UNIVERSITY OF
PLYMOUTH

PUSL2076

Data Programming in R

Naive Bayes Classifier

Assignment

L B H C Pathmakumara

Index number-10899186

Without Preprocessing the dataset

```
> library(e1071)
> library(caret)
> library(caTools)
```

- library(e1071)-to get the classifier(naive bay's)
- library(caret)-to train modesl
- library(caTools)-data set manipulation

```
> # Read the dataset
> ds <- read.csv("F:\\Year 2 Sem 1\\R programming\\Datasets\\student_portuguese_clean.csv")
```

- We used the above code to read the dataset.
- We didn’t do any preprocessing in this step.
-

```
> # Perform data splitting
> split_ratio <- sample.split(ds, SplitRatio = 0.75)

> training_dataset <- subset(ds, split_ratio == TRUE)

> testing_dataset <- subset(ds, split_ratio == FALSE)
```

- We split the dataset into two parts for training and testing. We used 75% of the dataset for training and the other part for the testing.

```
> # Ensure 'final_grade' is a factor in both datasets with the same levels
> training_dataset$final_grade <- as.factor(training_dataset$final_grade)

> testing_dataset$final_grade <- as.factor(testing_dataset$final_grade)
```

- We train and test the dataset to predict final grade

```
> # Create training and testing data frames with the selected columns
> training_ds <- training_dataset[, 1:33]

> testing_ds <- testing_dataset[, 1:33]
```

- We are using 1 to 33 columns for our prediction.

```
> # Train the Naive Bayes model
> set.seed(400)

> model <- naiveBayes(final_grade ~ ., data = training_dataset)
```

- We set a seed for train it 400 times to get an accurate.
- We used Naïve Bays Classifier

```
> # Make predictions on the testing dataset
> predicted_results <- predict(model, newdata = testing_ds)

> # Ensure predicted results are factors with the same levels as testing_dataset$final_grade
> predicted_results <- factor(predicted_results, levels = .... [TRUNCATED]
```

```
> # Ensure predicted results are factors with the same levels as testing_dataset$final_grade
> predicted_results <- factor(predicted_results, levels = .... [TRUNCATED]

> # Create confusion matrix
> confusionMatrix(predicted_results, testing_dataset$final_grade)
Confusion Matrix and Statistics
```

```
> # Create confusion matrix
> confusionMatrix(predicted_results, testing_dataset$final_grade)
Confusion Matrix and Statistics

          Reference
Prediction  0  6  7  8  9 10 11 12 13 14 15 16 17 18
        0   5  0  0  0  3  4  1  0  0  1  0  0  0  0
        6   0  0  0  0  0  0  0  0  0  0  0  0  0  0
        7   0  0  0  0  0  0  0  0  0  0  0  0  0  0
        8   0  1  1  7  1  0  0  0  0  0  0  0  0  0
        9   0  1  1  0  1  1  0  0  0  0  0  0  0  0
        10  0  0  0  2  4 10  9  1  0  0  0  0  0  0
        11  0  0  0  1  2  5  9  1  1  0  0  0  0  0
        12  0  0  0  0  0  0  2  3  1  1  0  0  0  0
        13  0  0  0  0  0  0  0  0  0  0  0  0  0  0
        14  0  0  0  0  0  2  9 14 24  8 14  1  1  0
        15  0  0  0  0  0  0  0  0  0  0  0  0  0  0
        16  0  0  0  0  0  0  0  0  0  0  0  1  0  0
        17  0  0  0  0  0  0  0  0  0  0  3  4  4  1
        18  0  0  0  0  0  0  0  0  0  0  0  0  2  3
```

