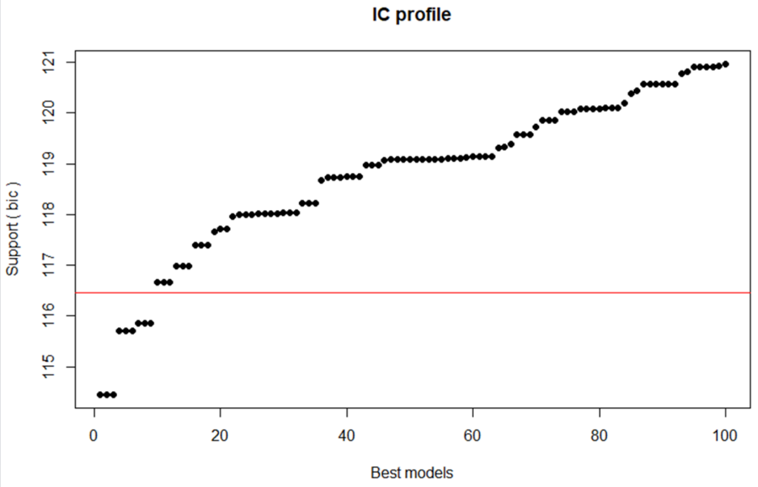
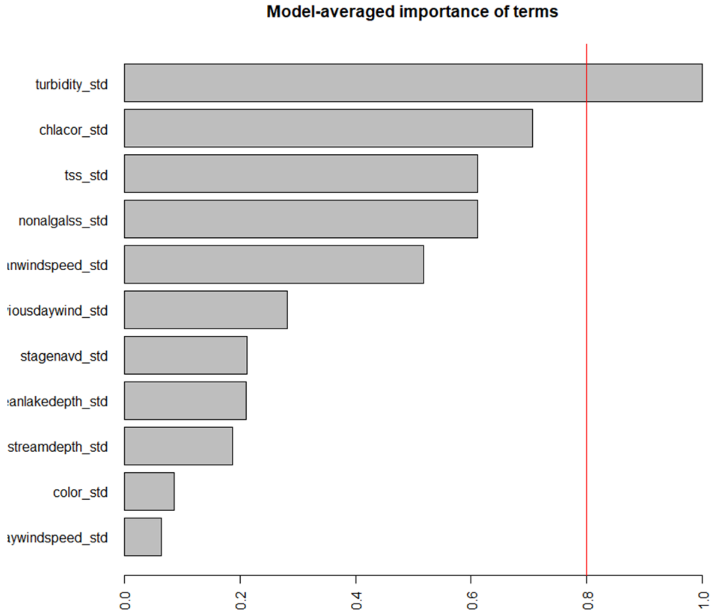
Advanced R:
Statistical Machine Learning

Dan Schmutz, MS
Chief Environmental Scientist

Zoom Workshop for SJRWMD
September 24, 2020

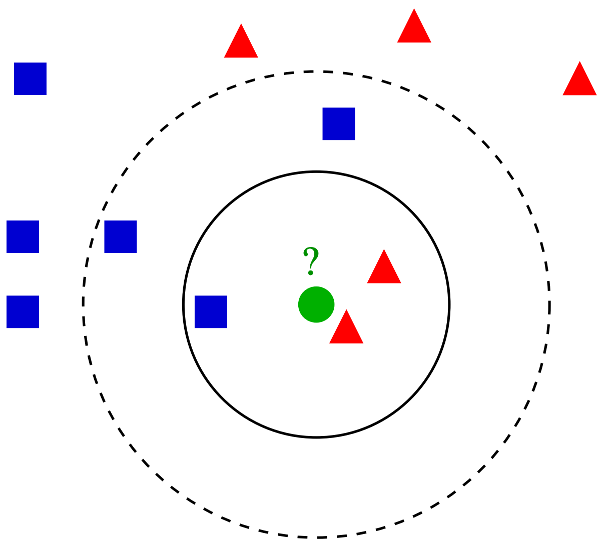




k-nearest neighbors

Birds of a feather flock together (in data space)

A Euclidian distance is calculated between observations (here example is just x and y space but could be many variables) and the average is assigned (regression) or majority vote (classification).



