

1 CODE

```
1 #EDA
2 #load all libraries
3 library(dummies)
4 library(corrplot)
5 library(DataExplorer)
6
7 #upload the dataset
8 data <- read.csv(file.choose())
9
10 # grouping output variable
11 for (i in 1:395){
12   if (data$G3[i]<5){
13     data$G3[i] = 5
14   }else{
15     if (data$G3[i]<9){
16       data$G3[i] = 4
17     }else{
18       if (data$G3[i]<13){
19         data$G3[i] = 3
20       }else{
21         if (data$G3[i]<17){
22           data$G3[i] = 2
23         }else{
24           if (data$G3[i]<21){
25             data$G3[i] = 1
26           }
27         }
28       }
29     }
30   }
31 }
32
33 # exploring data set
34 plot_str(data)
35 plot_missing(data)
36
37 ## continuous data
38 plot_histogram(data)
39 plot_density(data)
40 plot_correlation(data, type = "continuous")
41
42 ## categorical data
43 plot_bar(data)
44 plot_correlation(data, type = "all")
45 plot_correlation(data[, -33], type = "discrete")
46
47 # converting categorical data into numeric data
48 new_data <- dummy.data.frame(data, names = c("school", "sex", "
  address", "famsize", "Pstatus", "Mjob", "Fjob", "reason", "
  guardian", "schoolsup", "famsup", "paid", "activities", "
  nursery", "higher", "internet", "romantic"))
49 new_data <- new_data[, -c(2,4,7,9,11,18,23,27,30,35,37,39,
50   41,43,45,47,49, 59)]
51 # ensuring we have k-1 dummy variables to avoid multicollinearity
52
```

```

53 # correlation plot
54 library(corrplot)
55 corrplot(cor(new_data), type = "upper")
56
57 # finding training set
58 set.seed(2020)
59 train <- sample(1:395, 0.70*395)
60 table(data[,33])
61 table(data[train,33])
62
63 # checking for outliers in training set
64 Sx <- cov(new_data[train,])
65 D2 <- mahalanobis(new_data[train,], colMeans(new_data[train,]), Sx)
66
67 plot(density(D2, bw = 0.5),
68      main="Squared Mahalanobis distances") ; rug(D2)
69 qqplot(qchisq(ppoints(395), df = 40), D2,
70       main = expression("Q-Q plot of Mahalanobis" * ~D^2 *
71                          " vs. quantiles of" * ~ chi[3]^2))
72 abline(0, 1, col = 'gray')
73
74 library(ggplot2)
75 df <- data.frame(Index = 1:276, y = D2)
76 ggplot(df, aes(Index, sqrt(y))) + geom_point(colour = "black") +
77   labs(y = "Squared Mahalanobis Distances")
78
79 # converting classes in target variable into categorical variable
80 data$G3 <- as.factor(data$G3)

```

Listing 1: EDA code

```

1 #loading libraries
2 library(caret)
3 library(class)
4 library(ggplot2)
5 library(scales)
6 library(dummies)
7
8 # read data into r
9 data <- read.csv(file.choose())
10
11 # grouping output variable
12 for (i in 1:395){
13   if (data$G3[i]<5){
14     data$G3[i] = 5
15   }else{
16     if (data$G3[i]<9){
17       data$G3[i] = 4
18     }else{
19       if (data$G3[i]<13){
20         data$G3[i] = 3
21       }else{
22         if (data$G3[i]<17){
23           data$G3[i] = 2
24         }else{
25           if (data$G3[i]<21){
26             data$G3[i] = 1
27           }