- This is the confusion matrix of the predicted results and the testing dataset.

```
Overall Statistics

               Accuracy : 0.2982
                 95% CI : (0.2308, 0.3728)
    No Information Rate : 0.1754
    P-Value [Acc > NIR] : 5.735e-05

                  Kappa : 0.2375

 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   0.0000   0.0000  0.70000 0.090909   0.45455   0.30000   0.15789
Specificity           0.94578   1.0000   1.0000  0.98137 0.981250   0.89262   0.92908   0.97368
Pos Pred Value        0.35714      NaN      NaN  0.70000 0.250000   0.38462   0.47368   0.42857
Neg Pred Value        1.00000   0.9883   0.9883  0.98137 0.940120   0.91724   0.86184   0.90244
Prevalence            0.02924   0.0117   0.0117  0.05848 0.064327   0.12865   0.17544   0.11111
Detection Rate        0.02924   0.0000   0.0000  0.04094 0.005848   0.05848   0.05263   0.01754
Detection Prevalence  0.08187   0.0000   0.0000  0.05848 0.023392   0.15205   0.11111   0.04094
Balanced Accuracy     0.97289   0.5000   0.5000  0.84068 0.536080   0.67358   0.61454   0.56579
                     Class: 13 Class: 14 Class: 15 Class: 16 Class: 17 Class: 18
Sensitivity              0.000   0.80000   0.00000  0.166667   0.57143   0.75000
Specificity              1.000   0.59627   1.00000  1.000000   0.95122   0.98802
Pos Pred Value             NaN   0.10959       NaN  1.000000   0.33333   0.60000
Neg Pred Value           0.848   0.97959   0.90058  0.970588   0.98113   0.99398
Prevalence               0.152   0.05848   0.09942  0.035088   0.04094   0.02339
Detection Rate           0.000   0.04678   0.00000  0.005848   0.02339   0.01754
Detection Prevalence     0.000   0.42690   0.00000  0.005848   0.07018   0.02924
Balanced Accuracy        0.500   0.69814   0.50000  0.583333   0.76132   0.86901
>
```

- This shows the accuracy level of the of the without preprocessing the dataset. Accuracy level is 29.82%.

After Preprocessing the dataset

```
library(e1071)
library(caret)
library(caTools)

# Read the dataset
ds <- read.csv("F:\\Year 2 Sem 1\\R programming\\Datasets\\student_portuguese_clean.csv")
ds
```