When using KNN for classification, it is best to assess odd numbers for k to avoid ties in the event there is equal proportion of response levels

By Antti Ajanki AnAj - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=2170282

GPI

Creating analysis blueprint to process train and test with one-hot encoding

```
> # Create blueprint
> blueprint <- recipe(Survived2 ~ ., data = ttrain4) %>%
+   step_nzv(all_nominal()) %>% # remove sparse variables
+   step_dummy(all_nominal(), -all_outcomes(), one_hot = TRUE) %>%
+   step_center(all_numeric(), -all_outcomes()) %>%
+   step_scale(all_numeric(), -all_outcomes())
> blueprint
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor          7

Operations:

Sparse, unbalanced variable filter on all_nominal()
Dummy variables from all_nominal(), -all_outcomes()
Centering for all_numeric(), -all_outcomes()
Scaling for all_numeric(), -all_outcomes()
> 
```

GPI

Creating a crossvalidation scheme using caret trainControl function

```
# Create a resampling method
cv <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)

# Create a hyperparameter grid search
hyper_grid <- expand.grid(
  k = floor(seq(1, nrow(ttrain4)/3, length.out = 20))
)

# creating a tighter grid
hyper_grid <- expand.grid(
  k = floor(seq(1, nrow(ttrain4)/20, length.out = 20))
)
```

