A Binary Classification Problem: Assessing Water Quality and Detecting Potability

Nhan (Jimmy) Nguyen

Sarah Alqaysi

Sai Thiha

University of San Diego

ADS-503 Applied Predictive Modeling

28 June 2021

TABLE OF CONTENTS

Problem Statement

One of the most basic human rights in life is to have access to a drinkable water that adheres to all safety criteria and regulations. H.H. Mitchell from the Journal of Biological Chemistry 158 stated that the brain and heart are composed of 73% water. This means that dehydration can negatively affect several cognitive functions such as attention, memory, and mood. Both national and local authorities as well as private companies should be always monitoring and enforcing the right procedure when it comes to water filtration systems, down to every single house. Water potability is essential to human survival and should be prioritized for detecting safe drinking water. The goal of this project is to classify the safety and quality of drinkable water.

Justification

There are nine water quality metrics that describe the potability of a water sample. The pH value evaluates the acid-base range of water. This serves as the condition of water status for acidic or alkaline. Hardness is an observed metric of how much hardness is in raw water when coming in contact with calcium and magnesium salts. Total dissolved solids produce an unpleasant taste and diluted color in water through contact with minerals such as potassium, calcium or magnesium, etc. The amount of chlorine and chloramine in public water systems determines the safety for potable water. Sulfate concentrations are commonly found in sea water as a result from minerals, soils and rocks. A useful characteristic such as conductivity where the electrical conductivity value should not exceed 400 μS/cm according to WHO standards. Another strong indicator is the total amount of organic carbon compounds found in water sources which can determine the potable water quality. Whereas trihalomethanes levels up to 80 ppm are considered safe for drinking water. Lastly, turbidity describes the amount of solid matter present in the suspended state of water. Since potability characters may overlap with one another of these features, higher performance may be associated with more complex models, resulting in the expense of computing time to determine the potability.

Data Description

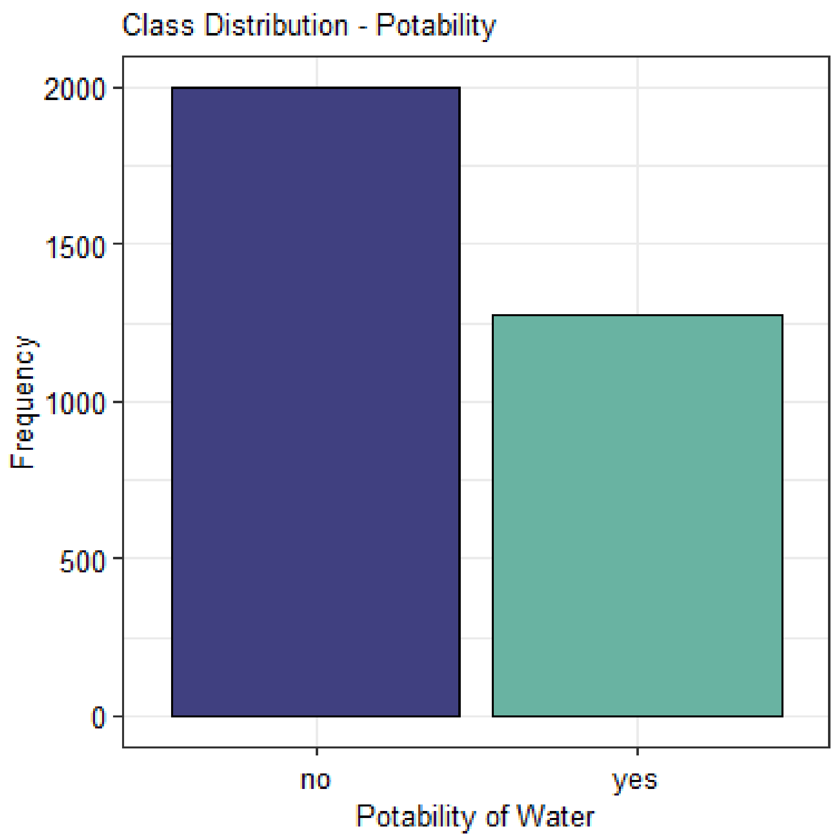
This water quality data set is obtained from Kaggle. A public data repository and platform for data scientists and machine learning practitioners to practice problem-solving. The data set has a small level of dimensionality with only 10 columns and 3,276 samples. In fact, 9 of the 10 variables are predictors describing the potability of these water samples. The data set is a comma separated values formatted file with only 512.88 kilobytes. The predictors are numerical variables containing continuous values on different scales and the response variable is a nominal categorical variable containing binary values of 0s and 1s. In summary, this data set does not suffer from a class imbalance problem or with high dimensionality.

Data Exploration

The first step in exploratory data analysis for this classification problem is to examine the class distribution for the variable of interest. The variable of interest is called '*potability*'. This variable as stated before is a categorical nominal variable consisting of values of *0* and *1*. The value of zero represents that a water sample is not potable or not safe for human consumption. On the other hand, the value of one represents the potable water quality. The class proportion of *0* is 61% whereas 1 has a proportion of 39%, as shown in Figure 1. This is a fairly balanced distribution between classes. The next step is to examine the predictors as a group using a correlation matrix to find any association between predictors.

Figure 1

Class Distributions - Potability



Note. potability of class (1) contained in 39% of the data set, while the non-potability class (0) contained in 61% of the data.

Data Wrangling

Pre-processing

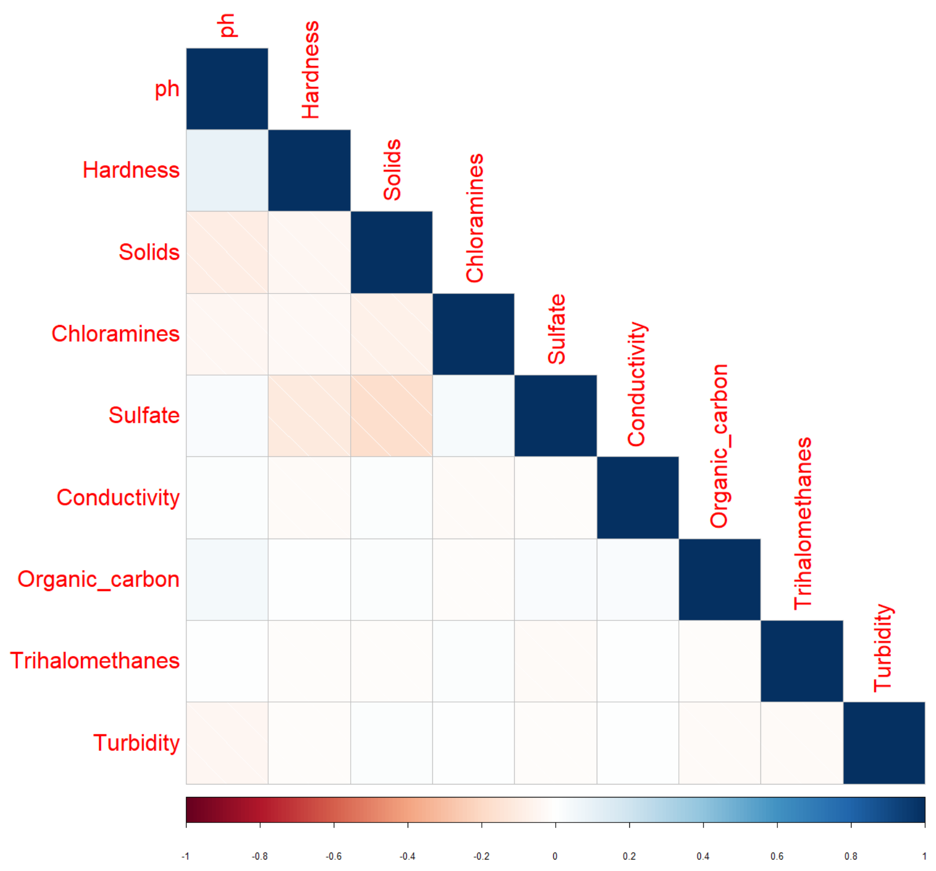
Since this data set contains only continuous predictors, the first pre-processing step is to search for predictors with degenerate distributions. This is important because the non-linear classification models may be affected with uninformative variables. After checking for any near zero-variance predictors, this data set does not contain such predictors. The next step is to look for between-predictor correlations. The cut-off point chosen is *0.75* and there is also no collinearity. Although the highest level of correlation is *0.2* between water samples' pH values and the hardness metric, as shown in Figure 2. This draws attention to a low degree for correlation. After attempting to remove collinearity, frequency distribution should be examined for any levels of skewness. The frequency distribution of predictors overlaid with the classes of potability displays fairly symmetrical distributions. Therefore, after careful examination of the skewness value for each predictor, all predictors follow a normal distribution.

Missing values

The last step in data wrangling is checking for missing values. The missingness pattern is apparent in this data structure across the predictors between potability. A missing values analysis shows potable water samples contain 27% more missing values than non-potable water samples. Strategies for dealing with missing values include replacing missing values with arbitrary value, mean or median, or a random sample. However, this may affect the performance of the models since it can lead to more noise. So the selected approach was to use a K-nearest neighbours imputation method to handle missing values. This concludes the data-wrangling stage where handling missing values was the only pre-processing step.

Figure 2

Correlation Matrix of Predictors



Data Splitting

Training, Validation, Test sets

There are 3,276 samples available and 80% will be used for training the models, while the remaining will be used to evaluate the final optimal model with the highest AUC value. Since this is a binary classification problem, the data were split by using a stratified random sampling to preserve class frequency distribution in the train and test split. When tuning for the optimal parameter(s) of each model, the search approach is 5 repeats of 10-fold cross-validation.

Model Strategies

Main Research Question

The main research question is based on the nine metrics of water samples, can a predictive model be used to correctly classify if a water sample is safe to drink for human consumption. The purpose of this project is to find a predictive model that can accurately classify water samples to change the way water quality is assessed for human consumption.

Linear Classification Models

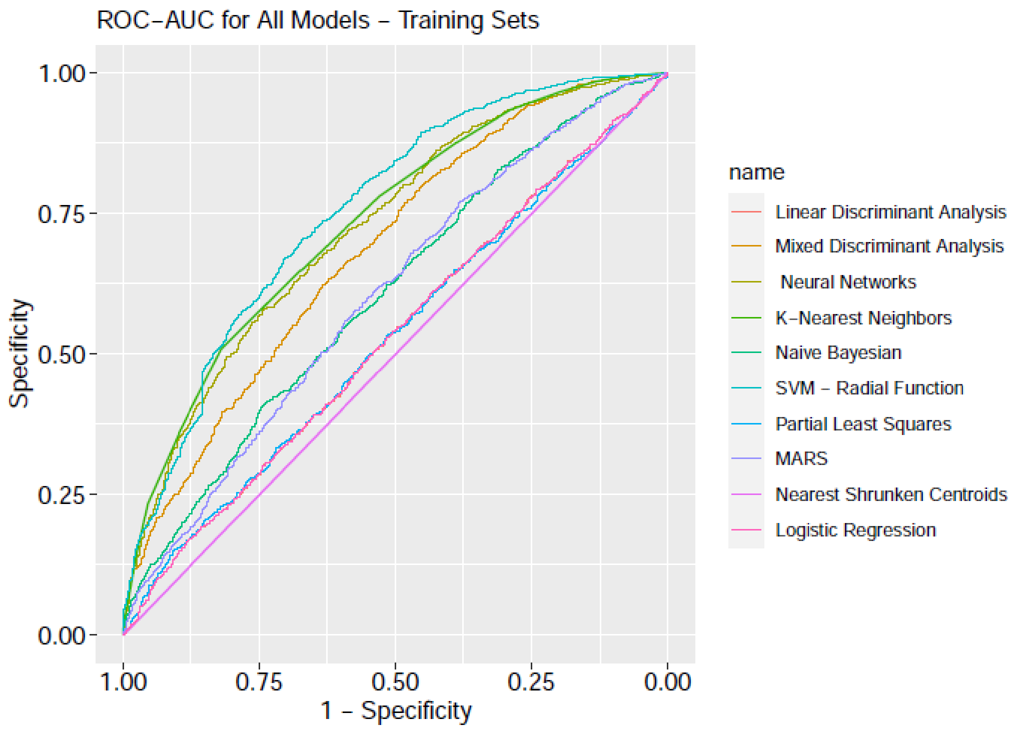
The creation of several linear classification models was built in an effort to beat an AUC baseline of 70% or more. These models include MARS, Partial Least Squares, Logistic Regression, Linear Discriminant Analysis, and Nearest Shrunken Centroids. However, these models failed to reach this level of prediction. A different approach was then taken with non-linear classification models, and it was found that *Support Vector Machines* tested the highest AUC curve with a value of 75.45%, it outperformed *Neural Networks, K-Nearest Neighbors* and all other non-linear classification models.

Nonlinear Classification Models

Mixed Discriminant Analysis, Neural Networks, K-Nearest Neighbours, Naive Bayesian and SVM-Radial Function models were performed for ROC-AUC fitness. And the following models, SVM, Neural Networks and K-Nearest came back with high AUC values. SVM predicted the highest AUC curve among the nonlinear classifiers with a value of 75.55%, it also predicted a high Specificity level of 0.9293. Different *tuneLength* values were also tested at 10 and 20, but they all predicted the same performance levels. Naive Bayes predicted the second highest AUC value of 73.35% and Neural Networks predicted the third highest AUC value of 72.4%. For the rest of models, given that AUC is closer to 0.5 than 1, it appears to have an unbalanced classification.

