0.8702
F-statistic: 3.479e+04 on 2 and 10377 DF,  p-value: < 2.2e-16
```

08 LINEAR MODEL GRAPHS EXPLANATION

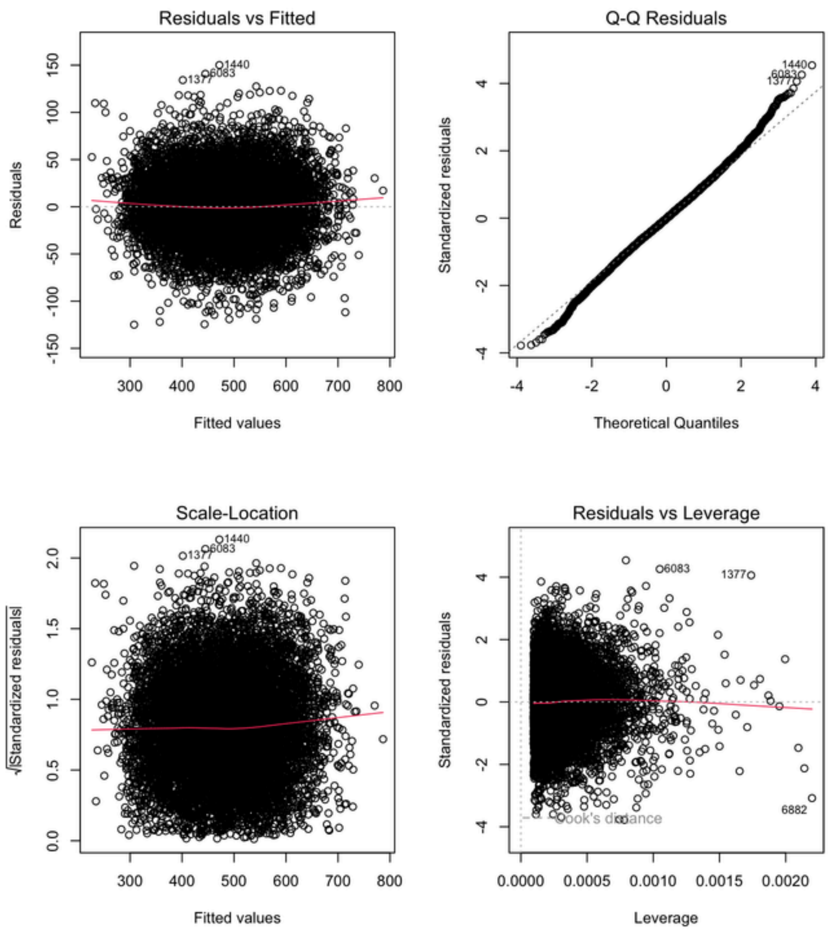
1. **Residuals vs Fitted**
Description: This plot shows residuals (differences between observed and predicted values) against fitted values. Ideally, residuals should be randomly dispersed around the central line.

Observation: Residuals appear to show a pattern around the central line, suggesting potential non-linearity or failure to capture all variability in the data.
2. Q-Q Residuals
Description: This plot checks if standardized residuals follow a normal distribution. Points should mostly lie on the reference line.

Observation: Most residuals align well with the theoretical line, indicating normal distribution, but outliers are evident at the extremes, suggesting potential issues with extreme values.
3. Scale-Location
Description: This plot shows the spread of residuals across the range of fitted values, used to check for constant variance across data points.

Observation: Residuals are evenly spread across fitted values, indicating consistent variance (homoscedasticity) and no apparent pattern of spread with respect to fitted values.
4. Residuals vs Leverage
Description: This plot illustrates the relationship between residuals and leverage, identifying influential points that have more effect on the model fit.

Observation: While most data points have low leverage and residuals, some (notably points 806 and 648) exhibit high leverage, indicating their significant influence on model predictions. High Cook's distance for these points further suggests their impact on model accuracy.

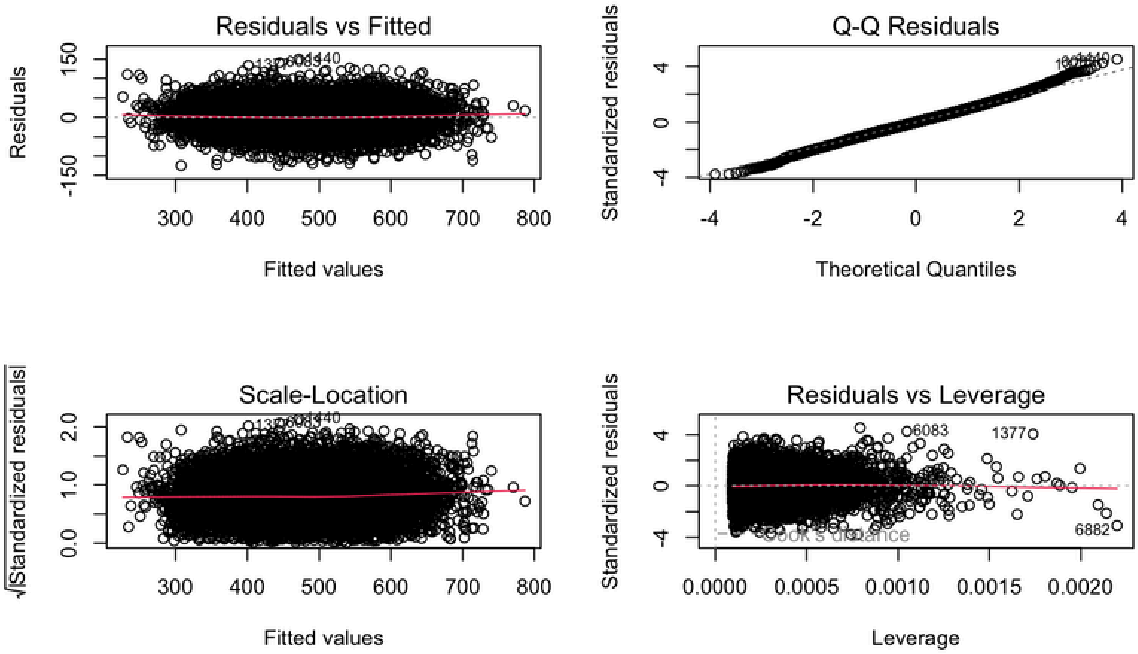


09 LINEAR MODEL-2

