

Milk Quality Prediction

Student Name: Md. Minhazul Bari Fahim

ID: 20-42176-1

Dept Name: CSE

Institute Name: American International University-Bangladesh

City, Country: Dhaka, Bangladesh

email address: minhazulbarifahim@gmail.com

Data set description: This dataset, which was generated through manual observations, is a vital resource for developing machine learning models to predict milk quality. The data set consists of 7 independent variables: pH, temperature, taste, odor, fat, turbidity, and color. The overall Grade or Quality of the milk is significantly influenced by these factors taken together. They perform an important role in facilitating predictive analysis for determining milk quality. The milk Grade, which can be binary in nature, is the desired variable in this situation. When the ideal conditions for Taste, Odor, Fat, and Turbidity are present, a value of 1 is assigned; otherwise, a value of 0 is given. The dataset also offers accurate pH and temperature readings.

- i. **pH:** Its define pH, ranges from 3 to 9.5 max: 6.25 to 6.90
- ii. **Temprature:** This column define temperature which ranges from 34'C to 90'C max: 34'C to 45.20'C
- iii. **Taste, Odor, Fat, Turbidity:** Those columns define Categorical data 0 (Bad) or 1 (Good). Max: 1 (Good)
- iv. **Colour:** This Column defines Colour of the milk which ranges from 240 to 255. Max: 255
- v. **Grade:** This Column defines Grade (Target) of the milk which is categorical data Where Low (Bad) or Medium (Moderate) High

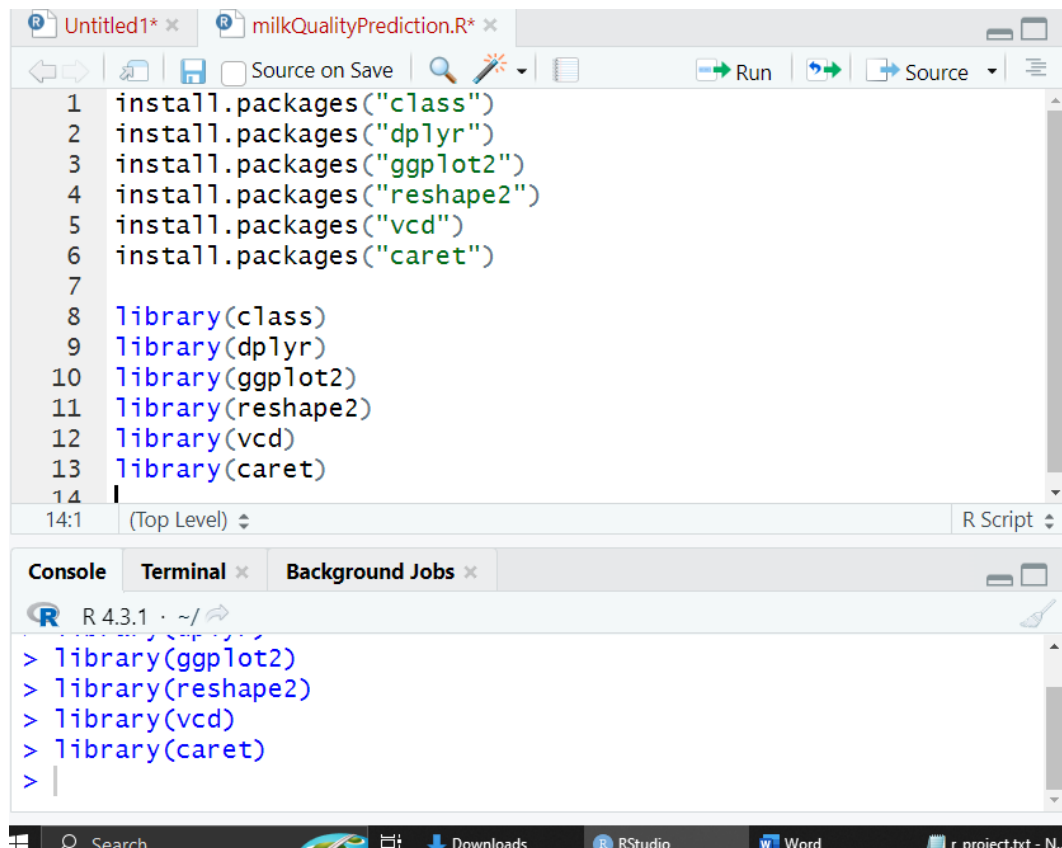
Methodology: The provided project performs the following steps:

- i. Data loading and preprocessing (handling missing values, converting categorical data, normalization, and feature selection).
- ii. Correlation analysis and visualization.
- iii. Data splitting into training and test sets.
- iv. Building a KNN classification model and evaluating its performance using accuracy.
- v. Performing 10-fold cross-validation to select the best KNN model.
- vi. Generating a confusion matrix and calculating precision and recall.

The project is a comprehensive implementation of a KNN classification workflow with data preprocessing, model training, evaluation, and performance analysis.

