ADS 503 - Final Project

Jimmy Nguyen, Sarah Alqaysi, Sai Thiha

Contents

Data Set

Table 1: Water Potability Data Set (continued below)

ph	Hardness	Solids	Chloramines	Sulfate	Conductivity
NA	204.9	20791	7.3	368.5	564.3
3.716	129.4	18630	6.635	NA	592.9
8.099	224.2	19910	9.276	NA	418.6
8.317	214.4	22018	8.059	356.9	363.3
9.092	181.1	17979	6.547	310.1	398.4
5.584	188.3	28749	7.545	326.7	280.5

Organic_carbon	Trihalomethanes	Turbidity	Potability
10.38	86.99	2.963	no
15.18	56.33	4.501	no
16.87	66.42	3.056	no
18.44	100.3	4.629	no
11.56	32	4.075	no
8.4	54.92	2.56	no

Data Set Total Number of Water Samples

Table 3: Total Number of Water Samples

Total
3276

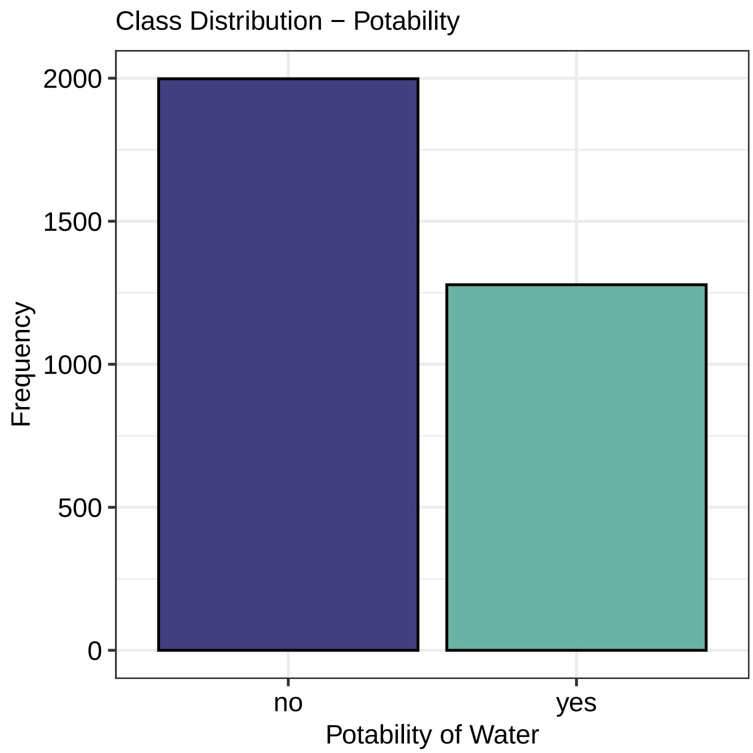
Data Set Total Number of Predictors

Table 4: Total Number of Water Characteristics

Total
9

Data Exploration

Class Distributions - Potability



Class Proportions

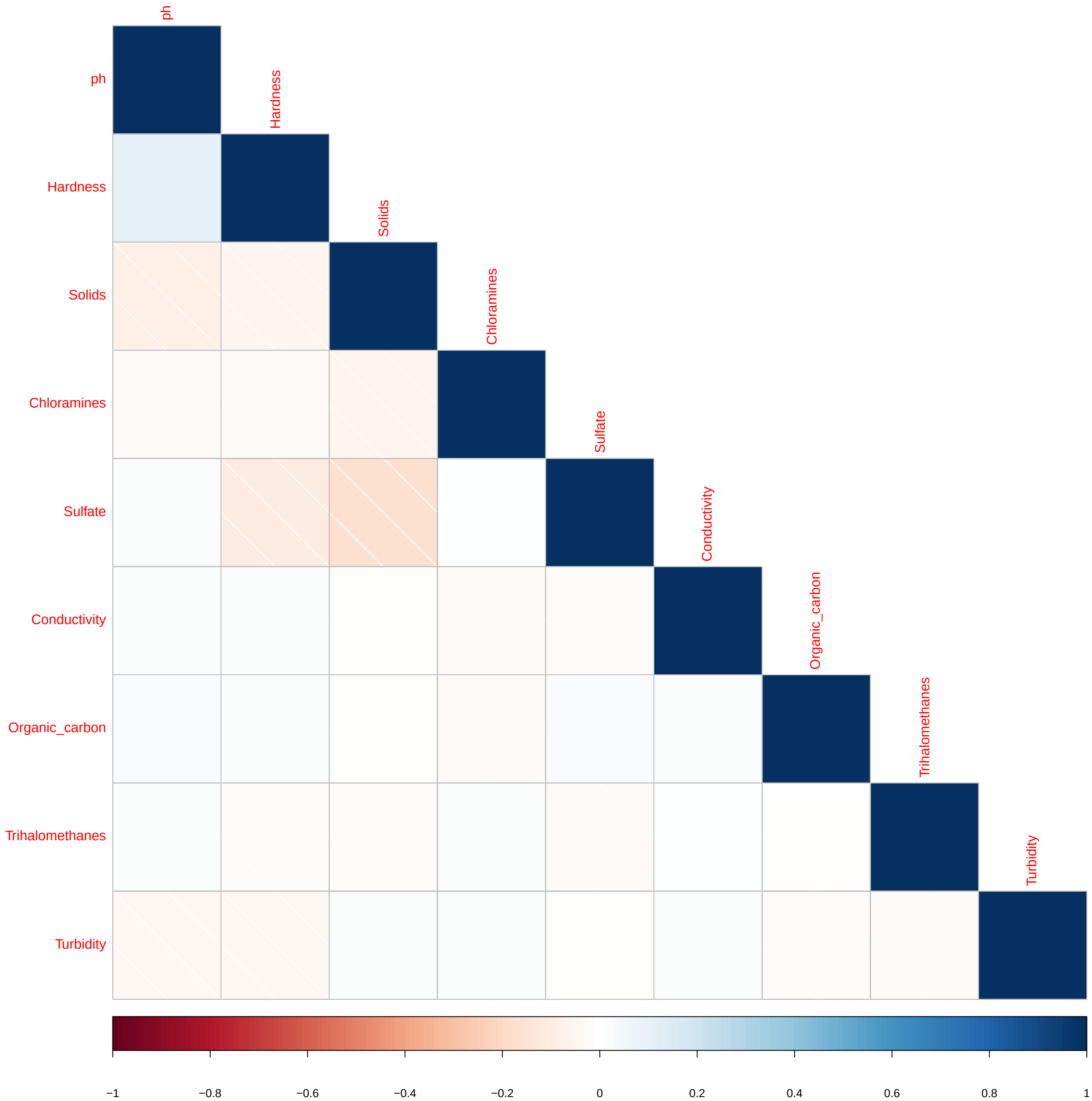
```
pota<- table(df$Potability)
round(prop.table(pota), digits = 3) %>% pander(style = "grid",
          caption = "Class Proportions")
```

Table 5: Class Proportions

no	yes
0.61	0.39

Findings - The potability class (1) contain 39% of the data set, while the non-potability class (0) contain 61% of the data.

Correlation Matrix of Predictors



Degree of correlation:

- Perfect: If the value is near ± 1, then it said to be a perfect correlation: as one variable increases, the other variable tends to also increase (if positive) or decrease (if negative).
- High degree: If the coefficient value lies between ± 0.50 and ± 1, then it is said to be a strong correlation.
- Moderate degree: If the value lies between ± 0.30 and ± 0.49, then it is said to be a medium correlation.
- Low degree: When the value lies below + .29, then it is said to be a small correlation.
- No correlation: When the value is zero.

Frequency Distribution of Predictors:

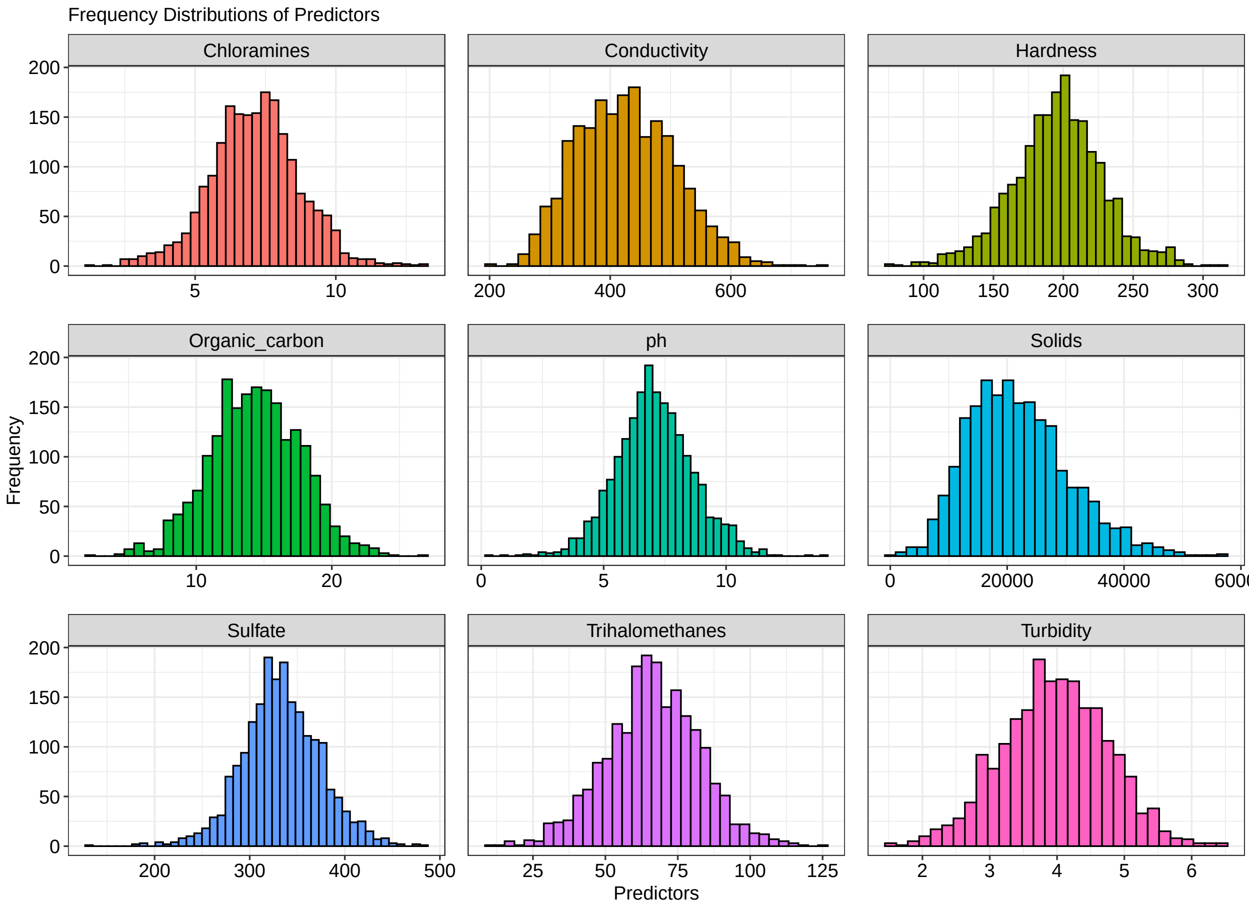


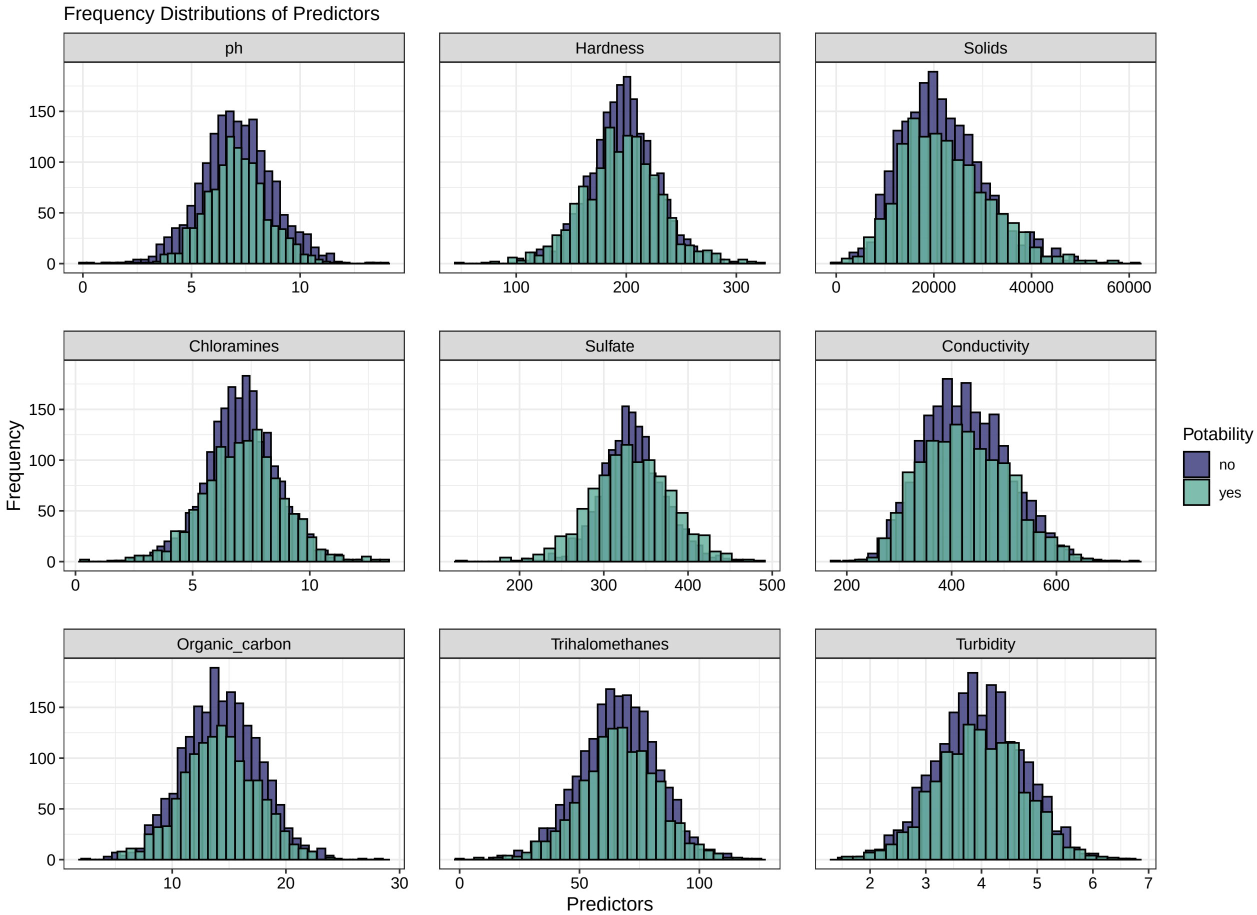
Table 6: Table continues below

ph	Hardness	Solids	Chloramines	Sulfate	Conductivity
0.04887	-0.08511	0.595	0.01296	-0.04649	0.2665

Organic_carbon	Trihalomethanes	Turbidity
-0.01999	-0.05135	-0.033

The rule of thumb seems to be: If the skewness is between -0.5 and 0.5, the data are fairly symmetrical. If the skewness is between -1 and – 0.5 or between 0.5 and 1, the data are moderately skewed. If the skewness is less than -1 or greater than 1, the data are highly skewed.

Frequency Distribution of Predictors with Response Overlaid:



Data Pre-processing

Zero-Variance Predictors :

```
nearZeroVar(predictors[complete.cases(predictors),])
```

```
## integer(0)
```

There are no predictors with degenerate distributions.

Remove Highly Correlated Predictors :

```
correlations <- cor(predictors[complete.cases(predictors),])
highCorr <- findCorrelation(correlations, cutoff = .75)
length(highCorr)
```

```
## [1] 0
```

There are no predictors with high collinearity with each other using a cut-off point of 0.75.

Check for Missing Values :

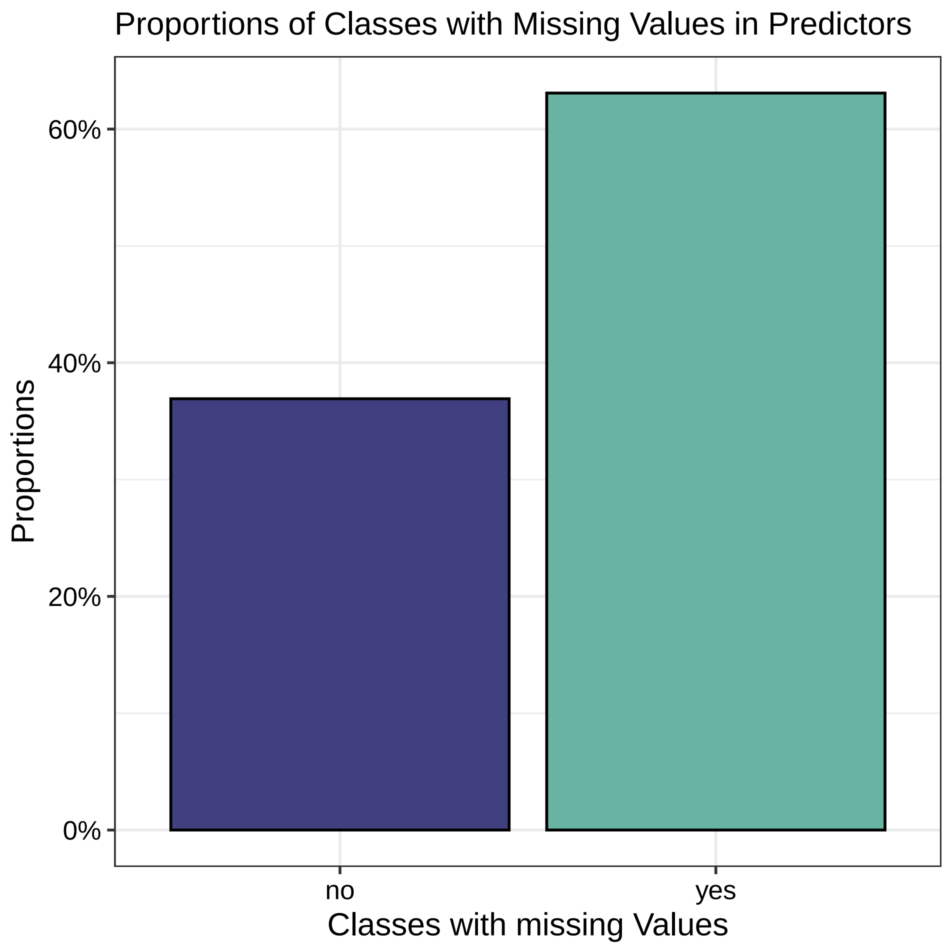


Table 8: Missing Values by Columns

Columns	Total.Missing.Values
Sulfate	781
ph	491
Trihalomethanes	162
Hardness	0
Solids	0
Chloramines	0
Conductivity	0
Organic_carbon	0
Turbidity	0
Potability	0

Table 9: Total Missing Values

Total
1434

Strategies to deal with Missing Values:

1. Replace the missing value with some constant pre-specified.
2. Replace the missing value with the mean/median of the predictor.
3. Replace the missing values with a value generated at random from the observed distribution of each predictor.
4. **The best method:** replace the missing values with imputated values based on the other characteristics of the record. Thus, a K-nearest neighbors and choosing the optimal number of neighbors as the tuning parameter.

However, imputation will not guarantee a better signal in the modeling process, since such techniques has uncertainty and bias. Also, this data set has a lot of missing values, this means that nearly every predictor would need to go through this imputation modeling technique. This is mainly because *class 1 (Potability)* is associated with high rates of missing values.

K-NN Imputation on Missing Values:

```
library(RANN)
impute <- preProcess(as.matrix(predictors), method = c("center", "scale", "knnImpute"))
predictors <- predict(impute, predictors)
```

Table 10: Imputated Predictors - Total Missing Values

Total
0

Data Splitting

```
# Stratified Random Sampling
set.seed(1)
trainingRows <- createDataPartition(response$Potability, p = .80, list = FALSE)

# training set
train <- predictors[trainingRows,]
train_class <- response[trainingRows,]

# test set
test <- predictors[-trainingRows,]
test_class <- response[-trainingRows,]

#resampling method
ctrl <- trainControl(summaryFunction = twoClassSummary,
                     classProbs = TRUE, savePredictions = TRUE,
                     method = "repeatedcv", repeats = 5)
```

Verify Data Partitions

Table 11: Data Split Proportions

Partitions	Proportions
Train Split	0.8
Test Split	0.2

Modeling

Get Model information

```
get_model_info <- function (model, set, set_class) {
  model_pred <-predict(model, set, type = "prob")

  model_df <- data.frame(pred = predict(model, set),
                         obs = set_class,
                         "yes"= model_pred[,"yes"],
                         "no" = model_pred[,"no"])
  return (model_df)}
```

Get ROC curve

```
get_roc <- function(model_df) {

  model_roc <- roc(response = model_df$obs,
                   predictor = model_df$yes,
                   levels = rev(levels(model_df$obs)))
  return (model_roc)}
```

Get Performance Metrics (AUC, Sensitivity, Specificity)

```
get_auc <- function(model_roc, model_df, set_class) {

  model_auc <- auc(model_roc)
  # yes will be used as the event of interests
  model_sens <- sensitivity(data = model_df$pred,
                            reference = set_class,
                            positive = "yes")

  model_spec <- specificity(data = model_df$pred,
                            reference = set_class,
                            negative = "no")


  metrics <- c("Area Under Curve", "Sensitivity", "Specificity")
  performance <- c(model_auc, model_sens, model_spec)

  model_results <- data.frame("Performance" = metrics, model = performance)
  return (model_results)}
```

Linear Discriminant Analysis

```
set.seed(476)
water_lda <- train(train,
                y = train_class,
                method = "lda",
                metric = "ROC",
                preProc = c("center", "scale"),
                trControl = ctrl,
                trace =  FALSE)


saveRDS(water_lda, "water_lda.rds")
```

```
water_lda <- readRDS("water_lda.rds")
lda_df <- get_model_info(water_lda, train, train_class)
lda_roc <- get_roc(lda_df)
lda_results <- get_auc(lda_roc, lda_df, train_class)
```