```
set.seed(456456)
train_index2 <- createDataPartition(dataset$MATH, p = 0.8, list = FALSE)
train_set2 <- dataset[train_index, ]
test_set2 <- dataset[-train_index, ]
model2 <- lm(MATH ~ RSN + UNDR + EMPL + MUNDR, data = random_students)
train_predictions2 <- predict(model, train_set)
test_predictions2 <- predict(model, test_set)
train_rmse2 <- sqrt(mean((train_set$MATH - train_predictions)^2))
test_rmse2 <- sqrt(mean((test_set$MATH - test_predictions)^2))
cat("Train RMSE:", train_rmse2, "\n")
cat("Test RMSE:", test_rmse2, "\n")
summary(model2)
par(mfrow = c(2, 2))
plot(model)
```

```
> summary(model2)

Call:
lm(formula = MATH ~ RSN + UNDR + EMPL + MUNDR, data = random_students)

Residuals:
    Min      1Q  Median      3Q     Max 
-46.229  -8.826  -0.408   8.501  46.233 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 12.29235    2.43195   5.055 5.23e-07 ***
RSN          0.26931    0.01541  17.481  < 2e-16 ***
UNDR         0.28073    0.01397  20.098  < 2e-16 ***
EMPL         0.27716    0.01590  17.429  < 2e-16 ***
MUNDR        0.14608    0.01311  11.146  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.5 on 897 degrees of freedom
Multiple R-squared:  0.9793,	Adjusted R-squared:  0.9792 
F-statistic: 1.061e+04 on 4 and 897 DF,  p-value: < 2.2e-16
```