Data set link: <https://www.kaggle.com/cpluzshrijayan/milkquality>

✚ Load all the necessary libraries for the dataset.



The screenshot shows the RStudio interface with a script editor and a console. The script editor contains the following code:

```

1 install.packages("class")
2 install.packages("dplyr")
3 install.packages("ggplot2")
4 install.packages("reshape2")
5 install.packages("vcd")
6 install.packages("caret")
7
8 library(class)
9 library(dplyr)
10 library(ggplot2)
11 library(reshape2)
12 library(vcd)
13 library(caret)
14

```

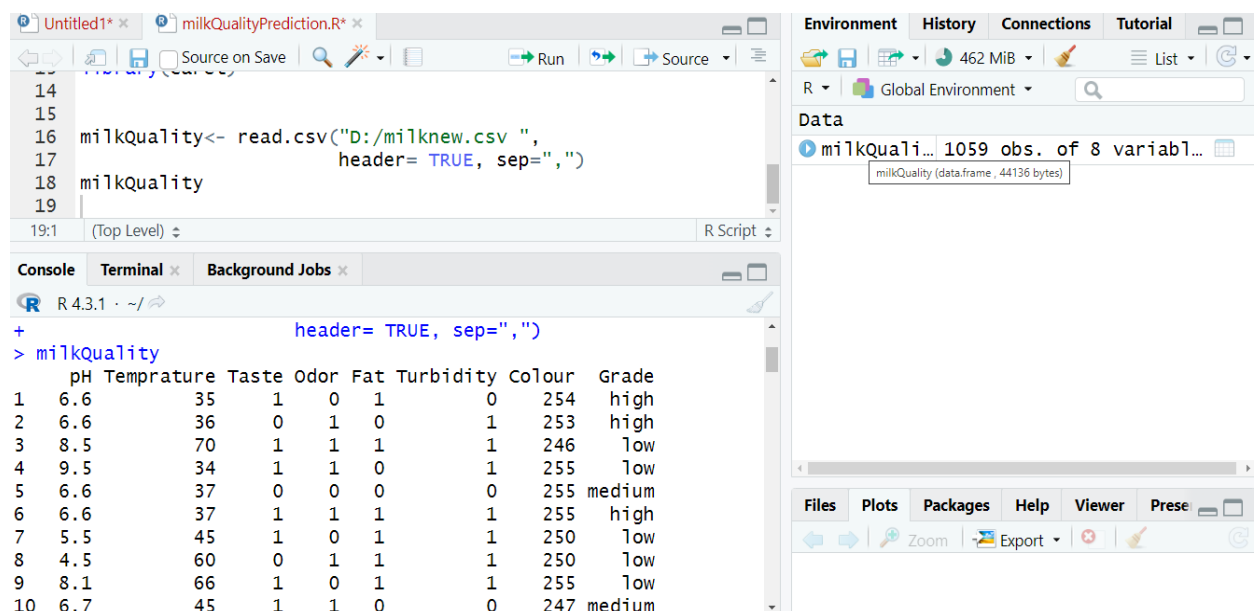
The console shows the execution of the library loading commands:

```

> library(ggplot2)
> library(reshape2)
> library(vcd)
> library(caret)
>

```

✚ Importing dataset (csv file) name as milkQuality and printing the dataset.



The screenshot shows the RStudio interface with a script editor, console, and environment pane. The script editor contains the following code:

```

14
15
16 milkQuality<- read.csv("D:/milknew.csv ",
17                       header= TRUE, sep=",")
18 milkQuality
19

```

The console shows the execution of the code, resulting in the following output:

```

+                               header= TRUE, sep=",")
> milkQuality
  pH Temperature Taste Odor Fat Turbidity Colour Grade
1  6.6          35    1  0  1          0    254  high
2  6.6          36    0  1  0          1    253  high
3  8.5          70    1  1  1          1    246  low
4  9.5          34    1  1  0          1    255  low
5  6.6          37    0  0  0          0    255  medium
6  6.6          37    1  1  1          1    255  high
7  5.5          45    1  0  1          1    250  low
8  4.5          60    0  1  1          1    250  low
9  8.1          66    1  0  1          1    255  low
10 6.7          45    1  1  0          0    247  medium

```

The environment pane on the right shows the Global Environment with the variable `milkQuality` (data.frame, 44136 bytes) loaded.

✚ Display detailed information about dataset.

