SAPIENZA
UNIVERSITÀ DI ROMA

ADVANCED STATISTICS PROJECT

GIUSEPPINA OREFICE

1 Introduction

The dataset has got 7 columns and 39 rows. Each row represents a different country, while the columns represent: GDP (that is the gross domestic product), CELLULAR, FERTILITY, INTERNET, CO2 and LITERACY. In order to predict the value of y, that is FERTILITY it's necessary the multiple linear regression since it is a quantitative variable.

```
#The first part is downloading all the libraries
library(MASS) °for linear regression
library(glmnet) #for cv
library(dplyr) #for the discretization
library(ggplot2) #for the plots
library(car)
library(tidyverse)
library(leaps) #for stepwise regression
library(class) #for classifiers
library(splines) #for regression splines
library(gam) #for gam
library(mgcv) #for gam with cubic spline
library(caret)
library(lmtest)

#Upload the dataset from the data
development <- read.csv("C:/Users/Giusy/Downloads/human_development.csv")
```

2 *Linear regression*

Multiple linear regression is a statistical technique that models the relationship between one dependent variable (also called the response variable) and two or more independent variables (also called predictors or explanatory variables). The goal is to fit a linear equation to observed data. The coefficients are estimated (β values) by using methods such as Ordinary Least Squares (OLS), which minimizes the sum of the squared residuals (the differences between observed and predicted values squared).

```
#Check the column names
colnames(development)

# Multiple regression by using fertility as response
lmfit <- lm(FERTILITY ~ CELLULAR + GDP + CO2 + INTERNET + LITERACY , data =
↪  development)
#Analyze the results of the model by regarding the coefficients
summary(lmfit)
coef(lmfit)
```

2.0.1 Results

```
lm(formula = FERTILITY ~ CELLULAR + GDP + CO2 + INTERNET + LITERACY,
    data = development)

Residuals:
     Min       1Q   Median       3Q      Max
-1.35920 -0.35214 -0.07364  0.31471  1.63459

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.6875995  0.5831313  11.468 4.71e-13 ***
CELLULAR     0.0005326  0.0088464   0.060    0.952
GDP         -0.0229975  0.0381223  -0.603    0.550
CO2          0.0215744  0.0452920   0.476    0.637
INTERNET     0.0073456  0.0141770   0.518    0.608
LITERACY    -0.0493994  0.0084930  -5.816 1.66e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7274 on 33 degrees of freedom
Multiple R-squared:  0.7036,Adjusted R-squared:  0.6586
F-statistic: 15.66 on 5 and 33 DF,  p-value: 6.66e-08
```

2.0.2 Explanation

Residuals are the differences between the observed values and the values predicted by the model. The summary provides the distribution of these residuals:

- **Min**: -1.35920
- **1Q (1st Quartile)**: -0.35214
- **Median**: -0.07364
- **3Q (3rd Quartile)**: 0.31471
- **Max**: 1.63459
- The range from the minimum (-1.35920) to the maximum (1.63459) residuals shows the variability in the model's predictions. A wider spread indicates greater variability in the residuals, meaning the model's predictions are less consistent.
- The interquartile range (from the 1st quartile to the 3rd quartile) is smaller (-0.35214 to 0.31471), indicating that 50% of the residuals are relatively close to zero. This suggests that for half of the observations, the model's predictions are fairly accurate
- If residuals exhibit a pattern (e.g., increasing or decreasing trend, clusters), it indicates potential problems with the model, such as missing predictors, non-linearity, or heteroscedasticity. A median close to zero suggests that the model's predictions are, on average, accurate, but the spread indicates the variability of the predictions.

We are going to analyse the predictors:

- CELLULAR: the coefficient is 0.000533, which is not statistically significant (p-value = 0.952).
- CO2: the coefficient is 0.021574, which is not statistically significant (p-value = 0.637).
- INTERNET: the coefficient is 0.007346, which is not statistically significant (p-value = 0.608).
- GDP: the coefficient is -0.022998, which is not statistically significant (p-value = 0.550).
- LITERACY: The coefficient is -0.049399, which is statistically significant (p-value = 1.66e-06). This suggests that higher literacy rates are associated with lower fertility rates.
- The *t-test* assesses if the coefficients are statistically different from the zero.

- Then, the *RSE* indicates the average distance that the observed values fall from the regression line. It's a measure of the residuals and it is evaluated as the square root of the mean squared error.
- *Multiple R-squared*, that is a measure of the model's accuracy since it takes into account the total sum of squares and the residual sum of squares and it explains how much variability is defined by the model, says that the 70 of the variability in fertility is explained and the*adjusted R-squared* which adjusts the R-squared value for the number of predictors in the model, suggesting a good fit.
- Also the *F-statistics*, that is a measure of how many evidences we have against or in favour of the null hypothesis, provides a good value 15.66 since if it is greater than 1, we accept the alternative hypothesis. [1]
- The *p-value* of 6.66e-08, indicating that the overall model is statistically significant.

2.0.3 Collinearity detection

One of the disadvantage of the multiple linear regression is the collinearity, when the predictors are correlated to other predictors. It may cause problems to understand the power of the predictor along the model. There are also problems of multicorrelation. The Variance Inflation Factor (VIF) values are used to detect multicollinearity among the predictors in the regression model. High VIF values indicate a high level of multicollinearity, which can affect the stability and interpretation of the regression coefficients.

```
#analyzing multicollinearity
vif_values <- vif(lmfit)
> print(vif_values)
```

```
CELLULAR       CO2  INTERNET       GDP  LITERACY
5.684117  3.217508  4.922550 11.720874  2.237146
```

The VIF values suggest that **GDP** has a high level of multicollinearity, which may be problematic. Multicollinearity can inflate the standard errors of the coefficients, making it difficult to determine the individual effect of each predictor. **CO2 a**nd **LITERACY** have low VIF values, indicating low multicollinearity . We could try to remove GDP or we could combine two collinear predictor in one term in order to improve the model's accuracy.