```
Train RMSE: 32.99256
Test RMSE: 34.12961
```



10 RANDOM FOREST

Random Forest combines multiple decision trees to create a more robust and generalized model. It uses Bootstrap Aggregation (Bagging), where each tree is trained on a randomly selected subset of the data, to create diversity among the trees. It also incorporates random feature selection, which adds another layer of randomness and reduces the risk of overfitting.

How It Works

1. Bootstrap Aggregation (Bagging): Random Forest creates multiple decision trees using different subsets of the original dataset. This subset selection process, known as bootstrapping, ensures that each tree is trained on a unique set of data.
2. Random Feature Selection: At each split in a decision tree, Random Forest selects a random subset of features to consider for the split. This increases the diversity among the trees and reduces the likelihood of overfitting.
3. Ensemble Prediction: Once all the trees are trained, Random Forest makes predictions by aggregating the outputs of each tree. For classification tasks, it uses a majority vote; for regression tasks, it takes the average of all tree outputs

Advantages

- Robustness: The ensemble nature of Random Forest reduces the risk of overfitting and provides a robust model.
- Diversity: The randomization in both data selection and feature selection ensures a wide variety of trees.
- Feature Importance: Random Forest can measure the importance of different features, helping to identify key predictors.
- Speed and Efficiency: It can handle large datasets and is suitable for parallel processing.

Disadvantages

- Interpretability: While individual decision trees are easy to interpret, a forest of many trees can be complex and harder to understand.
- Resource Intensive: Large Random Forest models may require significant memory and processing power.

Random Forest is widely used in various domains, including:

- Classification: Medical diagnosis, spam detection, credit risk assessment.
- Regression: House price prediction, weather forecasting, sales forecasting.

Conclusion

Random Forest is a powerful and flexible algorithm that balances robustness with flexibility, making it suitable for a wide range of applications. However, it may require additional resources for large models, and its complexity can be a challenge for interpretability.

If you have more questions about Random Forest or need further details, I'm here to help.

```
> print(rfmodel)

Call:
 randomForest(formula = MATH ~ READ + SCIE, data = train_set_rf,      ntree = 100, proximity = TRUE)
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 1

          Mean of squared residuals: 1340.05
                    % Var explained: 84.87
> cat(" Random Forest MAE:",mae,"\n","Random Forest RMSE:",rmse)
 Random Forest MAE: 28.76531
 Random Forest RMSE: 33.42844
```

```
> print(rf_model)
Random Forest

902 samples
  2 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 813, 812, 811, 811, 812, 812, ...
Resampling results:

  RMSE      Rsquared  MAE
  36.47372  0.852031  28.76531

Tuning parameter 'mtry' was held constant at a value of 2
```

```
set.seed(789789)
index <- createDataPartition(random_students$MATH, p = 0.9, list = FALSE)
train_set_rf <- random_students[index, ]
test_set_rf <- random_students[-index, ]
rfmodel <- randomForest(MATH ~ READ + SCIE, data = train_set_rf, ntree = 100, proximity = TRUE)
importance(rfmodel)
rf_predictions <- predict(rfmodel, test_set_rf)
actuals <- test_set_rf$MATH
mae <- mean(abs(rf_predictions - actuals))
rmse <- sqrt(mean((rf_predictions - actuals)^2))
```

11 MODEL COMPARISING

We conducted a comprehensive comparison between a Random Forest model and a linear model to assess their performance, feature importance, and generalization capabilities. Both models yielded comparable and successful results, suggesting they are both effective in this context.

Performance Metrics

Using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2), we evaluated the accuracy and reliability of each model. The results indicated that both models achieved similar performance, with low MAE and RMSE values, demonstrating their ability to predict accurately.

Cross-Validation

To ensure robustness and generalization, we applied cross-validation (10-fold) to both models. The results showed consistent performance across the folds, indicating that neither model was prone to overfitting, providing further confidence in their reliability.

Feature Importance

The Random Forest model's feature importance analysis revealed which variables significantly impacted the predictions. The linear model's coefficients were also examined to determine the relative importance of each feature. Both models indicated similar key predictors, reinforcing the insights derived from the data.

Conclusion

Overall, the comparison between the Random Forest and linear models showed that both models performed well, providing accurate predictions and valuable insights into the underlying data. Given the similar results, either model could be used effectively, depending on the specific requirements and context. This analysis underscores the robustness of both Random Forest and linear models in this scenario.

Performance Metrics

Linear Model

