Naive Bayes Classifier And Data Cleaning Assignment

Answer submission

Student ID – 10899486

Name - S.C.S.Sandanayake

Degree – Data science(Plymouth)

Question

Employ the provided dataset to forecast the "final grade" column utilizing the Naive Bayes algorithm. Execute the task through two distinct methodologies: initially, without engaging in any preprocessing, and subsequently, after applying data preprocessing techniques. Develop a concise report encompassing code snippets with succinct explanations. Ultimately, analyze and contrast the variances in model accuracies between the instances with and without preprocessing.

- Answer 1 – R script for get accuracy of the model without data preprocessing and cleaning

```r
library(e1071)
library(caret)

dataset <- read.csv("D:\\R\\week 9\\student_portuguese_clean.csv")
#View(dataset)

# Assuming 'final_grade' is the target column
target_col <- "final_grade"
x <- dataset[, setdiff(names(dataset), target_col)]
y <- dataset[[target_col]]

# Split the dataset into train and test
set.seed(600)
splitIndex <- createDataPartition(y, p = 0.75, list = FALSE)
train_data <- dataset[splitIndex, ]
test_data <- dataset[-splitIndex, ]

# train the Naive Bayes model
nb_model <- naiveBayes(final_grade ~ ., data = train_data)
# Make predictions on the test set
y_pred <- predict(nb_model, newdata = test_data)

#test data summary
summary(test_data)
#predicted results table
predicted_results <-predict(nb_model,newdata=test_data)
table(predicted_results)

#handling categorical variables for calculation
print(levels(y_pred))
print(levels(test_data$final_grade))
levels(factor(y_pred))
levels(factor(test_data$final_grade))
y_pred <- as.factor(y_pred)
test_data$final_grade <- as.factor(test_data$final_grade)
y_pred <- factor(y_pred)
test_data$final_grade <- factor(test_data$final_grade)

levels(y_pred) <- levels(test_data$final_grade)

#confusionMatrix calculation
confmatrix1<- confusionMatrix(y_pred,test_data$final_grade)
confmatrix1

# Evaluate accuracy of the model
accuracy <- sum(y_pred == test_data$final_grade) / length(test_data$final_grade)
cat("Accuracy without preprocessing:", accuracy, "\n")
```

The first R script is for get accuracy level without data preprocessing and data cleaning. I used 2 libraries in the 1st R script, library e1071 used for machine learning and statistical modeling tasks and caret library used for evaluating predictive models (Naive Bayes classification model training) etc...

1) First loaded the required libraries, then view and read the data set. After that I specify the target column and features. . I assume the target column as “final grade”. Then split data set into training set and testing sets .

This data set has 649 observations and 34 variables. Test data set includes 161 observations and 34 variables, train data set have 488 observations and 34 variables. The seed value is set to 600 because the table contain nearly 600 observations and the split index p value set to 7.5 (3/4) .

```
Data
  dataset        649 obs. of 34 variables
  splitIndex     int [1:488, 1] 1 2 3 4 5 6 8 9 10 12 ...
  test_data      161 obs. of 34 variables
  train_data     488 obs. of 34 variables
  x              649 obs. of 33 variables
Values
  target_col     "final_grade"
  y              int [1:649] 11 11 12 14 13 13 13 13 17 13 ...
```

2) Next build the Naive bayes model using the train data set. After that made predictions on the test data set (y_pred).

```
Values
  target_col     "final_grade"
  y              int [1:649] 11 11 12 14 13 13 13 13 17 13 ...
  y_pred         Factor w/ 17 levels "0","1","5","6",..: 12 15 12 12 9 12 11 15 1…
```