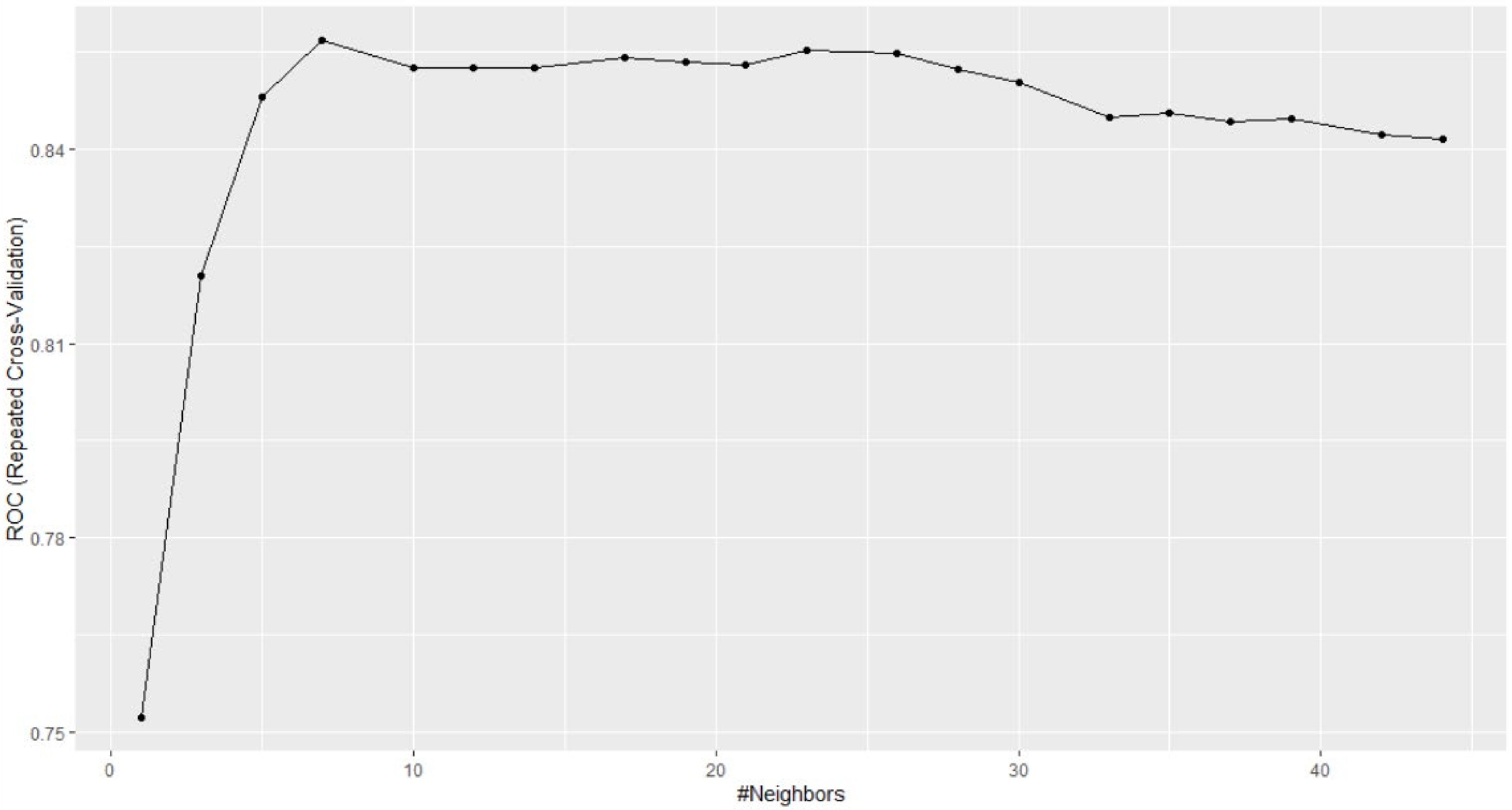
GPI

tuning the number of neighbors

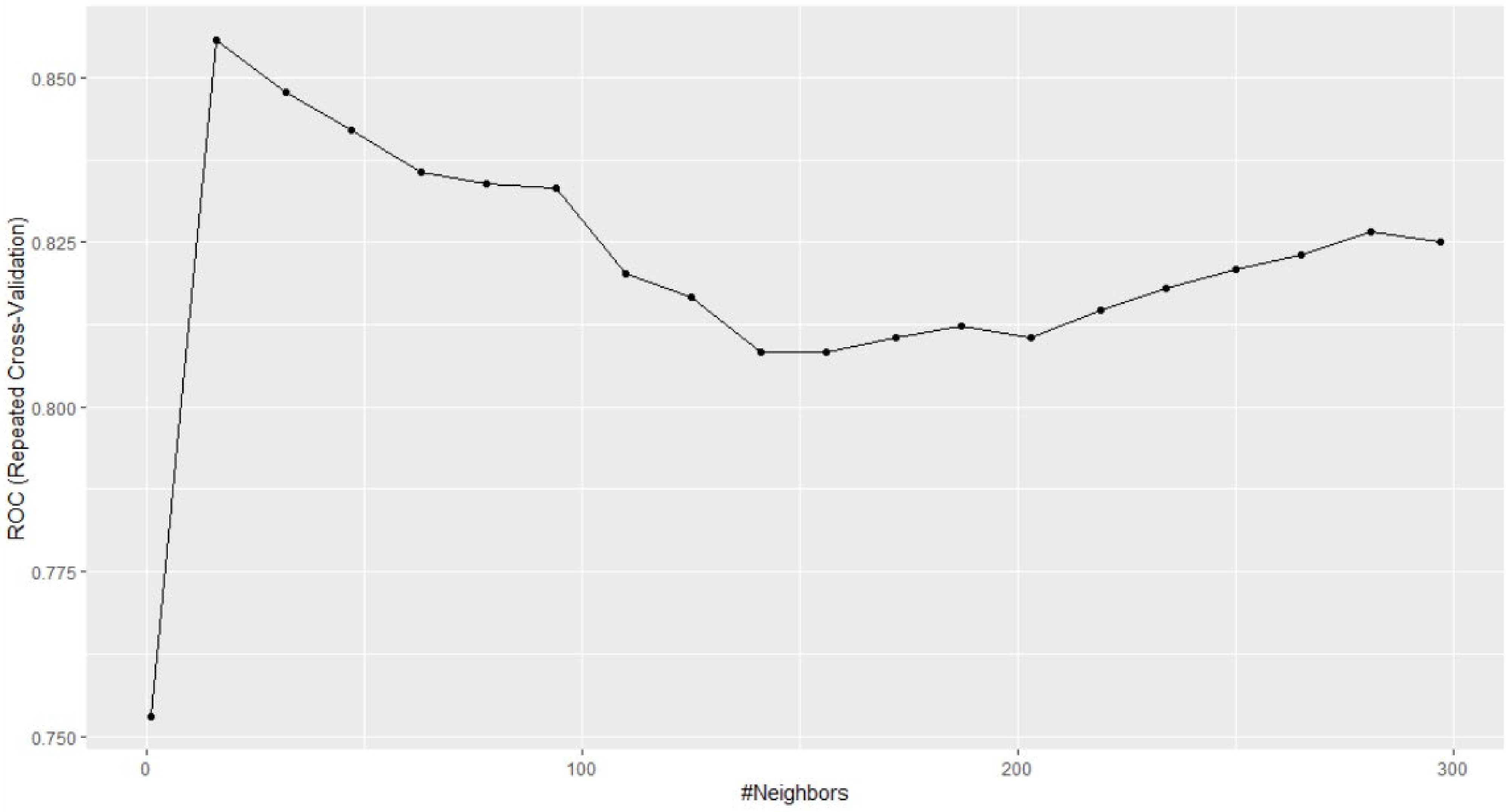
```
# Fit knn model and perform grid search
knn_grid <- train(
  blueprint,
  data = ttrain4knn,
  method = "knn",
  trControl = cv,
  tuneGrid = hyper_grid,
  metric = "ROC"
)

ggplot(knn_grid)
print(knn_grid)
```

