```
> library(faraway)
> library(psych)
> library(caret)
> library(tidyverse)
> library(pROC)
> library(class)
> library(e1071)
> library(reprex)
> library(DMwR)
> library(vegan)
> #Data Preprocessing
> str(pima)
'data.frame':   768 obs. of  9 variables:
 $ pregnant : int  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose  : int  148 85 183 89 137 116 78 115 197 125 ...
 $ diastolic: int  72 66 64 66 40 74 50 0 70 96 ...
 $ triceps  : int  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin  : int  0 0 0 94 168 0 88 0 543 0 ...
 $ bmi      : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ diabetes : num  0.627 0.351 0.672 0.167 2.288 ...
 $ age      : int  50 31 32 21 33 30 26 29 53 54 ...
 $ test     : int  1 0 1 0 1 0 1 0 1 1 ...
> pima$glucose[pima$glucose == 0] = NA
> pima$diastolic[pima$diastolic == 0] = NA
> pima$triceps[pima$triceps == 0] = NA
> pima$insulin[pima$insulin == 0] = NA
> pima$bmi[pima$bmi == 0] = NA
> pima$diabetes[pima$diabetes == 0] = NA
> pima$age[pima$age == 0] = NA
> sapply(pima, function(x) sum(is.na(x)))
 pregnant   glucose diastolic   triceps   insulin       bmi  diabetes
age      test
        0         5        35       227       374        11         0
0         0
> pima = pima[-manyNAs(pima),]
> sapply(pima, function(x) sum(is.na(x)))
 pregnant   glucose diastolic   triceps   insulin       bmi  diabetes
age      test
        0         1         0         0       140         1         0
0         0
> pima.clean = knnImputation(pima, k=10)
> pima.clean$test = as.factor(pima.clean$test)
> levels(pima.clean$test) = c("No", "Yes")
> table(pima.clean$test)
```

```
 No Yes
357 177
> #Creating the testing and training dataset
> set.seed(123)
> pima.train = sample_frac(tbl = pima.clean, replace=FALSE,
size=0.75)
> pima.test = anti_join(pima.clean, pima.train)
Joining, by = c("pregnant", "glucose", "diastolic", "triceps",
"insulin", "bmi", "diabetes", "age", "test")
> pima.train.lab = pima.train[,9]
> pima.test.lab = pima.test[,9]
> pima.train.data = pima.train[,1:8]
> pima.test.data = pima.test[,1:8]
> #Let us run a BLR first
> BLR = glm(test~., family=binomial, data=pima.train)
> summary(BLR)

Call:
glm(formula = test ~ ., family = binomial, data = pima.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8811  -0.6569  -0.3811   0.6329   2.4577

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.0860085  1.1663236  -7.790 6.69e-15 ***
pregnant     0.1371466  0.0507069   2.705  0.00684 **
glucose      0.0332173  0.0055084   6.030 1.64e-09 ***
diastolic   -0.0083542  0.0118990  -0.702  0.48262
triceps      0.0054582  0.0171733   0.318  0.75061
insulin      0.0004141  0.0013949   0.297  0.76660
bmi          0.0779741  0.0266807   2.922  0.00347 **
diabetes     1.1084110  0.3942004   2.812  0.00493 **
age          0.0241565  0.0167812   1.440  0.15001
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 498.43  on 399  degrees of freedom
Residual deviance: 351.31  on 391  degrees of freedom
AIC: 369.31

Number of Fisher Scoring iterations: 5
```

```
> #Improved BLR
> BLR2 = glm(test~pregnant+glucose+bmi+diabetes, family=binomial,
data=pima.train)
> summary(BLR2)

Call:
glm(formula = test ~ pregnant + glucose + bmi + diabetes, family =
binomial,
    data = pima.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8979  -0.6509  -0.3960   0.6130   2.5165

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.144281   1.005107  -9.098  < 2e-16 ***
pregnant     0.180351   0.039704   4.542 5.56e-06 ***
glucose      0.035315   0.004633   7.622 2.50e-14 ***
bmi          0.078391   0.020840   3.762 0.000169 ***
diabetes     1.190484   0.389996   3.053 0.002269 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 498.43  on 399  degrees of freedom
Residual deviance: 353.94  on 395  degrees of freedom