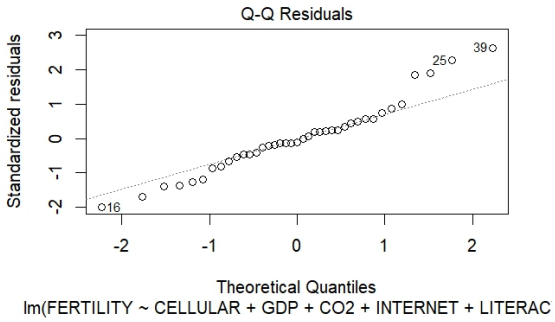
[1] In this case we could use the F-statistics for the model's since the observations are greater than the predictors.

```
if (any(vif_values > 5)) {
+   cat("Warning: Multicollinearity detected. Consider removing predictors with
↪  high VIF values.\n")
+ } else {
+   cat("No multicollinearity detected.\n")
+ }
```

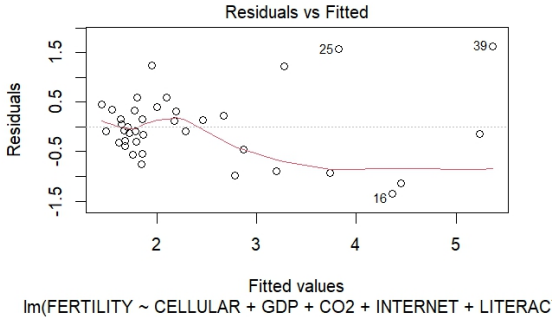
```
Warning: Multicollinearity detected.
Consider removing predictors with high VIF values.
```

2.0.4 PLOTS

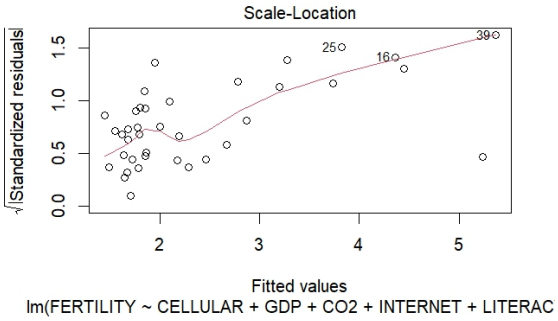
```
#Other general plots
plot(lmfit)
```



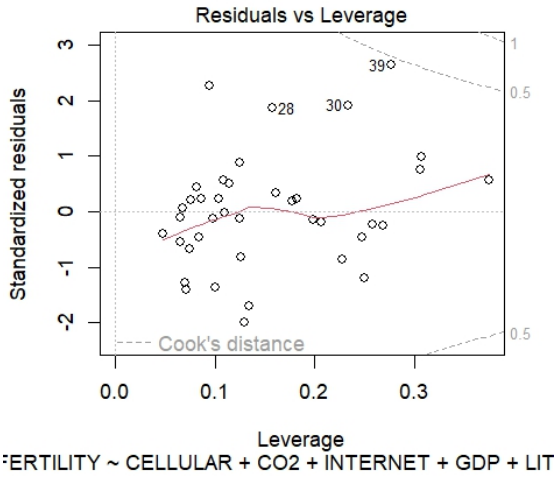
This plot assesses if the residuals follow a normal distribution. If the data are on the straight line, the residuals are normally distributed. Deviations from this line indicate that the residuals are not normally distributed. This can potentially seems an indicator of accuracy because: 1. the p-values cannot be reliable; 2. some data rely on the normality assumption.



This kind of curve can be explained by the fact that the most variance is not captured by the linear regression and the model's variance is not constant (heteroskedasticity) or the relationship between the independent and dependent variables is not linear.



This plot assesses well the homoskedasticity, the constance of the variance. If there is a straight line, there will be homoskedasticity. But since there is an upward curve, the variance increases as the predicted values increase explaining the heteroskedasticity.



This plot shows the residuals, but also the leverage that is a measure of how far is a point from the average. Outliers are all the residuals far from the average. If there are influential points, with high residuals and high leverage, these could affect the model's efficiency. To remove the outliers it's possible to consider the studentizied residuals; instead for the high leverage point, it's possible to use the leverage statistic.

2.0.5 Heteroscedasticity detection

Another issues of the linear regression is the heteroscedasticiy, the condiotion for which the variance of the errors is not constant and the magnitude of the residuals tends to increase with the fitted values. This violates one of the key assumptions of ordinary least squares (OLS) regression, which assumes homoscedasticity. Detecting and addressing heteroscedasticity is crucial for the following reasons:

1. **Inefficient Estimates** since the standard errors of the coefficients could be biased, leading to unreliable hypothesis tests (F-tests).
2. **Invalid Inference**: p-values may be incorrect, making it difficult to assess the true significance of the predictors.
3. **Model Misspecification**: it could indicate that the model is missing key variables, interactions, or nonlinearities that are important for explaining the variability in the dependent variable.

```r
# Perform the White test
ncvTest(lmfit)
```

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 22.34525, Df = 1, p = 2.2778e-06
```

The very small p-value (much less than 0.05) indicates strong evidence against the null hypothesis of homoscedasticity. Therefore, we reject the null hypothesis and conclude that heteroscedasticity is present in the model. In addition

```r
# Calculate robust standard errors
robust_se <- vcovHC(lmfit, type = "HC1")
# Display results with robust standard errors
coeftest(lmfit, vcov = robust_se)
```

```
t test of coefficients:

               Estimate  Std. Error t value  Pr(>|t|)
(Intercept)  6.68759946  0.98046998  6.8208 8.804e-08 ***
CELLULAR     0.00053255  0.00685091  0.0777 0.9385089
CO2          0.02157444  0.03612335  0.5972 0.5544210
INTERNET     0.00734559  0.01023466  0.7177 0.4779819
GDP         -0.02299749  0.02472459 -0.9301 0.3590499
LITERACY    -0.04939939  0.01256134 -3.9327 0.0004072 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The robust standard errors allows us to properly assess the significance of each predictor while mitigating the impact of non-constant variance on the standard error estimates. This approach leads to more trustworthy statistical inference, enhancing the overall quality of the regression analysis.

3 Lasso regression with cross-validation

3.0.1 Introduction

Lasso is a regression technique that performs both variable selection and regularization to enhance prediction accuracy and interpretability. The main reasons to use Lasso include:

- **Variable Selection**: Lasso can shrink some coefficients exactly to zero, effectively selecting a simpler model with fewer predictors.

- **Handling Multicollinearity**: By penalizing the absolute size of the coefficients, Lasso reduces the impact of multicollinearity among predictors.
- **Improved Prediction**: Regularization helps prevent overfitting, leading to better predictive performance on new data. It introduces bias by shrinking coefficients, but this can significantly reduce variance. The overall prediction error can be lower than that of the least squares method due to this trade-off.