- We loaded the above described libraries for analysis the dataset.

```
> #convert strings to  numeric-encoding
> str(ds$school)
 chr [1:649] "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" "GP" ...

> ds$school<-as.numeric(factor(ds$school))

> ds$school
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [50] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [99] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[148] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[197] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[246] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[295] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[344] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[393] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[442] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[491] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[540] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[589] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[638] 2 2 2 2 2 2 2 2 2 2 2 2

> str(ds$sex)
 chr [1:649] "F" "F" "F" "F" "F" "M" "M" "F" "M" "M" "F" "F" "M" "M" "M" "F" "F" "F" "M" "M" "M" ...

> ds$sex<-as.numeric(factor(ds$sex))

> ds$sex
  [1] 1 1 1 1 1 2 2 1 2 2 1 1 2 2 2 1 1 1 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 1 2 2 1 1 1 2 2 2 1 1 1 2 2
 [50] 1 1 1 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 2 2 1 2 1 2 2 2 1 2 1 2 2 2 1 2 1 1 1 1 2 2 1 1 1 1 2 1 2 1
 [99] 1 1 2 2 2 1 2 1 1 2 2 1 2 1 1 2 2 2 2 2 2 2 2 1 2 1 2 1 1 2 2 1 1 1 1 1 1 2 1 2 1 2 1 2 1 1 1 2
[148] 1 1 2 1 2 2 1 1 2 1 1 1 1 1 2 2 2 2 1 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 1 1 2 1 2 2 2 2 2
[197] 1 1 1 2 2 2 1 2 1 1 2 2 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 2 1 1 1 2 2 1 1
[246] 2 2 2 2 2 2 2 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 2 2 2 2 1 2 1 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 1 1
[295] 1 2 2 2 2 1 1 1 2 2 1 1 2 2 2 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 2 2 2 2 1 1 1 1 2 1 1 2 2 2 1 1 2 2 1 1
[344] 1 2 1 2 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 2 1 1 1 2 2 1 1 2 1 1 1 1
[393] 1 1 1 2 2 1 1 1 2 2 1 1 1 1 1 1 2 2 1 2 1 2 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 2
[442] 2 2 2 2 2 2 2 2 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2 1 2 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 2
[491] 1 1 1 1 1 1 1 1 2 1 1 2 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 2 1 2 1 1 2 2 2 1 2 2 2 2 1
[540] 1 1 1 1 2 2 2 1 2 1 1 2 2 2 2 1 1 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 1 2 1 2 2 2 2 2 1 1 1 1 1 1
[589] 1 1 1 1 1 1 1 1 2 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 2 1 1 1 2 1 1 1 1 1 1 1 2
```

```
> ds$family_size<-as.numeric(ds$family_size)

> ds$family_size
  [1] 1 1 2 1 1 2 2 1 2 1 1 1 2 1 1 1 1 1 1 2 1 1 2 2 1 1 1 1 2 1 1 1 1 2 1 1 2 2 1 1
 [50] 1 2 2 2 1 2 1 1 1 2 1 1 1 2 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 2 1 1 2 1 2 1 1 1
 [99] 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 1 1 2
[148] 1 1 1 1 1 2 1 2 1 1 2 1 1 1 1 2 2 2 1 1 1 1 2 1 1 1 2 2 1 2 1 2 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1
[197] 1 2 1 1 1 2 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 2 2 1 2
[246] 1 1 1 2 1 2 1 1 2 2 1 1 1 1 2 1 1 2 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 2 1 2
[295] 1 2 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 1 2 2 2 1 1 1 1 1 1 1 2 1 1 2 2 1 1 2
[344] 1 2 2 2 2 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 2 1 2 1 1 2 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1
[393] 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 2 2 2 1 2 2 2 1 1 2 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 1 1 2 2 2 1 1 1 1