The screenshot shows the RStudio interface with the script editor, console, and environment pane. The script editor contains the command `str(milkQuality)`. The console shows the output of this command, which is a detailed summary of the `milkQuality` dataset. The output indicates that the dataset is a data frame with 1059 observations and 8 variables. The variables are: pH (numeric), Temperature (integer), Taste (integer), Odor (integer), Fat (integer), Turbidity (integer), Colour (integer), and Grade (character). The output also shows the first few rows of the data.

```

20
21
22
23 str(milkQuality)
24
24:1 (Top Level)
R Script

R 4.3.1 ~ /
125 6.5      55      1      0      1      0      246      low
[ reached 'max' / getOption("max.print") -- omitted 934 rows ]
> str(milkQuality)
'data.frame':  1059 obs. of  8 variables:
 $ pH      : num  6.6 6.6 8.5 9.5 6.6 6.6 5.5 4.5 8.1 6.7 ...
 $ Temperature: int  35 36 70 34 37 37 45 60 66 45 ...
 $ Taste     : int  1 0 1 1 0 1 1 0 1 1 ...
 $ Odor      : int  0 1 1 1 0 1 0 1 0 1 ...
 $ Fat       : int  1 0 1 0 0 1 1 1 1 0 ...
 $ Turbidity : int  0 1 1 1 0 1 1 1 1 0 ...
 $ Colour    : int  254 253 246 255 255 255 250 250 255 247 ...
 $ Grade     : chr  "high" "high" "low" "low" ...
>

```

✚ Show the number of missing values in each column.

The screenshot shows the RStudio interface with the script editor, console, and environment pane. The script editor contains the command `colSums(is.na(milkQuality))`. The console shows the output of this command, which is a vector of zeros indicating that there are no missing values in any of the columns of the `milkQuality` dataset.

```

30
31 colSums(is.na(milkQuality))
32
33
33:1 (Top Level)
R Script

R 4.3.1 ~ /
$ Fat      : int  1 0 1 0 0 1 1 1 0 ...
$ Turbidity: int  0 1 1 1 0 1 1 1 0 ...
$ colour   : int  254 253 246 255 255 255 250 250 255
247 ...
$ Grade    : chr  "high" "high" "low" "low" ...
> colSums(is.na(milkQuality))
      pH Temperature      Taste      Odor      Fat
      0           0          0          0          0
Turbidity colour      Grade
      0           0          0
>

```

✚ Convert categorical to numeric data.

The screenshot shows an R script in the editor with the following code:

```

23
24
25 milkQuality$Grade<- as.numeric(factor(milkQuality$Grade))
26 gradeInNumeric <- c("high" = 3, "low" = 1, "medium" = 2)
27
28 milkQuality$Grade <- gradeInNumeric[milkQuality$Grade]
29 milkQuality
30

```

The console output shows the first 10 rows of the dataset:

	pH	Temperature	Taste	Odor	Fat	Turbidity	Colour	Grade
1	6.6	35	1	0	1	0	254	3
2	6.6	36	0	1	0	1	253	3
3	8.5	70	1	1	1	1	246	1
4	9.5	34	1	1	0	1	255	1
5	6.6	37	0	0	0	0	255	2
6	6.6	37	1	1	1	1	255	3
7	5.5	45	1	0	1	1	250	1
8	4.5	60	0	1	1	1	250	1

The Environment pane on the right shows the 'milkQuality' data frame with 1059 observations and 8 variables. The 'Values' pane shows the 'gradeInNu...' variable with values 3, 1, and 2.

✚ Use the min-max normalization after that save the normalize dataset name as n_milkQuality.