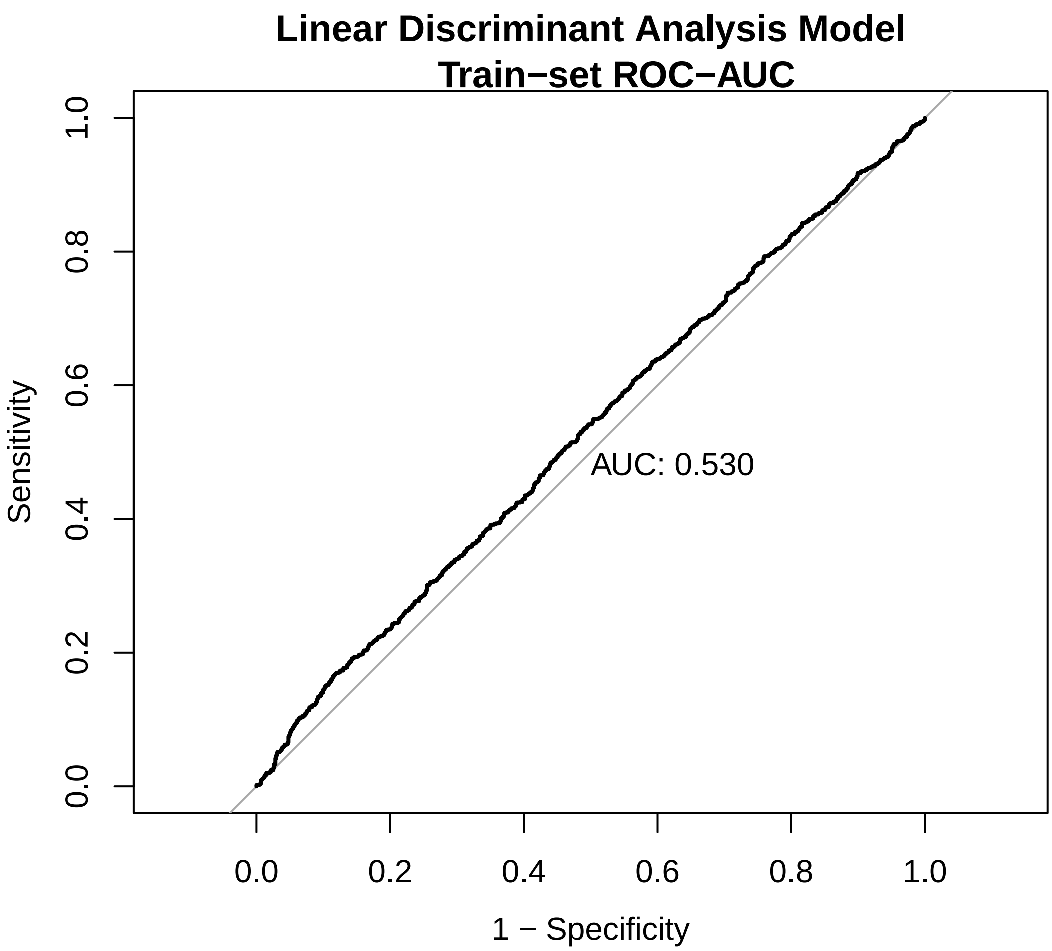


Table 12: LDA Model - Confusion Matrix

	no	yes
no	1599	0
yes	1023	0

Table 13: LDA Model - Training Results

Performance	model
Area Under Curve	0.5297
Sensitivity	0
Specificity	1

Mixed Discriminant Analysis Model

```
set.seed(476)
water_mda <- train(x = train,
                   y = train_class,
                   method = "mda",
                   metric = "ROC",
                   tuneGrid = expand.grid(.subclasses = 1:8),
                   trControl = ctrl)

saveRDS(water_mda, "water_mda.rds")
```

```
water_mda <- readRDS("water_mda.rds")
mda_df <- get_model_info(water_mda, train, train_class)
mda_roc <- get_roc(mda_df)
mda_results <- get_auc(mda_roc, mda_df, train_class)
```

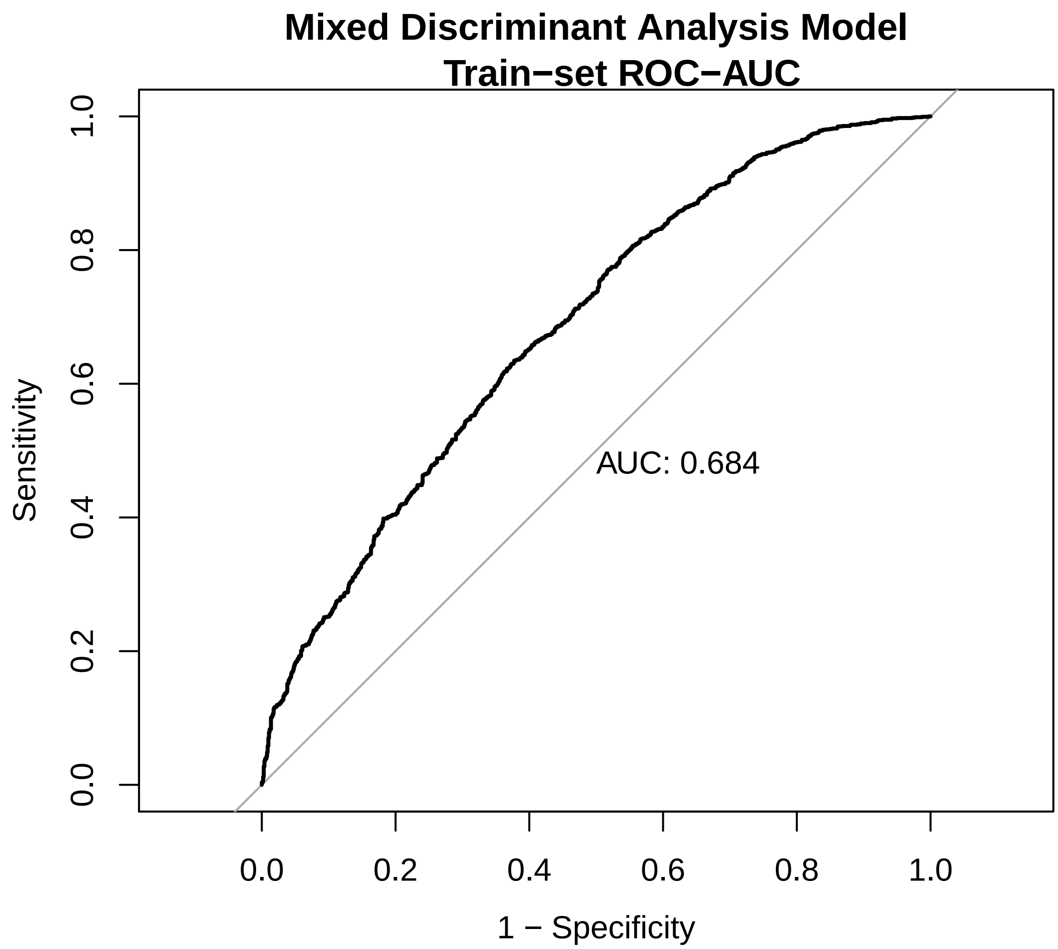


Table 14: MDA Model - Confusion Matrix

	no	yes
no	1441	158
yes	711	312

Table 15: Mixed Determinant Analysis Model - Training Results

Performance	model
Area Under Curve	0.6841
Sensitivity	0.305
Specificity	0.9012

Neural Networks

```
nnetGrid <- expand.grid(.size = 1:10,
                        .decay = c(0, .1, 1, 2))
maxSize <- max(nnetGrid$.size)
numWts <- 1*(maxSize * (length(train) + 1) + maxSize + 1)


water_nnet <- train(train, train_class, method = "nnet",metric = "ROC",
                  preProc = c("center", "scale", "spatialSign"),
                  tuneGrid = nnetGrid, trace = FALSE,
                  maxit = 2000, MaxNWts = numWts, trControl = ctrl)

saveRDS(water_nnet, "water_nnet.rds")
```

```
water_nnet <- readRDS("water_nnet.rds")
nnet_df <- get_model_info(water_nnet, train, train_class)
nnet_roc <- get_roc(nnet_df)
nnet_results <- get_auc(nnet_roc, nnet_df, train_class)
```

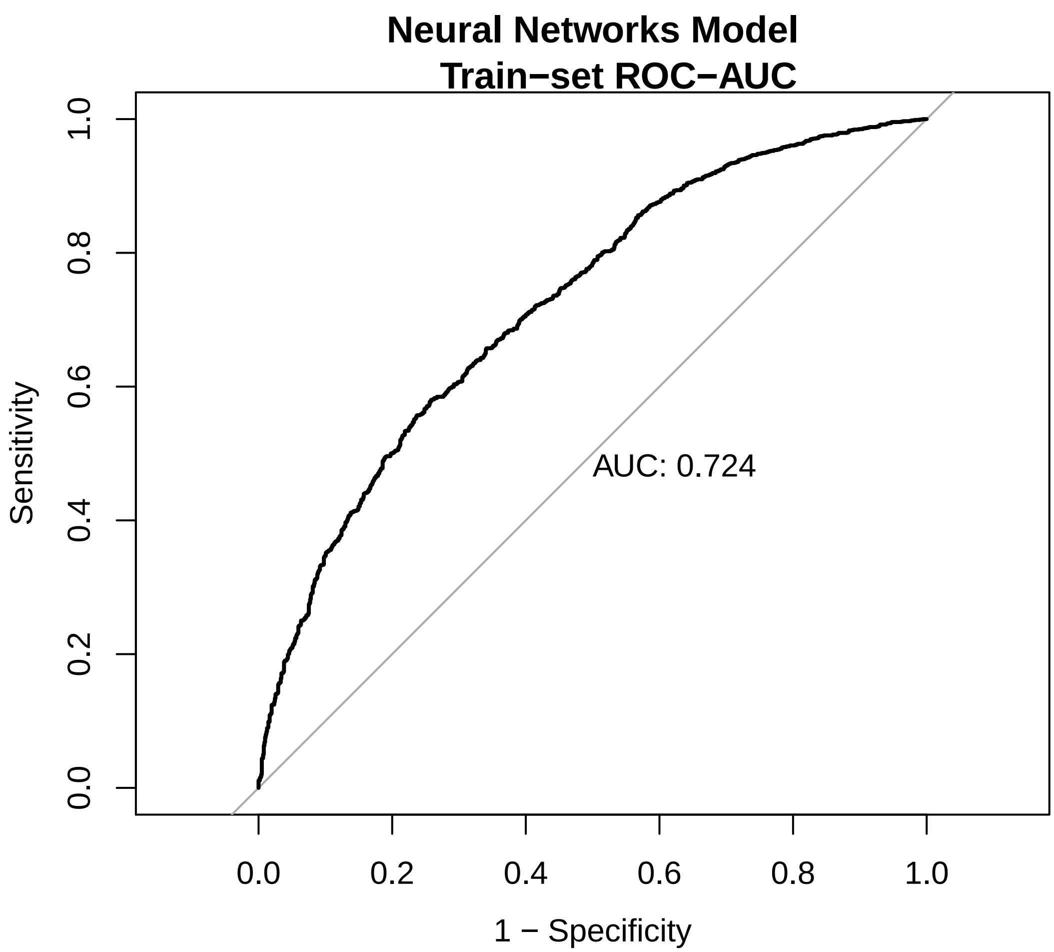


Table 16: Neural Networks Model - Confusion Matrix

	no	yes
no	1378	221
yes	591	432

Table 17: Neural Networks Model - Training Results

Performance	model
Area Under Curve	0.7241
Sensitivity	0.4223
Specificity	0.8618

K-NN

```
set.seed(476)
water_knn <- train(x = train,
                   y = train_class,
                   method = "knn",
                   metric = "ROC",
                   tuneLength = 10,
                   preProc = c("center", "scale"),
                   trControl = ctrl)

saveRDS(water_knn, "water_knn.rds")
```

```
water_knn <- readRDS("water_knn.rds")
knn_df <- get_model_info(water_knn, train, train_class)
knn_roc <- get_roc(knn_df)
knn_results <- get_auc(knn_roc, knn_df, train_class)
```