Figure 3

All Models ROC-AUC curve



Validation and Testing

Even though SVM scored the highest AUC value, followed by KNN, further performance metrics were measured to validate the accuracy of these models. Resamples of ROC values were taken for all models, to statistically compare model performance based on ROC values. R ranked the highest ROC values for SVM and interestingly for Neural Network. Figure 4 shows how SVM and Neural Network are the ones with the best ROC values. Figure 5 shows the average ROC value with standard error where you can see that SVM and Neural networks are the best performing models.

Results and Final Model Selection

To confirm the validation and testing findings, an ROC curve was calculated of the test set for both SVM and Neural Network, in addition to a confusion matrix and it was found that they have similar values, as shown in Figure 6. As a result, the Neural Network was chosen to be the preferred model for this dataset, as it also predicted more correct 'yes' samples that are actually 'yes' samples, as can be seen in Figure 7 and Figure 8 respectively.

Figure 4

All Models Resampled ROC Values

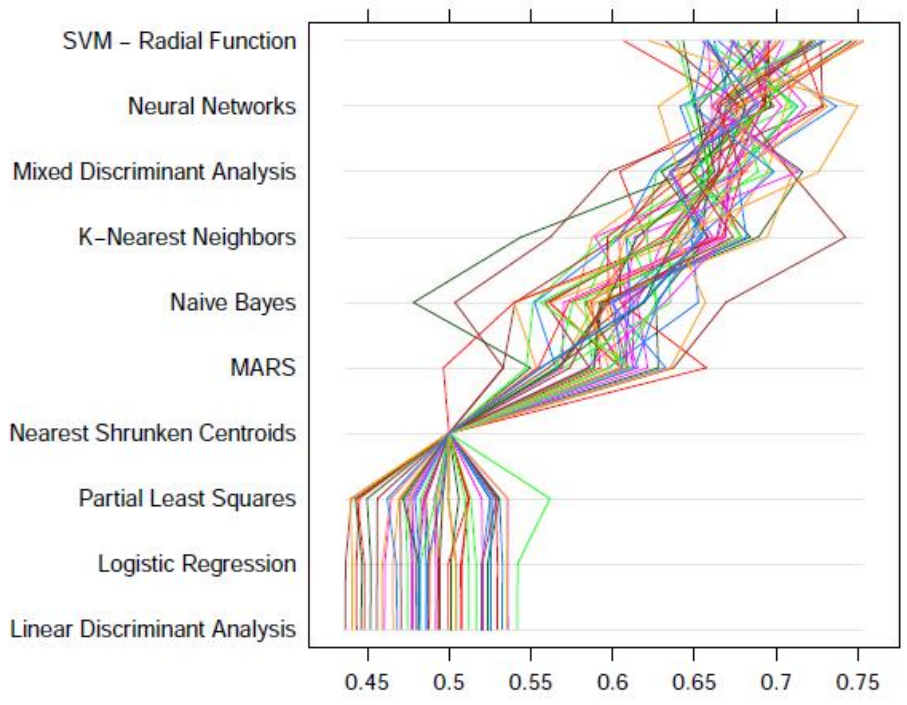


Figure 5

Average ROC value with standard error

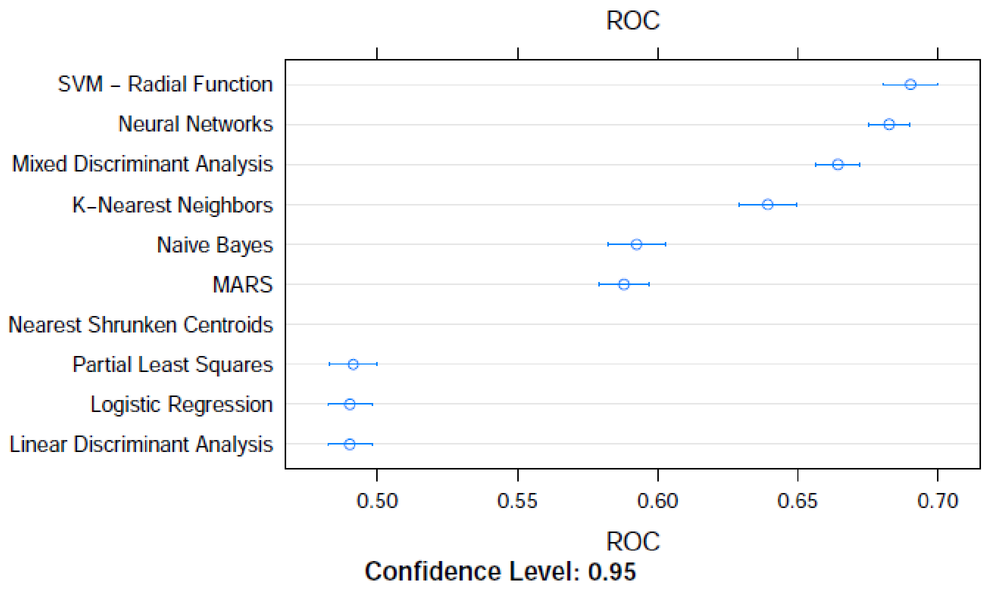


Figure 6

Final Models Test Set Performance

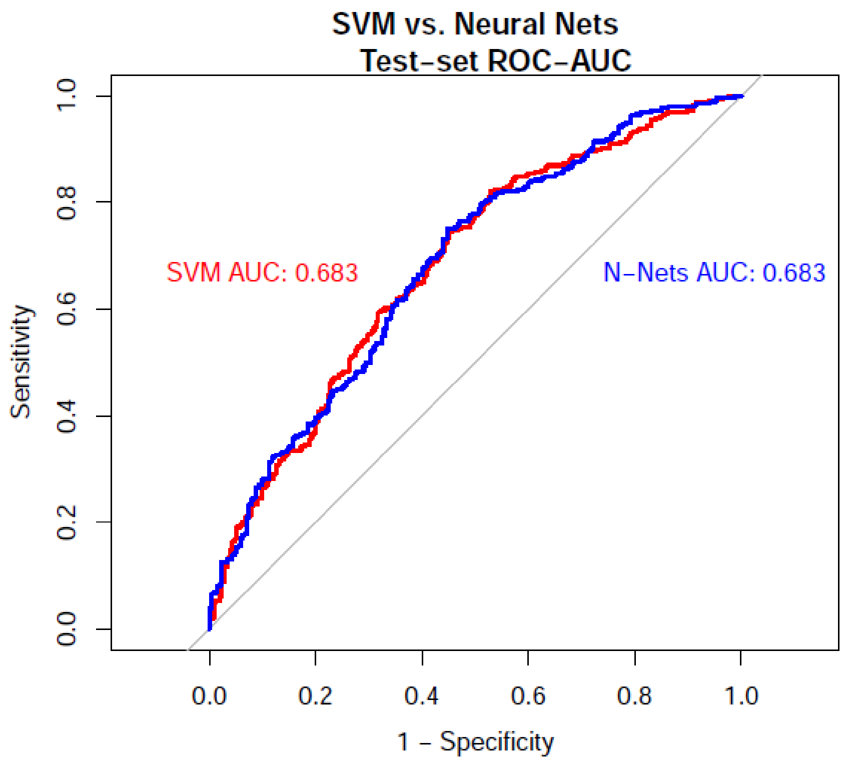


Figure 7

SVM Radial Function - Test Set Confusion Matrix

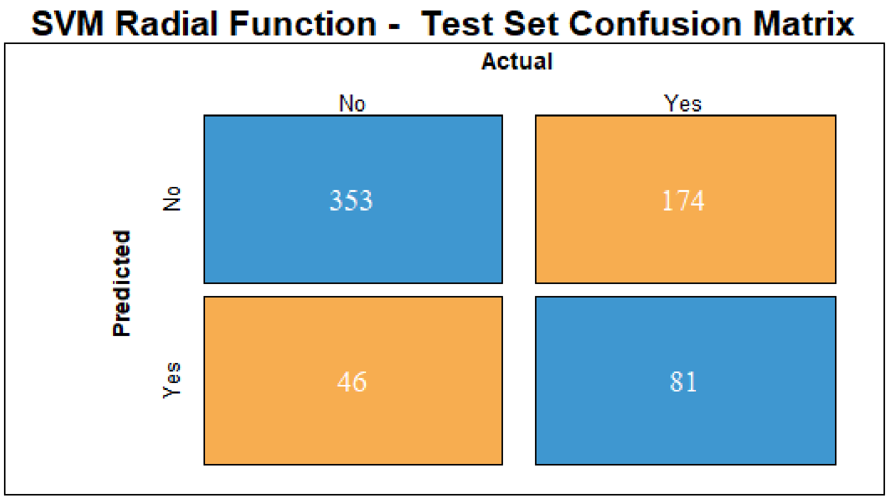
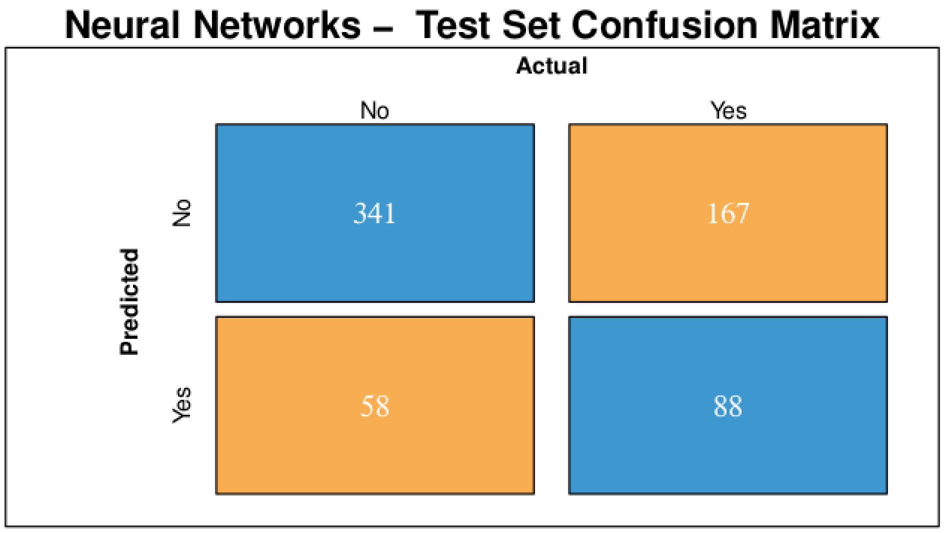


Figure 8

Neural Networks - Test Set Confusion Matrix



Conclusion

Findings

Based on WHO standard for the requirement of water potability in table A2 Appendix A, Chloramines, and Sulfate level shows 97% of the water sample is not potable. In terms of Carbon level and Hardness, 90% of the water sample is not potable. One of the difficulties is that 7 predictors show more than 90% of water is not potable but distribution of potability and non potability is 62% vs 38%. It seems that a possible discrepancy in the water dataset interferes with the models performance to reach a higher level of AUC. Although SVM and Neural Networks were tied under the AUC values, it should be noted that Neural Networks were selected as the final model due to its higher true positive value.

Suggestions

Predictors values should be adjusted to WHO standards to meet the requirements of water potability. Discrepancy of water data should be reviewed by experts before using them for nonlinear classification models.

References

Mitchell, H. Hamilton, T., Steggerda, F., Bean, H. (May 1945). THE CHEMICAL COMPOSITION OF THE ADULT HUMAN BODY AND ITS BEARING ON THE BIOCHEMISTRY OF GROWTH, Journal of Biological Chemistry, Volume 158, Issue 3, 1945, Pages 625-637, ISSN 0021-9258.https://doi.org/10.1016/S00219258(19)513394. https://www.sciencedirect.com/science/article/pii/S0021925819513394.

Hardness of Water. Usgs.gov. (2021). Retrieved 28 June 2021, from https://www.usgs.gov/special-topic/water-science-school/science/hardness-water?qt-science_center_objects=0#qt-science_center_objects.

Kadiwal, A. *Water Quality*. Kaggle.com. Retrieved 28 June 2021, from https://www.kaggle.com/adityakadiwal/water-potability.

