**Water Potability Investigation**

**Abby Lavoie**

**M448, Spring 2023**

**Abstract**

To label water as potable means it is safe for human consumption, bathing, and cooking. All of these uses of water can transmit disease and cause short-term or long-term illness and even death if the water is not safe. This means it is of utmost importance to be able to predict if water is safe or not. The current data set, taken from Kaggle, involves 9 physiochemical predictors and a binary outcome of potable or not. Multiple classification methods were applied to this data and compared for both variable importance and misclassification test error. Of these methods, the support vector machine with a polynomial kernel and degree 2 performed the best. Linear models performed the worst. Evidence piled up to cast doubt on the realness of this data set and in fact, it was concluded to be poorly manufactured. In the future a dataset gathered from real chemical analysis of natural bodies of water, wells, and rainwater collection would be sought to perform an investigation with real-world applications and meaning. In the future, the results of such an investigation could be used to help refine current at-home water tests which are difficult to use reliably.

**Introduction**

Potability refers to the safety of water for humans to drink, bathe in, and cook with. Non-potable water can cause immediate reactions like mouth burning from chemicals being in unsafe amounts, short-term disease like diarrhea due to the presence of coliform bacteria, long term disease like skin reactions, cholera, cancer due to pollutants like agricultural runoff, poisoning due to the presence of heavy metals, and even death (especially for infants and immunocompromised individuals). Many natural sources of water, including recycled water, contain microorganisms, toxic chemicals, and/or fecal matter. Even without these obvious sources that make water dangerous, even the basic natural chemical parts of water can harm human health.

In many parts of the world, such as island nations, regions prone to drought, or countries without maintained water infrastructure, access to clean water is a major issue. Even locally, those reliant on wells and recycled water can be exposed to unsafe water. It is important to have an affordable, at-home (or at the local government level in some places) water safety test. The current tests on the market are easy to misuse which easily causes a misunderstanding of the potability of the water source being tested. These tests seem to require a laboratory chemist to perform reliably which defeats the purpose of begin an at-home test. It is important to keep refining and investigating in order to provide a better, easier test to use.

Many of the chemical compounds found in natural water sources come from dissolved organic matter (for example, plants, animals, and animal waste), dissolved rocks, and soil. In places near agriculture the water can contain runoff that includes fertilizer, which is usually rich with nitrates and phosphates, and pesticides. Near populated areas the water sources can contain human fecal matter. Human activity can also cause an increase in salts and minerals. In times of drought, there is less water to buffer against these sources which naturally results in an increased concentration; the effect of this can range from unpleasant taste to harmful to human health. Measures of these different chemicals could provide rich information about the water being tested and the safety. Are there certain chemical measures that can predict potability reliably? If so, then these can the focus for new at-home tests. How reliably can physiochemical measures predict potability?

The water potability data set used in this investigation came from an unknown publisher on the Kaggle website. Unfortunately, with the information provided on the site, it seems to be synthetic rather than real data. Some discussion on the site suggests that the numbers provided are not plausible for real-life scenarios. These aspects will be pointed out in the following discussion. One of the variables very important to test for determining potability, coliform bacteria test, is missing from this data set. However, the data still provide a good opportunity to practice analytical methods in an attempt to answer the questions above. Moreover, the topic of the data set is an important one to shine light on. It seems especially important given the recent drought conditions in California and the search for methods of recycling water.

**Data Description**

This data set contains nine different continuous, quantitative physiochemical measures of water as predictors and a binary outcome of whether the water is potable or not. The type of problem presented by these data is one of classification. The predictor names are: pH, hardness, solids, chloramines, sulfate, conductivity, organic carbon, trihalomethanes, and turbidity (please see the table below for a description of these variables). There are 3276 observations with each indicating a different body of water and 1265 observations missing a value for at least one predictor. The only predictors with missing values are pH, Sulfate, and Trihalomethanes. If the water is potable then the outcome is listed as “1” and otherwise is “0”. The proportion of 1’s in this data set is 0.39 and 0’s is 0.61 which is slightly imbalanced. Please see the table below for more information on the variables, their acceptable levels, and range in this dataset.