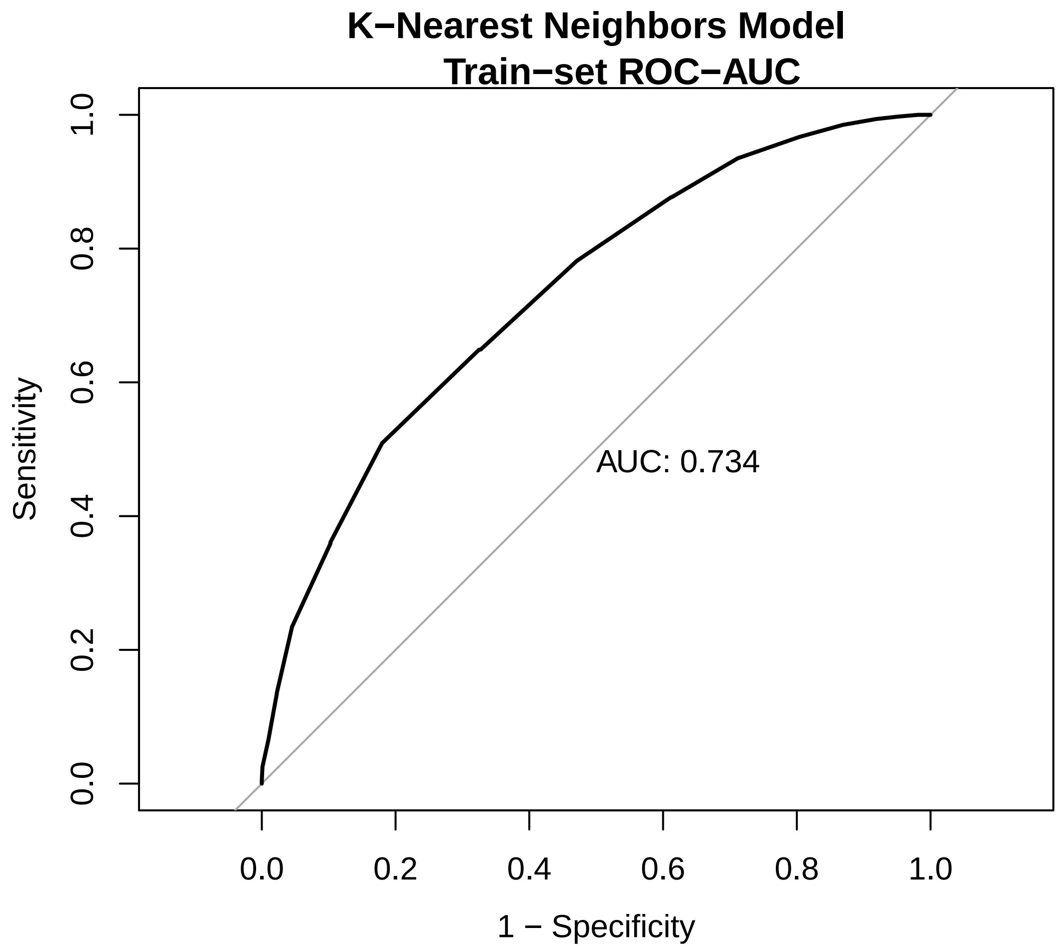


Table 18: KNN Model - Confusion Matrix

	no	yes
no	1495	104
yes	728	295

Table 19: K-Nearest Neighbors Model - Training Results

Performance	model
Area Under Curve	0.7335
Sensitivity	0.2884
Specificity	0.935

Naive-Bayes Model

```r
set.seed(476)
water_nb <- train(x = train,
                  y = train_class,
                  method = "nb",
                  preProc = c("center", "scale"),
                  metric = "ROC",
                  trControl = ctrl)

saveRDS(water_nb, "water_nb.rds")
```

```r
water_nb <- readRDS("water_nb.rds")
nb_df <- get_model_info(water_nb, train, train_class)
nb_roc <- get_roc(nb_df)
nb_results <- get_auc(nb_roc, nb_df, train_class)
```

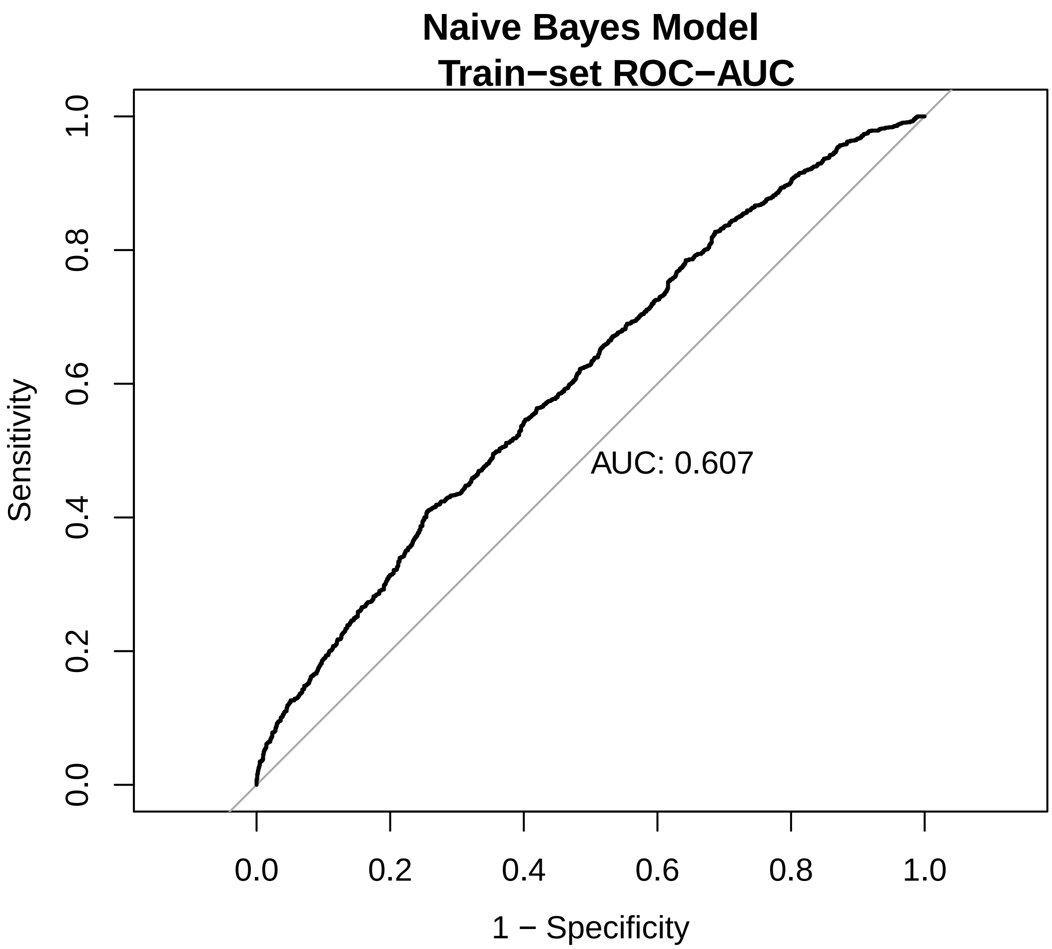


Table 20: Naive Bayes Model - Confusion Matrix

	no	yes
no	1402	197
yes	784	239

Table 21: Naive Bayes Model - Training Results

Performance	model
Area Under Curve	0.6072
Sensitivity	0.2336
Specificity	0.8768

SVM-Radial Function

```
set.seed(476)

sigmaRangeReduced <- sigest(as.matrix(train))
svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced[1], .C = 2^(seq(-4, 4)))

water_svm <- train(train, train_class, method = "svmRadial",
                   metric = "ROC",
                   preProc = c("center", "scale"),
                   tuneGrid = svmRGridReduced,
                   fit = FALSE,
                   trControl = ctrl)

saveRDS(water_svm, "water_svm.rds")
```

```
water_svm <- readRDS("water_svm.rds")
svm_df <- get_model_info(water_svm, train, train_class)
svm_roc <- get_roc(svm_df)
svm_results <- get_auc(svm_roc, svm_df, train_class)
```

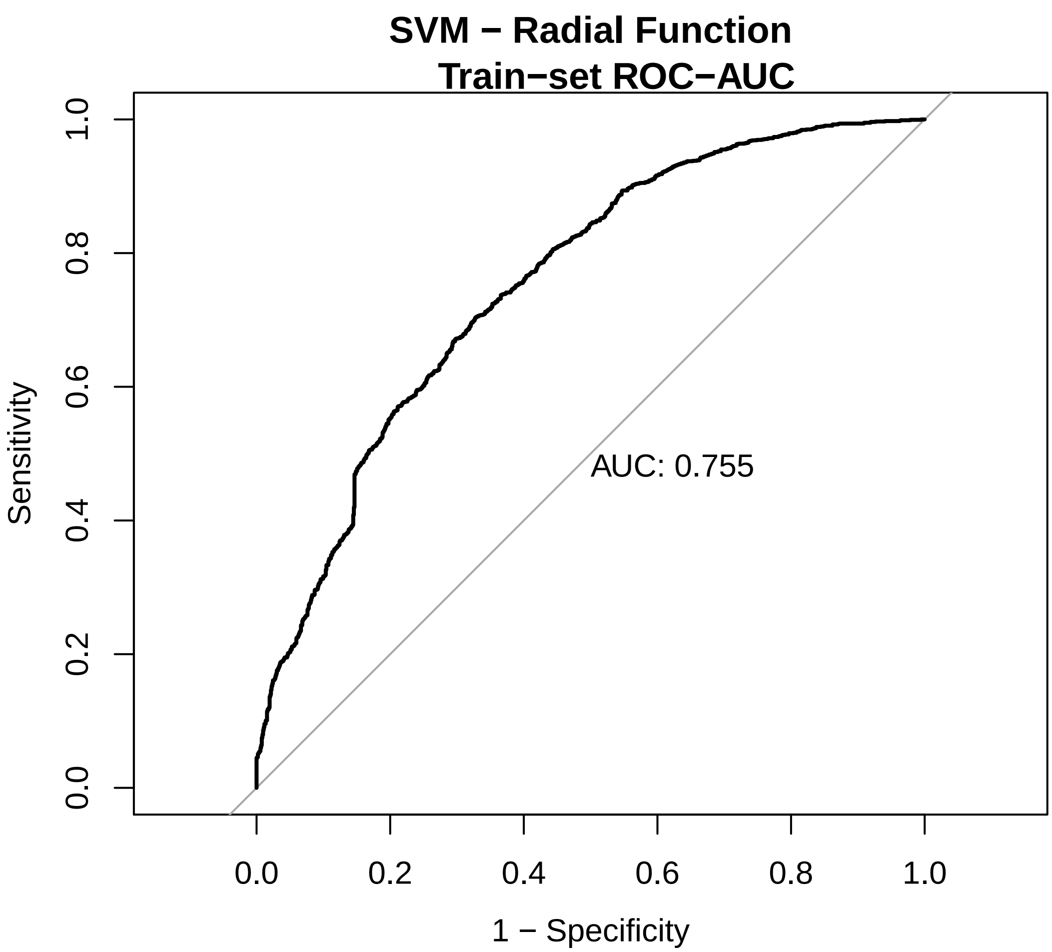


Table 22: SVM Radial Function - Confusion Matrix

	no	yes
no	1486	113
yes	640	383

Table 23: SVM Radial Function - Training Results

Performance	model
Area Under Curve	0.7545
Sensitivity	0.3744
Specificity	0.9293

PLS Model

```r
set.seed(476)

water_pls <- train(x = train, train_class,
                  method = "pls",
                  metric = "ROC",
                  preProc = c("center", "scale"),
                  tuneLength = 15,
                  trControl = ctrl)
saveRDS(water_pls, "water_pls.rds")
```

```r
water_pls <- readRDS("water_pls.rds")
pls_df <- get_model_info(water_pls, train, train_class)
pls_roc <- get_roc(pls_df)
pls_results <- get_auc(pls_roc, pls_df, train_class)
```