Enderlein, U., Enderlein, R., and Williams W. *Water Quality Requirements*. WHO. Retrieved 28 June 2021, from https://www.who.int/water_sanitation_health/resourcesquality/wpcchap2.pdf

Appendix A

Table A1

Information about the Data Set

Feature Name	Description
ph	pH of 1. water (0 to 14)
Hardness	Capacity of water to precipitate soap in mg/l
Solids	Total dissolved solids in ppm
Chloramines	Amount of Chloramines in ppm.
Sulfate	Amount of Sulfates dissolved in mg/L
Conductivity	Electrical conductivity of water in μS/cm
Organic_carbon	Amount of organic carbon in ppm
Trihalomethanes	Amount of Trihalomethanes in μg/L
Turbidity	Measure of light emitting property of water in NTU
Potability	Indicates if water is safe for human consumption. Potable - 1 and Not potable - 0

Table A2

Data Report for Continuous Data

Feature Name	Data Continuous Values: WHO standard of Acceptable Potable Water
pH Level	pH level between 7 to 7.5 is acceptable for drinkable water.
Hardness	Hardness level guideline is between 0 to 60 mg/L, 61 to 120 mg/L is considered

	moderately hard, and 121 to 180 mg/L is hard, more than 180 mg/L is very hard. Hardness is the result of compounds of calcium and magnesium, and other metals.
Solids	TDS level between 350-500 is acceptable. Concentration of dissolved particles or solids in water. TDS comprises inorganic salts such as calcium, magnesium, chlorides, sulfates, bicarbonates. TDS above 2000 is not acceptable for drinking water.
Chloramines	Chloramines levels up to 4 mg/L(milligrams per liter or 4 ppm (parts per million) are considered safe in drinking water
Sulfate	Sulfate level 250 mg/L is considered contaminant level. Sulfate is a substance that occurs naturally in drinking water.
Conductivity	Conductivity of 1412 μS/cm is standard level. Conductivity is to measure the ability to pass an electrical current. It is affected by temperature. Conductivity increases as temperature increases.
Organic_carbon	Drinking water's acceptable level of Organic Carbon is between 100 ppb to 10 ppm. Total Organic Carbon is a measure of the total amount of carbon in organic compounds in pure water.
Trihalomethanes	Trihalomethane's acceptable level in drinking water is up to 100 microgram per little.
Turbidity	Turbidity of acceptable level is not more than 5 NTU for potability of water.

Appendix B

Following is the R Markdown file for this project.

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	Organic_carbon	Trihalomethanes	Turbidity	Potability
NA	204.9	20791	7.3	368.5	564.3	10.38	86.99	2.963	no
3.716	129.4	18630	6.635	NA	592.9	15.18	56.33	4.501	no
8.099	224.2	19910	9.276	NA	418.6	16.87	66.42	3.056	no
8.317	214.4	22018	8.059	356.9	363.3	18.44	100.3	4.629	no
9.092	181.1	17979	6.547	310.1	398.4	11.56	32	4.075	no
5.584	188.3	28749	7.545	326.7	280.5	8.4	54.92	2.56	no

Data Set Total Number of Water Samples

Table 3: Total Number of Water Samples

Total
3276

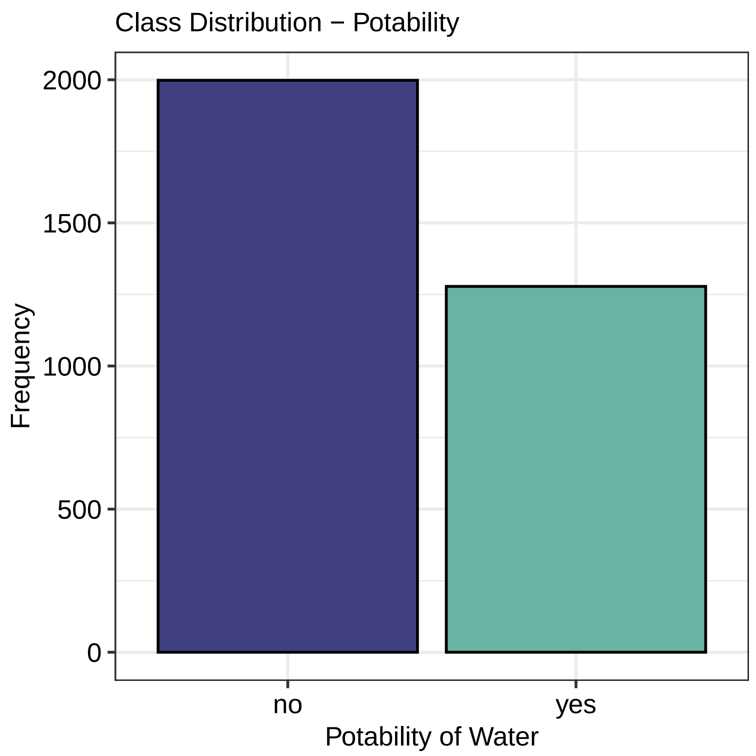
Data Set Total Number of Predictors

Table 4: Total Number of Water Characteristics

Total
9

Data Exploration

Class Distributions - Potability



Class Proportions

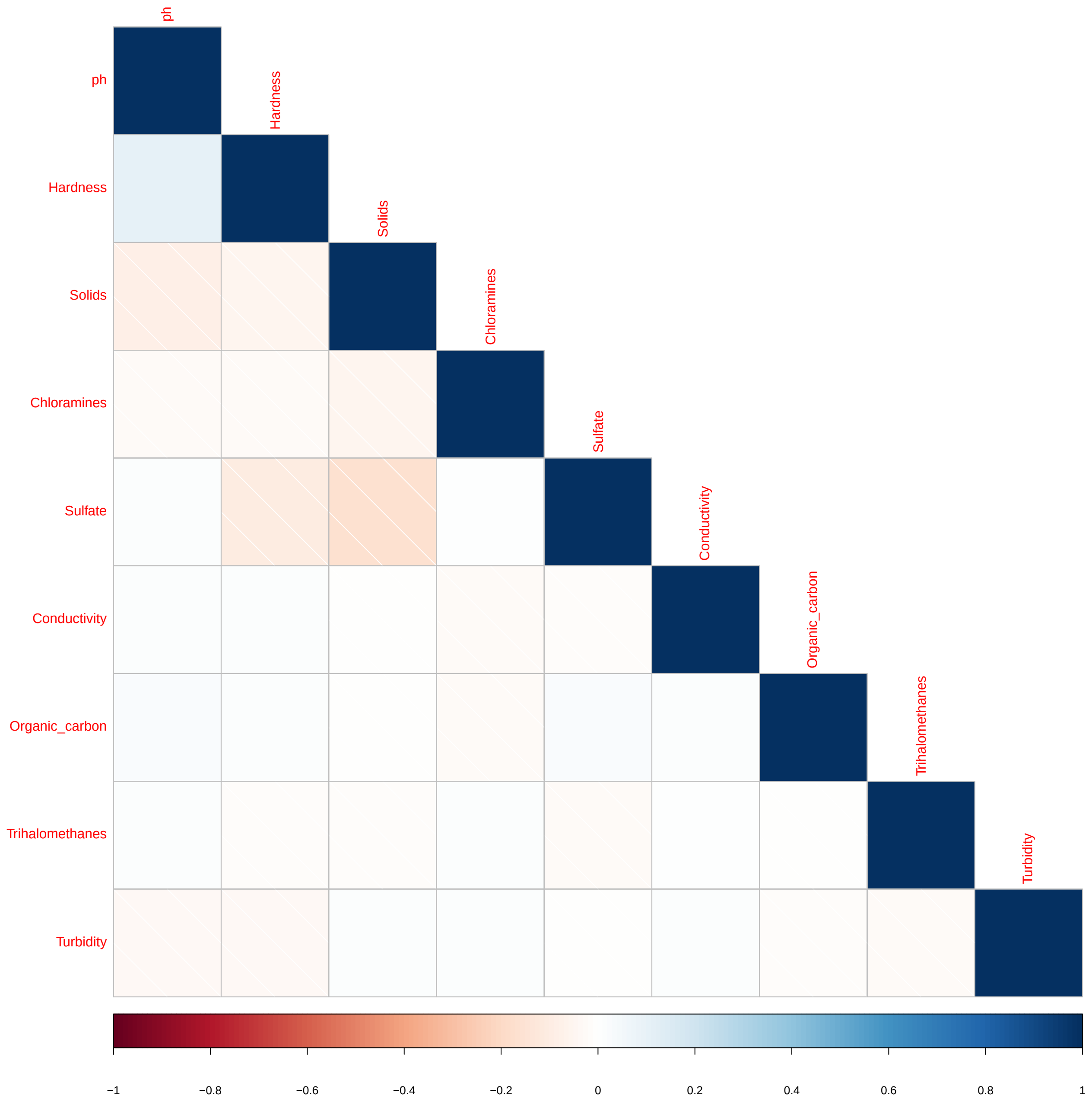
```
pota<- table(df$Potability)
round(prop.table(pota), digits = 3) %>% pander(style = "grid",
         caption = "Class Proportions")
```

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:

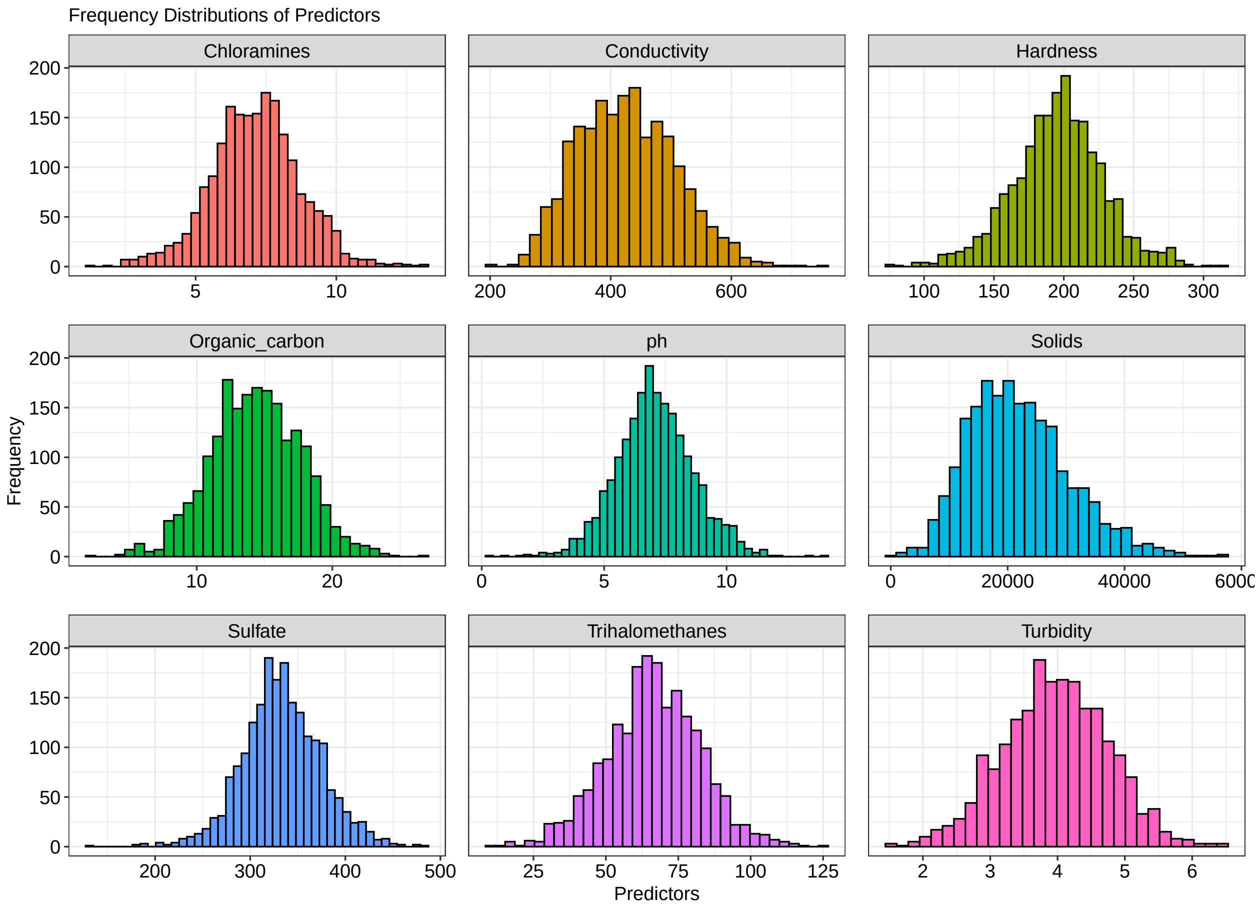


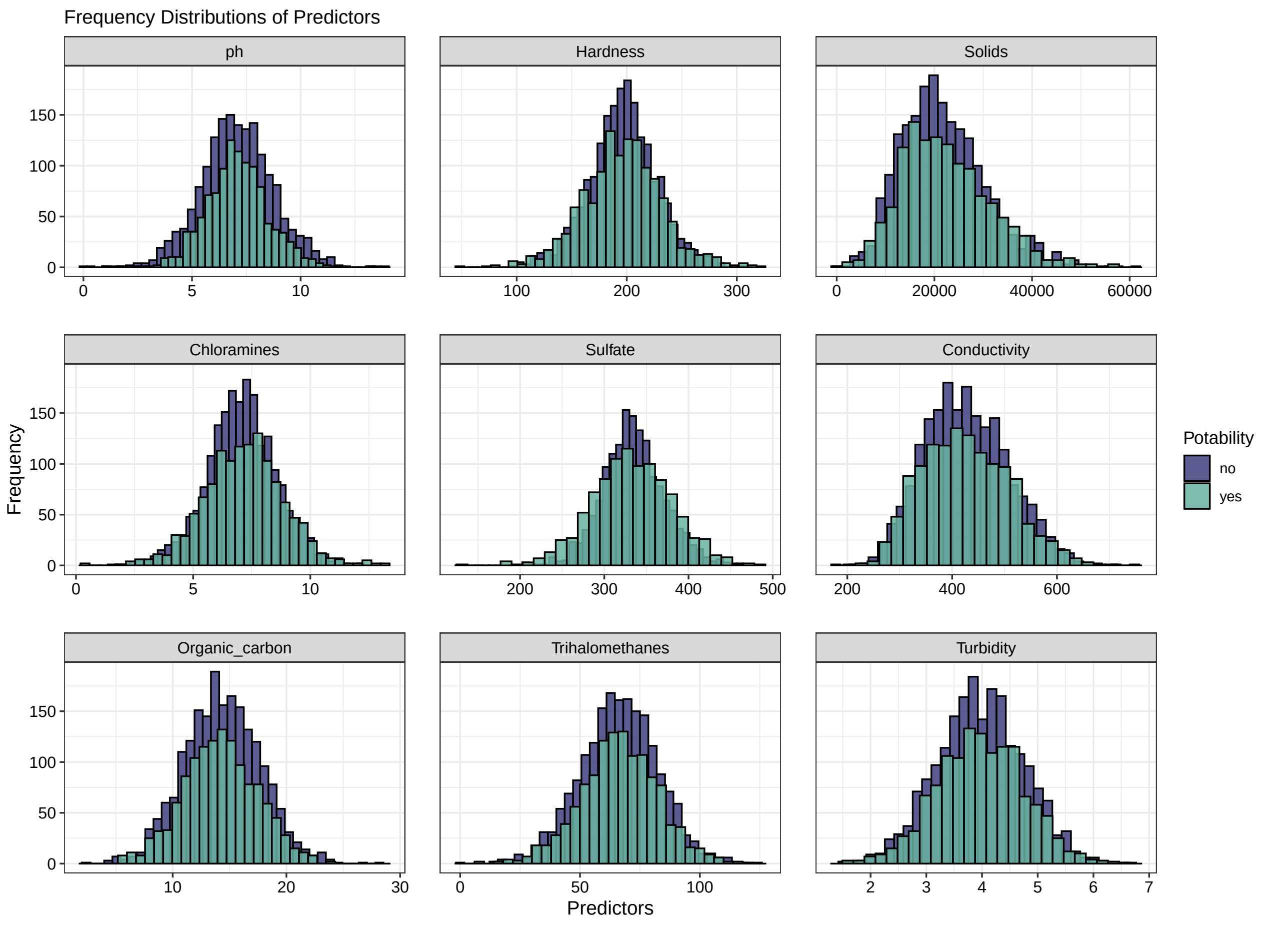
Table 6: Table continues below

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 :

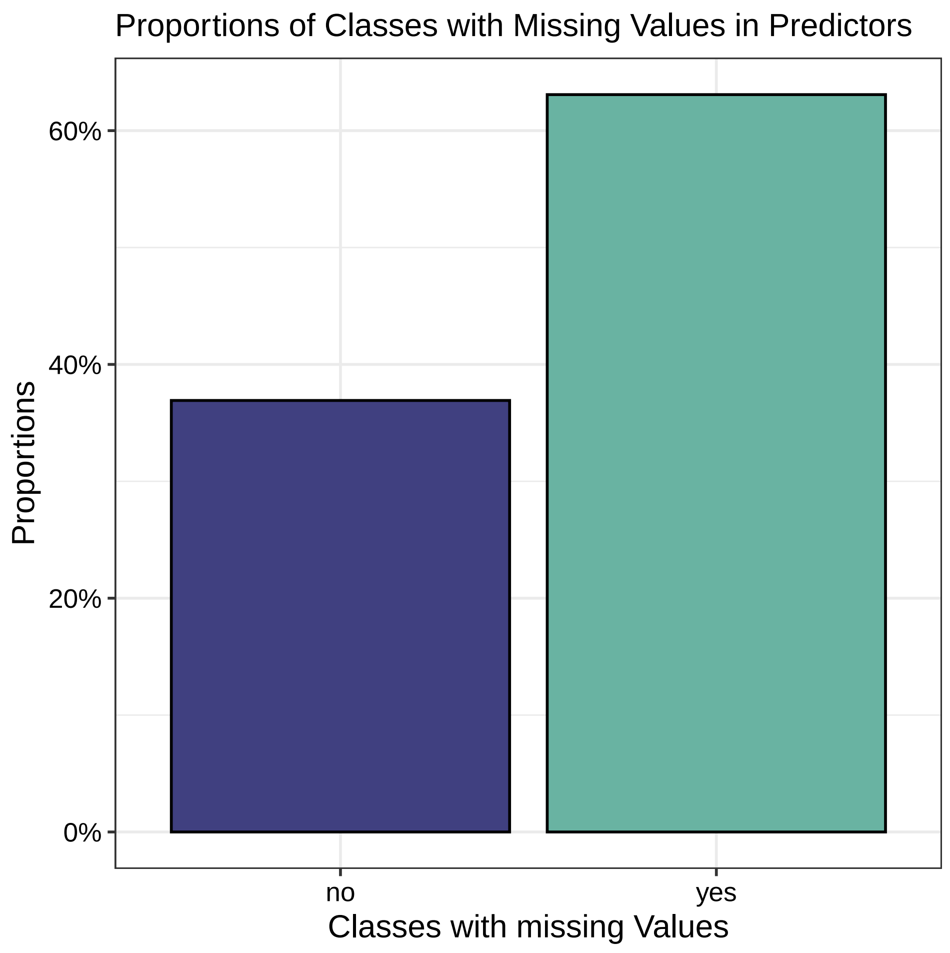


Table 8: Missing Values by Columns

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:

```r
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

```r
# 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

```r
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")
```

```r
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)
```

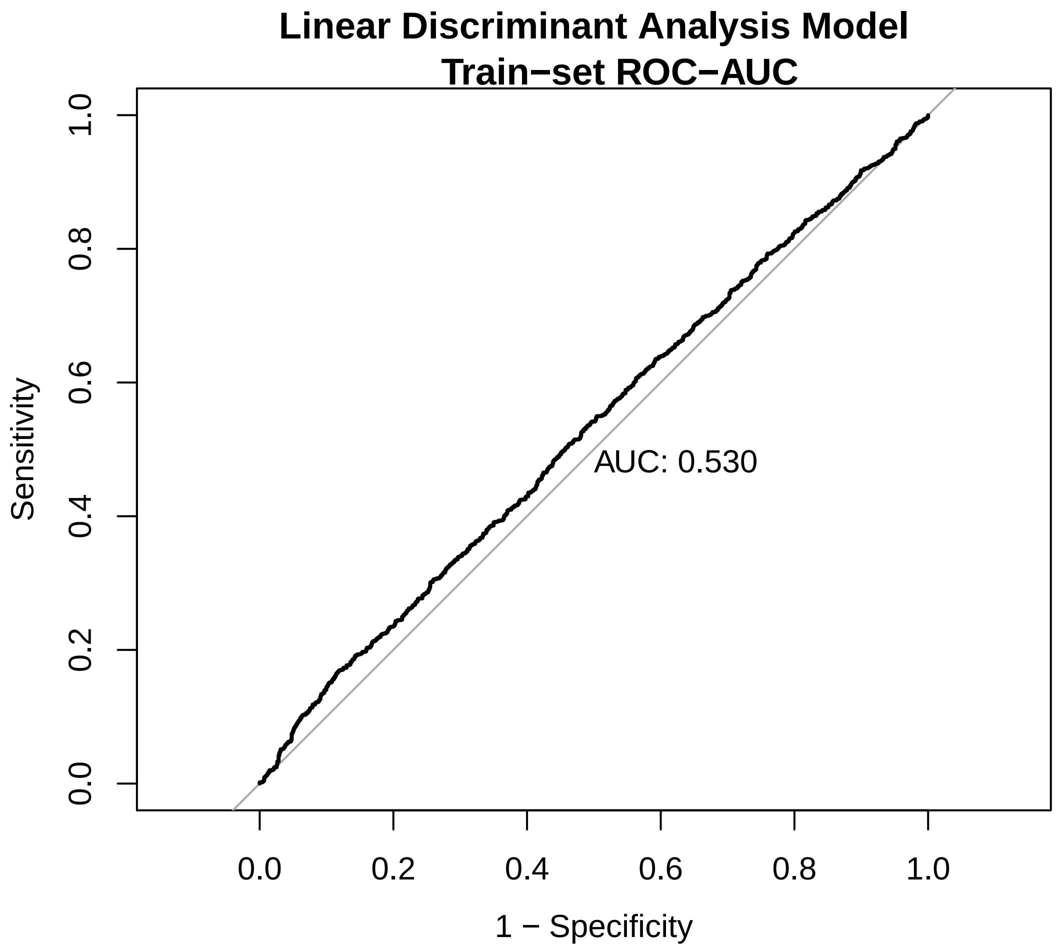


Table 12: LDA Model - Confusion Matrix

	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)
```

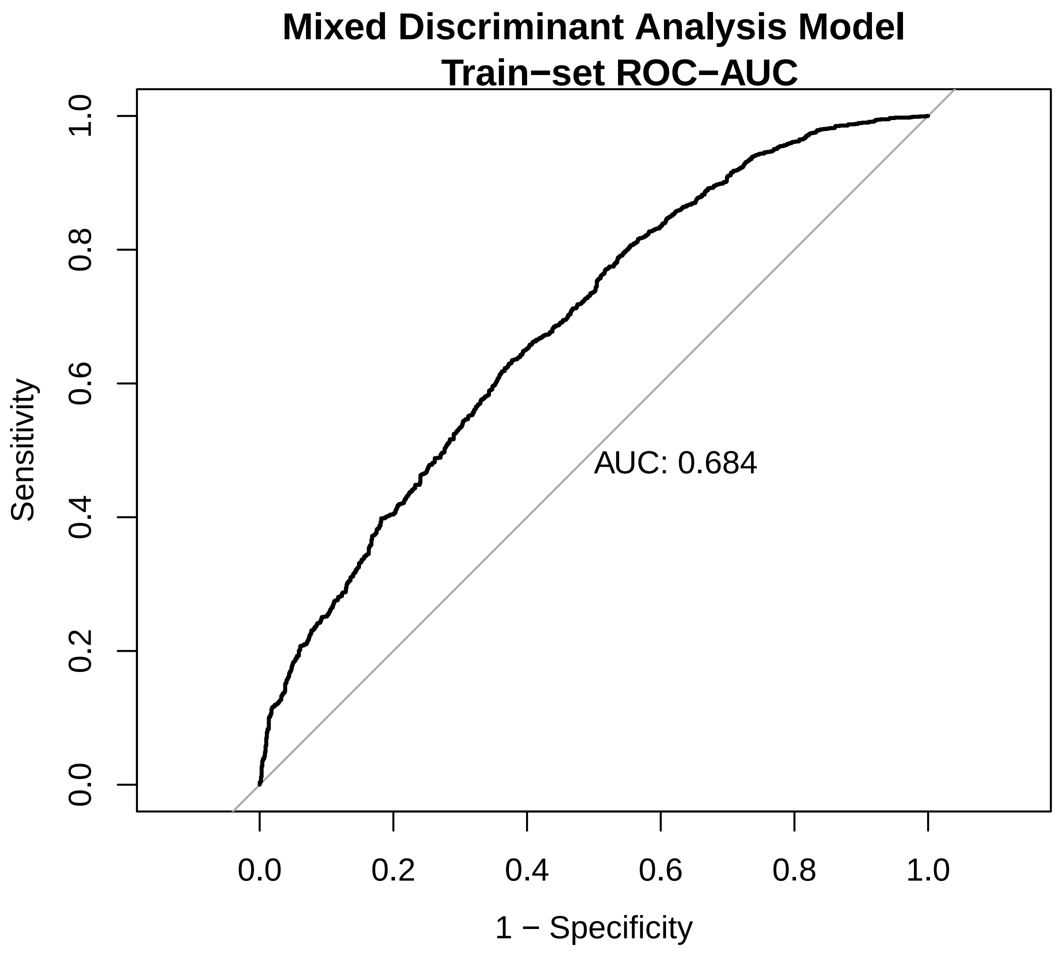


Table 14: MDA Model - Confusion Matrix

	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)
```