```

```

28     }
29   }
30 }
31 }
32 }
33 data$G3 <- as.factor(data$G3)
34
35 # converting categorical data into numeric data
36 new_data <- dummy.data.frame(data, names = c("school", "sex", "
      address", "famsize", "Pstatus", "Mjob", "Fjob", "reason", "
      guardian", "schoolsup", "famsup", "paid", "activities", "nursery
      ", "higher",
37 "internet", "romantic"))
38
39 # dividing new data
40 set.seed(2020)
41 train <- sample(1:395, 0.7*395)
42 pca.train <- new_data[train, -c(2, 4, 7, 9, 11, 18, 23, 27, 30,
43                                35, 37, 39, 41, 43, 45, 47, 49, 59)]
44 # ensuring we have k-1 dummy variables to avoid collinearity
45 pca.test <- new_data[-train, -c(2, 4, 7, 9, 11, 18, 23, 27, 30,
46                                35, 37, 39, 41, 43, 45, 47, 49, 59)]
47
48 # principal component analysis
49 pca2 <- prcomp(pca.train, scale=T, center=T)
50 pca1 <- prcomp(pca.train, scale=F, center=T)
51 pca <- prcomp(pca.train, scale=F, center=F)
52
53 # plotting resultatnt principal components
54 biplot(pca2, scale=0)
55 biplot(pca1, scale=0)
56 biplot(pca, scale=0)
57
58 # standard deviation of each principal component
59 std_dev2 <- pca2$sdev
60 std_dev1 <- pca1$sdev
61 std_dev <- pca$sdev
62
63 # compute variance
64 pca_var2 <- std_dev2^2
65 pca_var1 <- std_dev1^2
66 pca_var <- std_dev^2
67
68 # proportion of variance explained by each component
69 prop_var2 <- pca_var2/sum(pca_var2)
70 prop_var1 <- pca_var1/sum(pca_var1)
71 prop_var <- pca_var/sum(pca_var)
72
73 # scree plot
74 plot(prop_var[1:5], xlab="Principal Component",
75       ylab="Proportion of Variance Explained",
76       type="b", col="red")
77 lines(prop_var2[1:5], col="blue", type="b")
78 lines(prop_var1[1:5], col="green", type="b")
79
80 # plotting pca
81 dataset = data.frame(G3 = data[train, "G3"],

```

```

82         pca = pca$x)
83
84 ggplot(dataset) + geom_point(aes(pca.PC1, pca.PC2, colour = G3,
85     shape = G3), size = 2.5) +
86     labs(x = paste("First Component (", percent(prop_var[1]), "%)",
87         sep = "%"),
88         y = paste("Second Component (", percent(prop_var[2]), "%)",
89             sep = "%"))
89
90 # predictive modelling with PCA components
91 train_data <- data.frame(G3 = new_data$G3[train], pca$x)
92 train_data[,1:3]
93 test_data <- predict(pca, newdata = pca.test)
94 test_data <- as.data.frame(test_data)
95 test_data[,1:2]
96
97 #predicting using k-nearest neighbours
98 prediction <- knn(train_data[,2:3], test_data[,1], k=5)
99
100 #evaluating performance
101 confusionMatrix(prediction, new_data$G3[-train])

```

Listing 2: PCA code

```

1 # installing pcaMethods
2 if (!requireNamespace("BiocManager", quietly = TRUE))
3   install.packages("BiocManager")
4 BiocManager::install("pcaMethods")
5
6 #load packages
7 library(pcaMethods)
8 library(tidyverse)
9 library(caret)
10 library(class)
11
12 # read data into r
13 data <- read.csv(file.choose())
14
15 # grouping output variable
16 for (i in 1:395){
17   if (data$G3[i]<5){
18     data$G3[i] = 5
19   }else{
20     if (data$G3[i]<9){
21       data$G3[i] = 4
22     }else{
23       if (data$G3[i]<13){
24         data$G3[i] = 3
25       }else{
26         if (data$G3[i]<17){
27           data$G3[i] = 2
28         }else{
29           if (data$G3[i]<21){
30             data$G3[i] = 1
31           }
32         }
33       }
34     }
35   }
36 }

```

```

35 }
36 }
37
38 data$G3 <- as.factor(data$G3)
39
40 # converting categorical data into numeric data
41 new_data <- dummy.data.frame(data, names = c("school", "sex", "
    address", "famsize", "Pstatus", "Mjob", "Fjob", "reason", "
    guardian", "schoolsup", "famsup", "paid", "activities", "
    nursery", "higher", "internet", "romantic"))
42
43 # dividing new data
44 set.seed(2020)
45 train <- sample(1:395, 0.7*395)
46 rpca.train <- new_data[train, -c(2, 4, 7, 9, 11, 18, 23, 27, 30,
    35, 37, 39, 41, 43, 45, 47, 49, 59)]
47 # ensuring we have k-1 dummy variables to avoid collinearity
48 rpca.test <- new_data[-train, -c(2, 4, 7, 9, 11, 18, 23, 27, 30,
    35, 37, 39, 41, 43, 45, 47, 49, 59)]
49
50
51
52
53 rpca <- pca(rpca.train, nPcs = 10, method = "robustPca", center =
    FALSE) # unstandardized data
54 variation <- rpca@R2[1] + rpca@R2[3]
55
56
57 # plotting unstandardized components
58 require(ggplot2)
59 require(scales)
60 dataset = data.frame(G3 = data[train, "G3"],
    rpca = rpca@scores)
61
62
63 ggplot(dataset) + geom_point(aes(rpca.PC1, rpca.PC3, colour = G3,
    shape = G3), size = 2.5) +
64   labs(x = paste("First Principal Component (", percent(rpca@R2[1])
    , "%)", sep = "%"),
65     y = paste("Third Principal Component (", percent(rpca@R2[3])
    , "%)", sep = "%"))
66
67 # predictive modelling with PCA components
68 train_data <- data.frame(G3 = new_data$G3[train], rpca@scores)
69 train_data <- train_data[, c(1, 2, 4)]
70 test_data <- predict(rpca, newdata = rpca.test)
71 test_data <- as.data.frame(test_data$scores)
72 test_data <- test_data[, c(1, 3)]
73
74 # predicting using k-nearest neighbours
75 prediction <- knn(train_data[, 2:3], test_data, train_data[, 1], k=5)
76
77 # evaluating performance
78 confusionMatrix(prediction, new_data$G3[-train])