The screenshot shows an R script in the editor with the following code:

```

33
34
35 min_max_normalize <- function(x) {
36   (x - min(x)) / (max(x) - min(x))
37 }
38
39 n_milkQuality <- min_max_normalize(milkQuality)
40 print(n_milkQuality)
41

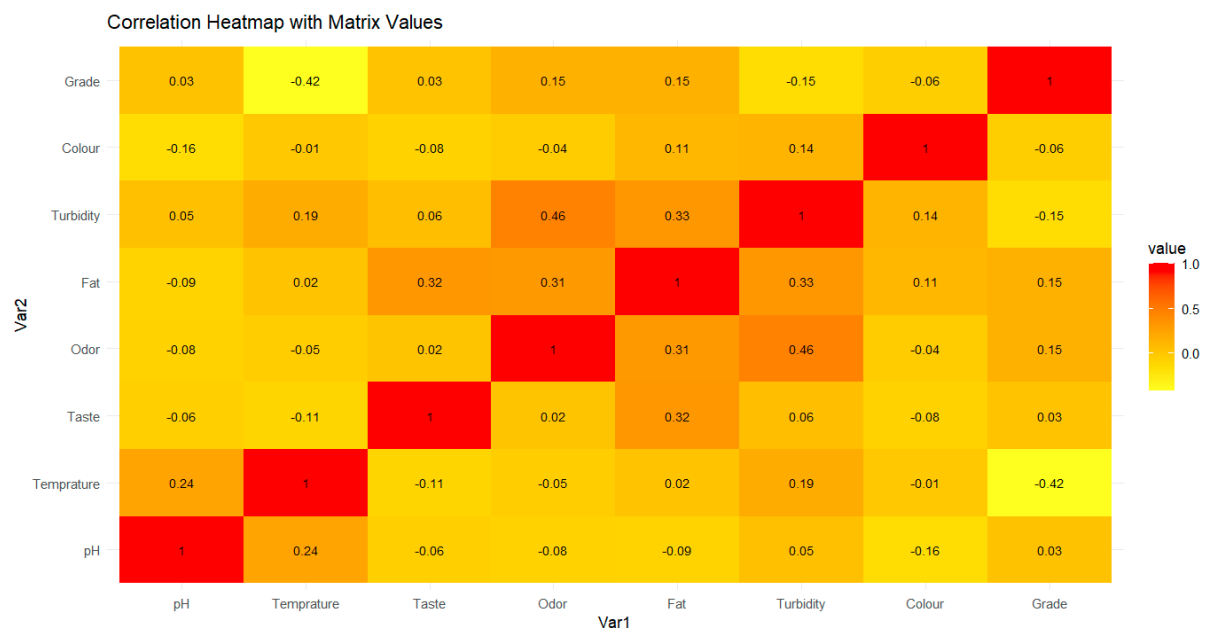
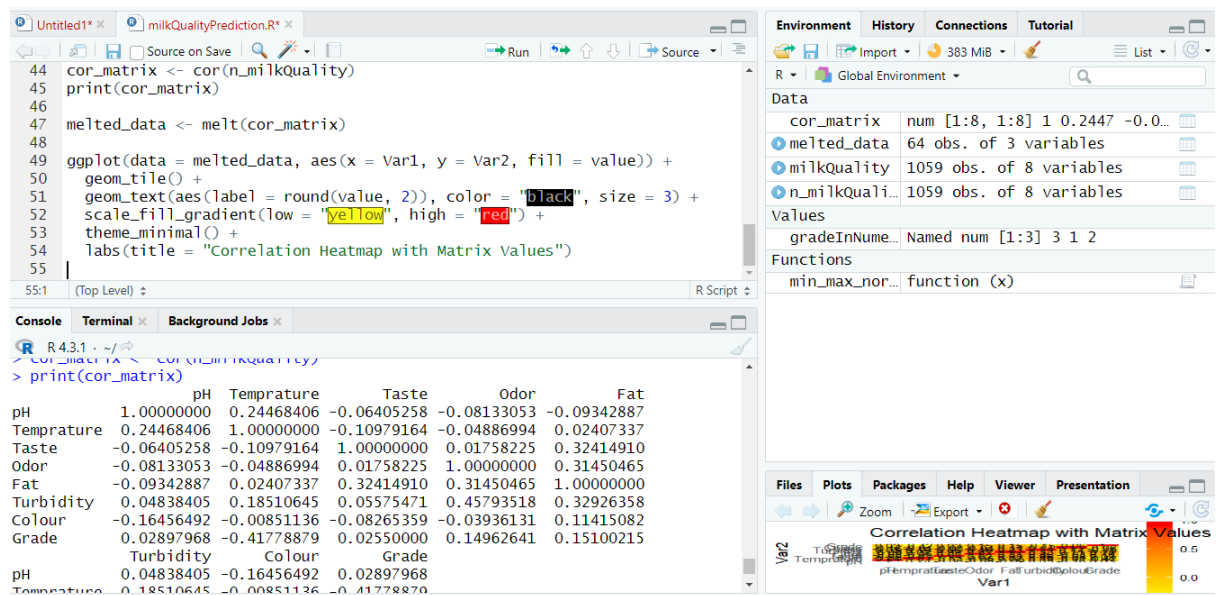
```

The console output shows the first 10 rows of the normalized dataset:

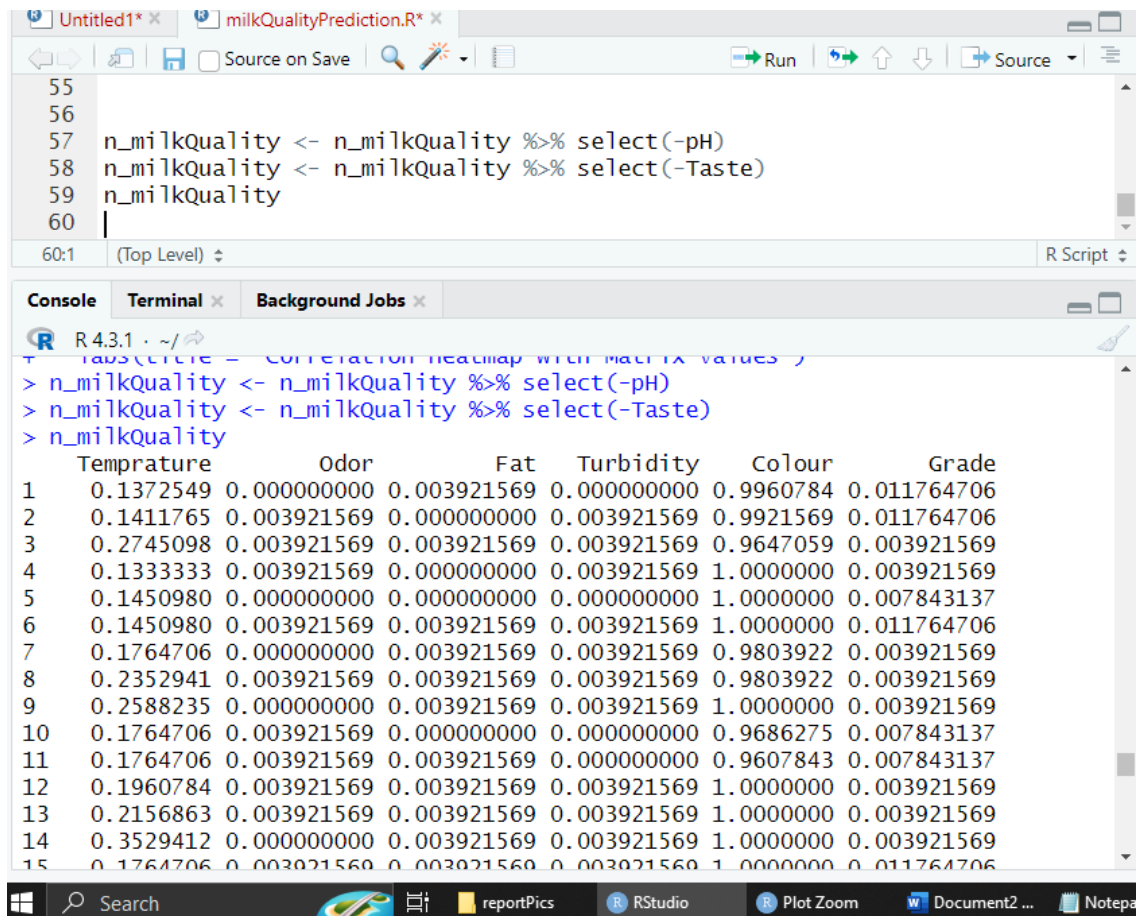
	pH	Temperature	Taste	Odor	Fat
1	0.02588235	0.1372549	0.003921569	0.000000000	0.003921569
2	0.02588235	0.1411765	0.000000000	0.003921569	0.000000000
3	0.03333333	0.2745098	0.003921569	0.003921569	0.003921569
4	0.03725490	0.1333333	0.003921569	0.003921569	0.000000000
5	0.02588235	0.1450980	0.000000000	0.000000000	0.000000000
6	0.02588235	0.1450980	0.003921569	0.003921569	0.003921569
7	0.02156863	0.1764706	0.003921569	0.000000000	0.003921569
8	0.01764706	0.2352941	0.000000000	0.003921569	0.003921569

The Environment pane on the right shows the 'n_milk' data frame with 1059 observations and 8 variables. The 'Functions' pane shows the 'min_max_n...' function.

Showing correlation heatmap with matrix values.



✚ After checking correlation, drop two column.



The screenshot shows the RStudio interface. The script editor contains the following code:

```

55
56
57 n_milkQuality <- n_milkQuality %>% select(-pH)
58 n_milkQuality <- n_milkQuality %>% select(-Taste)
59 n_milkQuality
60

```

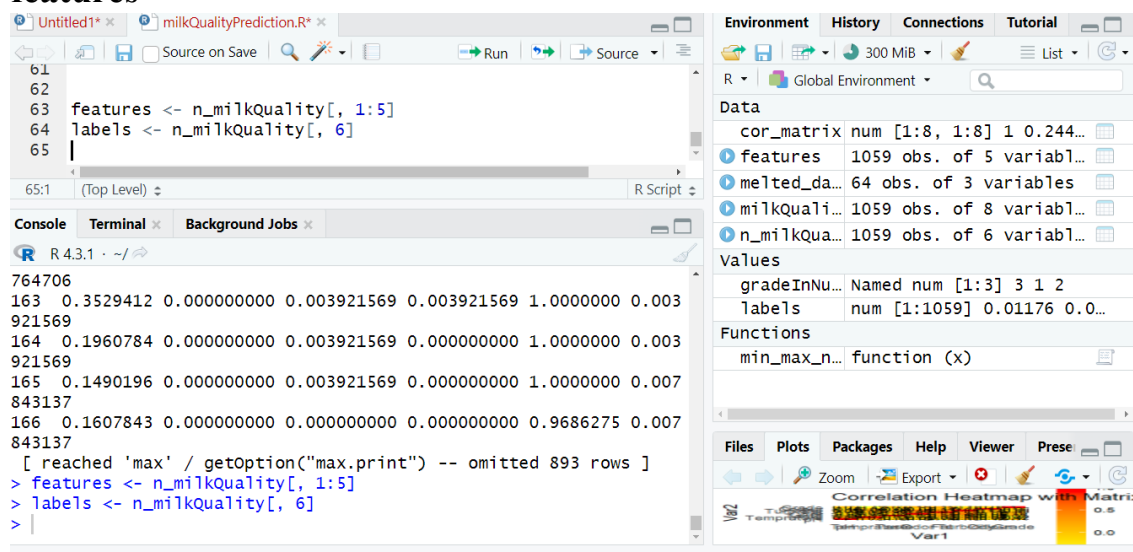
The console output shows the result of the code execution:

```

R 4.3.1 ~ /
> n_milkQuality <- n_milkQuality %>% select(-pH)
> n_milkQuality <- n_milkQuality %>% select(-Taste)
> n_milkQuality
  Temperature      Odor      Fat  Turbidity  Colour      Grade
1  0.1372549 0.000000000 0.003921569 0.000000000 0.9960784 0.011764706
2  0.1411765 0.003921569 0.000000000 0.003921569 0.9921569 0.011764706
3  0.2745098 0.003921569 0.003921569 0.003921569 0.9647059 0.003921569
4  0.1333333 0.003921569 0.000000000 0.003921569 1.0000000 0.003921569
5  0.1450980 0.000000000 0.000000000 0.000000000 1.0000000 0.007843137
6  0.1450980 0.003921569 0.003921569 0.003921569 1.0000000 0.011764706
7  0.1764706 0.000000000 0.003921569 0.003921569 0.9803922 0.003921569
8  0.2352941 0.003921569 0.003921569 0.003921569 0.9803922 0.003921569
9  0.2588235 0.000000000 0.003921569 0.003921569 1.0000000 0.003921569
10 0.1764706 0.003921569 0.000000000 0.000000000 0.9686275 0.007843137
11 0.1764706 0.003921569 0.003921569 0.000000000 0.9607843 0.007843137
12 0.1960784 0.003921569 0.003921569 0.003921569 1.0000000 0.003921569
13 0.2156863 0.003921569 0.003921569 0.003921569 1.0000000 0.003921569
14 0.3529412 0.000000000 0.003921569 0.003921569 1.0000000 0.003921569
15 0.1764706 0.003921569 0.003921569 0.003921569 1.0000000 0.011764706