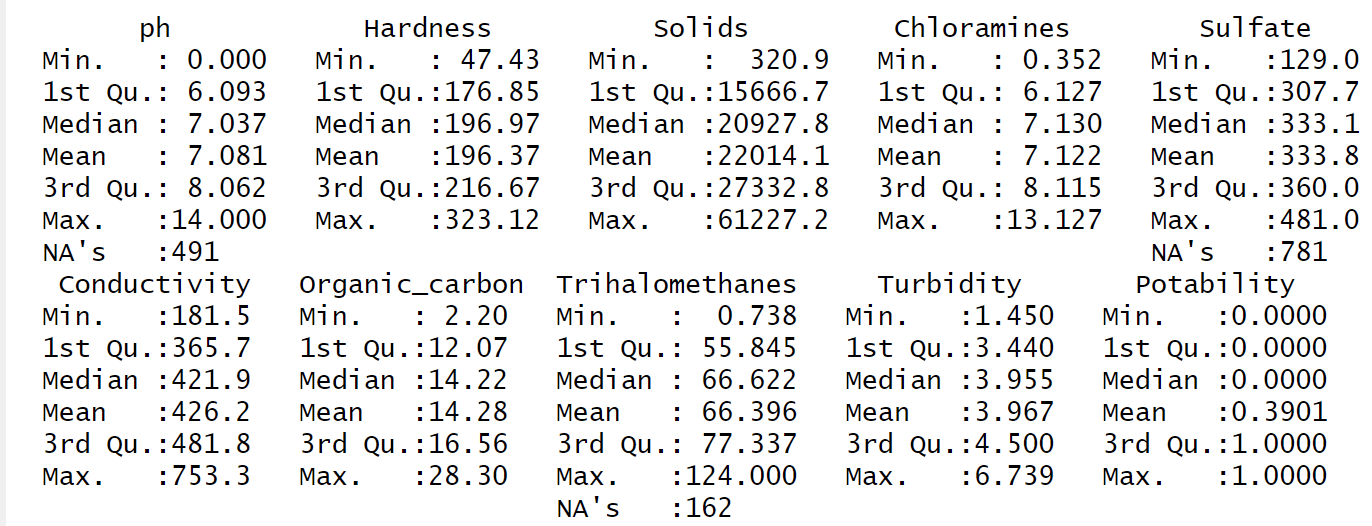
|  |  |  |  |
| --- | --- | --- | --- |
| **Water Potability Chemicals and Relevant Safety Guidelines** | | | |
| **Variable Name** | **Variable Description** | **Acceptable Levels** | **Range in Data** |
| pH | Acid/Base evaluator. | 6.5-8.5 (WHO) | 0-14 |
| Hardness | Measure of calcium and magnesium salts. Capacity of water to precipitate soap. | 60-120 mg/L (USGS) | 47.43-323.12 |
| Solids | Water dissolves many solids which can produce a bad taste and off color. | < 300 ppm (SDWF) | 320.9 – 61,227.2 |
| Chloramines | A major disinfectant in public water systems. Formed when ammonia is added. | ≤ 4 ppm (CDC) | 0.352-13.127 |
| Sulfate | Naturally occurring in minerals, rocks, and soil. Typical range is 3 to 30 mg/L. | < 250 mg/L (EPA) | 129 – 481 |
| Conductivity | Measure of ion concentration. Should not exceed 400 μS/cm. | < 1000 mS/cm (EPA) | 181.5 – 753.3 |
| Organic Carbon | Product of decaying organic matter. The EPA suggests <2mg/L in drinking water. | < 25 ppm | 2.20 – 28.30 |
| Trihalomethanes | Found in water treated with chlorine. | < 80 mcg/L (EPA) | 0.738 – 124 |
| Turbidity | Measure of solid matter in the water. Indicates quality of waste discharge. | < 5 NTU\* (WHO) | 1.450 – 6.739 |
| Potability | Indicator of whether water is safe to drink or not by humans. |  | 0 = Non-potable  1 = Potable |

\* NTU = Nephelometric Turbidity Units

To reiterate, since the outcome is categorical, the focus is on applying classification methods for the purposes of inference and prediction.