3) Then generate the summery and predicted results table on test data set.

```
   free_time        social       weekday_alcohol weekend_alcohol     health        absences     
 Min.   :1.00   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   : 0.000  
 1st Qu.:3.00   1st Qu.:3.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:3.00   1st Qu.: 0.000  
 Median :3.00   Median :3.000   Median :1.000   Median :2.000   Median :4.00   Median : 2.000  
 Mean   :3.19   Mean   :3.294   Mean   :1.528   Mean   :2.281   Mean   :3.61   Mean   : 3.544  
 3rd Qu.:4.00   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:5.00   3rd Qu.: 5.250  
 Max.   :5.00   Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :5.00   Max.   :26.000  
 NA's   :3      NA's   :1       NA's   :2       NA's   :1       NA's   :2      NA's   :1       
    grade_1         grade_2        final_grade   
 Min.   : 5.00   Min.   : 0.00   Min.   : 0.00  
 1st Qu.:10.00   1st Qu.:10.00   1st Qu.:10.00  
 Median :11.00   Median :11.00   Median :12.00  
 Mean   :11.38   Mean   :11.52   Mean   :12.01  
 3rd Qu.:13.00   3rd Qu.:13.00   3rd Qu.:14.00  
 Max.   :18.00   Max.   :18.00   Max.   :18.00  
 NA's   :1                                      
> #predicted results table
> predicted_results <-predict(nb_model,newdata=test_data)
> table(predicted_results)
predicted_results
 0  1  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 
10  0  0  0  2  7 11 14 25  2  2 68  0  0 18  2  0 
```

4) Next step is handling categorical values before doing the confusion matrix calculations.

```
> #handling categorical variables for calculation
> print(levels(y_pred))
 [1] "0"  "1"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16" "17" "18" "19"
> print(levels(test_data$final_grade))
NULL
> levels(factor(y_pred))
 [1] "0"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "17" "18"
> levels(factor(test_data$final_grade))
 [1] "0"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16" "17" "18"
> y_pred <- as.factor(y_pred)
> test_data$final_grade <- as.factor(test_data$final_grade)
> y_pred <- factor(y_pred)
> test_data$final_grade <- factor(test_data$final_grade)
> levels(y_pred) <- levels(test_data$final_grade)
```

5) Then get the confusion matrix calculations with statistics.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  6  7  8  9 10 11 12 13 14 15 16 17 18
        0   2  0  1  0  2  3  2  0  0  0  0  0  0  0
        6   0  0  1  0  1  0  0  0  0  0  0  0  0  0
        7   0  1  0  3  2  1  0  0  0  0  0  0  0  0
        8   0  0  1  3  2  3  2  0  0  0  0  0  0  0
        9   0  0  0  2  1  5  5  1  0  0  0  0  0  0
        10  0  0  0  1  1 13  6  3  0  0  0  1  0  0
        11  0  0  0  0  0  0  1  1  0  0  0  0  0  0
        12  0  0  0  0  0  0  0  1  1  0  0  0  0  0
        13  0  0  0  0  0  0  7 15 21 11 11  3  0  0
        14  0  0  0  0  0  0  0  0  0  3  3  3  4  5
        15  0  0  0  0  0  0  0  0  0  0  0  0  0  2
        16  0  0  0  0  0  0  0  0  0  0  0  0  0  0
        17  0  0  0  0  0  0  0  0  0  0  0  0  0  0
        18  0  0  0  0  0  0  0  0  0  0  0  0  0  0

Overall Statistics

               Accuracy : 0.2795
                 95% CI : (0.2117, 0.3556)
    No Information Rate : 0.1553
    P-Value [Acc > NIR] : 4.292e-05

                  Kappa : 0.1937

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 6 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 12
Sensitivity           1.00000 0.000000  0.00000  0.33333 0.111111   0.52000  0.043478  0.047619
Specificity           0.94969 0.987500  0.95570  0.94737 0.914474   0.91176  0.992754  0.992857
Pos Pred Value        0.20000 0.000000  0.00000  0.27273 0.071429   0.52000  0.500000  0.500000
Neg Pred Value        1.00000 0.993711  0.98052  0.96000 0.945578   0.91176  0.861635  0.874214
Prevalence            0.01242 0.006211  0.01863  0.05590 0.055901   0.15528  0.142857  0.130435
Detection Rate        0.01242 0.000000  0.00000  0.01863 0.006211   0.08075  0.006211  0.006211
Detection Prevalence  0.06211 0.012422  0.04348  0.06832 0.086957   0.15528  0.012422  0.012422
Balanced Accuracy     0.97484 0.493750  0.47785  0.64035 0.512792   0.71588  0.518116  0.520238
                     Class: 13 Class: 14 Class: 15 Class: 16 Class: 17 Class: 18
Sensitivity             0.9545   0.21429   0.00000   0.00000   0.00000   0.00000
Specificity             0.6619   0.89796   0.98639   1.00000   1.00000   1.00000
Pos Pred Value          0.3088   0.16667   0.00000       NaN       NaN       NaN
Neg Pred Value          0.9892   0.92308   0.91195   0.95652   0.97516   0.95652
Prevalence              0.1366   0.08696   0.08696   0.04348   0.02484   0.04348
Detection Rate          0.1304   0.01863   0.00000   0.00000   0.00000   0.00000
Detection Prevalence    0.4224   0.11180   0.01242   0.00000   0.00000   0.00000
Balanced Accuracy       0.8082   0.55612   0.49320   0.50000   0.50000   0.50000
```

6) Finally I evaluate the accuracy level for un-preprocessed instance.