Lasso adds a penalty equal to the absolute value of the magnitude of coefficients to the loss function. λ is a tuning parameter that controls the strength of the penalty. Larger λ values result in more coefficients being shrunk to zero. The selection of the tuning parameter can be done through the cross validation: we create a grid of lambda and through the cv, we will choose the value of lambda that minimizes the cv error. The cv provides also a way to make the subset selection, because as we know it directly estimates the test MSE.

3.0.2 Coding part

```r
# Define the x and the y variables
x <- model.matrix(FERTILITY ~ INTERNET + GDP + CO2 + CELLULAR + LITERACY, data = development)
y <- development$FERTILITY
# Lasso with cross-validation
fit.lasso=glmnet(x,y)

#Finding lasso with cv in order to have the best subset selection of predictors with the right lambda
cv.lasso <- cv.glmnet(x,y) #alpha 1 specifies that we are performing Lasso regression (as opposed to Ridge regression which would have alpha 0)
bestlam <- cv.lasso$lambda.min
```

```
[1] 0.09143874
```

```r
best_model_cv <- glmnet(x, y, alpha=1, lambda= bestlam)
coef(best_model_cv)
```

3.0.3 Results

```
 6 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  6.31749838
INTERNET     .
```

```
GDP           .
CO2           .
CELLULAR      .
LITERACY     -0.04561538
```

3.0.4 Explanation

The Lasso regression selected "LITERACY" as the most important predictors for fertility. Other predictors ("INTERNET," "CO2," "CELLULAR", "GDP") were shrunk to zero, indicating they do not significantly contribute to the model. The strong negative coefficient for "LITERACY" indicates that higher literacy rates are strongly associated with lower fertility rates, which aligns with the results from the linear regression model.

```
dim(coef(fit.lasso))
```

```
[1]  6 76
```

On the left side there are the number of predictors+intercept and on the right side there are the values of lambda found.

```
mse <- mean(cv.lasso\$cvm) print(paste("Cross-validated MSE:", mse))
```

[1] "Cross-validated MSE: 0.763188539942117"
This value shows the mse of the cross validation approach by using the lasso with cross validation. The cross-validated MSE reported is the average of these MSE values across all folds.

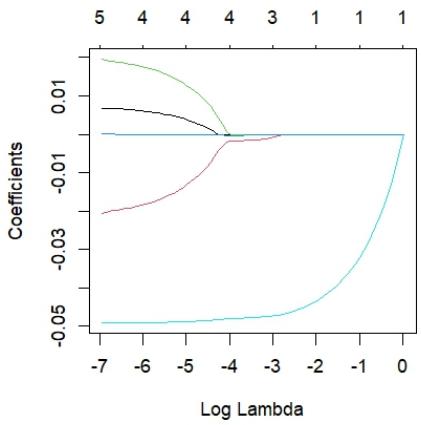


Figure 1: On the y-axis there are the coefficients and on the x-axis the log-lambda. It shows how the coefficients change after a change of the loglambda. As we can see, as the lasso will increase, it will enforce the shrinkage on the coefficients.

3.0.5 PLOTS

```
plot(fit.lasso,xvar="lambda",label=TRUE)
```

```
plot(cv.lasso)
```

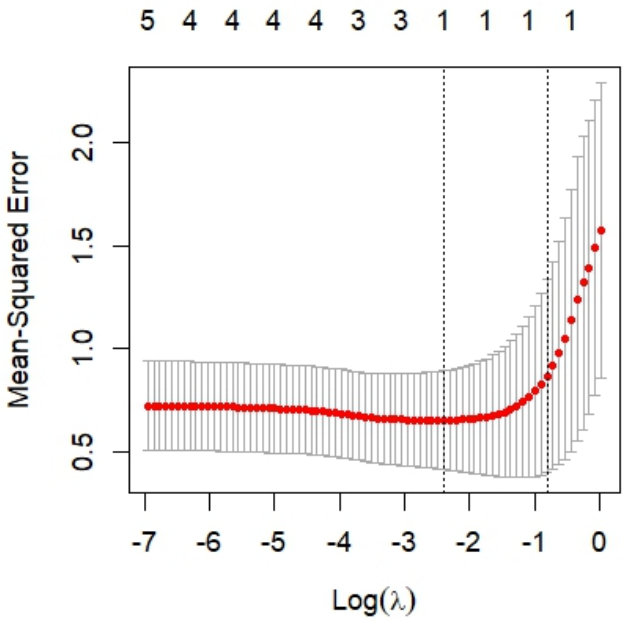


Figure 2: This plot represents the cross validation for a LASSO regression. It identifies the best value for lambda. The x-axis shows the values of the lamba parameter and y-axis shows the mean cross-validated error. The best lambda minimizes the mean cross validated error.

4 Forward stepwise with cross validation

Other ways to increase the model's accuracy by using the least squares approach are the subset selection (best subset selection, forward stepwise, backward and hybrid stepwise), the ridge regression and the dimensionality reduction (partial least squares and principal component analysis). In this coding part we are going to see the forward stepwise selection by implementing both the validation set approach and the k-folds cross validation. The forward stepwise approach is a good comprimise to avoid lots of calculation if we use the best subset selection, since it is based on the linear combination of the predictors. Instead, the forward stepwise starts from the null model and it adds one predictor at time. It has two drawbacks:

1. maybe we cannot end up to a good subset of predictors;
2. there is an overlap of predictors.

Therefore, it's useful to choose the training data and the testing data in order to apply the validation set approach, a method that allows us to evaluate the test mean squared error through. We first fit the model on the training data and then we evaluate the test mean squared error on the validation data.

4.1 Forward stepwise with validation set approach

```
#In order to apply the validation set approach, we divide the training set and
↪  the validation set
num_rows <- nrow(development)
train_size <- round(0.65 * num_rows)
dim(development)
```

[1] 39 7

Then, the regsubsets function is fundamental to use the forward stepwise regression.

```
set.seed(1)
train  <- sample(1:num_rows, train_size)
#Training a forward stepwise regression
regfit.fwd=regsubsets(FERTILITY ~ INTERNET + GDP + CO2 + CELLULAR + LITERACY,
                      data = development[train,],nvmax=5,method="forward")
summary(regfit.fwd)
```

The forward selection method works as follows:

1. **Start with no variables in the model**;

2. **add the variable that improves the model the most**, according to a chosen criterion (e.g., adjusted R-squared, AIC, BIC). In this specific case, the best subset is the one who minimizes the cross validation error;

3. **repeat step 2** until adding another variable does not significantly improve the model or until the maximum number of variables (`nvmax`) is reached.