```

Listing 3: RPCA code

```

1 #load packages
2 library(pcaMethods)
3 library(class)
4 library(caret)

```

```

5
6 # read data into r
7 data <- read.csv(file.choose())
8
9 # grouping output variable
10 for (i in 1:395){
11   if (data$G3[i]<5){
12     data$G3[i] = 5
13   }else{
14     if (data$G3[i]<9){
15       data$G3[i] = 4
16     }else{
17       if (data$G3[i]<13){
18         data$G3[i] = 3
19       }else{
20         if (data$G3[i]<17){
21           data$G3[i] = 2
22         }else{
23           if (data$G3[i]<21){
24             data$G3[i] = 1
25           }
26         }
27       }
28     }
29   }
30 }
31
32 data$G3 <- as.factor(data$G3)
33
34 # converting categorical data into numeric data
35 new_data <- dummy.data.frame(data, names = c("school", "sex", "
  address", "famsize", "Pstatus","Mjob", "Fjob", "reason", "
  guardian", "schoolsup", "famsup", "paid", "activities", "
  nursery", "higher", "internet", "romantic"))
36
37 # ensuring we have k-1 dummy variables to avoid collinearity
38 new_data <- new_data[, -c(2,4,7,9,11,18,23,27,30,
39   35,37,39,41,43,45,47,49)]
40
41 # creating missing values
42 set.seed(2020)
43 cols <- sample(1:41, 39, replace = T)
44 rows <- sample(1:395, 39)
45 miss <- cbind(rows,cols)
46 new_data[miss] <- NA
47
48
49 # dividing new data
50 set.seed(2020)
51 train <- sample(1:395,0.7*395)
52 ppca.train <- new_data[train,-42]
53 ppca.test <- new_data[-train,-42]
54
55
56 # conducting ppca
57 ppca <- pca(as.matrix(ppca.train), method = "ppca", nPcs = 2) #
  unstandardized data

```

```

58 ppca@R2cum
59
60 # plotting unstandardized components
61 require(ggplot2)
62 require(scales)
63 dataset = data.frame(G3 = data[train,"G3"],
64                      ppca = ppca@scores)
65
66 ggplot(dataset) + geom_point(aes(ppca.PC1, ppca.PC2, colour = G3,
67                                shape = G3), size = 2.5) +
67   labs(x = paste("First Principal Component (", percent(ppca@R2[1])
68     , ")", sep = ""),
69        y = paste("Second Principal Component (", percent(ppca@R2
70     [2]), ")", sep = ""))
71
72 # predictive modelling with PCA components
73 train_data <- data.frame(G3 = new_data$G3[train], ppca@scores)
74 test_data <- predict(ppca, newdata = ppca.test)
75 test_data <- as.data.frame(test_data$scores)
76
77 #predicting using k-nearest neighbours
78 prediction <- knn(train_data[,2:3], test_data, train_data[,1], k=5)
79
80 #evaluating performance
81 confusionMatrix(prediction, new_data$G3[-train])