```
> # Evaluate accuracy of the model
> accuracy <- sum(y_pred == test_data$final_grade) / length(test_data$final_grade)
> cat("Accuracy without preprocessing:", accuracy, "\n")
Accuracy without preprocessing: 0.2795031
> |
```

The accuracy level is 0.2795031 ~ 0.2795 .

- Answer 2 – R script for get accuracy of the model with data preprocessing and cleaning

```
library(e1071)
library(caret)
library(mice)

dataset <- read.csv("D:\\R\\week 9\\student_portuguese_clean.csv")
sum(is.na(dataset))
#View(dataset)

# Assuming the 'final_grade' is the target column
target_col <- "final_grade"
X <- dataset[, setdiff(names(dataset), target_col)]
y <- dataset[[target_col]]

# Impute missing values
imputed_data <- mice(X, method = "rf")
# Complete the imputation process
completed_data <- complete(imputed_data)

# Remove rows with missing values
dataset <- dataset[complete.cases(dataset), ]

# Split the dataset into training and testing sets
set.seed(600)
splitIndex <- createDataPartition(y, p = 0.75, list = FALSE)
train_data <- dataset[splitIndex, ]
test_data <- dataset[-splitIndex, ]

# Encode categorical variables
dummy_transform <- dummyVars(formula = paste(target_col, "~ ."), data = train_data, fullRank = TRUE)
train_data_processed <- predict(dummy_transform, newdata = train_data)
test_data_processed <- predict(dummy_transform, newdata = test_data)

#Check Categorical Variables
sapply(train_data, function(x) length(unique(x)))

head(train_data_processed)

# Preprocess the data
preprocess_params <- preProcess(train_data[, setdiff(names(train_data), target_col)], method=c("center", "scale"))
train_data_processed <- predict(preprocess_params, train_data)
test_data_processed <- predict(preprocess_params, test_data)

# Initialize and train the Naive Bayes model
nb_model <- naiveBayes(train_data_processed[, setdiff(names(train_data_processed), target_col)], train_data$final_grade)

# Make predictions on the test set
y_pred <- predict(nb_model, newdata = test_data_processed)

summary(test_data)
predicted_results <-predict(nb_model,newdata=test_data)
table(predicted_results)

```

```r

#handling categorical variables for calculation
print(levels(y_pred))
print(levels(test_data$final_grade))
levels(factor(y_pred))
levels(factor(test_data$final_grade))
y_pred <- as.factor(y_pred)
test_data$final_grade <- as.factor(test_data$final_grade)
y_pred <- factor(y_pred)
test_data$final_grade <- factor(test_data$final_grade)

levels(y_pred) <- levels(test_data$final_grade)

#confusionMatrix calculation
confmatrix1<- confusionMatrix(y_pred,test_data$final_grade)
confmatrix1

# Evaluate the accuracy of the model
accuracy <- sum(y_pred == test_data_processed$final_grade) / length(test_data_processed$final_grade)
cat("Accuracy with preprocessing:", accuracy, "\n")
```

The second R script is for take the accuracy level with data preprocessing and data cleaning. I used 3 libraries in the 2st R script. Library e1071 , caret library , and mice library which is used for multiple imputations.

1) First load the required libraries and Then read the data set and check for the missing values, then view and read the data set. After that I specify the target column and features. . I assume the target column as “final grade”.