In this case, the algorithm started with an empty model and added variables one by one, each time selecting the variable that resulted in the best improvement in model performance, up to a maximum of 5 variables.

The `regsubsets` function then evaluates all possible subsets of the independent variables up to the specified maximum size (`nvmax = 5`) using the forward selection approach. The output indicates that for each subset size from 1 to 5, there is one model that has been considered the best according to the selection criteria.

The results of this process are typically used to identify the best subset of predictors for a regression model, helping to balance model complexity with predictive performance.

```
regsubsets.formula(FERTILITY ~ INTERNET + GDP + CO2 + CELLULAR +
    LITERACY, data = development[train, ], nvmax = 5, method = "forward")
5 Variables  (and intercept)
          Forced in Forced out
INTERNET      FALSE      FALSE
GDP           FALSE      FALSE
CO2           FALSE      FALSE
CELLULAR      FALSE      FALSE
LITERACY      FALSE      FALSE
1 subsets of each size up to 5
Selection Algorithm: forward
          INTERNET GDP CO2 CELLULAR LITERACY
1  ( 1 ) " "      " " " " " " "       "*"
2  ( 1 ) " "      "*" " " " " "       "*"
3  ( 1 ) " "      "*" "*" " " "       "*"
4  ( 1 ) " "      "*" "*" "*"       "*"
5  ( 1 ) "*"      "*" "*" "*"       "*"
```

Then, the next step is testing the model,

```r
#we run a loop and for each i we extract the coefficients
val.errors <- rep(NA, 5)
x.test <- model.matrix(FERTILITY ~ INTERNET + GDP + CO2 + CELLULAR + LITERACY,
↪  data = development[-train, ])


for (i in 1:5) {
  coefi <- coef(regfit.fwd, id = i)
```

```
  pred_names <- intersect(names(coefi), colnames(x.test))  # Identifies the
  ↪  common names between the coefficients and the columns of the test data.
  ↪  This ensures that only the relevant predictors are used for making
  ↪  predictions.

  pred <- x.test[, pred_names] %*% coefi[pred_names]
  val.errors[i] <- mean((development$FERTILITY[-train] - pred)^2)
}
#print the errors
print(val.errors)
```

```
[1] 0.5676430 0.5869293 0.5766257 0.5801661 0.5786546
```

The best model will be the model who minimizes the cv error.

```
best_model_size1 <- which.min(val.errors)
cat("Best model size:", best_model_size1, "\n")
```

Best model size: 1

```
best_model <- coef(regfit.fwd, best_model_size1)
print(best_model)
```

```
(Intercept)    LITERACY
 7.1497396  -0.0549907
```

4.1.1 Explanation of the results

The forward stepwise regression identified a model with only `LITERACY` as a predictor, suggesting it alone might be sufficient to explain variations in `FERTILITY` among the selected predictors. The coefficients (`best_model`) provide insights into the direction and magnitude of the relationships between each predictor and the response variable (`FERTILITY`). For example, an increase in `LITERACY` is associated with a decrease in `FERTILITY` by approximately `-0.054`. The validation errors and subsequent selection of the best model size ensure that the model generalizes well to unseen data, minimizing overfitting. Therefore, the final model (`best_lm`) encapsulates this process, providing a refined understanding of the relationship between predictors and `FERTILITY` in your dataset.

```r
#it %*% performs matrix multiplication between x.test[, pred_names] (matrix of predictors in the test set) and best_model[pred_names] (vector of coefficients).
pred_names <- intersect(names(best_model), colnames(x.test))
pred <- x.test[, pred_names] %*% best_model[pred_names]
mse_forward <- mean((development$FERTILITY[-train] - pred)^2)
cat("MSE (Forward Stepwise Regression):", mse_forward, "\n")
```

MSE (Forward Stepwise Regression): 0.567643

4.2 Forward stepwise with k-folds cv

Another method to choose the best subset is k-folds cross validation, that has got lots of advantage. The validation set approach seen above has got some drawbacks related to:

1. high variability, because the training dataset and the validation dataset are arbitrarly chosen;
2. the training error is estimated on few observations and the test error can be overstimated.

Therefore, the leave-one-out cross validation or the k-folds cross validation help us to face off these issues. The number of k is chosen. In addition it is necessary to indicate the predict.rugsubsets as a function in order to apply regsubsets. The script performs a 10-fold cross-validation to identify the best subset of predictors for the `FERTILITY` regression model using forward selection:

1. It randomly splits the data into 10 folds.
2. For each fold, it fits the model on the training data and evaluates the model on the test data.
3. It calculates the mean squared error for each model size and identifies the best model size based on the lowest error.
4. It refits the best model size on the entire dataset and extracts the coefficients.