```

Listing 4: PPCA code

```

1 #load packages
2 library(kernlab)
3 library(dummies)
4
5 # read data into r
6 data <- read.csv(file.choose())
7
8 # grouping output variable
9 for (i in 1:395){
10   if (data$G3[i]<5){
11     data$G3[i] = 5
12   }else{
13     if (data$G3[i]<9){
14       data$G3[i] = 4
15     }else{
16       if (data$G3[i]<13){
17         data$G3[i] = 3
18       }else{
19         if (data$G3[i]<17){
20           data$G3[i] = 2
21         }else{
22           if (data$G3[i]<21){
23             data$G3[i] = 1
24           }
25         }
26       }
27     }
28   }
29 }

```

```

30
31 data$G3 <- as.factor(data$G3)
32
33 # converting categorical data into numeric data
34 new_data <- dummy.data.frame(data, names = c("school", "sex", "
    address", "famsize", "Pstatus", "Mjob", "Fjob", "reason", "
    guardian", "schoolsup", "famsup", "paid", "activities", "
    nursery", "higher", "internet", "romantic"))
35
36 # dividing new data
37 set.seed(2020)
38 train <- sample(1:395, 0.7*395)
39 kpca.train <- new_data[train, -c(2, 4, 7, 9, 11, 18, 23, 27, 30,
40                                35, 37, 39, 41, 43, 45, 47, 49, 59)]
41 # ensuring we have k-1 dummy variables to avoid collinearity
42 kpca.test <- new_data[-train, -c(2, 4, 7, 9, 11, 18, 23, 27, 30,
43                                 35, 37, 39, 41, 43, 45, 47, 49, 59)]
44
45 # kernel principal component analysis
46 kpca <- kpca(as.matrix(kpca.train), kernel = "rbfdot", kpar = list(
47     sigma = c(100, 10, 1, 0.1, 0.001, 0.0001, 0.00001, 0.0000001)))
48 kpca@eig
49
50 # plotting kpca
51 dataset = data.frame(G3 = data[train, "G3"],
52                      kpca = kpca@pcv)
53 require(ggplot2)
54 require(scales)
55
56 prop_var <- kpca@eig/sum(kpca@eig)
57 sum(prop_var[1:2])
58
59 ggplot(dataset) + geom_point(aes(kpca.1, kpca.2, colour = G3, shape
60                                = G3), size = 2.5) +
61   labs(x = paste("First Component (", percent(prop_var[1]), "%)",
62                 sep = "%"),
63        y = paste("Second Component (", percent(prop_var[2]), "%)",
64                 sep = "%"))

```

Listing 5: KPCA code

```

1 #load libraries
2 library(kernlab)
3 library(caret)
4 library(dummies)
5 library(class)
6
7 # read data into r
8 data <- read.csv(file.choose())
9
10 # grouping output variable
11 for (i in 1:395){
12   if (data$G3[i]<5){
13     data$G3[i] = 5
14   }else{
15     if (data$G3[i]<9){
16       data$G3[i] = 4
17     }else{

```



```

18     if (data$G3[i]<13){
19         data$G3[i] = 3
20     }else{
21         if (data$G3[i]<17){
22             data$G3[i] = 2
23         }else{
24             if (data$G3[i]<21){
25                 data$G3[i] = 1
26             }
27         }
28     }
29 }
30 }
31 }
32 data$G3 <- as.factor(data$G3)
33
34 # converting categorical data into numeric data
35 new_data <- dummy.data.frame(data, names = c("school", "sex", "
    address", "famsize", "Pstatus", "Mjob", "Fjob", "reason", "
    guardian", "schoolsup", "famsup", "paid", "activities", "
    nursery", "higher", "internet", "romantic"))
36
37 # dividing new data
38 set.seed(2020)
39 train <- sample(1:395, 0.7*395)
40 kpca.data <- new_data[, -c(2,4,7,9,11,18,23,27,30,
41     35,37,39,41,43,45,47,49,59)]
42 # ensuring we have k-1 dummy variables to avoid collinearity
43
44 # kernel principal component analysis
45 kpca <- kpca(as.matrix(kpca.data), kernel = "rbfdot", kpar = list(
46     sigma = c(100, 10, 1, 0.1, 0.001, 0.0001, 0.00001, 0.0000001)))
47 kpca@eig
48
49 # predictive modelling with PCA components
50 Z <- kpca@rotated[,1:2]
51 train_data <- as.data.frame(Z[train,])
52 test_data <- as.data.frame(Z[-train,])
53
54 #using knn for predicting
55 prediction <- knn(train_data, test_data, new_data$G3[train], k=5)
56
57 #evaluating performance
58 confusionMatrix(prediction, new_data$G3[-train])

```

Listing 6: KPCA predicting code