```

✚ Assuming the last column is the target variable and the rest are features



The screenshot shows the RStudio interface. The script editor contains the following code:

```

61
62
63 features <- n_milkQuality[, 1:5]
64 labels <- n_milkQuality[, 6]
65

```

The console output shows the result of the code execution:

```

R 4.3.1 ~ /
163 0.3529412 0.000000000 0.003921569 0.003921569 1.0000000 0.003
921569
164 0.1960784 0.000000000 0.003921569 0.000000000 1.0000000 0.003
921569
165 0.1490196 0.000000000 0.003921569 0.000000000 1.0000000 0.007
843137
166 0.1607843 0.000000000 0.000000000 0.000000000 0.9686275 0.007
843137
[ reached 'max' / getOption("max.print") -- omitted 893 rows ]
> features <- n_milkQuality[, 1:5]
> labels <- n_milkQuality[, 6]
>

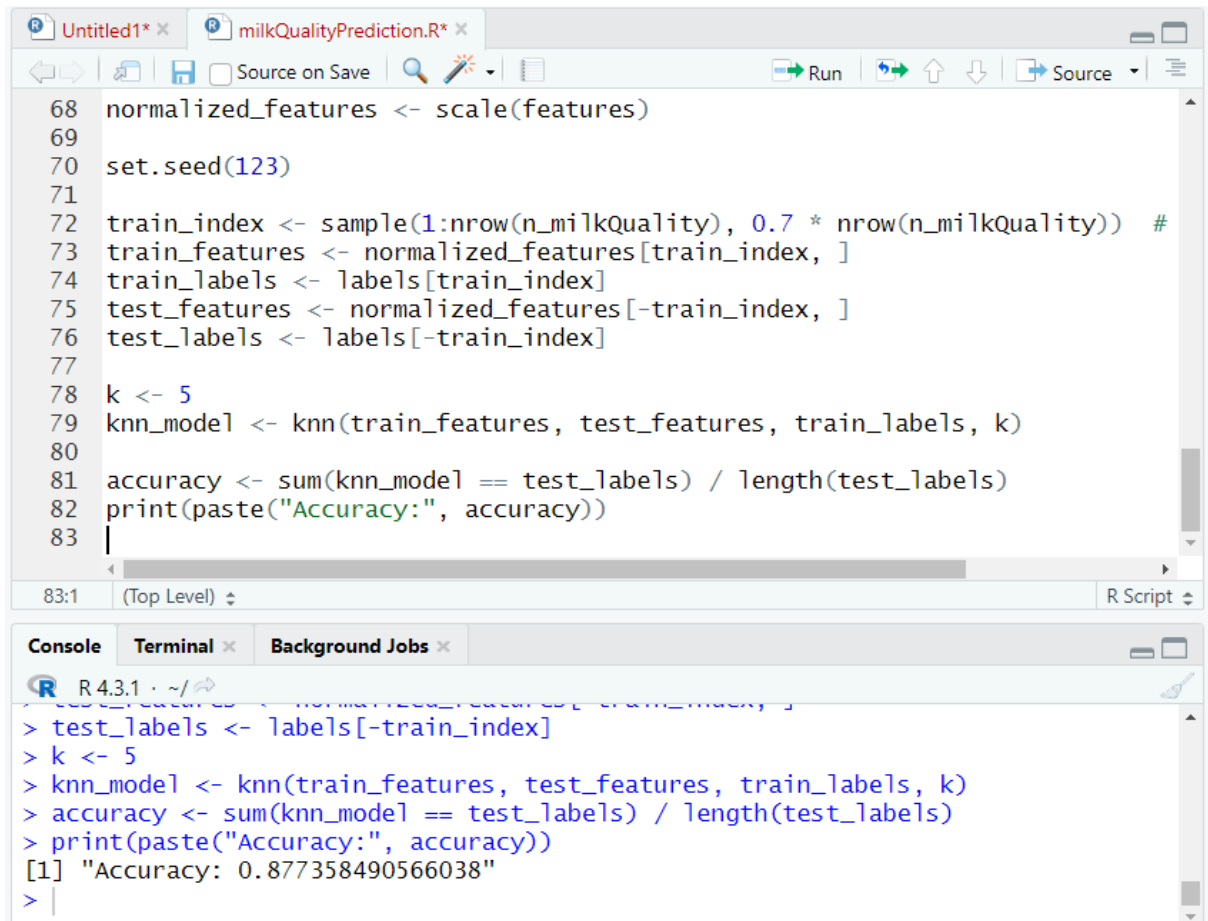
```