The best model size is 1, meaning the model with the lowest cross-validation error includes only one predictor (`LITERACY`).

```r
k <- 10
num_rows <- nrow(development)
folds <- sample(rep(1:k, length= num_rows))  #Randomly assigns each row in the dataset to one of the 10 folds.
dim(development)
```

```r
set.seed(1)
 #This initializes a matrix to store the cross-validation errors. It has 10 rows (one for each fold) and 5 columns (one for each model size from 1 to 5).
cv.errors <- matrix(NA, k, 5,
                    dimnames = list(NULL, paste(1:5)))
#A custom function to make predictions for models fitted with the regsubsets function
predict.regsubsets <- function(object, newdata, id, ...) {
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id = id)
  xvars <- names(coefi)
  mat[, xvars] %*% coefi
}
#fits the model on the training set (excluding the j-th fold) creating a loop
for(j in 1:k) {
  best.fit <- regsubsets(FERTILITY ~ INTERNET + GDP + CO2 + CELLULAR + LITERACY, data = development [folds != j, ],
                         nvmax = 5)

  for (i in 1:5) {
    pred.folds <- predict.regsubsets(best.fit, development [ folds == j,] , id =i)
    cv.errors [j, i] <-
      mean((development$FERTILITY[folds == j] - pred.folds)^2) #makes predictions on the validation set (the j-th fold).
  }
}

mean.cv.errors <- apply(cv.errors, 2, mean) #2 is because it evaluates the error on the column
print(mean.cv.errors)
```

```
     1          2          3          4          5
0.5370764 0.5786005 0.6115552 0.6104865 0.6105225
```

```r
# Identify the best model size (the one with the lowest cross-validation error)
best.model.size2 <- which.min(mean.cv.errors)
print(paste("Best model size:", best.model.size2))
```

```
[1] "Best model size: 1"
```

```r
# Refit the best model size on the entire dataset
best.fit <- regsubsets(FERTILITY ~ INTERNET + GDP + CO2 + CELLULAR + LITERACY,
                       data = development, nvmax = 5)

# Extract the coefficients for the best model
best.coefficients <- coef(best.fit, id = best.model.size2)
print(best.coefficients)
```

```
(Intercept)    LITERACY
 6.70232382 -0.05007295
```

```r
x.full <- model.matrix(FERTILITY ~ INTERNET + GDP + CO2 + CELLULAR + LITERACY,
↪  data = development)
# Predict the fertility using the best model coefficients
pred.full <- x.full[, names(best.coefficients)] %*% best.coefficients #The %*%
↪  operator performs matrix multiplication between the selected columns of
↪  x.full and the best.coefficients. This matrix multiplication (X where X is
↪  the predictor matrix and  is the vector of coefficients) is a standard way
↪  to compute linear model predictions.
mse_cv <- mean((development$FERTILITY - pred.full)^2)
cat("MSE (Cross-Validation):", mse_cv, "\n")
```

MSE (Cross-Validation): 0.4553044

5 *GAM MODEL*

Since, as we have already seen, the relationship between the response and the predictors, could be not linear, it's necessary moving beyond the linearity. In this section we are going to see the Generalized additive model.[2] A key feature of GAMs is their ability to capture non-linear relationships through the use of smooth functions. They are an extension of the generalized linear function. These act as the bridges, capturing the non-linear relationships between predictors and the response. Splines, like cubic or natural cubic, are popular choices for building these smooth curves. It is important to say that they mantain the additivity feature of the multiple predictors.

```r
gam_model_cubic <- gam(GDP ~ s(FERTILITY) + s(LITERACY, bs = "cr"), data =
↪  development)
gam_model_ns <- gam(GDP ~ ns(FERTILITY) + ns(LITERACY), data = development)
AIC_cubic <- AIC(gam_model_cubic)
```

[2]Other model are the step functions, the polynomial regression, the regression splines and the local regression.

```r
AIC_ns <- AIC(gam_model_ns)
summary(gam_model_cubic)
```

```
Family: gaussian
Link function: identity

Formula:
GDP ~ s(FERTILITY) + s(LITERACY, bs = "cr")

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  15.9933     0.8387   19.07   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
               edf Ref.df     F  p-value
s(FERTILITY) 1.000  1.000 0.006    0.938
s(LITERACY)  5.245  6.165 9.857 3.44e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.756   Deviance explained = 79.6%
GCV = 33.693  Scale est. = 27.434    n = 39
```

5.1 Explanation of the results

The edf are the estimated degrees of freedom and they tell us about the flexibility of the smooth term. Higher edf indicates more flexibility. The *F-statistic* and its corresponding p-value for each smooth term indicates that the smooth term significantly contributes to the model. *The adjusted R-squared* value is 0.756, suggesting that approximately 75.6 of the variability in GDP is explained by the model after adjusting for the number of predictors, while the GCV (Generalized Cross-Validation) is a measure of model fit. Lower values indicate a better fit. The *deviance* is a measure of fit that compares the fitted model with the perfect model. Higher level indicates a better fit.

```r
summary(gam_model_ns)
```

```
Family: gaussian
Link function: identity
```

```
Formula:
GDP ~ ns(FERTILITY) + ns(LITERACY)

Parametric coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)     -3.880      8.517  -0.456  0.65139
ns(FERTILITY)   -3.199     13.246  -0.242  0.81051
ns(LITERACY)    31.341      9.832   3.188  0.00296 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


R-sq.(adj) =  0.488   Deviance explained = 51.5%
GCV = 62.331  Scale est. = 57.536    n = 39
```

```
AIC_results <- c("cubic" = AIC_cubic, "natural spline" = AIC_ns)
BIC_cubic <- BIC(gam_model_cubic)
BIC_ns <- BIC(gam_model_ns)
BIC_results <- c("cubic" = BIC_cubic, "natural spline" = BIC_ns)
AIC_results
```

```
cubic natural spline
     248.3119       273.5997
```

```
BIC_results
```

```
cubic natural spline
       262.0275       280.2539
```

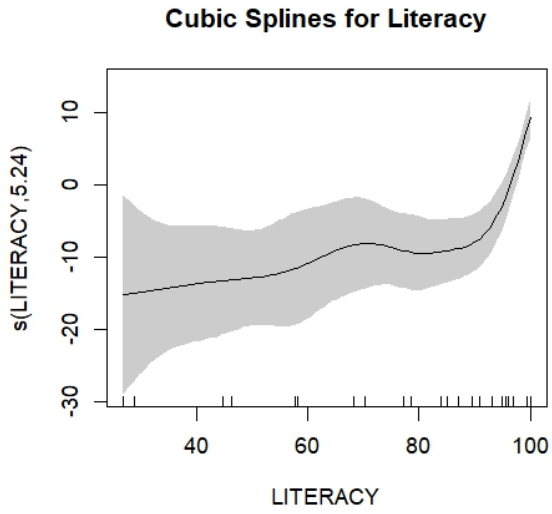


Figure 3: This is the plot for the gam model by using the cubic spline for literacy. The gray area is the confidence interval of the fitted smooth. This interval reflects the uncertainty for the estimation.