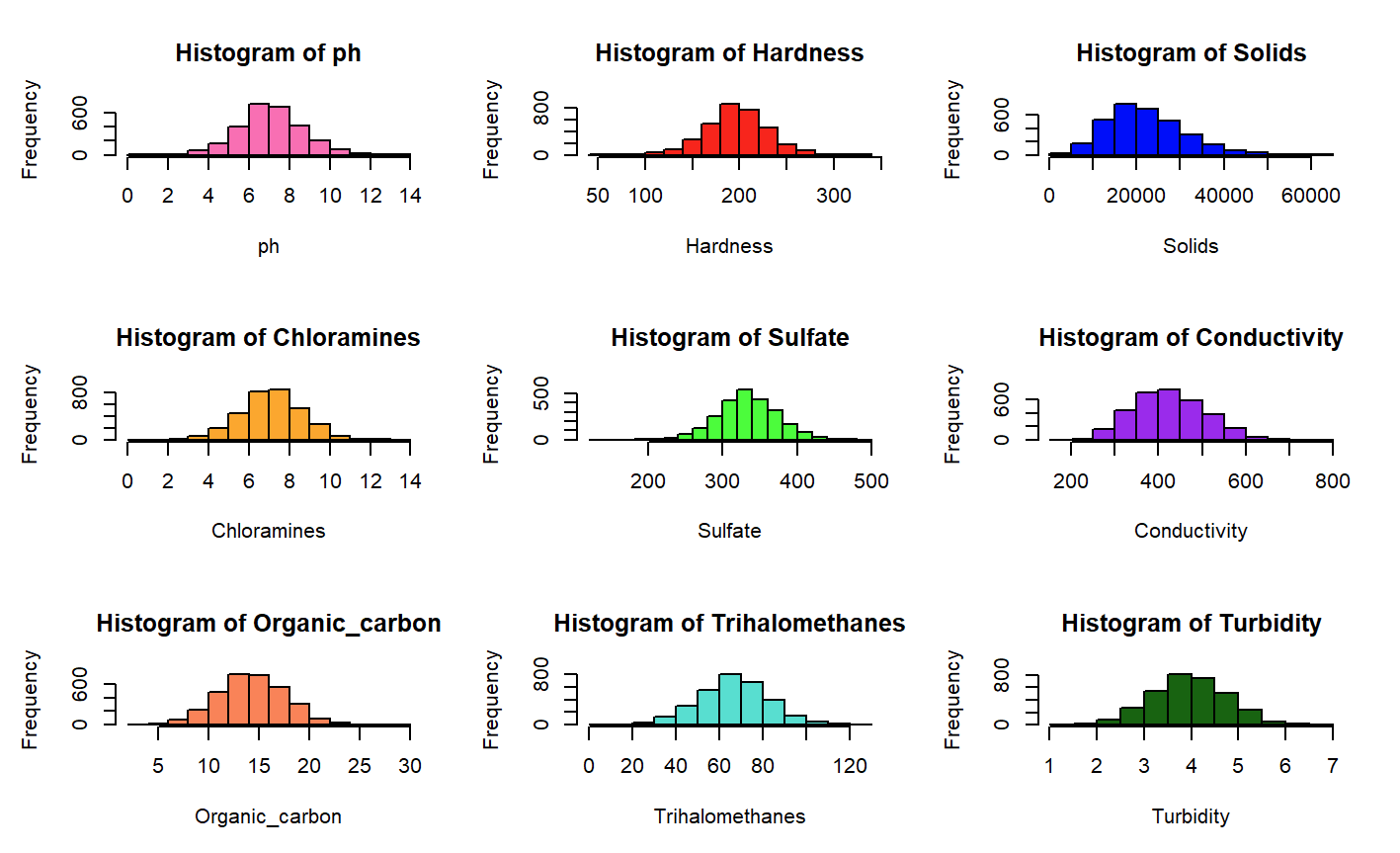
**Data Exploration: Summary, Visualization, and Observations**

The summary of the data:

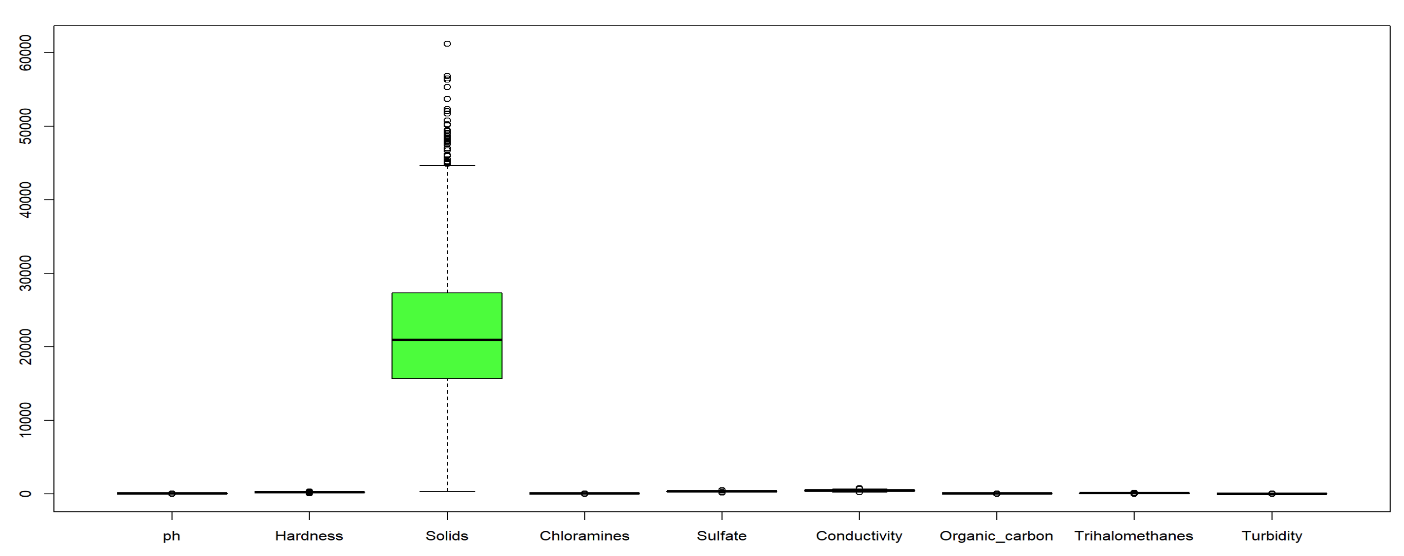


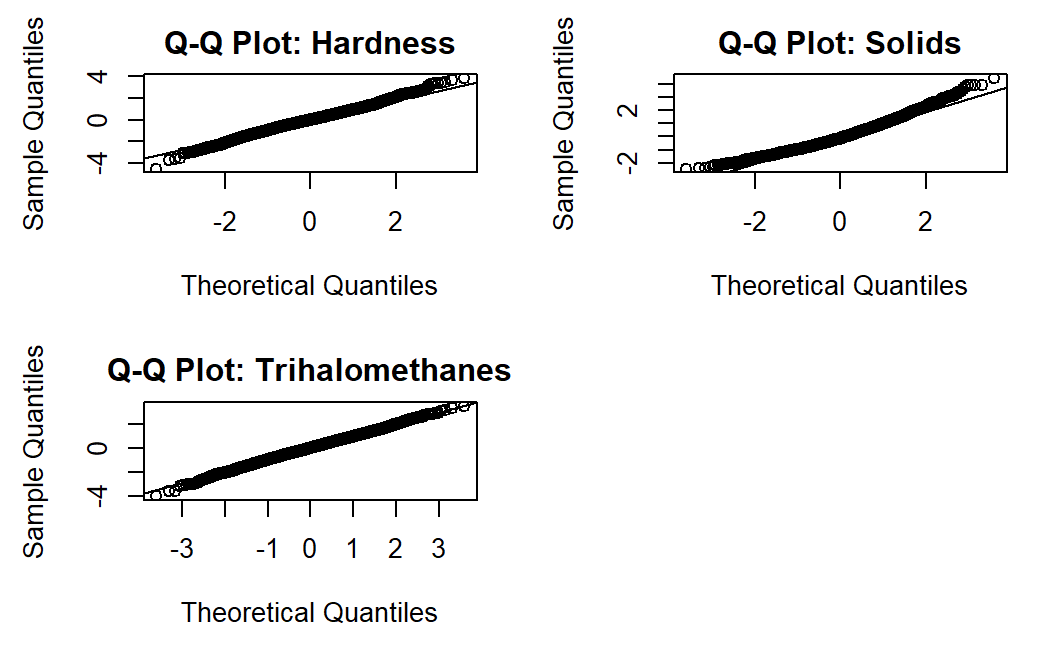
Without affecting the data as imported into R, we can see that the outcome is labeled as numerical rather than a factor. My first step of preparation is to change the label using as.factor(). After using this command and rerunning the summary function, Potability is properly labeled as a categorical outcome with 1278 1’s (potable bodies of water) and 1998 0’s (non-potable bodies of water). The above summary shows the difference in scale of the different predictors, especially how much larger the measurements for the Solids variable are than the rest.