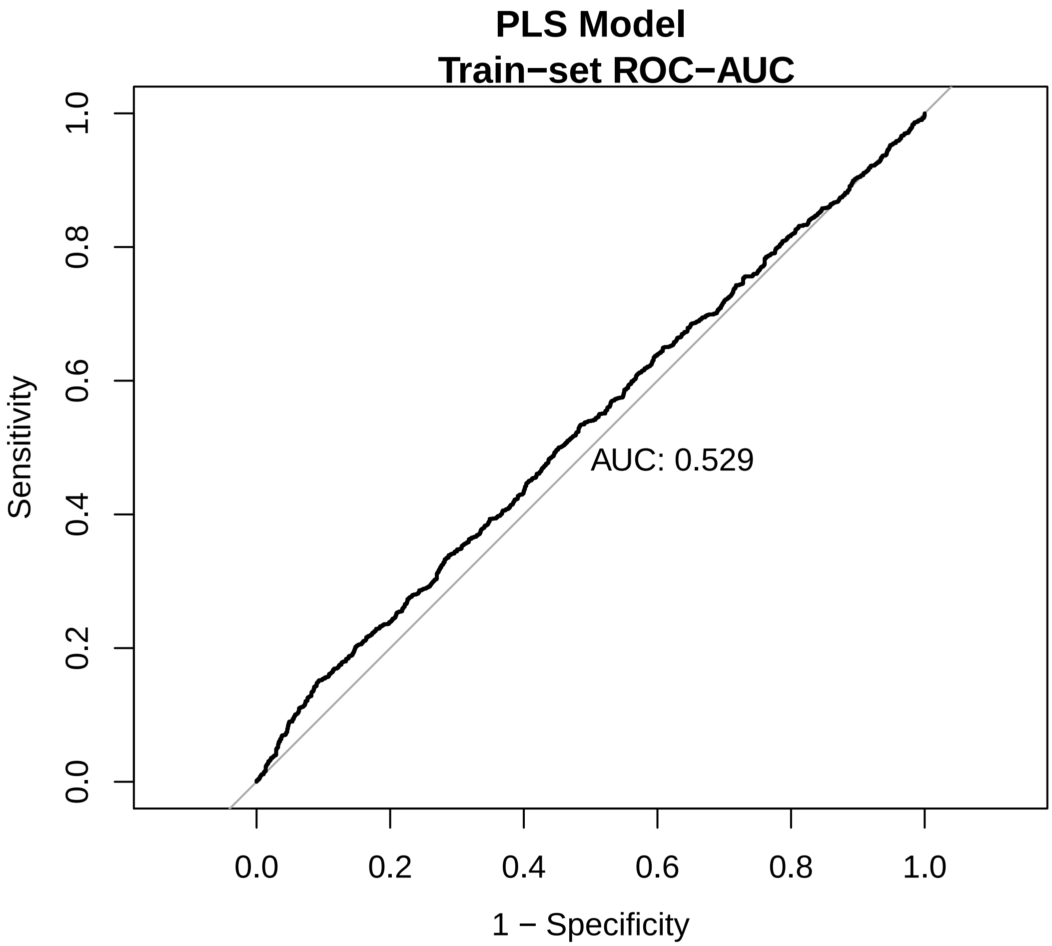


Table 24: PLS Model - Confusion Matrix

	no	yes
no	1599	0
yes	1023	0

Table 25: PLS Model - Training Results

Performance	model
Area Under Curve	0.5292
Sensitivity	0
Specificity	1

MARS

```r
set.seed(476)

water_mars <- train(x = train, train_class,
                    method = "earth",
                    metric = "ROC",
                    tuneGrid = expand.grid(.degree = 1,
                                           .nprune = 2:25),
                    trControl = ctrl)
saveRDS(water_mars, "water_mars.rds")
```

```r
water_mars <- readRDS("water_mars.rds")
mars_df <- get_model_info(water_mars, train, train_class)
mars_roc <- get_roc(mars_df)
mars_results <- get_auc(mars_roc, mars_df, train_class)
```

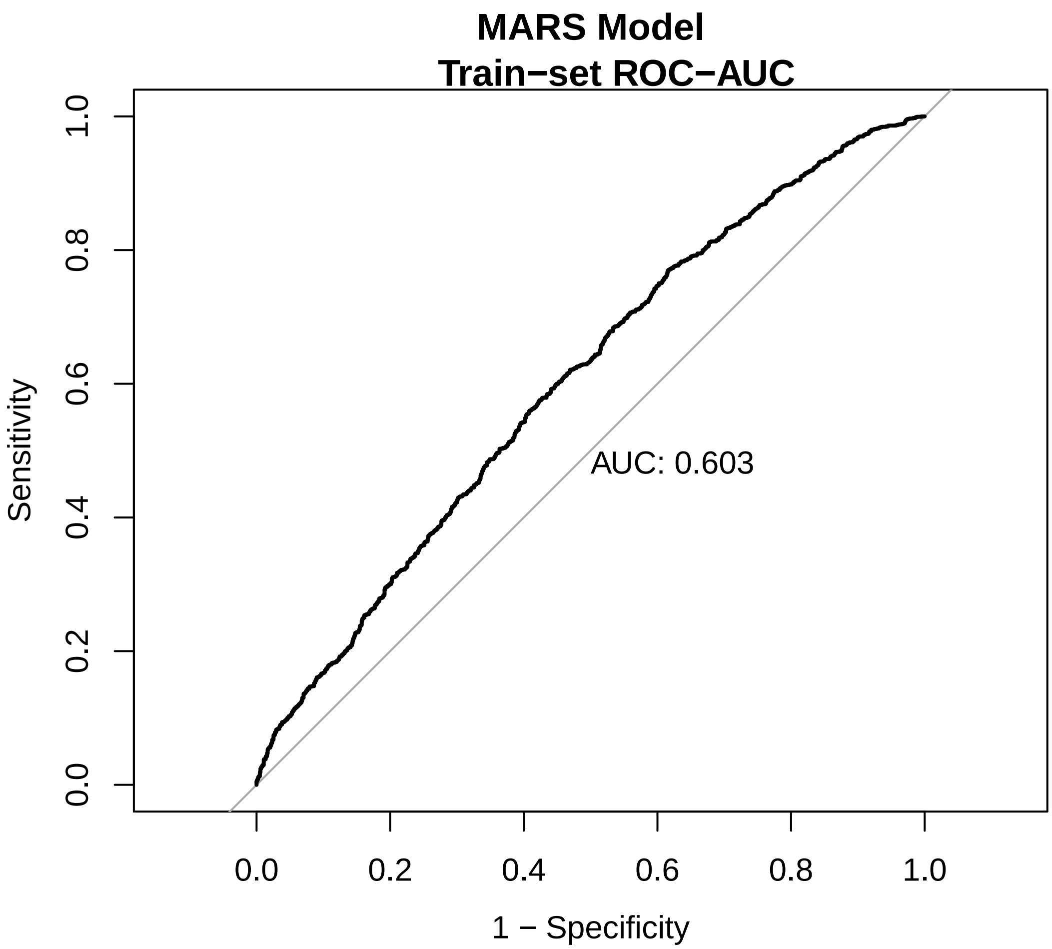


Table 26: MARS Model - Confusion Matrix

	no	yes
no	1471	128
yes	852	171

Table 27: MARS Model - Training Results

Performance	model
Area Under Curve	0.6032
Sensitivity	0.1672
Specificity	0.9199

Nearest Shrunken Centroids

```
set.seed(476)

water_nsc <- train(x = train, train_class,
                   method = "pam",
                   metric = "ROC",
                   preProc = c("center", "scale"),
                   tuneGrid = data.frame(.threshold = 0:25),
                   trControl = ctrl)
saveRDS(water_nsc, "water_nsc.rds")
```

```
water_nsc <- readRDS("water_nsc.rds")
nsc_df <- get_model_info(water_nsc, train, train_class)
nsc_roc <- get_roc(nsc_df)
nsc_results <- get_auc(nsc_roc, nsc_df, train_class)
```

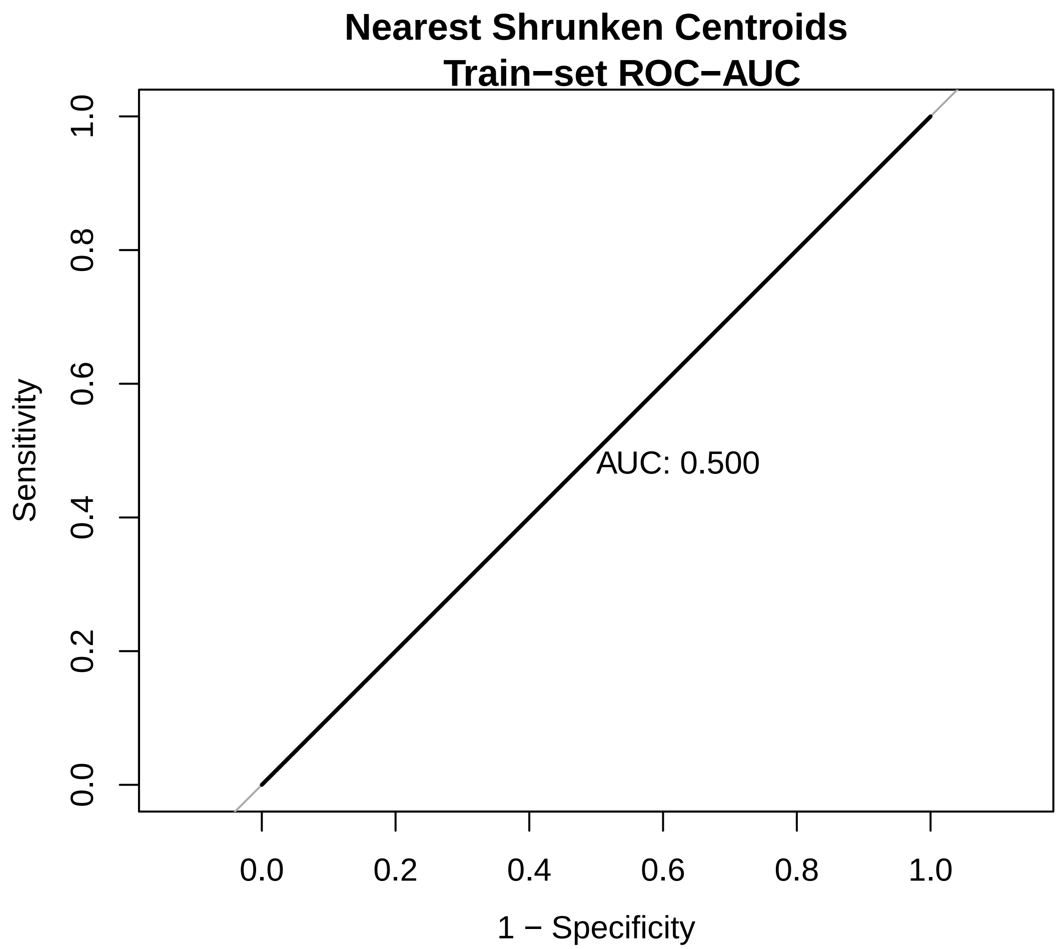


Table 28: Nearest Shrunken Centroids - Confusion Matrix

	no	yes
no	1599	0
yes	1023	0

Table 29: Nearest Shrunken Centroids - Training Results

Performance	model
Area Under Curve	0.5
Sensitivity	0
Specificity	1

Logistic Regression

```
set.seed(476)

water_log <- train(x = train, train_class,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)
saveRDS(water_log, "water_log.rds")
```

```
water_log <- readRDS("water_log.rds")
log_df <- get_model_info(water_log, train, train_class)
log_roc <- get_roc(log_df)
log_results <- get_auc(log_roc, log_df, train_class)
```