AIC estimates the relative quality of statistical models for a given dataset. It deals with the trade-off between the goodness of fit of the model and the complexity of the model. Lower AIC values indicate a better model. BIC is similar to AIC but includes a stronger penalty for models with more parameters. Lower BIC values indicate a better model. Therefore, the cubic is slightly better than the natural.

```
# Plot for cubic spline model
plot(gam_model_cubic, select = 2, scheme = 1, main = "Cubic Splines for
↪  Literacy")
```

5.2 Regression splines

Regression splines offer another solution, capturing the hidden curves and bends in your data. It works by dividing the X-axis into segments and fitting a separate polynomial function within each segment. The data range is divided into sections using specific points called knots. Few knots leads to a smoother spline, resembling a straight line if there are very few. More knots allows the spline to bend and curve more, potentially capturing intricate non-linear relationships. However, too many knots can lead to overfitting and abrupting changes. As the

number of knots increases, so does the degrees of freedom. These polynomial segments are then stitched together at the knots, ensuring smoothness across the entire spline. The assumptions is the continuity on the first and the second derivative. These popular splines offer flexibility but can be prone to overfitting if too many knots are used.

```r
#SPLINES FOR LITERACY
fit <- lm(GDP ~ bs(LITERACY, knots= c(25, 40, 60)), data=development) #cubic splines with 7 degree of freedom
#A sequence of values from the minimum to the maximum LITERACY in the dataset is created for prediction.
literacy_seq <- seq(min(development$LITERACY), max(development$LITERACY), length.out = 100)

# Predict GDP values using the model
predicted_gdp <- predict(fit, newdata = data.frame(LITERACY = literacy_seq))

# Plot the original data
plot(development$LITERACY, development$GDP,
     main = "Cubic spline fit",
     xlab = "LITERACY", ylab = "GDP", pch = 19, col = "blue")

# Add the fitted spline to the plot
lines(literacy_seq, predicted_gdp, col = "red", lwd = 2)
```

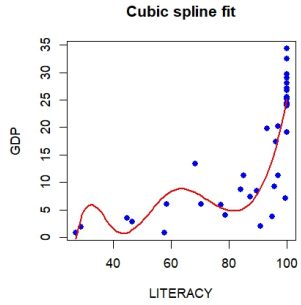


Figure 4: The figure 4 shows the cubic spline fit for literacy. Since the model has got only 7 degrees of freedom, it looks like to can't achieve all the points. We should increase the degrees of freedom in order to get more flexibility. It is influenced by the outliers, that seem a lot.

6 *Classifiers*

In order to apply the logistic regression, the linear discriminant analysis and the k nearest neighbors, we should discretize the fertility variable. As we know, we could have quantitative variables, qualitative variables and counts. Applying these classifiers means that the variable should be categorical.

```
#discretization of the variabile FERTILITY by adding a new column in the dataset.
#ifelse(FERTILITY > 2, ">2", "<2") is a conditional statement that checks the value of Fertility: if is greater than 2, fertility_class is set to ">2".
Otherwise, fertility_class is set to "<2".
development.1 <- development %>%
  mutate(FERTILITY_CLASS = ifelse(FERTILITY > 2, ">2", "<2"))
# Select the first 25 observations for training
training_data <- development.1[1:25, ]
test_data <- development.1[26:nrow(development.1), ]
# Convert FERTILITY_CLASS to a binary factor in order to define it as a categorical variable
training_data$FERTILITY_CLASS <- factor(training_data$FERTILITY_CLASS, levels = c("<2", ">2"))
```

6.1 Logistic Regression

Logistic regression is a statistical method used for binary classification problems. *It predicts the probability that a given input point belongs to a certain class.* Unlike linear regression, which is used for continuous target variables, logistic regression deals with categorical target variables. It models the *probability*

of a binary outcome using a logistic function. The goal is to find the best-fitting model to describe the relationship between the dependent binary variable and one or more independent variables.
Training a logistic regression model involves finding the best coefficients β that minimize the difference between the predicted probabilities and the actual outcomes. This is typically done using maximum likelihood estimation (MLE).
It establishes a threshold value (¿0.5) and it uses the sigmoid function, also known as the logistic function to map any real-valued number into the (0, 1) interval, making it suitable for probability prediction. It has got a s-shaped curve. In order to shift into a linear representation, we could use the log odds.

6.1.1 Coding part

```
glmfit <- glm(FERTILITY_CLASS ~ INTERNET + GDP + CO2 + CELLULAR + LITERACY,
                      data = training_data,
                      family = binomial) #it specifies that the model is a
                      ↪  logistic regression, suitable for binary outcomes.
summary(glmfit)
```

```
summary(glmfit)

Call:
glm(formula = FERTILITY_CLASS ~ INTERNET + GDP + CO2 + CELLULAR +
    LITERACY, family = binomial, data = training_data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.56138    6.23775   0.892    0.373
INTERNET     0.00206    0.08059   0.026    0.980
GDP         -0.39984    0.30233  -1.323    0.186
CO2          0.05377    0.49338   0.109    0.913
CELLULAR     0.03400    0.05076   0.670    0.503
LITERACY    -0.01802    0.08014  -0.225    0.822

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34.296  on 24  degrees of freedom
Residual deviance: 13.369  on 19  degrees of freedom
AIC: 25.369

Number of Fisher Scoring iterations: 6
```

Deviance measures the goodness of fit of a model:

- **Null deviance**: 34.296 on 24 degrees of freedom.

- This represents the deviance of a model with no predictors (only the intercept). It measures how well the response variable is predicted by a model with only the intercept.
- The decrease from the null deviance (34.296) to the residual deviance (13.369) indicates that the model with predictors explains more of the variability in the response variable than a model with only the intercept.
- **Residual deviance**: 13.369 on 19 degrees of freedom.
- This represents the deviance of the fitted model with predictors included. It measures how well the response variable is predicted by the full model.
-
- **Degrees of freedom**: Reflect the number of independent pieces of information available to estimate parameters. For the null deviance, it's the number of observations minus 1. For the residual deviance, it's the number of observations minus the number of estimated parameters.
- The AIC is a measure of the relative quality of a statistical model for a given set of data. It balances model fit and complexity by penalizing the number of parameters. Lower AIC values indicate a better model, considering both fit and complexity.
- The **AIC** value of 25.369 provides a way to compare models; a lower AIC value indicates a preferable model when comparing multiple models.
-
- **Number of Fisher Scoring iterations** indicates how quickly the model converges to the final parameter estimates. Fewer iterations suggest faster convergence.