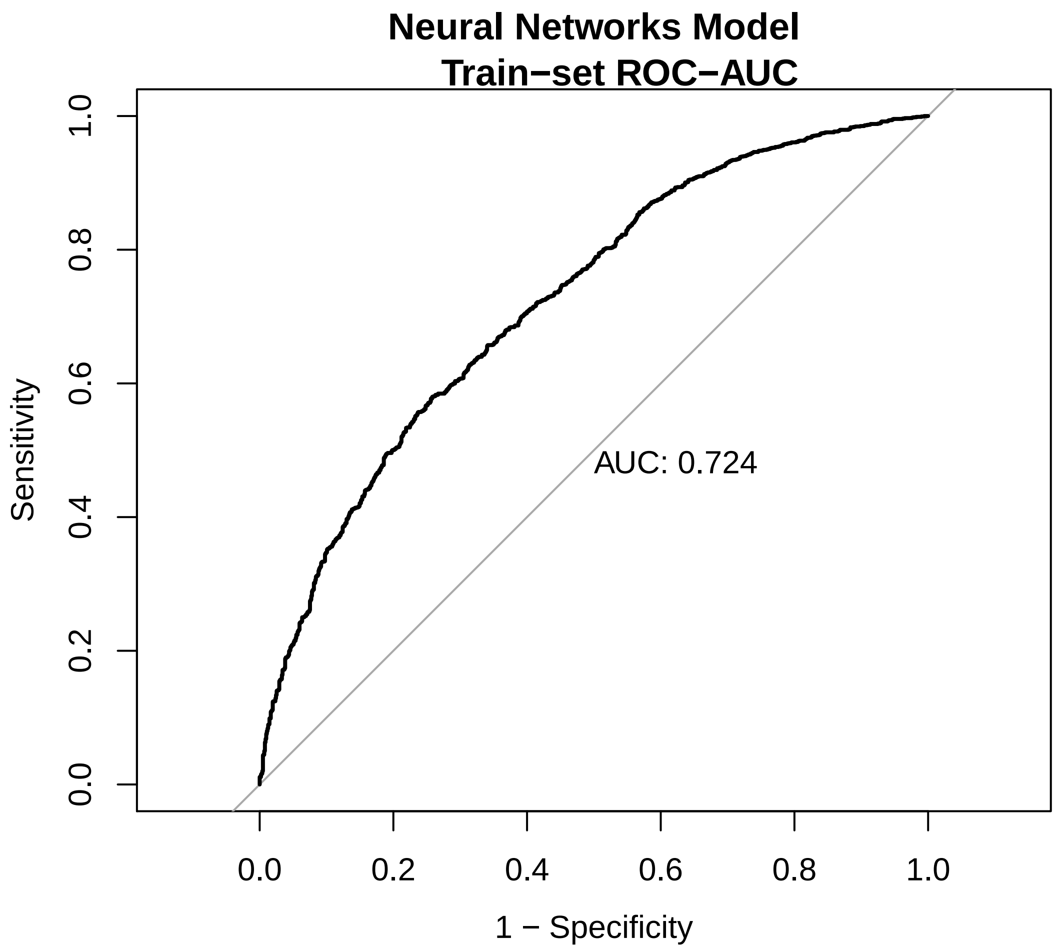


Table 16: Neural Networks Model - Confusion Matrix

	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

```r
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")
```

```r
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)
```

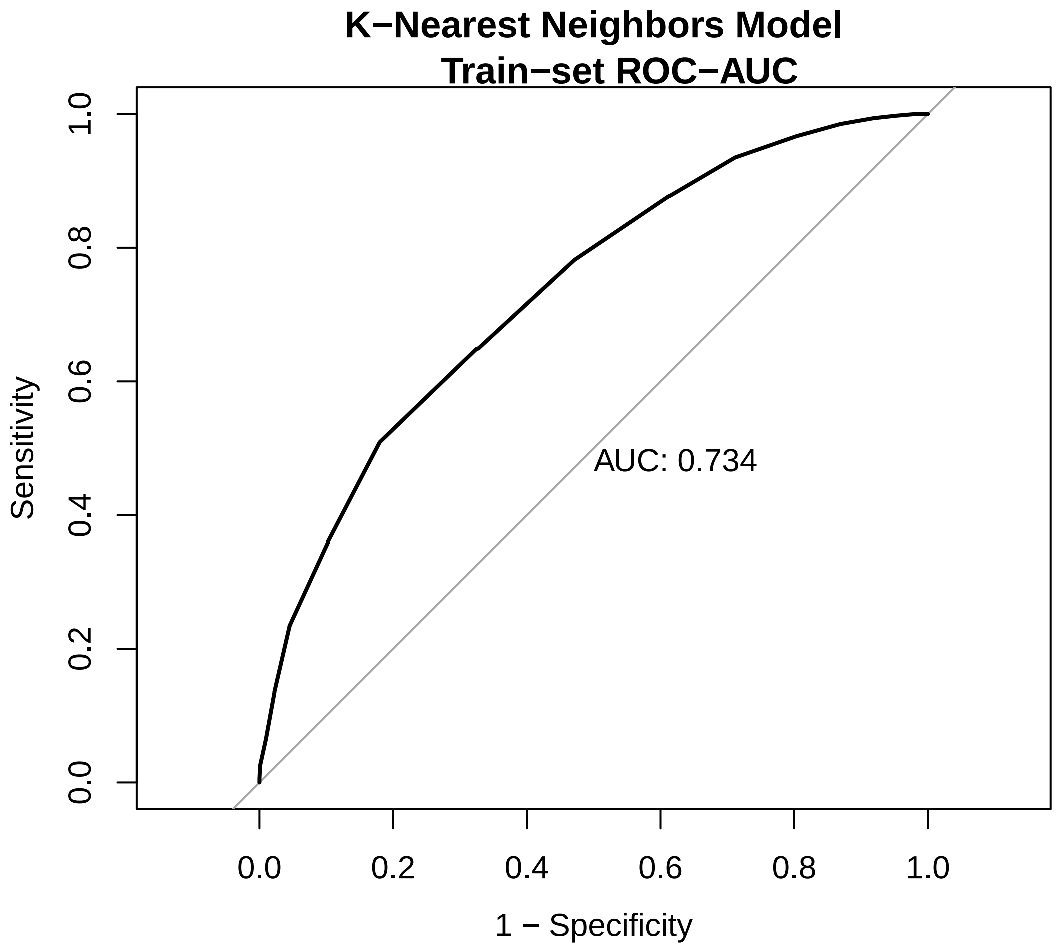


Table 18: KNN Model - Confusion Matrix

	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

```
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")
```

```
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)
```

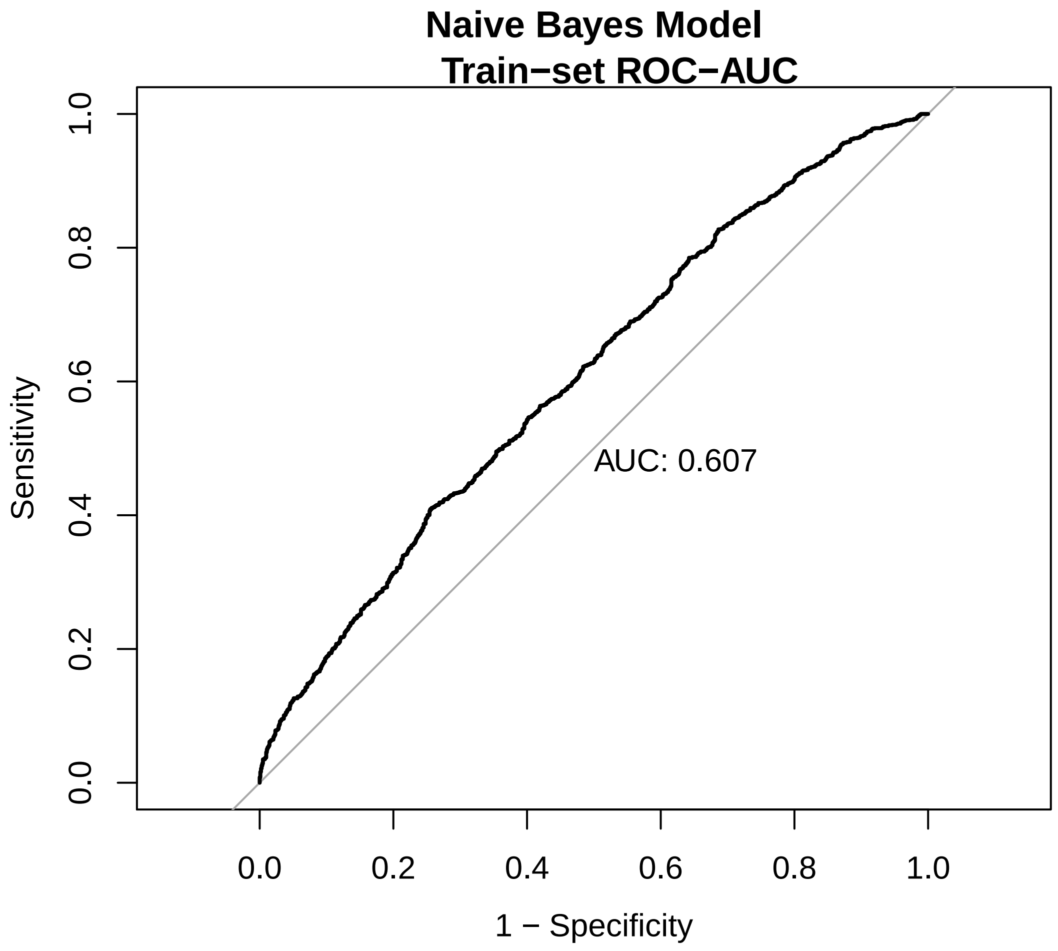


Table 20: Naive Bayes Model - Confusion Matrix

	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)
```

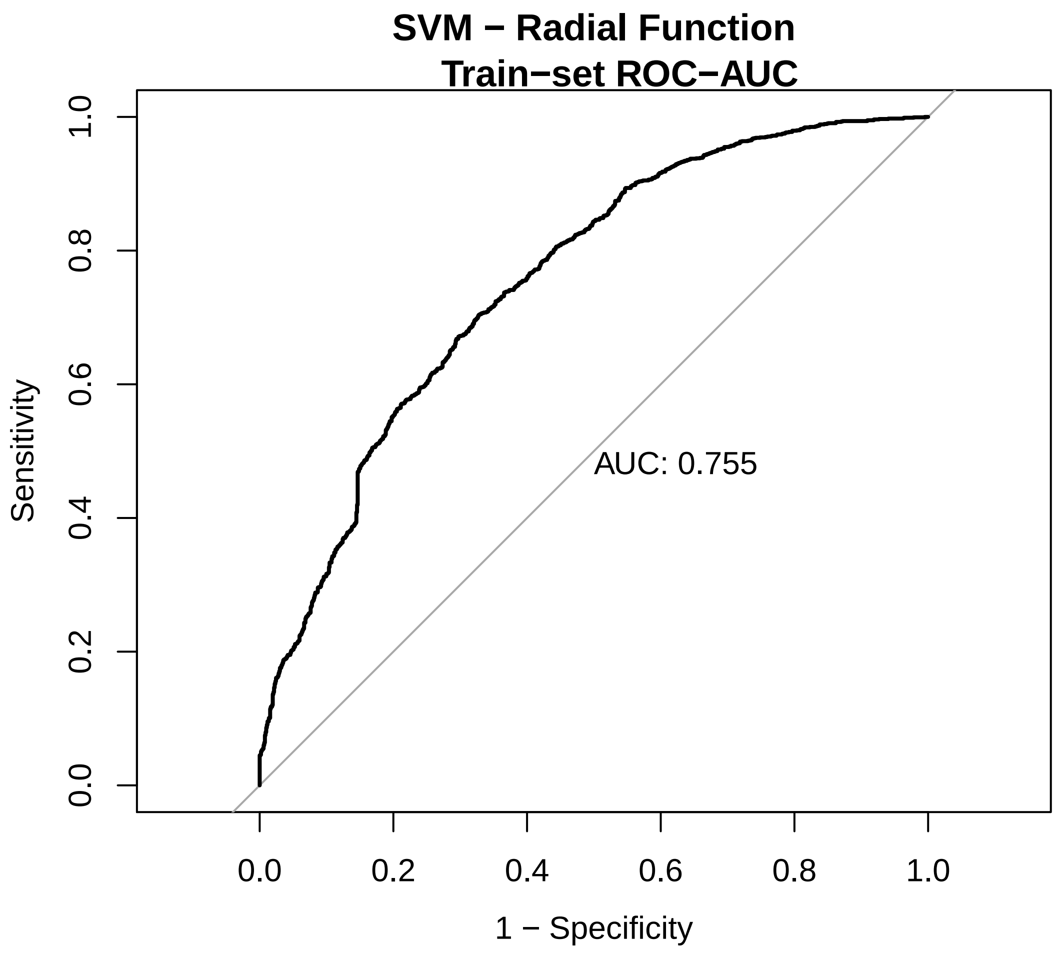


Table 22: SVM Radial Function - Confusion Matrix

	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