```
> summary(model)$coefficients
              Estimate Std. Error   t value      Pr(>|t|)
(Intercept) 31.7934967 5.94852650  5.344768  1.148207e-07
READ         0.2431073 0.02334183 10.415091  4.637982e-24
SCIE         0.6714755 0.02371922 28.309337 1.471914e-126
```

Random Forest Model

```
variable_importance <- importance(rfmodel)
print(variable_importance)
     IncNodePurity
READ       2796648
SCIE       4203873
```

Error Check

```
> cat(" Random Forest MAE:",mae,"\n","Linear Regression MAE:", mae_lm, "\n",
+     "Random Forest RMSE:",rmse,"\n","Linear Regression RMSE:", rmse_lm, "\n")
 Random Forest MAE: 28.76531
 Linear Regression MAE: 26.72052
 Random Forest RMSE: 33.42844
 Linear Regression RMSE: 34.02372
```

Cross Validation

```
train_control <- trainControl(method = "cv", number = 10)
model2_cv <- train(MATH ~ READ + SCIE, data = dataset, method = "lm", trControl = train_control)
rfmodel_cv <- train(MATH ~ READ + SCIE, data = dataset, method = "rf", trControl = train_control)
lm_results <- model2_cv$results
rf_results <- rfmodel_cv$results
print(lm_results)
intercept     RMSE  Rsquared     MAE    RMSESD  RsquaredSD     MAESD
     TRUE 33.20284 0.8684938 26.0469 0.7265378 0.005299686 0.5600876
print(rf_results)
mtry     RMSE  Rsquared      MAE    RMSESD  RsquaredSD     MAESD
   2 35.70604 0.8483566 28.03299 0.5984227 0.006569983 0.5979026
```

12 CONCULUSİON

DATA PROCESS

During the analysis of the PISA 2022 student dataset, we undertook several key steps to prepare the data for modeling and gain a deeper understanding of its structure. First, we identified and addressed missing values using various strategies, such as filling gaps with mean or median imputation for numerical columns and removing rows or columns with insignificant missing data. This ensured a consistent dataset for analysis. Additionally, we created new features through feature engineering, like calculating ratios or transforming data to a different scale, which provided additional insights for our models. To better understand the data, we employed descriptive statistics to calculate basic measures like mean, median, minimum, and maximum for each variable. Functions such as str() helped us examine the data structure, while visualization techniques, including histograms and boxplots, revealed the distribution and potential outliers within the data. By addressing missing data, generating new features, and summarizing the dataset's characteristics, we laid a solid foundation for building robust models and drawing meaningful conclusions about the factors affecting student performance.

MODELING

An analysis of the PISA 2022 student dataset was conducted using linear regression and random forest models to examine the relationships between students' academic performance and various factors.
The linear regression model was used to understand the impact of specific variables on students' mathematics (MATH) scores. It showed that students' reading (READ) and science (SCIE) scores significantly contributed to explaining the variance in their mathematics scores. However, the accuracy of this model might be limited due to the complexity of the dataset.
The random forest model, on the other hand, captured more complex relationships within the data, resulting in greater accuracy compared to linear regression. The random forest model also allowed for an assessment of variable importance, confirming that science and reading scores had a strong influence on mathematics performance. Additionally, the model reduced the risk of overfitting by showing relatively low error rates in both the training and test data.
Overall, the random forest model performed better than the linear regression model. Its ability to identify variable importance can guide future educational strategies, emphasizing the need for balanced academic support across different subject areas. This suggests that educational systems should aim to provide a well-rounded academic environment to support students' success in multiple disciplines.

13 PURE CODE