```
> dataset <- read.csv("D:\\R\\week 9\\student_portuguese_clean.csv")
> sum(is.na(dataset))
[1] 51
```

```
Data
 dataset          649 obs. of 34 variables
 x                649 obs. of 33 variables
Values
 target_col       "final_grade"
 y                int [1:649] 11 11 12 14 13 13 13 13 17 13 ...
```

2) After that impute the missing values by using random forest method . The mice library was used for that purpose.

```
imputed_data            Large mids (22 elements,  712.4 kB)
   $ data          :'data.frame':       649 obs. of  33 variables:
    ..$ student_id          : int [1:649] 1 2 3 4 5 6 7 8 9 10 ...
    ..$ school              : chr [1:649] "GP" "GP" "GP" "GP" ...
    ..$ sex                 : chr [1:649] "F" "F" "F" "F" ...
    ..$ age                 : int [1:649] 18 17 15 15 16 16 16 17 15 15 ...
    ..$ address_type        : chr [1:649] "Urban" "Urban" "Urban" "Urban" ...
    ..$ family_size         : chr [1:649] "Greater than 3" "Greater than 3" "Less than or equal to…
    ..$ parent_status       : chr [1:649] "Apart" "Living together" "Living together" "Living toge…
    ..$ mother_education    : chr [1:649] "higher education" "primary education (4th grade)" "prim…
    ..$ father_education    : chr [1:649] "higher education" "primary education (4th grade)" "prim…
    ..$ mother_job          : chr [1:649] "at_home" "at_home" "at_home" "health" ...
    ..$ father_job          : chr [1:649] "teacher" "other" "other" "services" ...
```

```
  $ imp            :List of 33
   ..$ student_id         :'data.frame':       0 obs. of  5 variables:
   .. ..$ 1: logi(0)
   .. ..$ 2: logi(0)
   .. ..$ 3: logi(0)
   .. ..$ 4: logi(0)
   .. ..$ 5: logi(0)
   ..$ school             :'data.frame':       0 obs. of  5 variables:
   .. ..$ 1: logi(0)
   .. ..$ 2: logi(0)
   .. ..$ 3: logi(0)
   .. ..$ 4: logi(0)
   .. ..$ 5: logi(0)
   ..$ sex                :'data.frame':       0 obs. of  5 variables:
   .. ..$ 1: logi(0)
   .. ..$ 2: logi(0)
   .. ..$ 3: logi(0)
   .. ..$ 4: logi(0)
   .. ..$ 5: logi(0)
   ..$ age                :'data.frame':       3 obs. of  5 variables:
   .. ..$ 1: int [1:3] 17 15 15
   .. ..$ 2: int [1:3] 16 15 15
   .. ..$ 3: int [1:3] 15 15 15
   .. ..$ 4: int [1:3] 16 15 17
   .. ..$ 5: int [1:3] 16 15 15
```

3) Then I split data set into training set and testing sets . The seed value is set to 600 that nearly equal to total table observations and the split index p value set to 7.5 (3/4) .

```
Data
 completed_data     649 obs. of 33 variables
 dataset            604 obs. of 34 variables
 imputed_data       Large mids (22 elements,  712.4 kB)
 splitIndex         int [1:488, 1] 1 2 3 4 5 6 8 9 10 12 ...
 test_data          153 obs. of 34 variables
 train_data         488 obs. of 34 variables
 x                  649 obs. of 33 variables
Values
 target_col         "final_grade"
 y                  int [1:649] 11 11 12 14 13 13 13 13 17 13 ...
```

4) Next encode the categorical values into numerical values . caret package is used to create dummy variables for categorical predictors. It transforms categorical variables into a binary (0 or 1) format.

5) And after that I checked the categorical variables.