The Environment pane on the right shows the following objects:

- Data**
 - cor_matrix: num [1:8, 1:8] 1 0.244...
 - features: 1059 obs. of 5 variabl...
 - melted_da...: 64 obs. of 3 variables
 - milkQuali...: 1059 obs. of 8 variabl...
 - n_milkQua...: 1059 obs. of 6 variabl...
- Values**
 - gradeInNu...: Named num [1:3] 3 1 2
 - labels: num [1:1059] 0.01176 0.0...
- Functions**
 - min_max_n...: function (x)

At the bottom, a small heatmap titled "Correlation Heatmap with Matrix" is visible, showing the correlation between variables.

Built KNN model and find the accuracy.



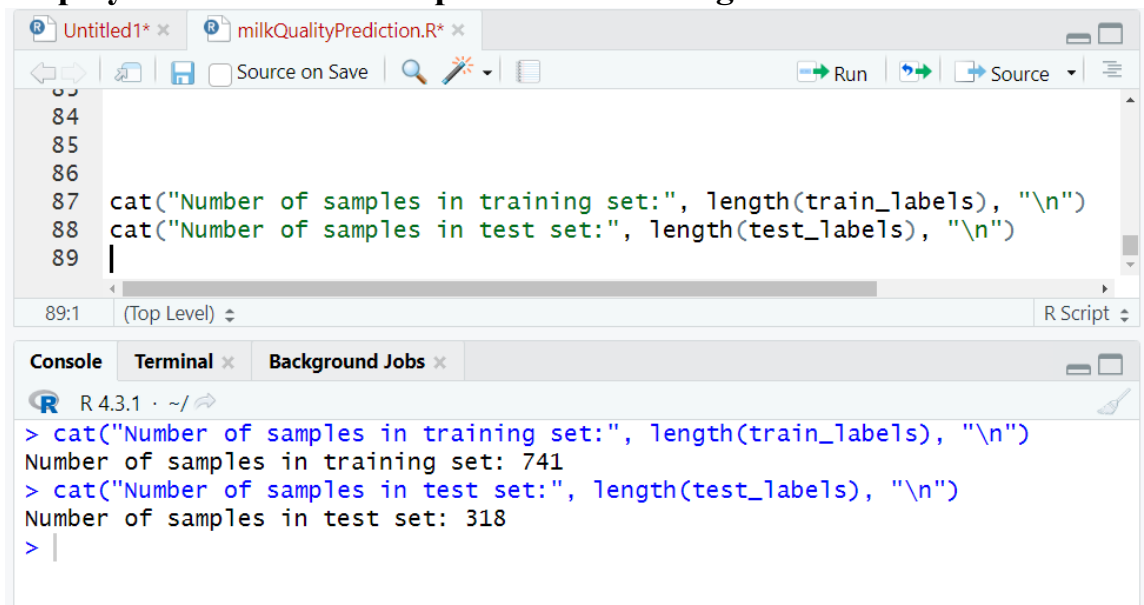
The screenshot shows an RStudio window with a script file named 'milkQualityPrediction.R'. The script contains the following R code:

```
68 normalized_features <- scale(features)
69
70 set.seed(123)
71
72 train_index <- sample(1:nrow(n_milkQuality), 0.7 * nrow(n_milkQuality)) #
73 train_features <- normalized_features[train_index, ]
74 train_labels <- labels[train_index]
75 test_features <- normalized_features[-train_index, ]
76 test_labels <- labels[-train_index]
77
78 k <- 5
79 knn_model <- knn(train_features, test_features, train_labels, k)
80
81 accuracy <- sum(knn_model == test_labels) / length(test_labels)
82 print(paste("Accuracy:", accuracy))
83
```