AIC: 363.94

Number of Fisher Scoring iterations: 5

> AIC(BLR, BLR2)
     df      AIC
BLR   9 369.3126
BLR2  5 363.9405
> #ROC Curve
> test.prob = predict(BLR2, newdata=pima.test, type="response")
> test.roc = roc(pima.test$test~test.prob, plot=TRUE, print.auc =
TRUE)
Setting levels: control = No, case = Yes
Setting direction: controls < cases
> #Normalize the data first
> pima.train.norm = decostand(pima.train.data, "normalize")
> pima.test.norm = decostand(pima.train.data, "normalize")
> #let's try using CV, then train the model
> pima.cv = trainControl(method="repeatedcv", number=10, repeats=6)
> train.knn = train(test~., data=pima.train, method="knn", trControl
= pima.cv)
> train.knn
k-Nearest Neighbors

400 samples
  8 predictor
  2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 359, 359, 360, 361, 360, 359, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.7620672  0.4295923
  7  0.7548562  0.4224442
  9  0.7507312  0.4063231

Accuracy was used to select the optimal model using the largest
value.
The final value used for the model was k = 5.
> #optimal k value = 5 (accuracy of 76.206%)
> pima.knn = knn(pima.train.norm, pima.test.norm, cl=pima.train.lab,
k=5, prob=TRUE)
> summary(pima.knn)
 No Yes
298 102

> confusionMatrix(pima.knn, pima.train.lab)
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  252  46
       Yes  22  80

               Accuracy : 0.83
                 95% CI : (0.7895, 0.8655)
    No Information Rate : 0.685
    P-Value [Acc > NIR] : 3.198e-11
```

```
                  Kappa : 0.5847

 Mcnemar's Test P-Value : 0.005285

            Sensitivity : 0.9197
            Specificity : 0.6349
         Pos Pred Value : 0.8456
         Neg Pred Value : 0.7843
             Prevalence : 0.6850
         Detection Rate : 0.6300
   Detection Prevalence : 0.7450
      Balanced Accuracy : 0.7773

       'Positive' Class : No




> #Now let's try a SVM
> x.pima = subset(pima.train, select=-test)
> y.pima = pima.train.lab
> pima.svm = svm(x.pima, y.pima, kernel="linear")
> summary(pima.svm)

Call:
svm.default(x = x.pima, y = y.pima, kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  192

 ( 97 95 )


Number of Classes:  2

Levels:
 No Yes
```

```
> pima.svm.pred = predict(pima.svm, x.pima)
> pima.tab = table(pima.svm.pred, y.pima)
> prop.table(pima.tab)
             y.pima
pima.svm.pred    No   Yes
          No  0.625 0.140
          Yes 0.060 0.175
> mean(pima.svm.pred == y.pima)
[1] 0.8
> #Let's use a CV (tune) to optimize the model
> pima.svm.tune <- tune(svm, x.pima, y.pima, kernel = "radial",
ranges = list(cost = 1^(-1:2), gamma = c(.5,1,2)))
> summary(pima.svm.tune)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.5

- best performance: 0.2275

- Detailed performance results:
   cost gamma  error dispersion
1     1   0.5 0.2275 0.06061032
2     1   0.5 0.2275 0.06061032
3     1   0.5 0.2275 0.06061032
4     1   0.5 0.2275 0.06061032
5     1   1.0 0.2975 0.05706965
6     1   1.0 0.2975 0.05706965
7     1   1.0 0.2975 0.05706965
8     1   1.0 0.2975 0.05706965
9     1   2.0 0.3175 0.05779514
10    1   2.0 0.3175 0.05779514
11    1   2.0 0.3175 0.05779514
12    1   2.0 0.3175 0.05779514

> pima.svm.2 = svm(x.pima, y.pima, kernel="radial",cost=1, gamma=0.5)
> summary(pima.svm.2)

Call:
svm.default(x = x.pima, y = y.pima, kernel = "radial", gamma = 0.5,
cost = 1)
```

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  297

 ( 171 126 )


Number of Classes:  2

Levels:
 No Yes

> pima.svm.pred2 = predict(pima.svm.2, x.pima)
> pima.svm.table = table(pima.svm.pred2, y.pima)
> prop.table(pima.svm.table)
              y.pima
pima.svm.pred2    No   Yes
           No  0.680 0.055
           Yes 0.005 0.260
> mean(pima.svm.pred2 == y.pima)
[1] 0.94
```