```
> #Check Categorical Variables
> sapply(train_data, function(x) length(unique(x)))
         student_id              school                 sex                 age        address_type         family_size
                452                   3                   3                   9                   3                   3
      parent_status    mother_education    father_education          mother_job          father_job school_choice_reason
                  3                   6                   6                   6                   6                   5
           guardian         travel_time          study_time      class_failures      school_support      family_support
                  4                   5                   5                   5                   3                   3
  extra_paid_classes          activities      nursery_school           higher_ed     internet_access romantic_relationship
                  3                   3                   3                   3                   3                   3
family_relationship           free_time              social     weekday_alcohol     weekend_alcohol              health
                  6                   6                   6                   6                   6                   6
           absences             grade_1             grade_2         final_grade
                 21                  18                  16                  18
```

```
dummy_transform          List of  9
    $ call      : language dummyVars.default(formula = paste(target_col, "~ ."), data =…
    $ form      :Class 'formula'  language final_grade ~ .
     .. ..- attr(*, ".Environment")=<environment: 0x0000027b30109f48>
    $ vars      : chr [1:34] "final_grade" "student_id" "school" "sex" ...
    $ facVars   : NULL
    $ lvls      : NULL
    $ sep       : chr "."
    $ terms     :Classes 'terms', 'formula'  language final_grade ~ student_id + school…
     .. ..- attr(*, "variables")= language list(final_grade, student_id, school, sex, a…
     .. ..- attr(*, "factors")= int [1:34, 1:33] 0 1 0 0 0 0 0 0 0 0 ...
     .. .. ..- attr(*, "dimnames")=List of 2
     .. .. .. ..$ : chr [1:34] "final_grade" "student_id" "school" "sex" ...
     .. .. .. ..$ : chr [1:33] "student_id" "school" "sex" "age" ...
     .. ..- attr(*, "term.labels")= chr [1:33] "student_id" "school" "sex" "age" ...
```

6) Now display head of process data. It displays the first few rows of the processed training data, which now includes the dummy variables for categorical predictors.

```
> head(train_data_processed)
  student_id schoolMS sexM age address_typeUrban family_sizeLess than or equal to 3 parent_statusLiving together
1          1        0    0  18                 1                                  0                            0
2          2        0    0  17                 1                                  0                            1
3          3        0    0  15                 1                                  1                            1
5          5        0    0  16                 1                                  0                            1
6          6        0    1  16                 1                                  1                            1
8          8        0    0  17                 1                                  0                            0
  mother_educationhigher education mother_educationnone mother_educationprimary education (4th grade)
1                                1                    0                                             0
2                                0                    0                                             1
3                                0                    0                                             1
5                                0                    0                                             0
6                                1                    0                                             0
8                                1                    0                                             0
  mother_educationsecondary education father_educationhigher education father_educationnone
1                                   0                                1                    0
2                                   0                                0                    0
3                                   0                                0                    0
5                                   1                                0                    0
6                                   0                                0                    0
8                                   0                                1                    0
  father_educationprimary education (4th grade) father_educationsecondary education mother_jobhealth mother_jobother m
1                                             0                                   0                0               0
2                                             1                                   0                0               0
3                                             1                                   0                0               0
5                                             0                                   1                0               1
6                                             0                                   1                0               0
8                                             0                                   0                0               1
  mother_jobteacher father_jobhealth father_jobother father_jobservices father_jobteacher school_choice_reasonhome
1                 0                0               0                  0                 1                        0
2                 0                0               1                  0                 0                        0
3                 0                0               1                  0                 0                        0
```

7) The next step is preprocessing the data .

```
preprocess_params          List of  21
  $ dim              : int [1:2] 451 33
  $ bc               : NULL
  $ yj               : NULL
  $ et               : NULL
  $ invHyperbolicSine: NULL
  $ mean             : Named num [1:12] 333.71 16.78 0.255 3.951 3.182 ...
   ..- attr(*, "names")= chr [1:12] "student_id" "age" "class_failures" "family_relationship" ...
  $ std              : Named num [1:12] 183.417 1.181 0.646 0.932 1.066 ...
   ..- attr(*, "names")= chr [1:12] "student_id" "age" "class_failures" "family_relationship" ...
  $ ranges           : NULL
  $ rotation         : NULL
  $ method           :List of 3
   ..$ center: chr [1:12] "student_id" "age" "class_failures" "family_relationship" ...
   ..$ scale : chr [1:12] "student_id" "age" "class_failures" "family_relationship" ...
   ..$ ignore: chr [1:21] "school" "sex" "address_type" "family_size" ...
  $ thresh           : num 0.95
  $ pcaComp          : NULL
  $ numComp          : NULL
  $ ica              : NULL
  $ wildcards        :List of 2
   ..$ PCA: chr(0)
   ..$ ICA: chr(0)
  $ k                : num 5
  $ knnSummary       :function (x, ...)
```