None of the predictors (`INTERNET`, `GDP`, `CO2`, `CELLULAR`, `LITERACY`) have p-values less than 0.05. This means that, based on the given data and model, there is no statistically significant evidence to suggest that any of these predictors are related to the response variable `FERTILITY_CLASS`. The training dataset only contains 25 observations, which is quite small. This small sample size can lead to high variability in the estimates and reduced power to detect significant effects.

```
1 plot(glmfit)
```

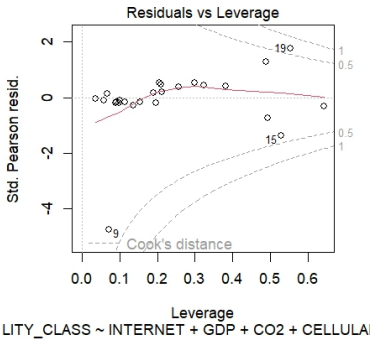


Figure 5: This plot shows that there are some influential points.

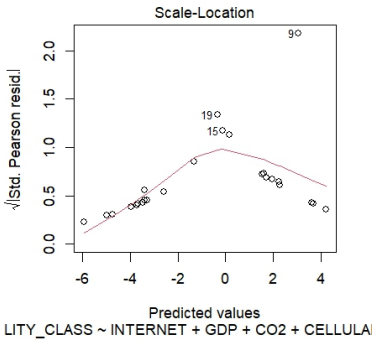


Figure 6: As we can see, there is a rend in the data showing heteroskedasticity.

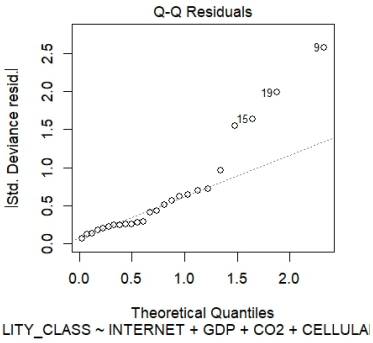


Figure 7: In general the data fall on a straight diagonal line, so we can conclude by saying that the residuals are normally distributed. Some of them are not normally distributed.

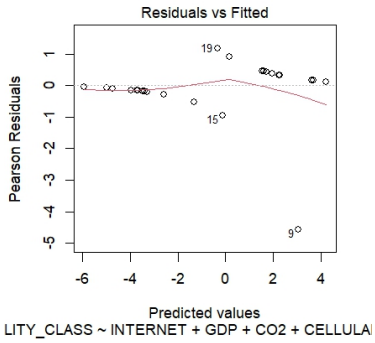


Figure 8: This little curve shows that the relationship between independent variable and dependent variable is not linear.

```r
glm.probs <- predict(glmfit, test_data, type = "response")
glm.pred <- ifelse(glm.probs > 0.5, ">2", "<2")
predicted_classes <- ifelse(glm.pred > 0.5, 1 , 0)
#confusion matrix
table(glm.pred, test_data$FERTILITY_CLASS)
```

```
glm.pred <2 >2
      <2  5  1
      >2  1  7
```

```r
mean(glm.pred == test_data$FERTILITY_CLASS)
```

```
[1] 0.8571429
```

```r
mean(glm.pred != test_data$FERTILITY_CLASS)
```

```
[1] 0.1428571
```

6.1.2 Explanation of the results

- The logistic regression model has an accuracy of approximately 85.7% on the test dataset.
- The misclassification rate is approximately 14.3%, meaning that 14.3% of the predictions are incorrect.
- The confusion matrix shows that the model correctly predicts both classes quite well, with only one instance each of false positives and false negatives.

6.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a supervised learning method that assumes predictors follow a multivariate normal distribution and that each class has the same covariance matrix. LDA maximizes class separability by using a linear discriminant function to maximize the distance between class means while keeping class variances minimal. This results in new data being assigned to the class based on proximity to the class mean, adjusted for the common covariance structure. The discriminant score essentially measures the distance of the new data point from the class means, adjusted for the covariance structure and prior probabilities. It can be interpreted as the log of the posterior probabilities (up to an additive constant). Thus, LDA assigns the new data point to the class with the highest posterior probability.

The decision boundary is linear, which can coincide with the Bayes classifier under LDA assumptions. LDA typically has high bias but low variance, achieving maximum separability between classes.

The **ROC curve** is crucial for evaluating binary classifiers, showing the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) across different thresholds. The **Area Under the ROC Curve (AUC)** indicates classifier performance, with higher AUC values signifying better classifiers.

```
#applying the linear discriminant analysis by using Fertility_class we are
↪  trying to maximize the separation between the <2 and >2
lda.fit <- lda(FERTILITY_CLASS ~ INTERNET + GDP + CO2 + CELLULAR + LITERACY,
               data = training_data)
coef(lda.fit)
```

```
              LD1
INTERNET -0.013096914
GDP      -0.164076888
CO2       0.014951834
CELLULAR  0.005683002
LITERACY -0.001400256
```

```
#we're assigning new observations to each class by using test data and the
↪  linear discriminant analysis.
lda.pred <- predict(lda.fit, test_data)
lda.class <- lda.pred$class
table(lda.class, test_data$FERTILITY_CLASS)
```

```
lda.class <2 >2
      <2  5  1
      >2  1  7
```

The covariance matrix shows that:

- 5 observations were correctly classified as <2.
- 7 observations were correctly classified as >2.
- 1 observation was misclassified as <2.
- 1 observation was misclassified as >2.

```
#this evaluates the accuracy of our prediction.
mean(lda.class == test_data$FERTILITY_CLASS)
```

```
[1] 0.8571429
```

```r
mean(lda.class != test_data$FERTILITY_CLASS)
```

```
[1] 0.1428571
```

6.2.1 Decision boundary

```r
# Fit LDA model using only INTERNET and GDP
lda.fit <- lda(FERTILITY_CLASS ~ INTERNET + GDP, data = training_data)

# Create a grid of values
grid <- expand.grid(
  INTERNET = seq(min(training_data$INTERNET), max(training_data$INTERNET),
  ↪  length = 100),
  GDP = seq(min(training_data$GDP), max(training_data$GDP), length = 100)
)

# Predict class for each point in the grid
grid_pred <- predict(lda.fit, grid)$class

# Add predictions to the grid
grid$FERTILITY_CLASS <- grid_pred

# Plot decision boundaries and data points
ggplot() +
  geom_tile(data = grid, aes(x = INTERNET, y = GDP, fill = FERTILITY_CLASS),
  ↪  alpha = 0.3) +
  geom_point(data = training_data, aes(x = INTERNET, y = GDP, color =
  ↪  FERTILITY_CLASS)) +
  labs(title = "LDA Decision Boundary", x = "INTERNET", y = "GDP") +
  theme_minimal()