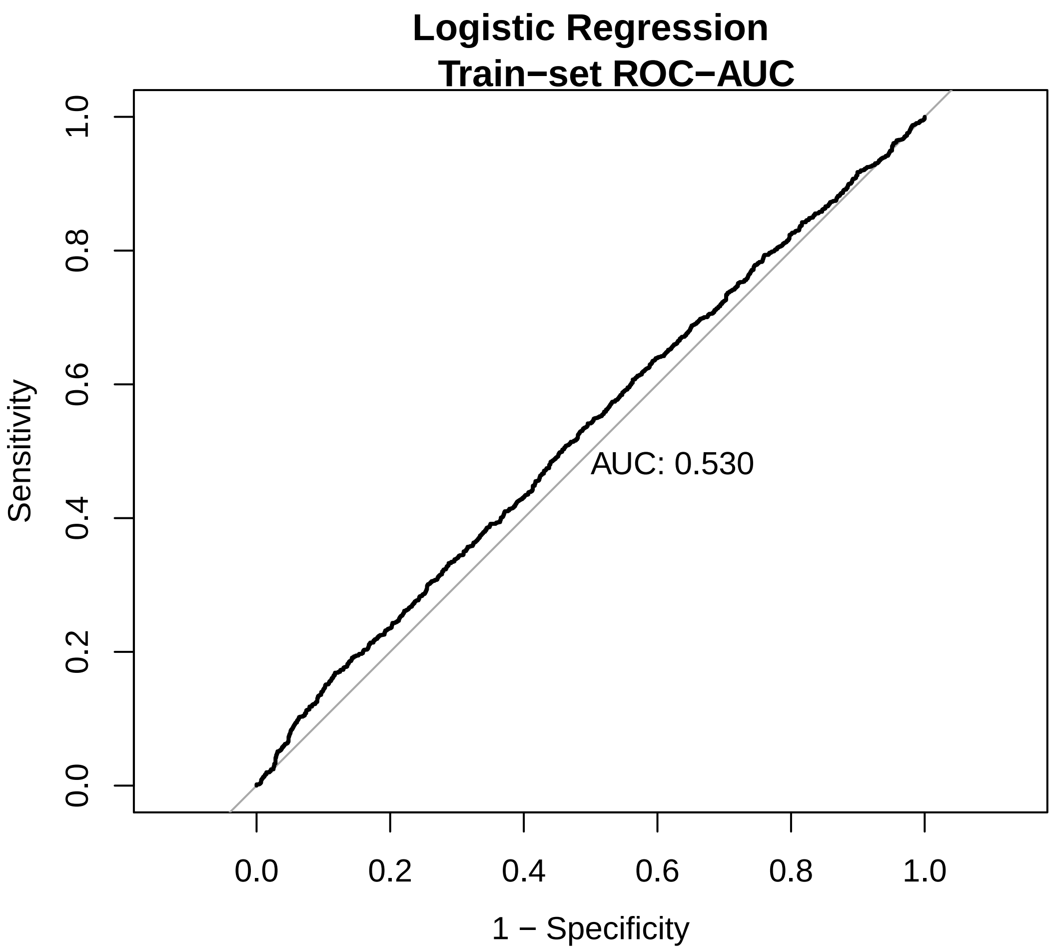


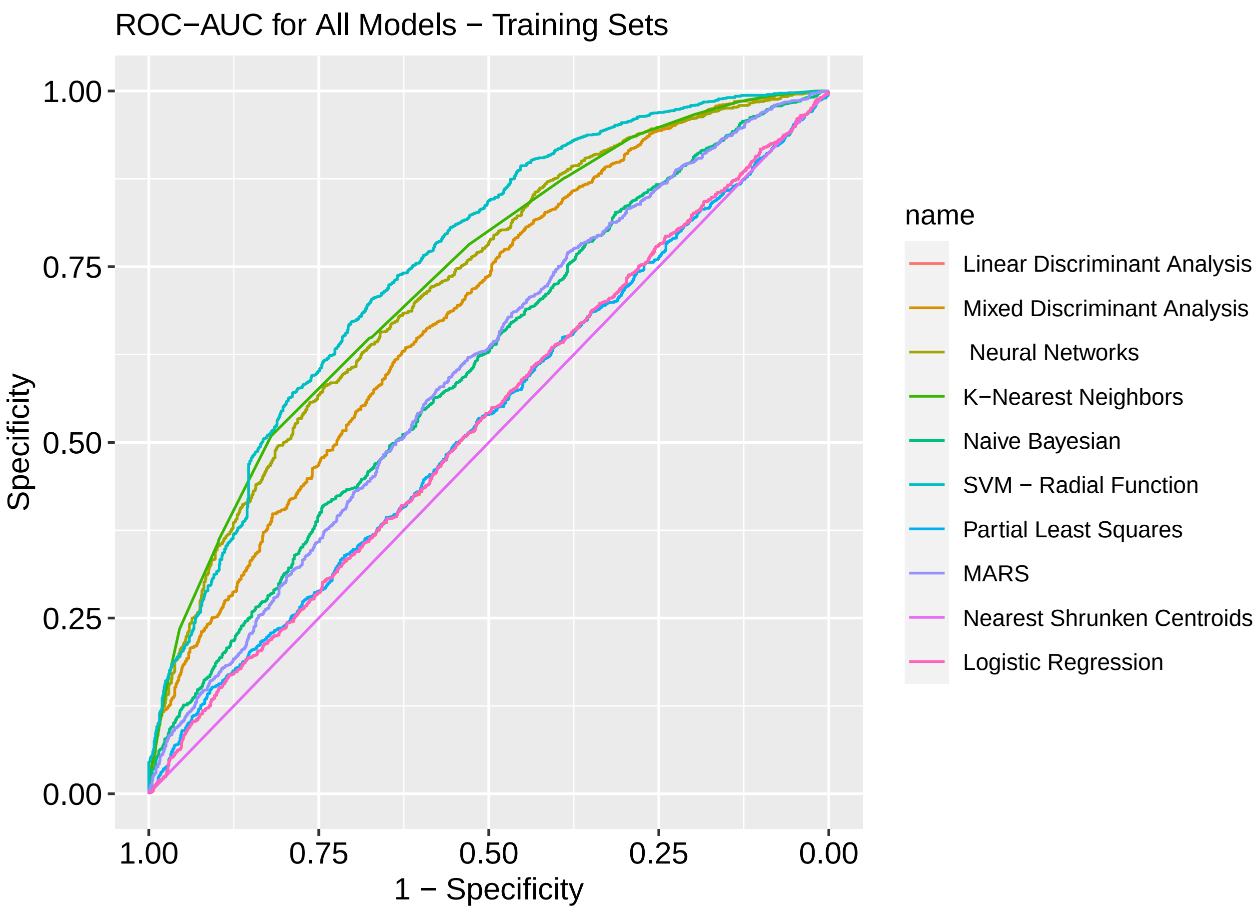
Table 30: Logistic Regression - Confusion Matrix

	no	yes
no	1599	0
yes	1023	0

Table 31: Logistic Regression - Training Results

Performance	model
Area Under Curve	0.5297
Sensitivity	0
Specificity	1

All Models ROC-AUC curve:

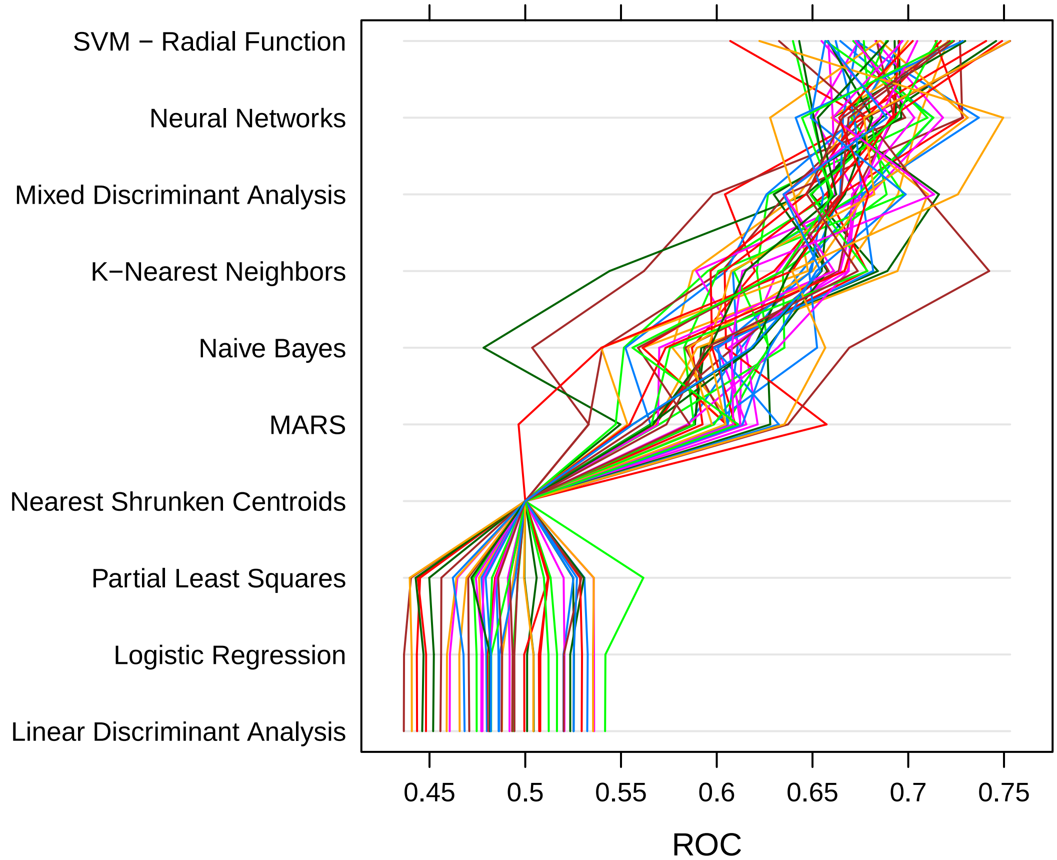


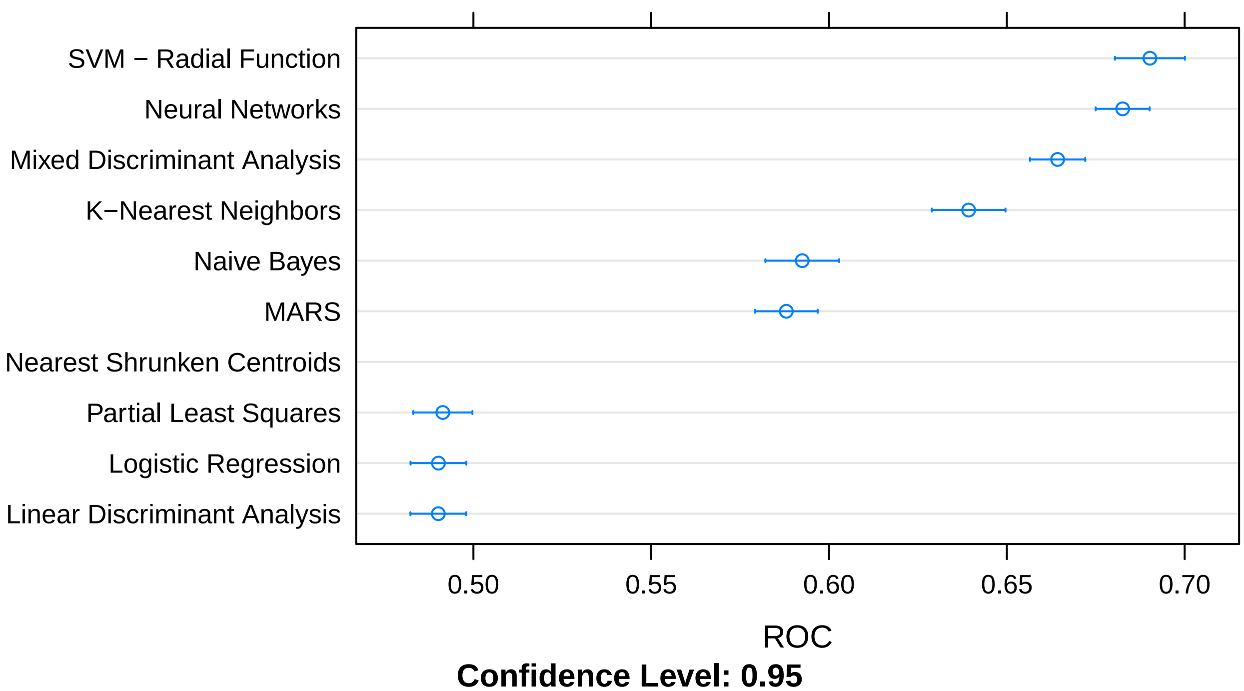
All Models AUC Values:

Table 32: All Models (Training Set) - AUC

Models	AUC	Sensitivity	Specificity
Linear Discriminant Analysis	0.5297	0	1
Mixed Discriminant Analysis	0.6841	0.305	0.9012
Neural Networks	0.7241	0.4223	0.8618
K-Nearest Neighbors	0.7335	0.2884	0.935
Naive Bayes	0.6072	0.2336	0.8768
SVM - Radial Function	0.7545	0.3744	0.9293
Partial Least Squares	0.5292	0	1
MARS	0.6032	0.1672	0.9199
Nearest Shrunken Centroids	0.5	0	1
Logistic Regression	0.5297	0	1

Resampled ROC values





Results

SVM Radial Function - Best Tuning Parameters

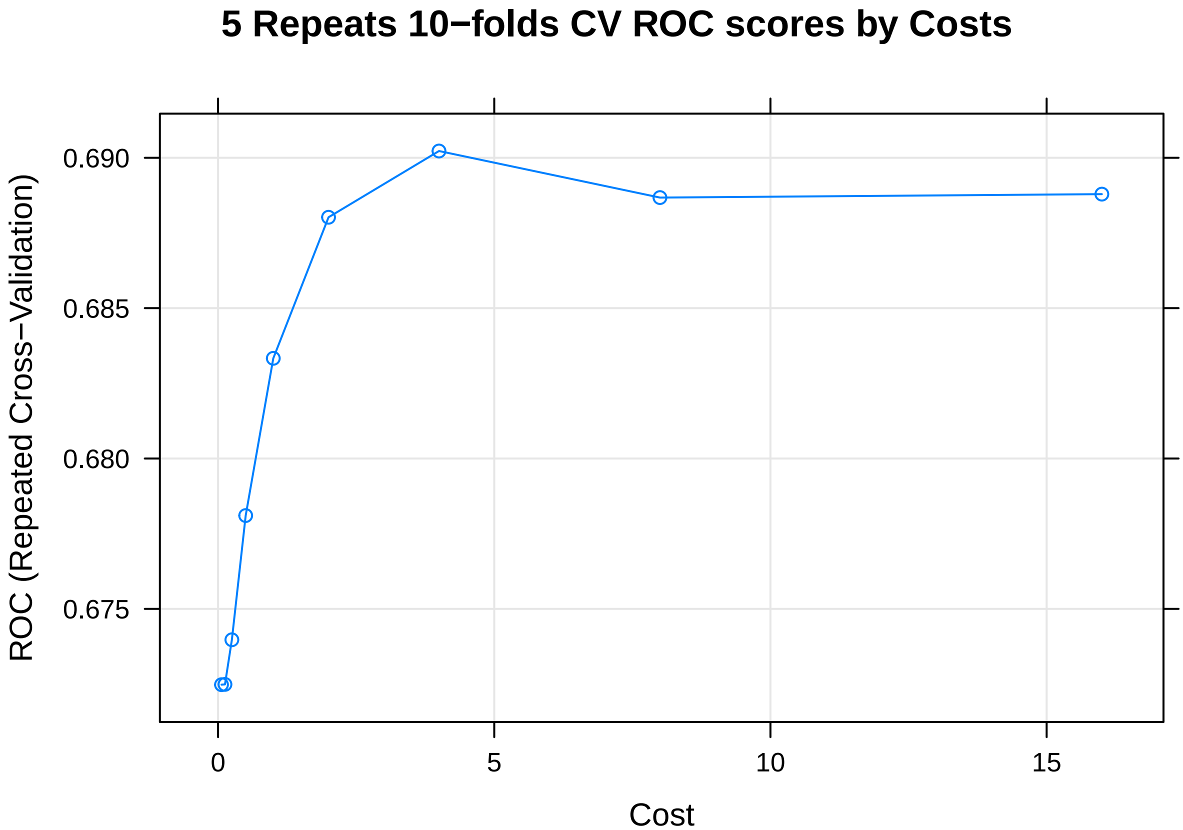


Table 33: Best Tuning Parameter based on ROC values

	sigma	C
7	0.03307	4

Neural Networks - Best Tuning Parameters

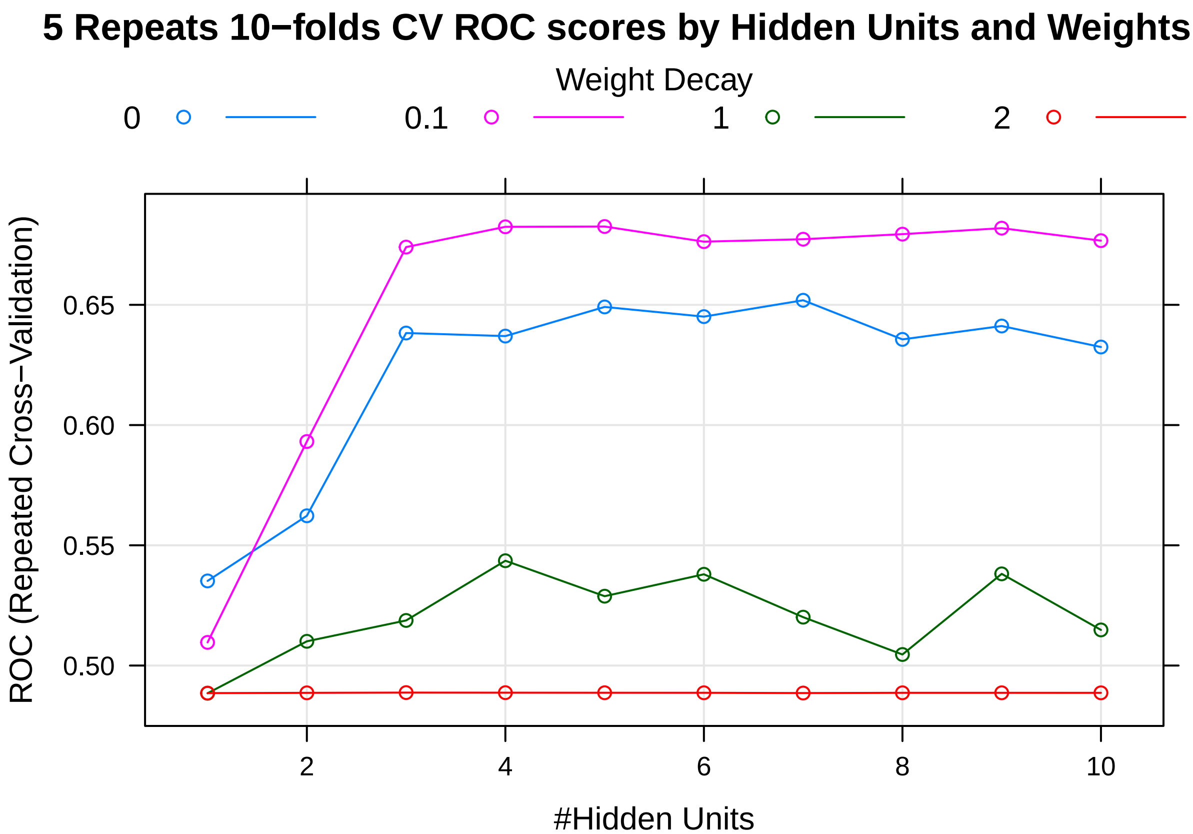
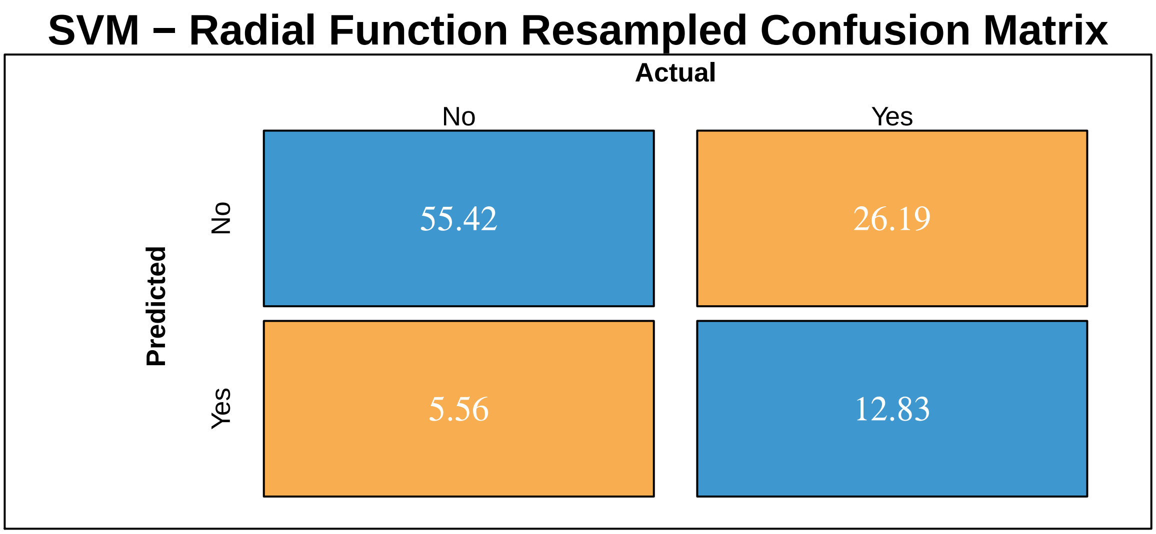