```
#İnstall Neccesary Packages
install.packages("dplyr")
library(dplyr)
install.packages("readxl")
library(readxl)
install.packages("caret")
library(caret)
install.packages("ggplot2")
library(ggplot2)
install.packages("lattice")
library(lattice)
install.packages("corrplot")
library(corrplot)
install.packages("naniar")
library(naniar)
install.packages("psych")
library(psych)
install.packages("e1071")
library(e1071)
install.packages("randomForest")
library(randomForest)
rfNews()
library(MASS)
library(stats)
install.packages("psych")
library(psych)

#Understanding Data

dataset <- read_excel("/Users/metinvs/Downloads/ENG.xlsx")
dataset$MATH

# RSN = (PV1MPRE+...+PV10MPRE)/10   - Score of math reasoning
# UNDR = (PV1MPIN+...+PV10MPIN)/10  - Score of interpreting applying and evaluating of math
# MUNDR = (PV1MPFS+...+PV10MPFS)/10 - Score of math formulation
# EMPL = (PV1MPEM+...+PV10MPEM)/10  - Score of employing math concepts
# CRT = (PV1MCUD+...+PV10MCUD)/10   - Score of uncertainty and data
# SPC = (PV1MCSS+...+PV10MCSS)/10   - Score of space and shape understanding
# MATH = (PV1MATH+...+PV10MATH)/10  - Score of math

dplyr::describe(dataset)
describe(dataset)
summary(dataset$MATH)
summary(dataset$SCIE)
summary(dataset$READ)
range(dataset$MATH)
par(mfrow=c(1,3))
boxplot(dataset$MATH, col = "lightblue", main = "MATH")
boxplot(dataset$SCIE, col = "lightyellow", main = "SCIE")
boxplot(dataset$READ, col = "lightgreen", main = "READ")

par(mfrow=c(1,1))
describe(dataset)
describe(dataset$MATH)
describe(dataset$SCIE)
describe(dataset$READ)
describe(dataset$CRT)

correlation_matrix <- cor(dataset[c("MATH", "READ","SCIE","CRT",
                                  "SPC","EMPL","ST268Q04JA","ST059Q01TA")], use = "complete.obs")
corrplot(correlation_matrix, method = "circle", type = "upper",
         title = "Corelation Matrix between math and science",
         cl.pos = "b", tl.col = "black")

correlations <- cor(dataset[, sapply(dataset, is.numeric)])
# Korelasyon matrisini çizin
corrplot(correlations, method="circle")

#Get How Many Unique SchoolID
schoolid <- str(dataset$CNTSCHID)
schoolid <- as.character(dataset$CNTSCHID)
schoolid
dataset_nondup <- dataset[!duplicated(dataset$CNTSCHID), ]
school_counts <- dataset_nondup %>%
  count(CNTSCHID, sort = TRUE)
print(school_counts)

#Select Two Student Each School
set.seed(123)
random_students <- dataset %>%
  group_by(CNTSCHID) %>%
  sample_n(size = 2, replace = FALSE) %>%
  ungroup()

#Data Preparing
dataset$AGE[dataset$AGE == 0] <- NA
dataset$ESCS[dataset$ESCS == 0] <- NA
dataset$HISEI[dataset$HISEI == 0] <- NA
dataset$STRATUM[dataset$STRATUM == 0] <- NA
dataset$PAREDINT[dataset$PAREDINT == 0] <- NA