```
test_data_processed   153 obs. of 34 variables
train_data            488 obs. of 34 variables
train_data_processed  488 obs. of 34 variables
  $ student_id           : num  -1.81 -1.81 -1.8 -1.79 -1.79 ...
  $ school               : chr  "GP" "GP" "GP" "GP" ...
  $ sex                  : chr  "F" "F" "F" "F" ...
  $ age                  : num  1.033 0.186 -1.508 -0.661 -0.661 ...
  $ address_type         : chr  "Urban" "Urban" "Urban" "Urban" ...
  $ family_size          : chr  "Greater than 3" "Greater than 3" "Less than o…
  $ parent_status        : chr  "Apart" "Living together" "Living together" "L…
  $ mother_education     : chr  "higher education" "primary education (4th gra…
  $ father_education     : chr  "higher education" "primary education (4th gra…
  $ mother_job           : chr  "at_home" "at_home" "at_home" "other" ...
  $ father_job           : chr  "teacher" "other" "other" "other" ...
  $ school_choice_reason : chr  "course" "course" "other" "home" ...
  $ guardian             : chr  "mother" "father" "mother" "father" ...
  $ travel_time          : chr  "15 to 30 min." "<15 min." "<15 min." "<15 min…
  $ study_time           : chr  "2 to 5 hours" "2 to 5 hours" "2 to 5 hours" "…
  $ class_failures       : num  -0.395 -0.395 -0.395 -0.395 -0.395 ...
```

8) Now we can initialize and train the naive bayes model.

```
nb_model                    List of  5
  $ apriori  : 'table' int [1:17(1d)] 11 1 1 2 8 27 23 63 75 48 ...
   ..- attr(*, "dimnames")=List of 1
   .. ..$ train_data$final_grade: chr [1:17] "0" "1" "5" "6" ...
  $ tables   :List of 33
   ..$ student_id            : num [1:17, 1:2] 1.335 -0.876 -0.293 -0.617 0.396…
   .. ..- attr(*, "dimnames")=List of 2
   .. .. ..$ train_data$final_grade: chr [1:17] "0" "1" "5" "6" ...
   .. .. ..$ student_id             : NULL
   ..$ school                : 'table' num [1:17, 1:2] 0 1 1 1 0.375 ...
   .. ..- attr(*, "dimnames")=List of 2
   .. .. ..$ train_data$final_grade: chr [1:17] "0" "1" "5" "6" ...
   .. .. ..$ school                 : chr [1:2] "GP" "MS"
   ..$ sex                   : 'table' num [1:17, 1:2] 0.545 0 0 0 0.625 ...
   .. ..- attr(*, "dimnames")=List of 2
   .. .. ..$ train_data$final_grade: chr [1:17] "0" "1" "5" "6" ...
```

9) In this step make predictions form the test data set.

```
> predicted_results <-predict(nb_model,newdata=test_data)
> table(predicted_results)
predicted_results
 0  1  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
20  0  0  0 15  0 12  1 87  0  0  0 18  0  0  0  0
>
```

10) Next step is handling categorical values before doing the confusion matrix calculations.