GPI

Initial grid search



Tighter grid



- Optimal k=7

```
> print(knn_grid)
k-Nearest Neighbors

891 samples
  7 predictor
  2 classes: 'X0', 'X1'

Recipe steps: nzv, dummy, center, scale
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 802, 802, 802, 802, 802, 801, ...
Resampling results across tuning parameters:

  k   ROC        Sens       Spec
   1  0.7522112  0.7998788  0.6972605
   3  0.8203702  0.8534949  0.7066050
   5  0.8481334  0.8669966  0.7096134
   7  0.8568601  0.8794007  0.6960840
  10  0.8526195  0.8885253  0.6545546
  12  0.8525994  0.8932795  0.6304874
  14  0.8525433  0.9063906  0.6141513
  17  0.8541171  0.9092727  0.6141345
  19  0.8534421  0.9100269  0.6065546
  21  0.8530624  0.9140471  0.6053445
  23  0.8552967  0.9216700  0.6012269
  26  0.8549390  0.9264108  0.5983193
  28  0.8524725  0.9318923  0.5861008
  30  0.8504276  0.9333535  0.5667899
  33  0.8449102  0.9322424  0.5516303
  35  0.8456760  0.9348013  0.5545546
  37  0.8443148  0.9340673  0.5492605
  39  0.8447129  0.9384242  0.5434118
  42  0.8422789  0.9380875  0.5463697
  44  0.8416627  0.9395354  0.5451597

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 7.
>
```

GPI

Preparing datasets for evaluation using knn=7

- Baking in the variable transformations
- Scaling very important for knn to avoid different scales having undue influence on proximity

```
> # baking to make prepared train and test
> trained_rec <- prep(blueprint, training = ttrain4)
> train_tknn <- bake(trained_rec, new_data = ttrain4)
> test_tknn  <- bake(trained_rec, new_data = ttest4)
> str(train_tknn)
tibble [891 x 13] (S3: tbl_df/tbl/data.frame)
 $ Age          : num [1:891] -0.565 0.663 -0.258 0.433 0.433 ...
 $ SibSp        : num [1:891] 0.433 0.433 -0.474 0.433 -0.474 ...
 $ Parch        : num [1:891] -0.473 -0.473 -0.473 -0.473 -0.473 ...
 $ Fare         : num [1:891] -0.502 0.786 -0.489 0.42 -0.486 ...
 $ Survived2    : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
 $ Embarked_C   : num [1:891] -0.482 2.073 -0.482 -0.482 -0.482 ...
 $ Embarked_Q   : num [1:891] -0.307 -0.307 -0.307 -0.307 -0.307 ...
 $ Embarked_S   : num [1:891] 0.615 -1.623 0.615 0.615 0.615 ...
 $ Gender_female: num [1:891] -0.737 1.355 1.355 1.355 -0.737 ...
 $ Gender_male  : num [1:891] 0.737 -1.355 -1.355 -1.355 0.737 ...
 $ Pclass2_1    : num [1:891] 0.902 -1.107 0.902 -1.107 0.902 ...
 $ Pclass2_2    : num [1:891] -0.51 -0.51 -0.51 -0.51 -0.51 ...
 $ Pclass2_3    : num [1:891] -0.565 1.767 -0.565 1.767 -0.565 ...
> 
```

GPI

knn=7 does well on the training set

train_tknn_mod<-knn(train = train_tknn, test = train_tknn,cl = train_tknn$Survived2, k=7)

- 96% accurate on train!

```
> confusionMatrix(train_tknn_mod,train_tknn$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 530  16
         1  19 326

               Accuracy : 0.9607
                 95% CI : (0.9458, 0.9725)
    No Information Rate : 0.6162
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9171

 Mcnemar's Test P-Value : 0.7353

            Sensitivity : 0.9654
            Specificity : 0.9532
         Pos Pred Value : 0.9707
         Neg Pred Value : 0.9449
             Prevalence : 0.6162
         Detection Rate : 0.5948
   Detection Prevalence : 0.6128
      Balanced Accuracy : 0.9593

       'Positive' Class : 0
```

knn=7 does great on the test set too!

```
test_tknn_mod<-knn(train = train_tknn, test = test_tknn,cl = train_tknn$Survived2, k=7)
```

- 92% accurate on test!

Note there’s no real “model”! The result for test is a vote of the 7 statistically closest train neighbors.

```
> confusionMatrix(test_tknn_mod,test_tknn$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 229  11
         1  21 134

               Accuracy : 0.919
                 95% CI : (0.8876, 0.9439)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8281

 Mcnemar's Test P-Value : 0.1116

            Sensitivity : 0.9160
            Specificity : 0.9241
         Pos Pred Value : 0.9542
         Neg Pred Value : 0.8645
             Prevalence : 0.6329
         Detection Rate : 0.5797
   Detection Prevalence : 0.6076
      Balanced Accuracy : 0.9201

       'Positive' Class : 0
```

GPI