```

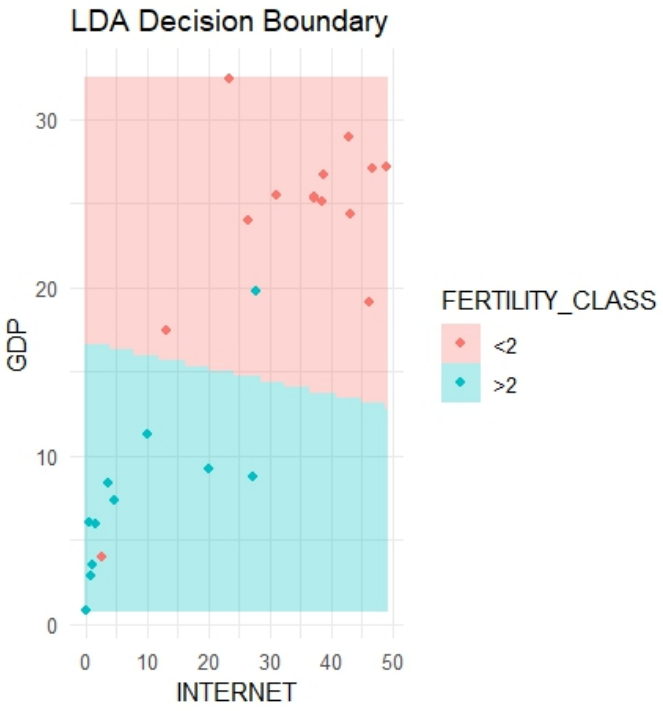


Figure 9: This plot show the decision boundary by only using GDP and internet. The blue class is ¿2, while the red class is ¡2. There are only two misclassified points.

6.3 K-Nearest neighbors

K-Nearest Neighbors (kNN) is a non-parametric, supervised machine learning algorithm widely used for classification tasks. Its ease of understanding and implementation make it a popular choice for various applications.

At its core, kNN classifies new data points by finding the k most similar instances (neighbors) in the training data with known labels.
It then assigns the most frequent class label among those neighbors to the new data point. Data points are represented as *feature vectors*, where each feature corresponds to an attribute of the data.
kNN needs a way to measure how "close" two data points are. This is where the distance metric comes in.
Common choices include Euclidean distance (straight-line distance) or Manhattan distance (sum of absolute differences in features). It calculates the distance between the new data point and all the labeled points in the training data and after choosing a value for "k" (the number of nearest neighbors), it assigns the new data to the most frequent class.
If k is small, noise can appear; if k is large we could lose local patterns. There-

fore, the valòue of k must be chosen through cross validation.

```
#KNN
train_x <- training_data[, c("INTERNET", "CELLULAR", "GDP", "CO2", "LITERACY")]
train_y <- training_data$FERTILITY_CLASS
test_x <- test_data[, c("INTERNET", "CELLULAR", "GDP", "CO2", "LITERACY")]
test_y <- test_data$FERTILITY_CLASS
train_x1 <- scale(train_x)
test_x1 <- scale(test_x)
knnmodel1 <- knn(train = train_x1, test = test_x1, cl = train_y, k = 10)
knnmodel2 <- knn(train = train_x1, test = test_x1, cl = train_y, k = 15)
knnmodel3 <- knn(train = train_x1, test = test_x1, cl = train_y, k = 5)
mean(knnmodel1 == test_y)
```

```
[1] 0.8571429
```

```
mean(knnmodel2 == test_y)
```

```
[1] 0.8571429
```

```
mean(knnmodel3 == test_y)
```

```
[1] 0.8571429
```

```
mean(knnmodel1 != test_y)
```

```
[1] 0.1428571
```

```
mean(knnmodel2 != test_y)
```

```
[1] 0.1428571
```

```
mean(knnmodel3 != test_y)
```

```
[1] 0.1428571
```

These results can be explained by this plot.

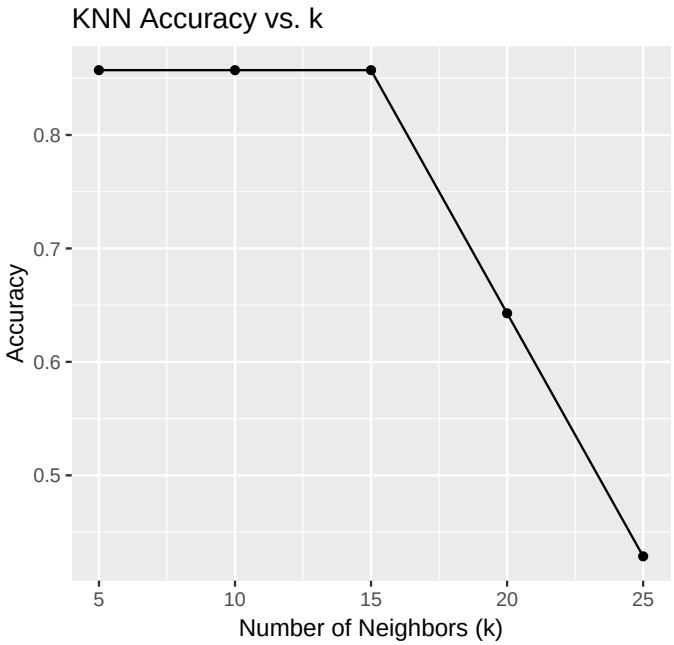


Figure 10: This plot analyzes the level of accuracy with different values of k. As we can see, for k=5, 10 and 15 the level of accuracy is almost the same; instead for k=20, 25 the level of accuracy is becoming low.

```r
# Evaluate KNN models with different values of k
k_values <- c(5, 10, 15, 20, 25)
accuracy_knn <- sapply(k_values, function(k) {
  knn_model <- knn(train_x, test_x, train_y, k = k)
  mean(knn_model == test_y)
})

# Plot accuracy vs. k
accuracy_df <- data.frame(k = k_values, Accuracy = accuracy_knn)

ggplot(accuracy_df, aes(x = k, y = Accuracy)) +
  geom_line() +
  geom_point() +
  labs(title = "KNN Accuracy vs. k", x = "Number of Neighbors (k)", y =
  ↪  "Accuracy")
```