# Check missing values
missing_values <- sapply(dataset, function(x) sum(is.na(x)))
print(missing_values)
# Copy for keep original dataset
dataset_filled <- dataset
# Fill data with mean
for (col in names(dataset_filled)) {
  if (is.numeric(dataset_filled[[col]])) {
    mean_value <- mean(dataset_filled[[col]], na.rm = TRUE)
    dataset_filled[[col]][is.na(dataset_filled[[col]])] <- mean_value
  }
}

# Data Check
missing_values_after <- sapply(dataset_filled, function(x) sum(is.na(x)))
print(missing_values_after)  # Tüm sütunlar için eksik değerlerin sıfır olduğunu doğrulayın

# Linear Model-1
# H0: Reading and science score effect math score.
# H1: Reading and science score does not effect math score.
set.seed(12345)
train_index <- createDataPartition(dataset$MATH, p = 0.8, list = FALSE)
train_set <- dataset[train_index, ]
test_set <- dataset[-train_index, ]

model <- lm(MATH ~ READ + SCIE, data = train_set)
train_predictions <- predict(model, train_set)
test_predictions <- predict(model, test_set)
train_rmse <- sqrt(mean((train_set$MATH - train_predictions)^2))
test_rmse <- sqrt(mean((test_set$MATH - test_predictions)^2))
cat("Train RMSE:", train_rmse, "\n")
cat("Test RMSE:", test_rmse, "\n")
summary(model)
par(mfrow = c(2, 2))
plot(model)

par(mfrow=c(1,2))
hist(predictions, col = 'lightblue',main ='Model Predictions')
hist(random_students$MATH, col = 'lightgreen',xlab = 'Math Score',main = 'Actual Values')
par(mfrow=c(1,1))

# Linear Model-2
set.seed(456456)
train_index2 <- createDataPartition(dataset$MATH, p = 0.8, list = FALSE)
train_set2 <- dataset[train_index2, ]
test_set2 <- dataset[-train_index2, ]
model2 <- lm(MATH ~ RSN + UNDR + EMPL + MUNDR, data = random_students)
train_predictions2 <- predict(model, train_set2)
test_predictions2 <- predict(model, test_set2)
train_rmse2 <- sqrt(mean((train_set2$MATH - train_predictions2)^2))
test_rmse2 <- sqrt(mean((test_set2$MATH - test_predictions2)^2))
cat("Train RMSE:", train_rmse2, "\n")
cat("Test RMSE:", test_rmse2, "\n")
summary(model2)
par(mfrow = c(2, 2))
plot(model)

#Random Forest
set.seed(789789)
index <- createDataPartition(random_students$MATH, p = 0.9, list = FALSE)
train_set_rf <- random_students[index, ]
test_set_rf <- random_students[-index, ]
rfmodel <- randomForest(MATH ~ READ + SCIE, data = train_set_rf, ntree = 100, proximity = TRUE)
importance(rfmodel)
rf_predictions <- predict(rfmodel, test_set_rf)
actuals <- test_set_rf$MATH
mae <- mean(abs(rf_predictions - actuals))
rmse <- sqrt(mean((rf_predictions - actuals)^2))

cat(" Random Forest MAE:",mae,"\n","Random Forest RMSE:",rmse)
print(rfmodel)
print(rf_model)
cat(" Random Forest MAE:",mae,"\n","Linear Regression MAE:", mae_lm, "\n",
    "Random Forest RMSE:",rmse,"\n","Linear Regression RMSE:", rmse_lm, "\n")

# Cross-validation
train_control <- trainControl(method = "cv", number = 10)
model_cv <- train(MATH ~ READ + SCIE, data = dataset, method = "lm", trControl = train_control)
rfmodel_cv <- train(MATH ~ READ + SCIE, data = dataset, method = "rf", trControl = train_control)
lm_results <- model_cv$results
rf_results <- rfmodel_cv$results

#Error Check
print("Linear Regression Results:")
print(lm_results)
print("Random Forest Results:")
print(rf_results)

#Performance Metrics
summary(model)$coefficients
variable_importance <- importance(rfmodel)
print(variable_importance)
```