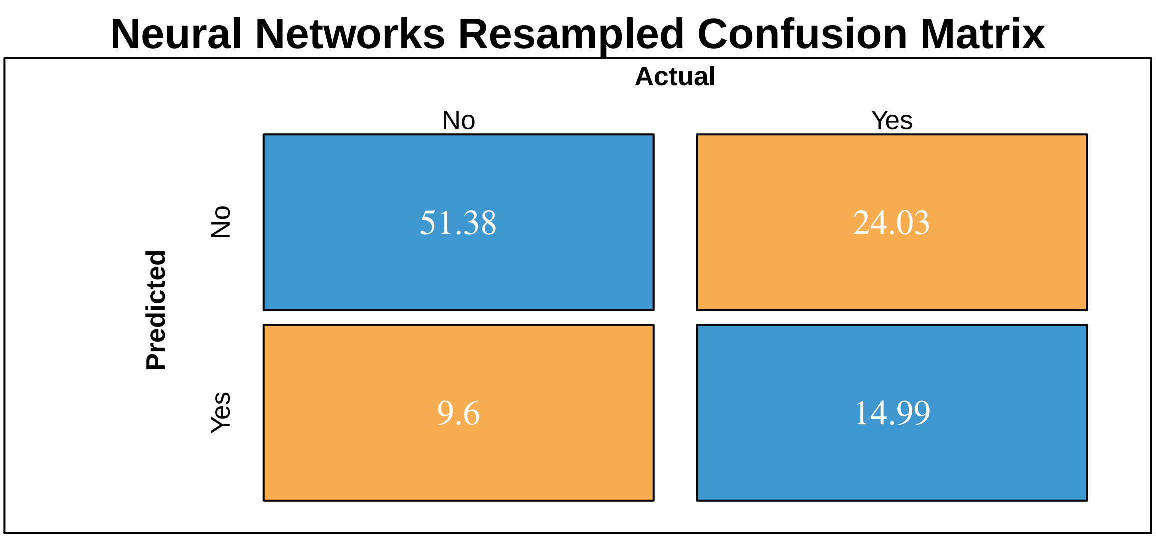
Table 34: Best Tuning Parameter based on ROC values

	size	decay
18	5	0.1

Resampled Confusion Matrix (Training Set) - SVM vs. Neural Networks




```
draw_confusion_matrix(nnet_cm, "Neural Networks")
```

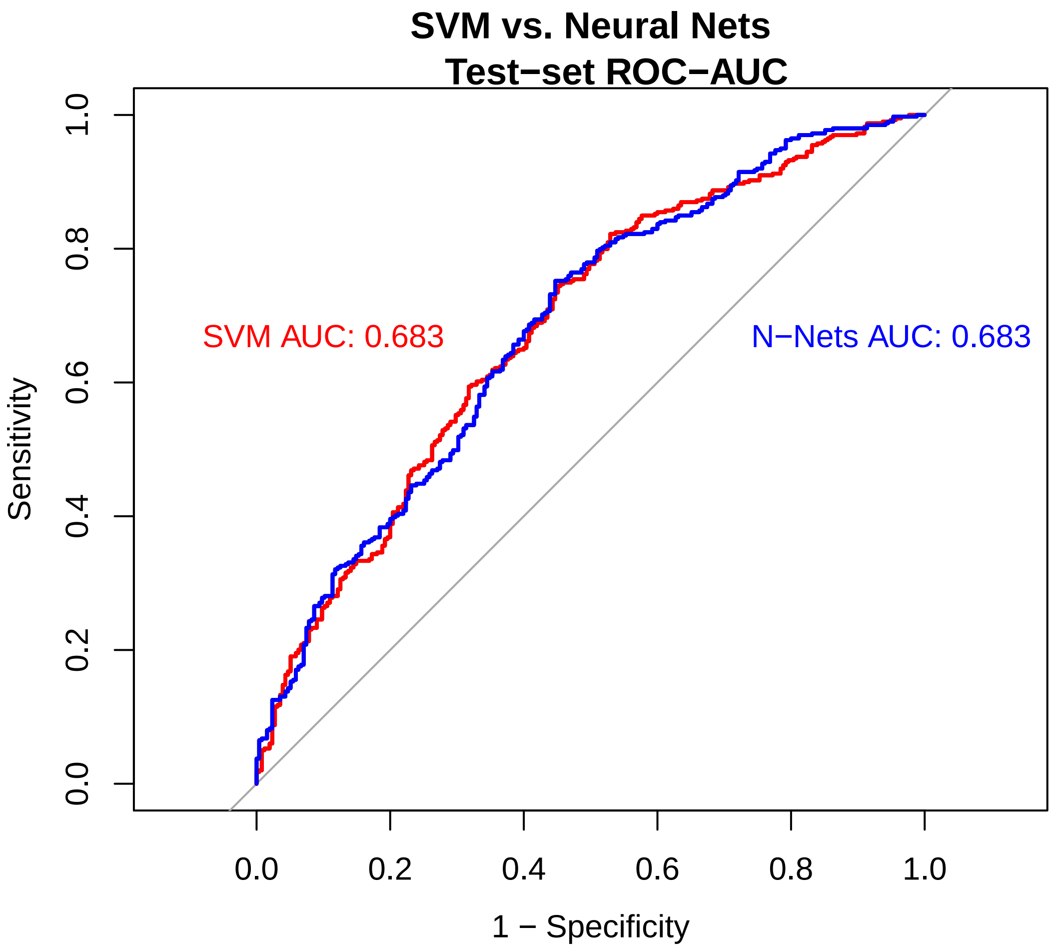


Test Sets - SVM vs. Neural Networks

```
svm_test_df <- get_model_info(water_svm, test, test_class)
svm_test_roc <- get_roc(svm_test_df)
svm_test_results <- get_auc(svm_test_roc, svm_test_df, test_class)
```

```
nnet_test_df <- get_model_info(water_nnet, test, test_class)
nnet_test_roc <- get_roc(nnet_test_df)
nnet_test_results <- get_auc(nnet_test_roc, nnet_test_df, test_class)
```

Final Models Test Set Performance



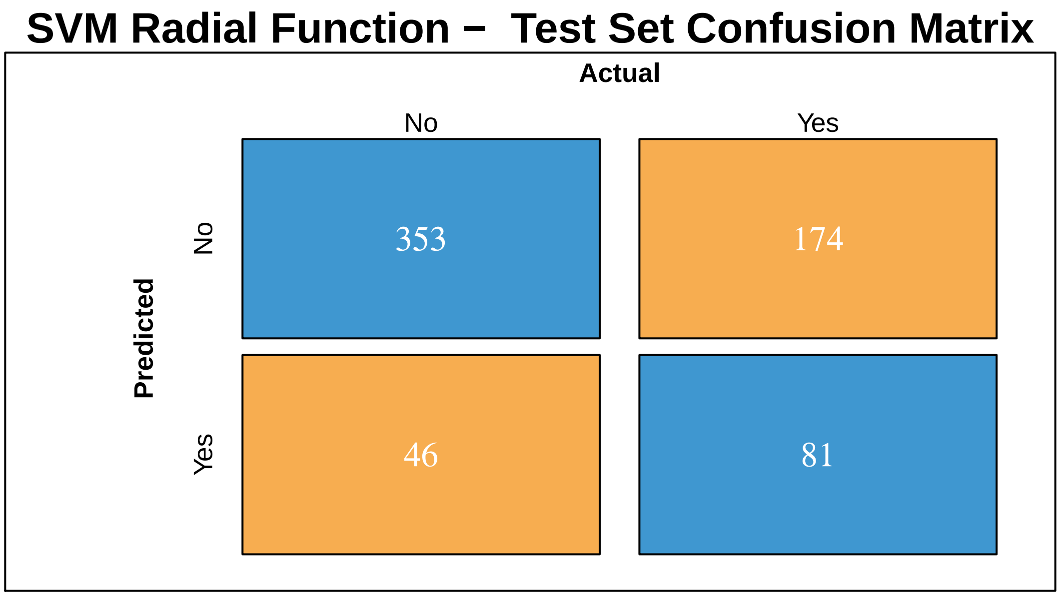


Table 35: SVM - Test Set Performance

Performance	model
Area Under Curve	0.6831
Sensitivity	0.3176
Specificity	0.8847

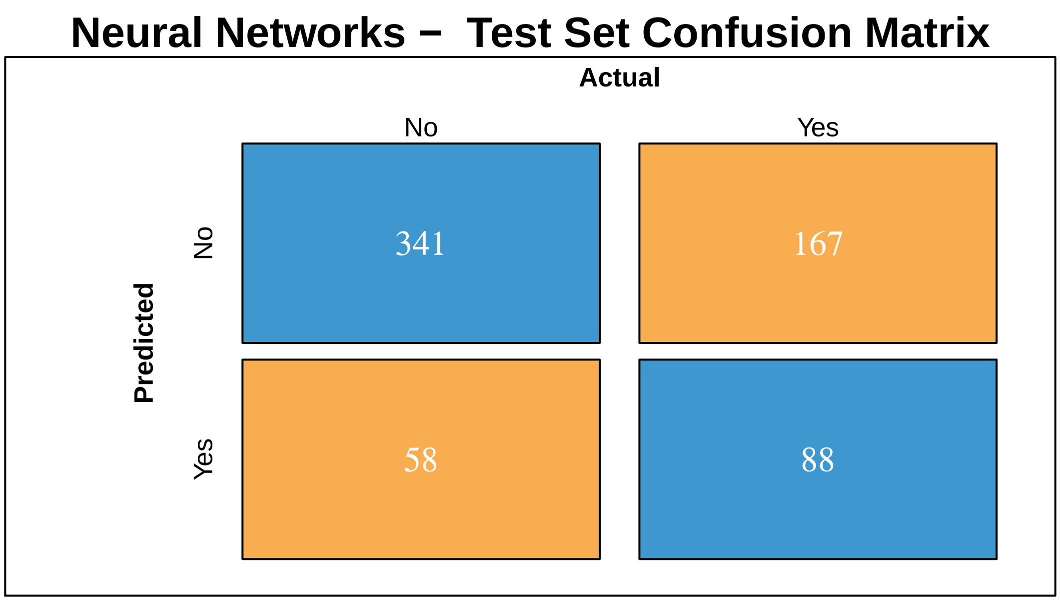


Table 36: N-Nets - Test Set Performance

Performance	model
Area Under Curve	0.6833
Sensitivity	0.3451
Specificity	0.8546