```
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")
```

```
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)
```

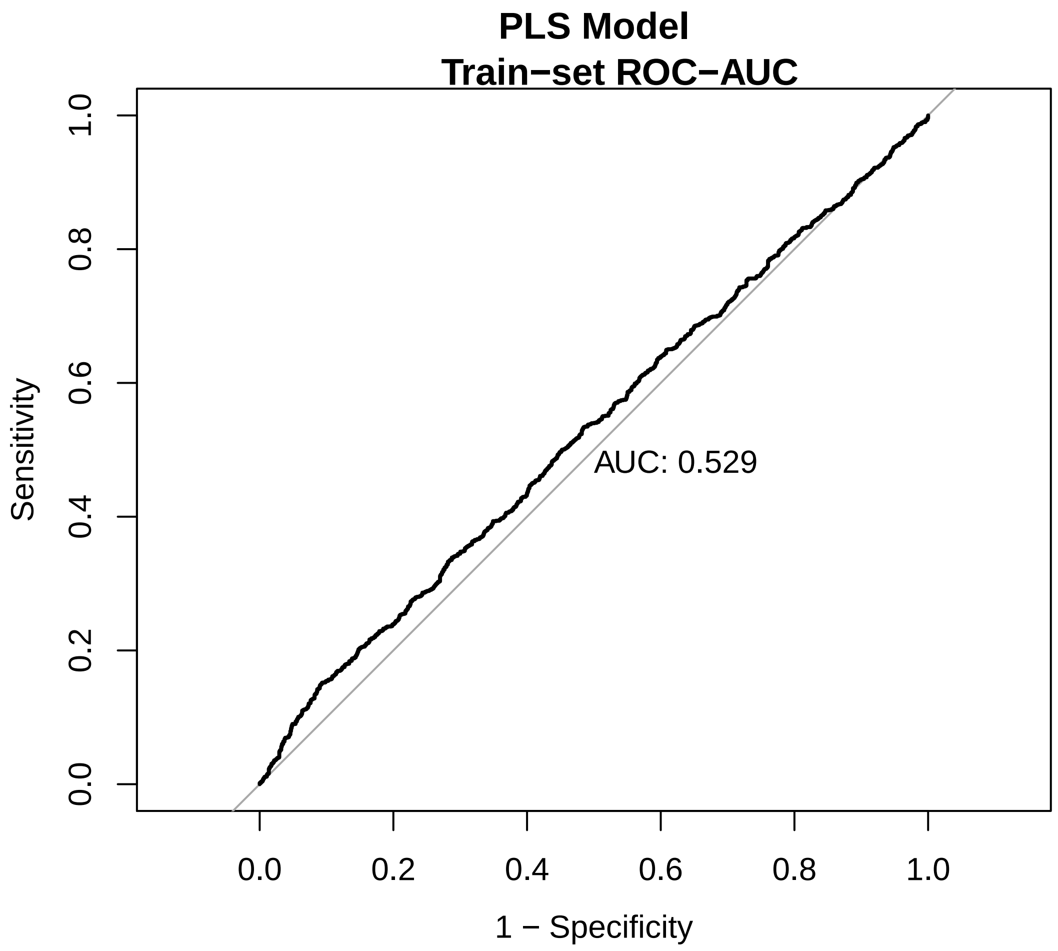


Table 24: PLS Model - Confusion Matrix

	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

```
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")
```

```
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)
```

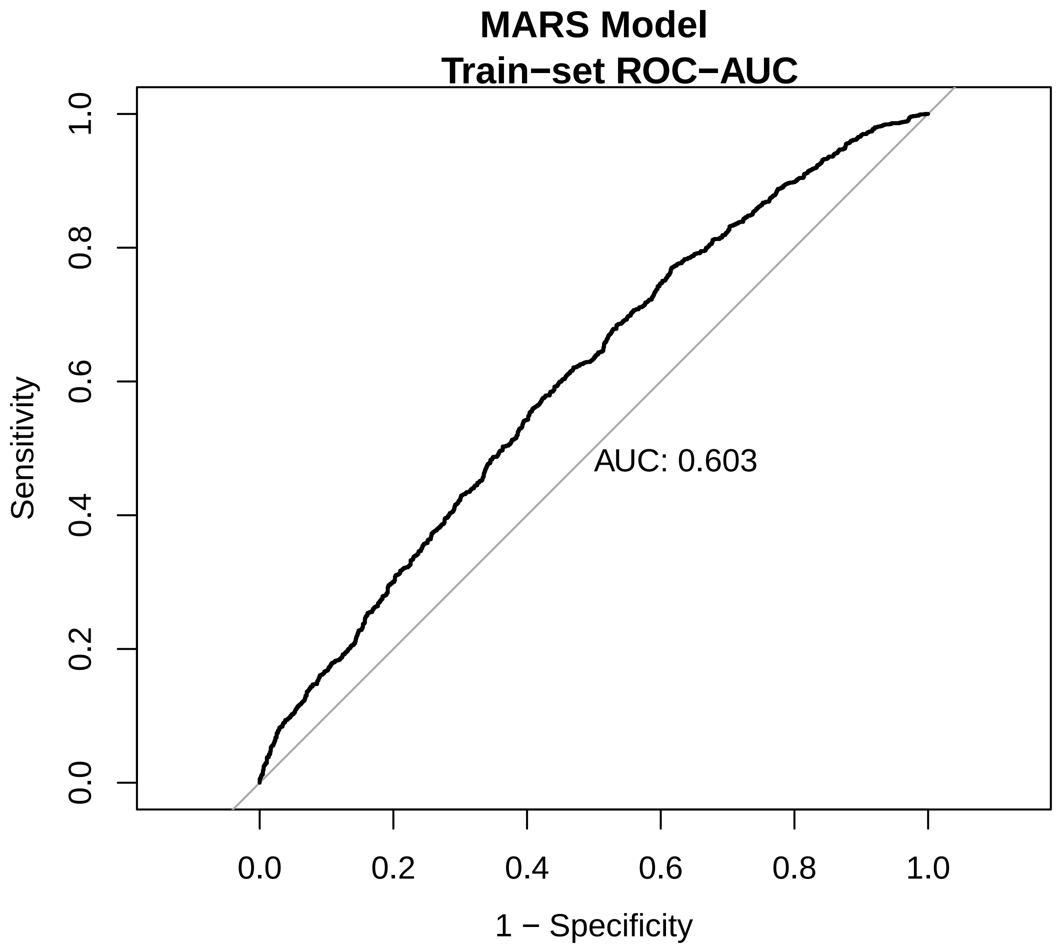


Table 26: MARS Model - Confusion Matrix

	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

```r
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")
```

```r
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)
```

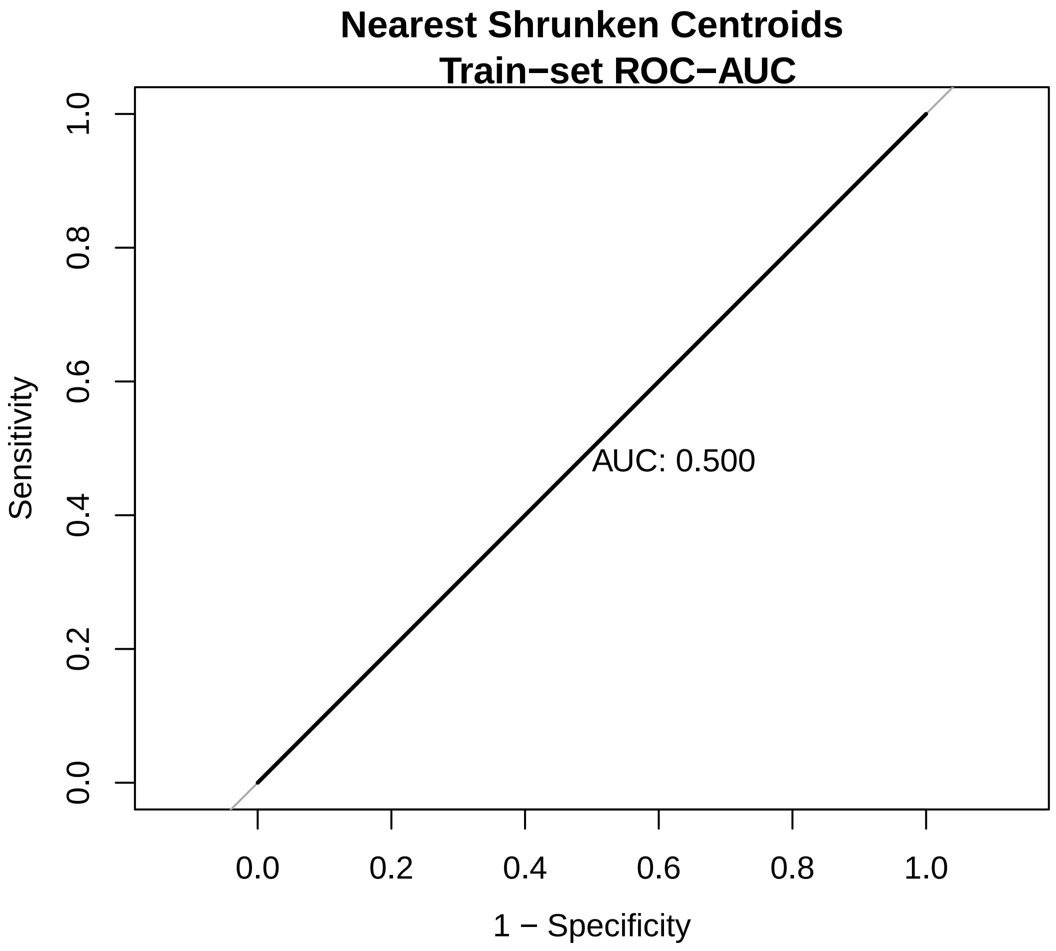


Table 28: Nearest Shrunken Centroids - Confusion Matrix

	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

```r
set.seed(476)

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

```r
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)
```