[442] 1 2 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 2 1 2 1 1 1 1 1 2 1 2 1 1 2 1 1
[491] 1 1 1 1 1 2 1 2 1 1 1 1 2 2 1 2 1 1 2 1 2 1 1 1 1 1 2 1 2 1 1 2 1 1 2 2 1 2 1 2 1 1 1 1 2 2 1 2 2 2
[540] 2 1 1 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 1 1 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 1 2
[589] 1 2 1 1 1 2 1 2 1 1 2 1 1 1 2 2 1 1 1 2 2 1 1 1 1 1 1 2 2 2 2 1 2 1 1 2 1 1 1 2 1 1 2 1 1 2 1 1 1
[638] 1 1 1 1 1 1 1 1 2 1 2 2

> str(ds$parent_status)
 chr [1:649] "Apart" "Living together" "Living together" "Living together" "Living together" ...

> ds$parent_status<-factor(ds$parent_status,levels = c("Apart","Living together"))

> ds$parent_status<-as.numeric(ds$parent_status)

> ds$parent_status
  [1] 1 2 2 2 2 2 2 1 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 1 2 2
 [50] 2 2 2 1 2 1 1 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
 [99] 2 2 2 2 2 2 2 1 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 1 2 1 2 2 2 2 2 2 1 2 1 2
[148] 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[197] 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
[246] 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2
[295] 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2
[344] 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[393] 2 2 1 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[442] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2
[491] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2
[540] 2 2 2 2 2 2 2 2 2 1 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 1 1 2 2 1 2 2

> ds$address_type <- factor(ds$address_type, levels = c("Urban", "Suburban", "Rural"))

> ds$address_type <- as.numeric(ds$address_type)

> str(ds$address_type)
 num [1:649] 1 1 1 1 1 1 1 1 1 1 ...

> ds$address_type<-as.numeric(ds$address_type)

> ds$address_type
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 3 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1
 [50] 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 3 3 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 1
 [99] 1 1 1 1 1 1 1 1 1 1 3 1 1 3 1 1 3 1 1 1 3 1 1 1 1 1 1 1 1 3 1 3 1 1 1 1 1 1 1 3 1 3 3 1 1 3 1 1
[148] 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 3 3 3 3 1 1 3 3 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1
[197] 1 1 1 1 1 1 1 3 1 1 1 3 1 1 1 3 1 1 1 1 1 3 3 1 1 1 1 3 1 1 1 1 3 1 1 1 1 3 1 1 1 1 3 1 1 1 1 1
[246] 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 1 1 3 1 1 3 1 1 1 3 1 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 1 1 1 1 1 3 1
[295] 3 1 1 3 1 3 1 1 3 1 1 1 3 1 3 1 3 1 1 1 3 1 1 1 3 1 1 1 3 1 1 1 1 3 1 1 1 1 1 1 1 1 1 3 3 1 1 1
[344] 1 1 1 1 3 1 1 3 1 3 1 1 1 3 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 3 1 1 1 1 3 1 3 1 1 1 3 1
[393] 1 1 3 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 3 3 1 1 1 3 1 1 1 1 1 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 1
[442] 1 3 3 3 3 1 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 3 3 1 1 3 1 1 3 3 3 1 3 1 3 1 3 1 1 3 3 3 1 3 1 3 1 3
[491] 3 1 3 1 3 1 3 1 1 1 1 1 3 1 1 3 1 1 3 3 3 1 3 1 1 1 3 3 1 3 1 3 1 1 1 3 1 1 1 1 1 1 3 1 1 1 3
[540] 1 1 3 1 3 3 1 1 3 1 3 1 1 1 1 3 3 3 3 3 3 3 1 3 1 3 3 3 3 1 1 3 1 1 1 3 3 3 1 1 3 1 1 1 3 3 3 1 3
[589] 1 1 1 1 1 1 1 1 1 3 1 3 3 1 1 3 1 1 1 3 1 1 3 3 3 1 3 1 1 1 3 1 1 3 3 1 3 1 3 3 3 1 3 3 3 3 3 1 3 1
[638] 3 1 3 3 3 1 3 3 1 1 1 3

> str(ds$address_type)
 num [1:649] 1 1 1 1 1 1 1 1 1 1 ...

> str(ds$family_size)
 chr [1:649] "Greater than 3" "Greater than 3" "Less than or equal to 3" "Greater than 3" ...

> ds$family_size<-factor(ds$family_size,levels = c("Greater than 3","Less than or equal to 3"))

> ds$family_size<-as.numeric(ds$family_size)
```