```
> #handling categorical variables for calculation
> print(levels(y_pred))
 [1] "0"  "1"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16" "17" "18" "19"
> print(levels(test_data$final_grade))
NULL
> levels(factor(y_pred))
 [1] "0"  "7"  "8"  "9"  "10" "11" "12" "14" "16" "17" "18"
> levels(factor(test_data$final_grade))
 [1] "0"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16" "17" "18" "19"
> y_pred <- as.factor(y_pred)
> test_data$final_grade <- as.factor(test_data$final_grade)
> y_pred <- factor(y_pred)
> test_data$final_grade <- factor(test_data$final_grade)
>
> levels(y_pred) <- levels(test_data$final_grade)
>
```

11) After that get the confusion matrix and statistics values.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  6  7  8  9 10 11 12 13 14 15 16 17 18 19
        0   0  0  1  1  1  2  0  0  0  0  0  0  0  0  0
        6   0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
        7   1  0  0  6  0  4  0  0  0  0  0  0  0  0  0
        8   0  1  0  1  3  5  0  0  0  0  0  0  0  0  0
        9   1  0  0  0  3 10  5  1  0  0  0  0  0  0  0
        10  0  0  0  0  1  6  6  0  0  2  0  1  0  0  0
        11  0  0  0  0  0  0  3  3  0  1  0  0  0  0  0
        12  0  0  0  0  0  1  6 17 13 14  7  4  0  0  0
        13  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0
        14  0  0  0  0  0  0  0  0  0  2  2  5  5  2  1
        15  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0
        16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
        17  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
        18  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
        19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

Overall Statistics

               Accuracy : 0.2092
                 95% CI : (0.1477, 0.2822)
    No Information Rate : 0.183
    P-Value [Acc > NIR] : 0.229

                  Kappa : 0.1115

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 6 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
Sensitivity           0.00000 0.000000 0.000000 0.125000  0.33333   0.21429   0.15000    0.8095   0.00000   0.09524   0.00000   0.00000
Specificity           0.96689 0.993421 0.927632 0.937931  0.88194   0.92000   0.96992    0.6591   0.98571   0.88636   0.98611   1.00000
Pos Pred Value        0.00000 0.000000 0.000000 0.100000  0.15000   0.37500   0.42857    0.2742   0.00000   0.11765   0.00000       NaN
Neg Pred Value        0.98649 0.993421 0.992958 0.951049  0.95489   0.83942   0.88356    0.9560   0.91391   0.86029   0.94040   0.93464
Prevalence            0.01307 0.006536 0.006536 0.052288  0.05882   0.18301   0.13072    0.1373   0.08497   0.13725   0.05882   0.06536
Detection Rate        0.00000 0.000000 0.000000 0.006536  0.01961   0.03922   0.01961    0.1111   0.00000   0.01307   0.00000   0.00000
Detection Prevalence  0.03268 0.006536 0.071895 0.065359  0.13072   0.10458   0.04575    0.4052   0.01307   0.11111   0.01307   0.00000
Balanced Accuracy     0.48344 0.496711 0.463816 0.531466  0.60764   0.56714   0.55996    0.7343   0.49286   0.49080   0.49306   0.50000
                     Class: 17 Class: 18 Class: 19
Sensitivity            0.00000   0.00000  0.000000
Specificity            1.00000   1.00000  1.000000
Pos Pred Value             NaN       NaN       NaN
Neg Pred Value         0.96078   0.98039  0.993464
Prevalence             0.03922   0.01961  0.006536
Detection Rate         0.00000   0.00000  0.000000
Detection Prevalence   0.00000   0.00000  0.000000
Balanced Accuracy      0.50000   0.50000  0.500000
```

12) Finally evaluate the accuracy level of the model.

```
> # Evaluate the accuracy of the model
> accuracy <- sum(y_pred == test_data_processed$final_grade) / length(test_data_processed$final_grade)
> cat("Accuracy with preprocessing:", accuracy, "\n")
Accuracy with preprocessing: 0.2091503
```

The new accuracy level after the preprocessing data is 0.2091503 ~ 0.2092

Summery and conclusion

Conclusion:

The accuracy level before preprocessing the data is = 0.2795031 ~ 0.2795 .

The new accuracy level after the preprocessing is = 0.2091503 ~ 0.2092.

According to calculations after preprocessing the data the accuracy has been dropped by, 0.0703528.

Summary:

The first R script was coded to train a Naive Bayes classifier without the use of any preprocessing steps. It included the fundamental data segmentation, model training, and accuracy calculation steps. The model's accuracy on the test set was determined, offering an indication of performance without the need for data preparation. This accuracy score shows the Naive Bayes model's baseline performance on the given dataset.

The second R script extended the analysis by using data preprocessing steps. It has handling missing values using imputation, encoding categorical variables and scaling numeric predictors. After that the naïve bayes model was trained on preprocessed data. And the new accuracy calculated.