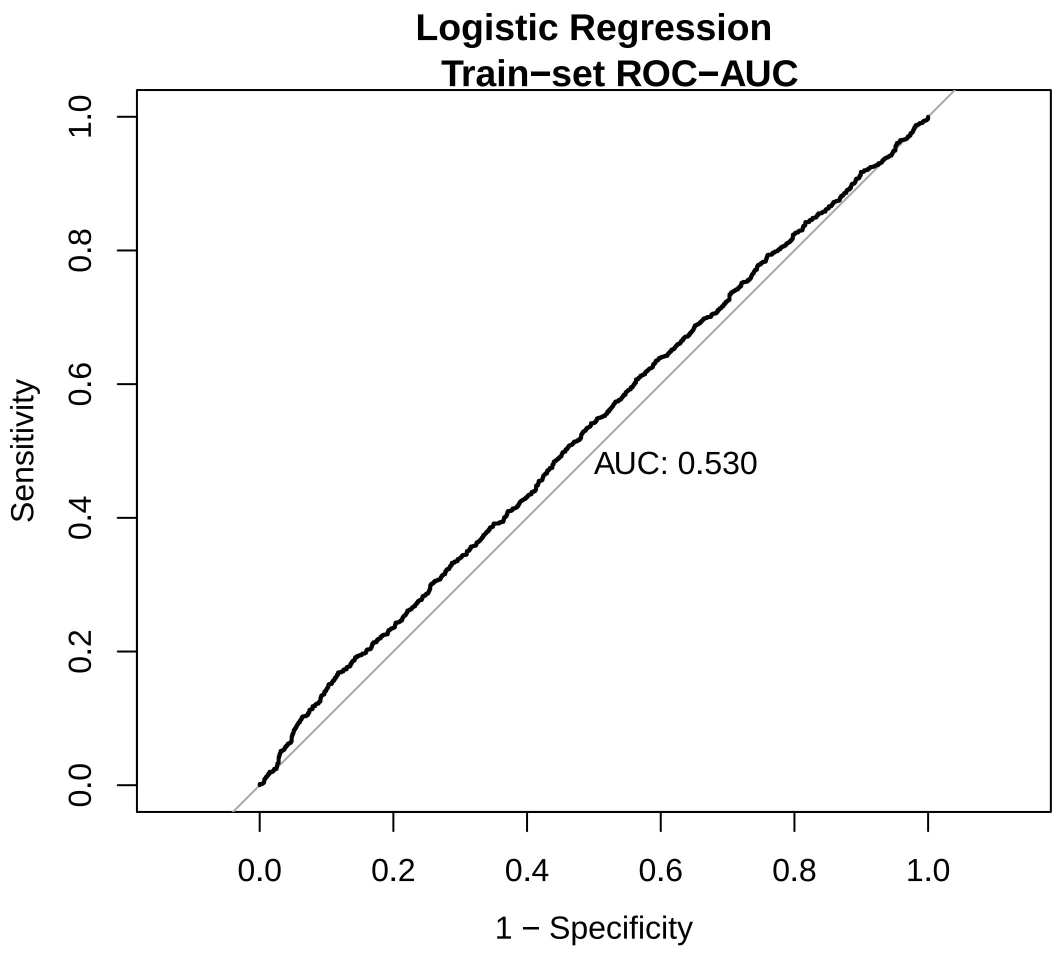


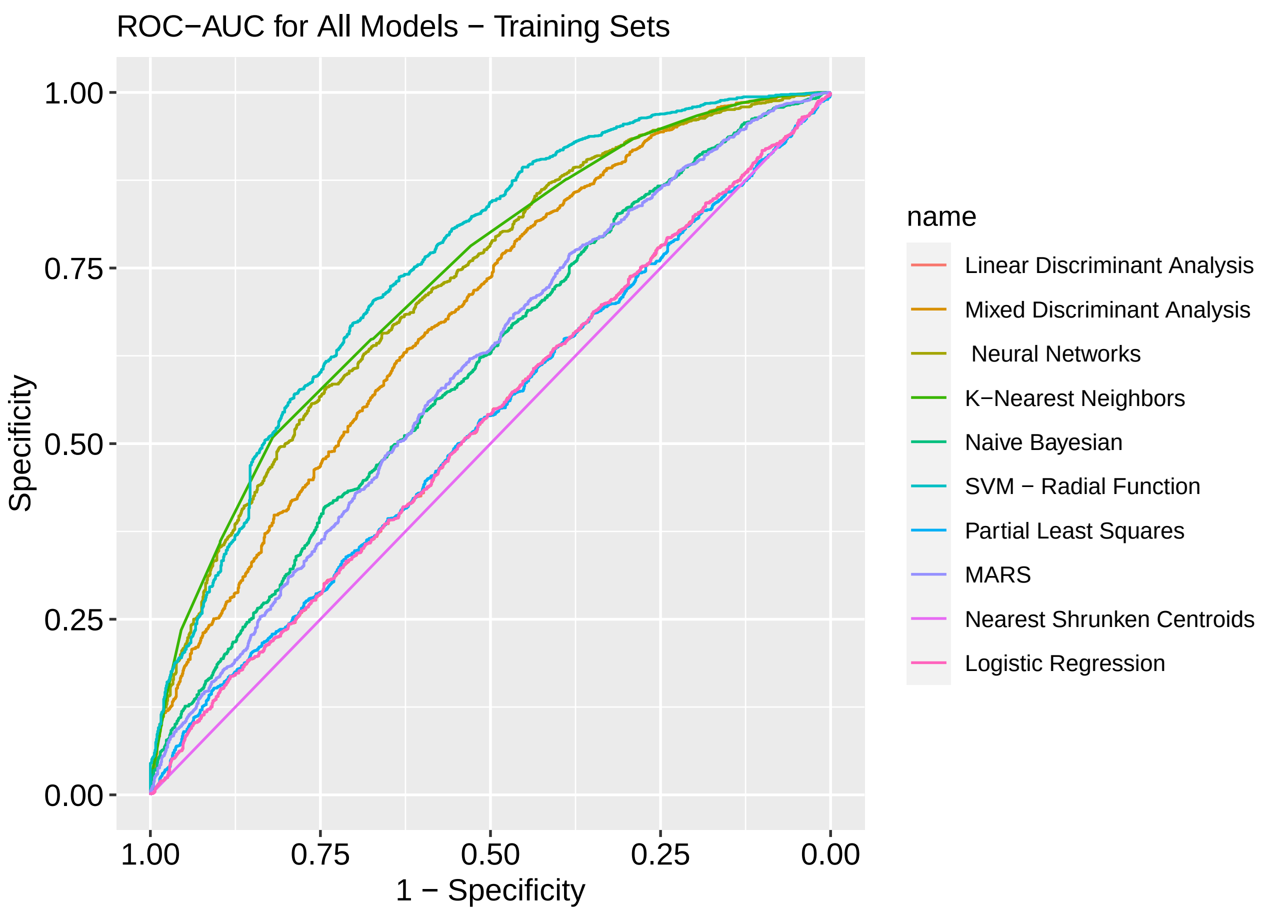
Table 30: Logistic Regression - Confusion Matrix

	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:

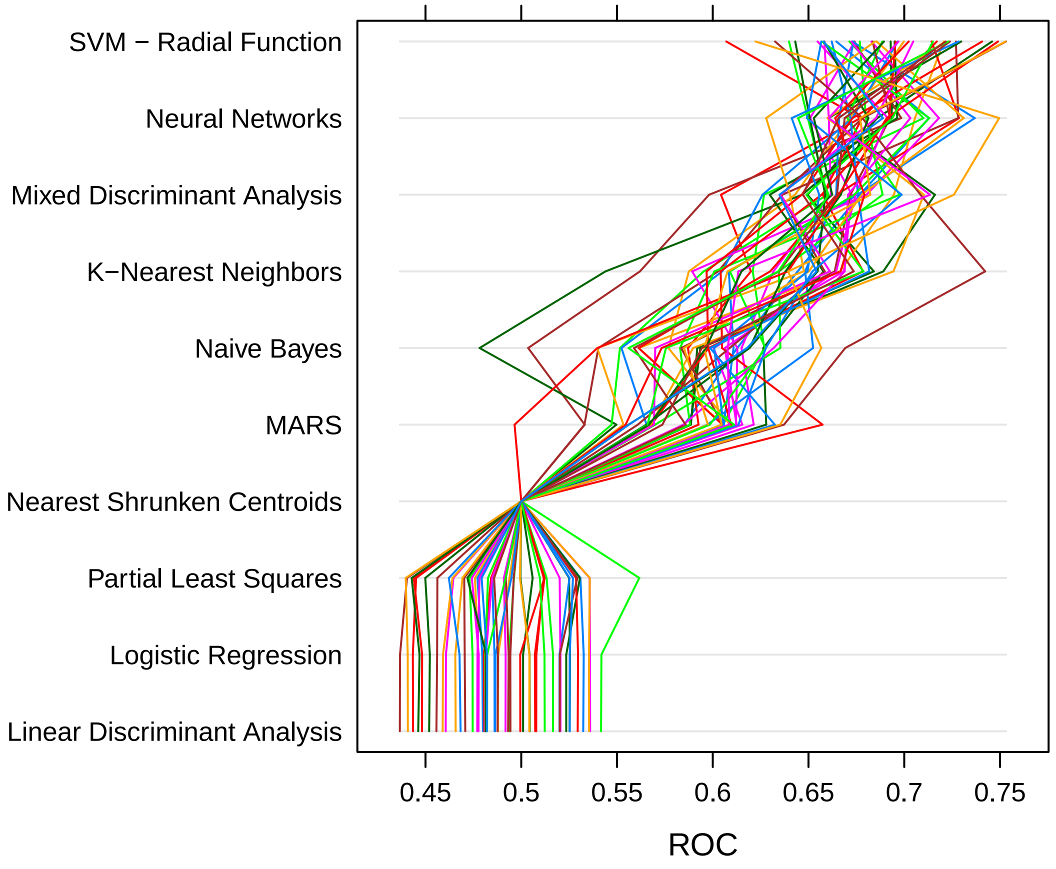


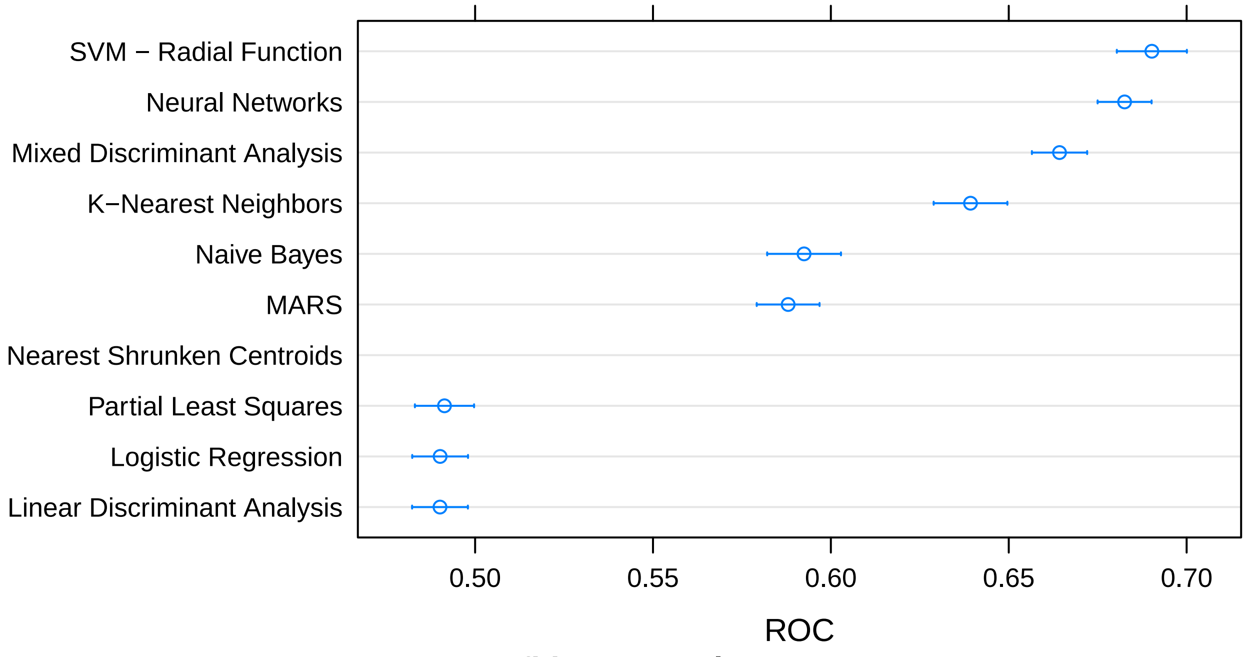
All Models AUC Values:

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





Confidence Level: 0.95

Results

SVM Radial Function - Best Tuning Parameters

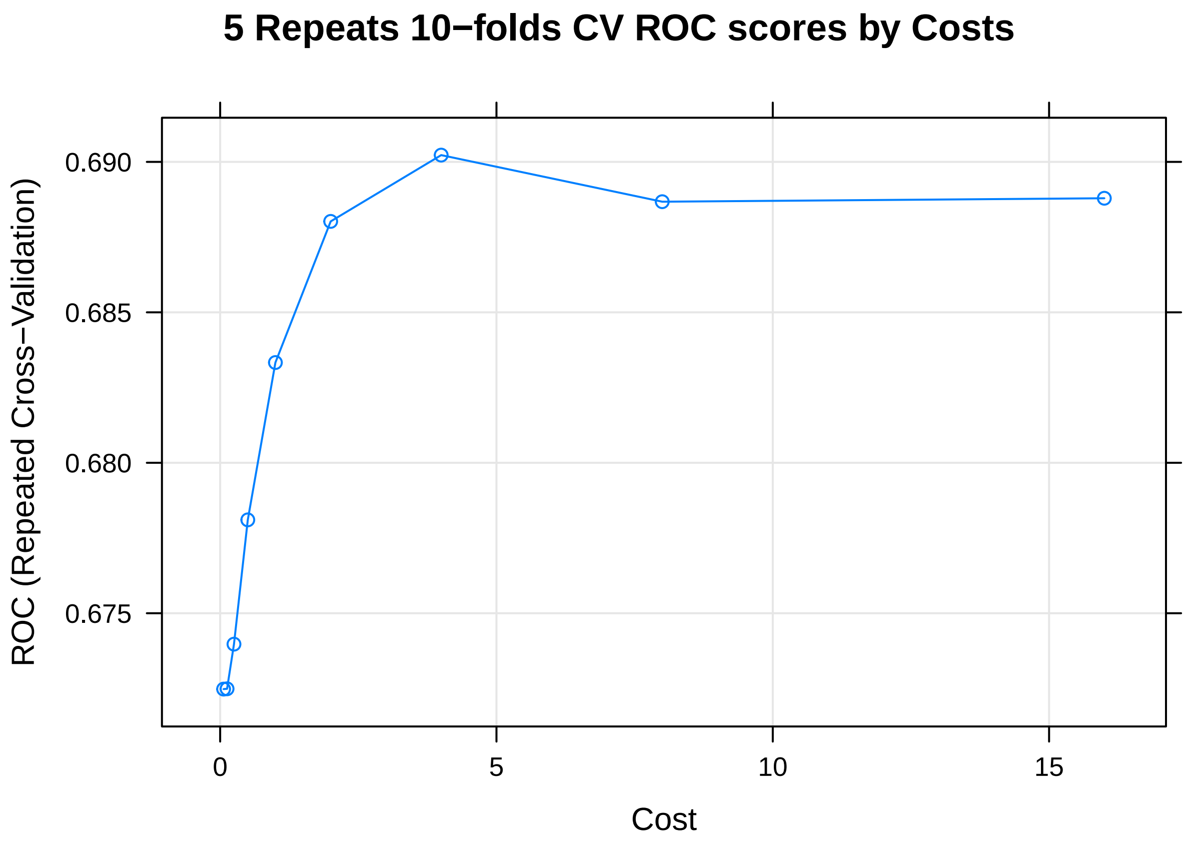


Table 33: Best Tuning Parameter based on ROC values

	sigma	C
7	0.03307	4

Neural Networks - Best Tuning Parameters

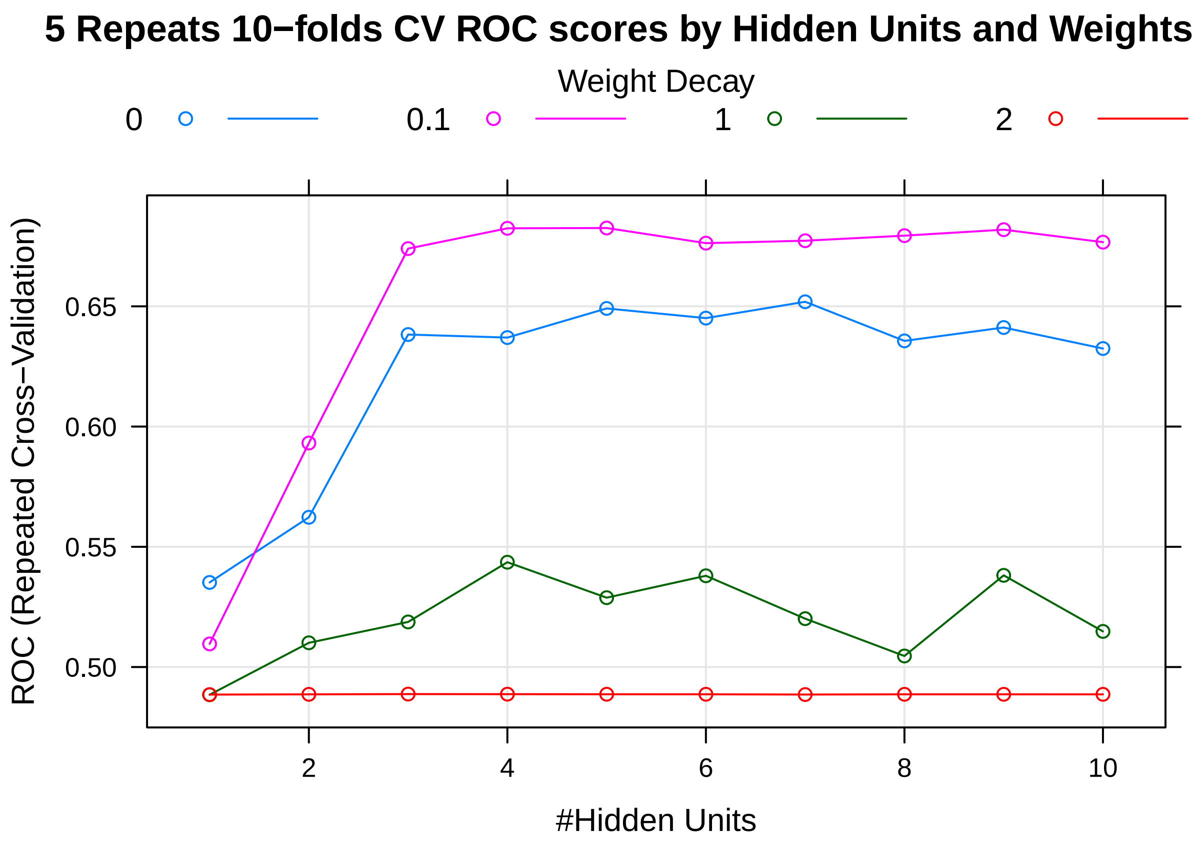
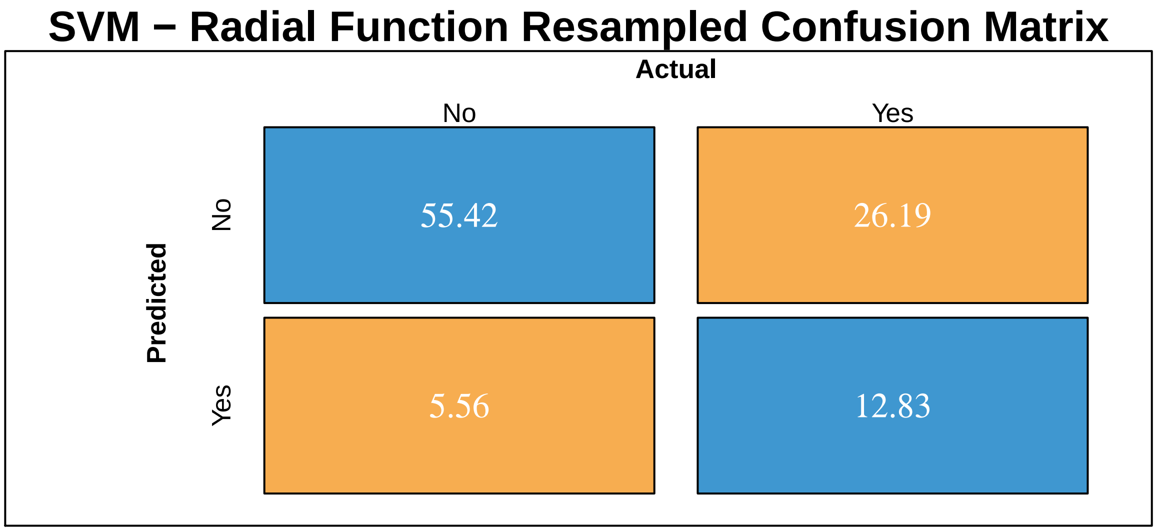


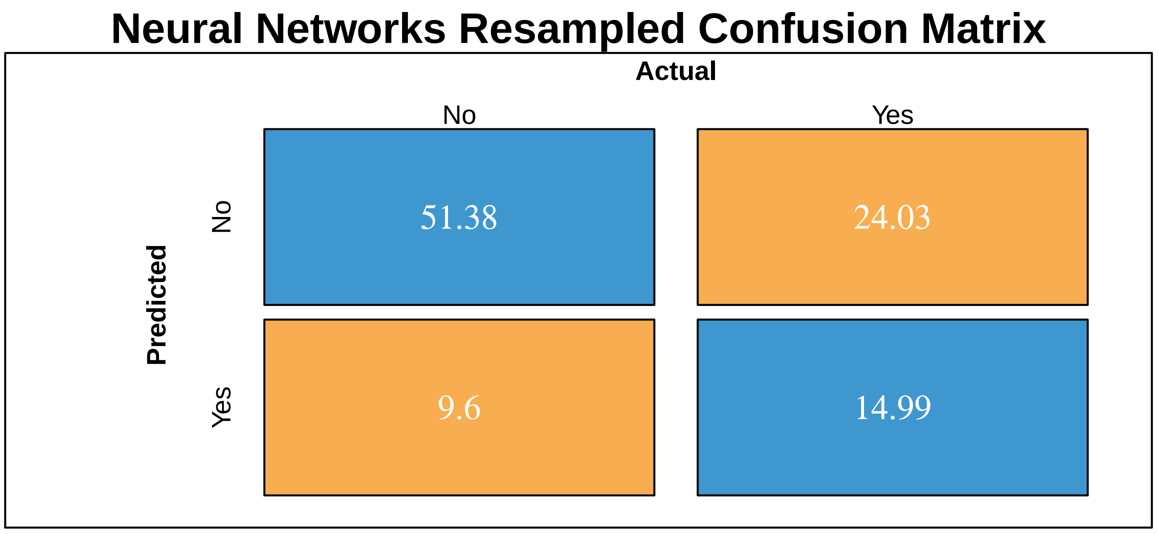
Table 34: Best Tuning Parameter based on ROC values

	size	decay
18	5	0.1

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



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

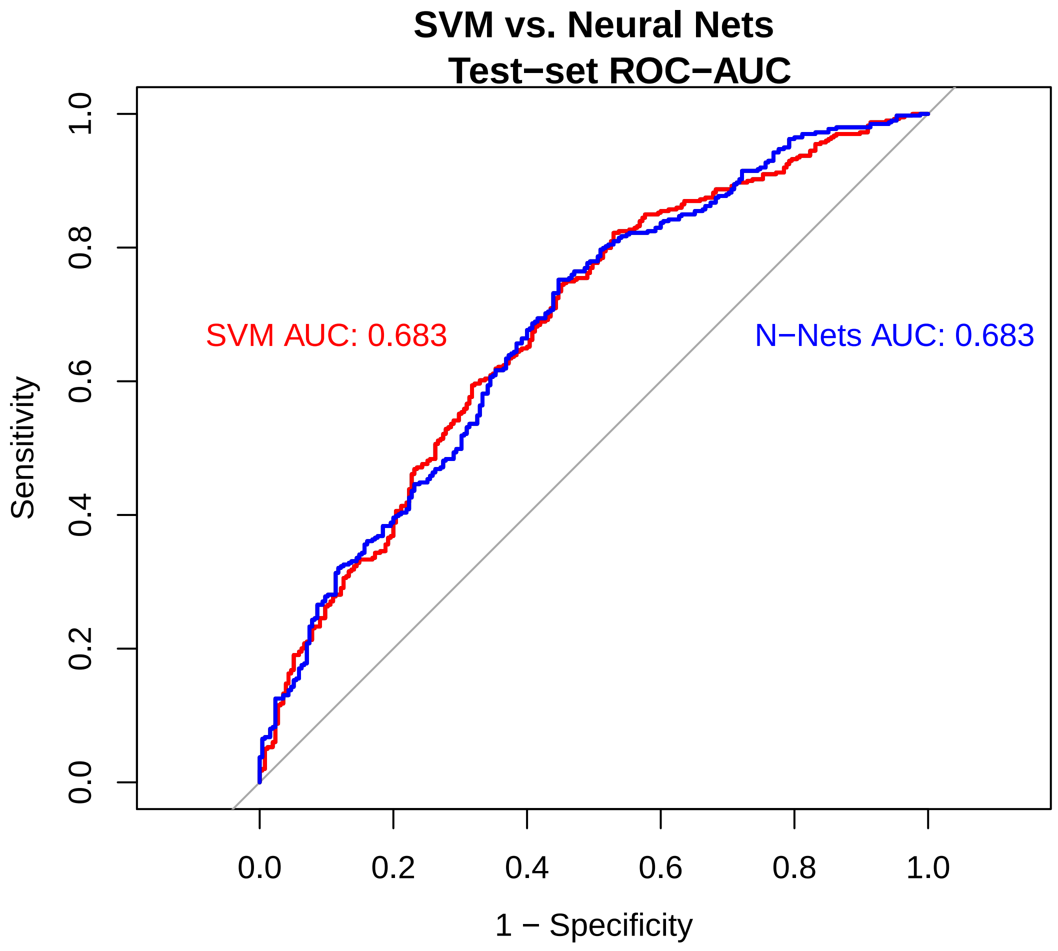


Test Sets - SVM vs. Neural Networks

```
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



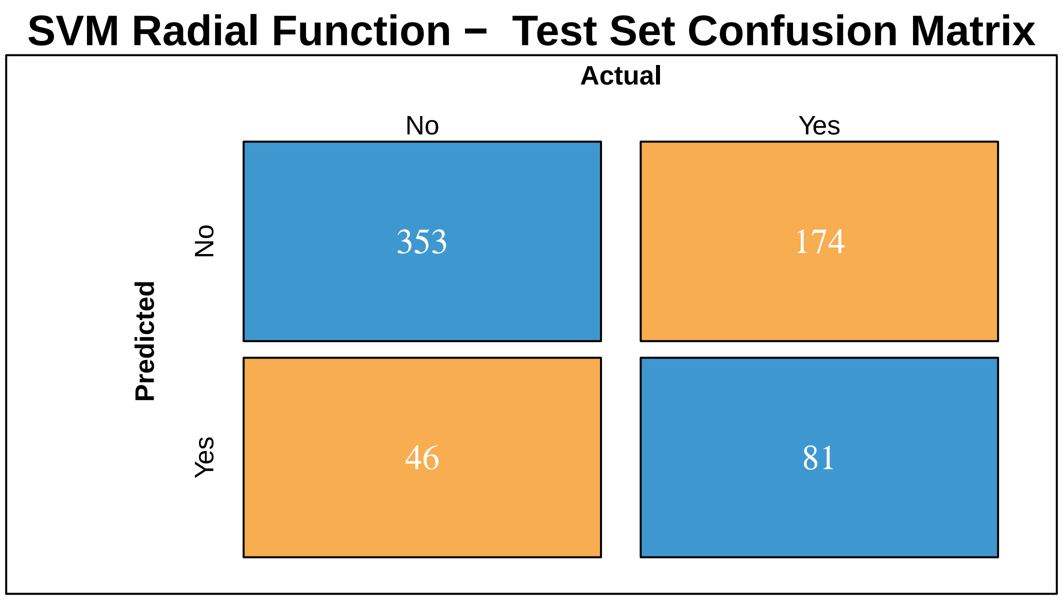


Table 35: SVM - Test Set Performance

Performance	model
Area Under Curve	0.6831
Sensitivity	0.3176
Specificity	0.8847

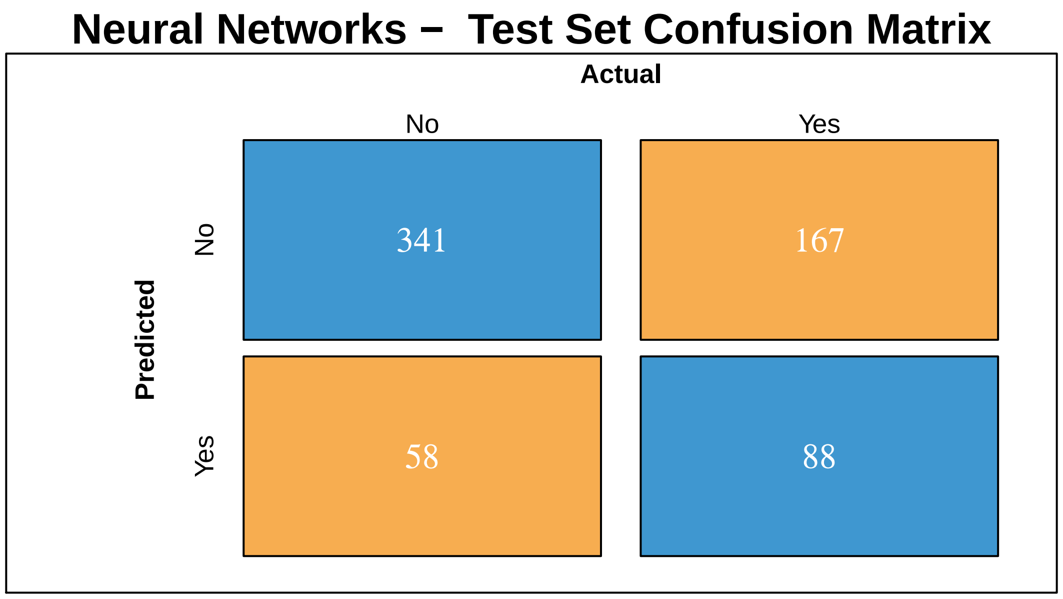


Table 36: N-Nets - Test Set Performance

Performance	model
Area Under Curve	0.6833
Sensitivity	0.3451
Specificity	0.8546