- We converted all the categorical data in to numeric as a part of the data preprocessing.

```
> #handling missing values
> #sum(is.na(ds))
>
> missing_cols <- colnames(ds)[colSums(is.na(ds)) >0]

> missing_cols
 [1] "age"                "mother_education"    "travel_time"         "study_time"
 [5] "class_failures"     "family_relationship" "free_time"           "social"
 [9] "weekday_alcohol"    "weekend_alcohol"     "health"              "absences"
[13] "grade_1"            "grade_2"

> # Impute missing values with column means
> for (col in missing_cols) {
+   ds[[col]][is.na(ds[[col]])] <- mean(ds[[col]], na.rm = TRUE)
+ }

> #remove rows that with missing values
> #na.omit(ds)
>
```

- Then we find all the missing values and take a mean of them. So its very useful for to make predictions.

```
> # Perform data splitting
> split_ratio <- sample.split(ds, SplitRatio = 0.75)

> training_dataset <- subset(ds, split_ratio == TRUE)

> testing_dataset <- subset(ds, split_ratio == FALSE)

> # Ensure 'final_grade' is a factor in both datasets with the same levels
> training_dataset$final_grade <- as.factor(training_dataset$final_grade)

> training_dataset$
+ testing_dataset$final_grade <- as.factor(testing_dataset$final_grade)

> # Create training and testing data frames with the selected columns
> training_ds <- training_dataset[, 1:34]

> testing_ds <- testing_dataset[, 1:34]

> # Train the Naive Bayes model
> set.seed(400)

> #model <- naiveBayes(final_grade ~ ., data = training_dataset)
> # Train the Naive Bayes model using the correct data frame
> # Train the Naive Baye .... [TRUNCATED]

> # Make predictions on the testing dataset
> predicted_results <- predict(model, newdata = testing_ds)

> # Convert predicted_results to a factor with the same levels as training_ds$final_grade
> predicted_results <- factor(predicted_results, levels = le .... [TRUNCATED]

> # Convert testing_dataset$final_grade to a factor with the same levels as predicted_results
> testing_dataset$final_grade <- factor(testing_dataset$ .... [TRUNCATED]

> # Create confusion matrix
> confusionMatrix(predicted_results, testing_dataset$final_grade)
Confusion Matrix and Statistics
```

- After preprocessing the data we apply the naïve bays classifier.

```
> # Create confusion matrix
> confusionMatrix(predicted_results, testing_dataset$final_grade)
Confusion Matrix and Statistics

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

- Then we took the confusion matrix of the preprocessed data.

```
Overall Statistics

               Accuracy : 0.1228
                 95% CI : (0.0777, 0.1816)
    No Information Rate : 0.2105
    P-Value [Acc > NIR] : 0.999

                  Kappa : 0.0792

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 5 Class: 6 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
Sensitivity           0.66667       NA       NA 0.000000  0.40000  0.00000  0.00000  0.027778   0.00000
Specificity           0.85119        1        1 0.882353  0.95181  1.00000  0.98780  0.992593   0.97368
Pos Pred Value        0.07407       NA       NA 0.000000  0.20000      NaN  0.00000  0.500000   0.00000
Neg Pred Value        0.99306       NA       NA 0.993377  0.98137  0.95322  0.95858  0.792899   0.88623
Prevalence            0.01754        0        0 0.005848  0.02924  0.04678  0.04094  0.210526   0.11111
Detection Rate        0.01170        0        0 0.000000  0.01170  0.00000  0.00000  0.005848   0.00000
Detection Prevalence  0.15789        0        0 0.116959  0.05848  0.00000  0.01170  0.011696   0.02339
Balanced Accuracy     0.75893       NA       NA 0.441176  0.67590  0.50000  0.49390  0.510185   0.48684
                     Class: 12 Class: 13 Class: 14 Class: 15 Class: 16 Class: 17 Class: 18 Class: 19
Sensitivity           0.000000  0.040000   0.20000   0.28571  0.090909   0.62500   0.42857        NA
Specificity           0.993506  0.993151   0.82609   0.87898  1.000000   0.77301   0.98780    0.9883
Pos Pred Value        0.000000  0.500000   0.06667   0.17391  1.000000   0.11905   0.60000        NA
Neg Pred Value        0.900000  0.857988   0.94326   0.93243  0.941176   0.97674   0.97590        NA
Prevalence            0.099415  0.146199   0.05848   0.08187  0.064327   0.04678   0.04094    0.0000
Detection Rate        0.000000  0.005848   0.01170   0.02339  0.005848   0.02924   0.01754    0.0000
Detection Prevalence  0.005848  0.011696   0.17544   0.13450  0.005848   0.24561   0.02924    0.0117
Balanced Accuracy     0.496753  0.516575   0.51304   0.58235  0.545455   0.69900   0.70819        NA
>
```

- So the accuracy level of the preprocessed dataset any how decreased to 12.28%.

Full Code: https://github.com/HChandeepa/Data-Set-Analysis-Using-NaiveBayesClassifier