The console output shows the execution of the script, resulting in the following output:

```
> test_labels <- labels[-train_index]
> k <- 5
> knn_model <- knn(train_features, test_features, train_labels, k)
> accuracy <- sum(knn_model == test_labels) / length(test_labels)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.877358490566038"
>
```

Display the number of samples in the training and test sets



The screenshot shows an RStudio window with a script file named 'milkQualityPrediction.R'. The script contains the following R code:

```
84
85
86
87 cat("Number of samples in training set:", length(train_labels), "\n")
88 cat("Number of samples in test set:", length(test_labels), "\n")
89
```

The console output shows the execution of the script, resulting in the following output:

```
> cat("Number of samples in training set:", length(train_labels), "\n")
Number of samples in training set: 741
> cat("Number of samples in test set:", length(test_labels), "\n")
Number of samples in test set: 318
>
```


✚ Using 10-fold cross validation and predicting accuracy.

```

86
87 ctrl <- trainControl(method = "cv", number = 10)
88 knn_model_cv <- train(
89   x = train_features,
90   y = train_labels,
91   method = "knn",
92   trControl = ctrl,
93   tuneGrid = expand.grid(k = k)
94 )
95 print(knn_model_cv)
96 best_k <- knn_model_cv$bestTune$k
97 knn_model <- knn(train_features, test_features, train_labels, best_k)
98 accuracy <- sum(knn_model == test_labels) / length(test_labels)
99 print(paste("Accuracy (Test Set):", accuracy))
100
94:2 (Top Level)

```

Console Terminal Background Jobs

R 4.3.1 · ~/

no pre-processing
 Resampling: Cross-Validated (10 fold)
 Summary of sample sizes: 667, 666, 668, 668, 667, 667, ...
 Resampling results:

RMSE	Rsquared	MAE
0.001875978	0.6270674	0.0009379829

Tuning parameter 'k' was held constant at a value of 5
 > best_k <- knn_model_cv\$bestTune\$k
 > knn_model <- knn(train_features, test_features, train_labels, best_k)
 > accuracy <- sum(knn_model == test_labels) / length(test_labels)
 > print(paste("Accuracy (Test Set):", accuracy))
 [1] "Accuracy (Test Set): 0.877358490566038"
 >

✚ Confusion matrix and statistics

```

100
101
102 test_labels <- factor(test_labels, levels = levels(knn_model))
103
104 conf_matrix <- confusionMatrix(data = knn_model, reference = test_labels)
105 print(conf_matrix)
106
99:1 (Top Level) R Script

```

Console Terminal Background Jobs

R 4.3.1 · ~/

> print(conf_matrix)
 Confusion Matrix and Statistics

	Reference		
Prediction	0.00392156862745098	0.00784313725490196	0.0117647058823529
0.00392156862745098	115	1	7
0.00784313725490196	1	98	6
0.0117647058823529	12	12	66

Overall Statistics

Accuracy : 0.8774


```

R 4.3.1 ~ /
Confusion Matrix and Statistics

              Reference
Prediction    0.00392156862745098 0.00784313725490196 0.0117647058823529
0.00392156862745098              115                1                7
0.00784313725490196                1               98                6
0.0117647058823529                12              12              66

Overall Statistics

              Accuracy : 0.8774
              95% CI : (0.8362, 0.9113)
    No Information Rate : 0.4025
    P-Value [Acc > NIR] : <2e-16

              Kappa : 0.8138

    McNemar's Test P-Value : 0.3455

Statistics by Class:

              Class: 0.00392156862745098 Class: 0.00784313725490196
Sensitivity                0.8984                0.8829
Specificity                0.9579                0.9662
Pos Pred Value             0.9350                0.9333
Neg Pred Value             0.9333                0.9390
Prevalence                 0.4025                0.3491
Detection Rate             0.3616                0.3082
Detection Prevalence       0.3868                0.3302
Balanced Accuracy          0.9282                0.9245
              Class: 0.0117647058823529
Sensitivity                0.8354
Specificity                0.8996
Pos Pred Value             0.7333
Neg Pred Value             0.9430
Prevalence                 0.2484
Detection Rate             0.2075
Detection Prevalence       0.2830
Balanced Accuracy          0.8675
>

```

Finding the precision and recall.

```

Untitled1* x milkQualityPrediction.R* x
Source on Save Run
107
108
109
110 precision <- conf_matrix$table[2, 2] / sum(conf_matrix$table[, 2])
111 recall <- conf_matrix$table[2, 2] / sum(conf_matrix$table[2, ])
112
113 cat("Precision:", precision, "\n")
114 cat("Recall:", recall, "\n")
115 |
116
117
115:1 (Top Level) R Script
Console Terminal x Background Jobs x
R 4.3.1 ~ /
> precision <- conf_matrix$table[2, 2] / sum(conf_matrix$table[, 2])
> recall <- conf_matrix$table[2, 2] / sum(conf_matrix$table[2, ])
> cat("Precision:", precision, "\n")
Precision: 0.8828829
> cat("Recall:", recall, "\n")
Recall: 0.9333333
>

```