The normality of the variables must be checked as normality is an underlying assumption of some methods. As seen in the histograms below, the variables are all dclose to normally distributed with the appearance of only very slight skew to hardness, solids, and trihalomethanes. Using the skewness() function in R resulted in values between -1 and 1 for all variables which can be interpreted as only moderate skew at the worst. This confirms the visual analysis using the histograms below. It is odd to see the predictors normally distributed without any transformations which is an indicator of the manufactured nature of this dataset.



To investigate this further I used boxplots for all the variables and Q-Q plots on the three previously mentioned variables (please see the plots below). The Q-Q plots of hardness and trihalomethanes show they are normally distributed while the plot of solids is very close to normally distributed. The boxplots of the unstandardized variables clearly show a difference in scale. Individually they all show some number of outliers.



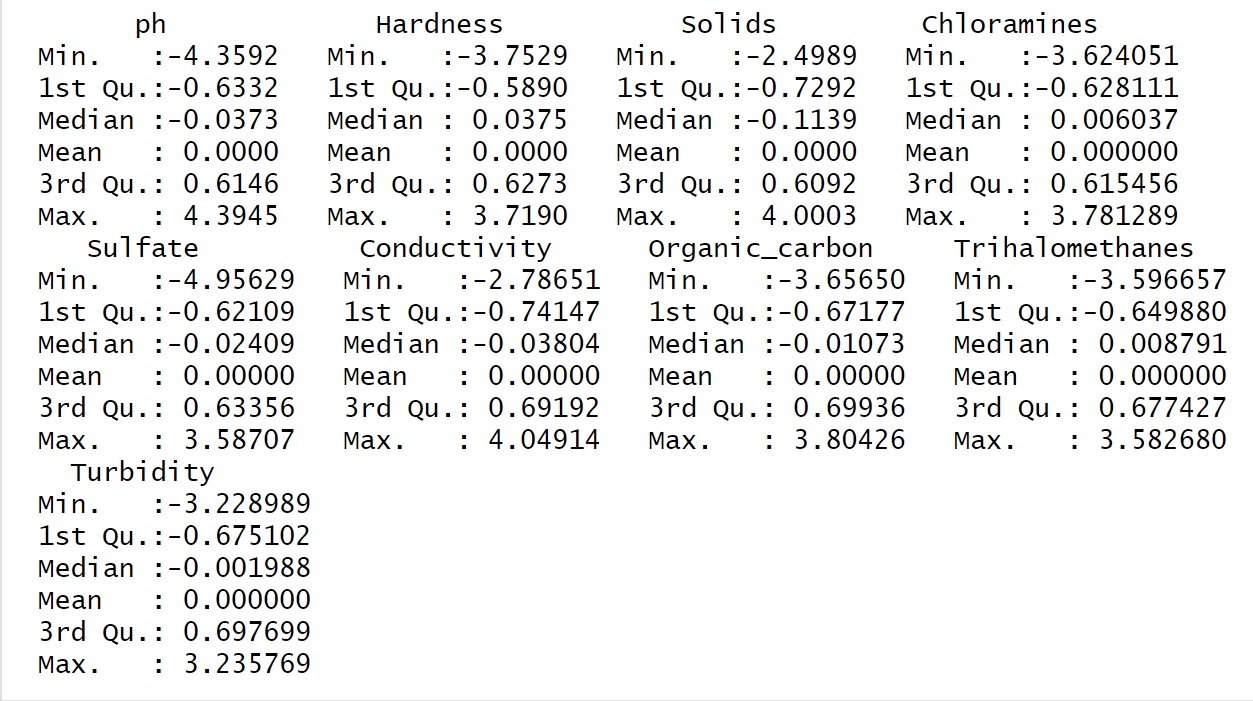


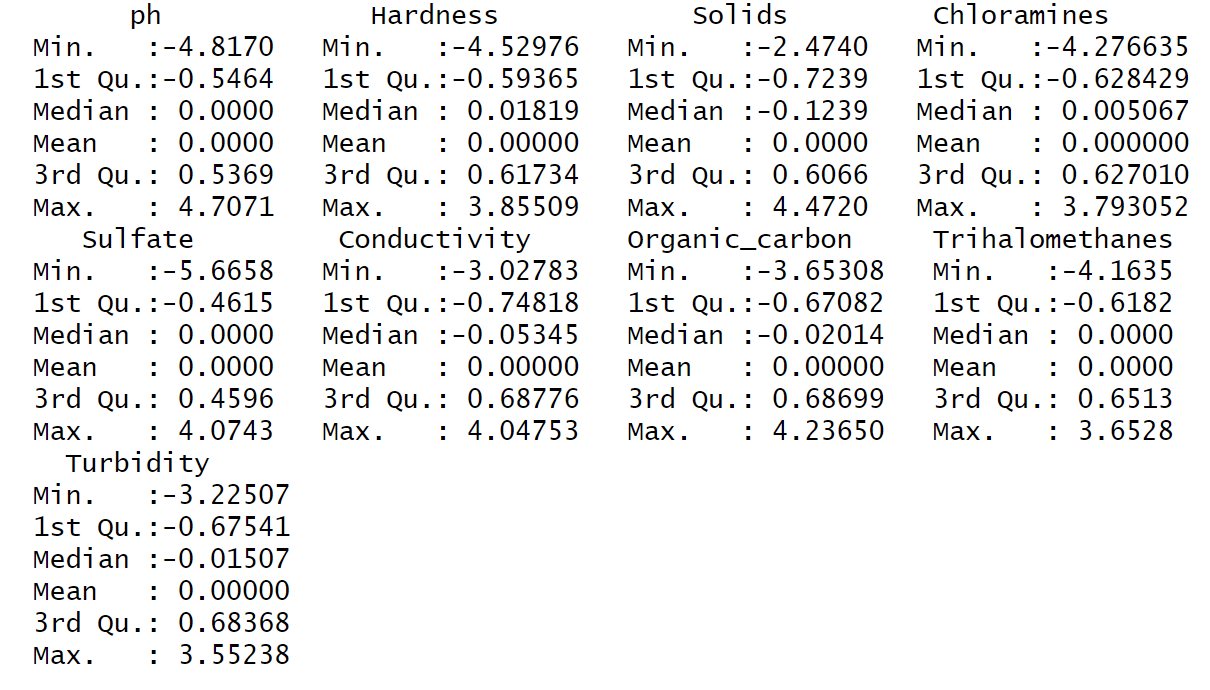
Another important aspect to the data that the summary shows is the number of missing values. So, my next step was to take care of these. Only three of the nine variables were missing observations: pH, sulfates and trihalomethanes. This data set contains a large number of observations (3276) so dropping a few would probably not matter much, but there also 1265 observations with missing data. Due to this, I was split on how best to prepare my data table, so I decided to try two different methods: omission and imputation. From this point on I have two data sets that I can use to compare their performance on different tests. One data set I have removed all the observations with missing values which reduced the sample size from 3276 to 2011. The other data set I imputed the missing values by replacing NA’s with the column mean (the variable’s mean was used to replace all the missing values for each variable that was missing values) and thus retained the original sample size of 3276. The data set created by removing the observations with missing data will be referred to as the “reduced data set” or “omitted” and the one created by replacing missing values will be known as the “imputed data set”.

I was interested in how these methods of dealing with the missing values affected the variance of each variable. In the table below, I first show the variance of the variables in the initial data set without the missing observations included for the three variables that have them. Next, I show the variances of the variables after omitting the observations with missing data. Then I show the variances for the data set where I imputed the missing values. Lastly, I show the variances of the standardized variables which is the same for both data sets. From this table we can see the impact of deleting observations and imputing missing data on the variance of the different predictors. The variance did not reduce for all variables with omission or imputation suggesting that this is not a good way of determining which way to treat the missing observations. None of the variables have near-zero variance which supplements my understanding from the histograms of the normality of the distributions of each predictor especially after standardizing the variables which ensured the variance was equal to one for all predictors.

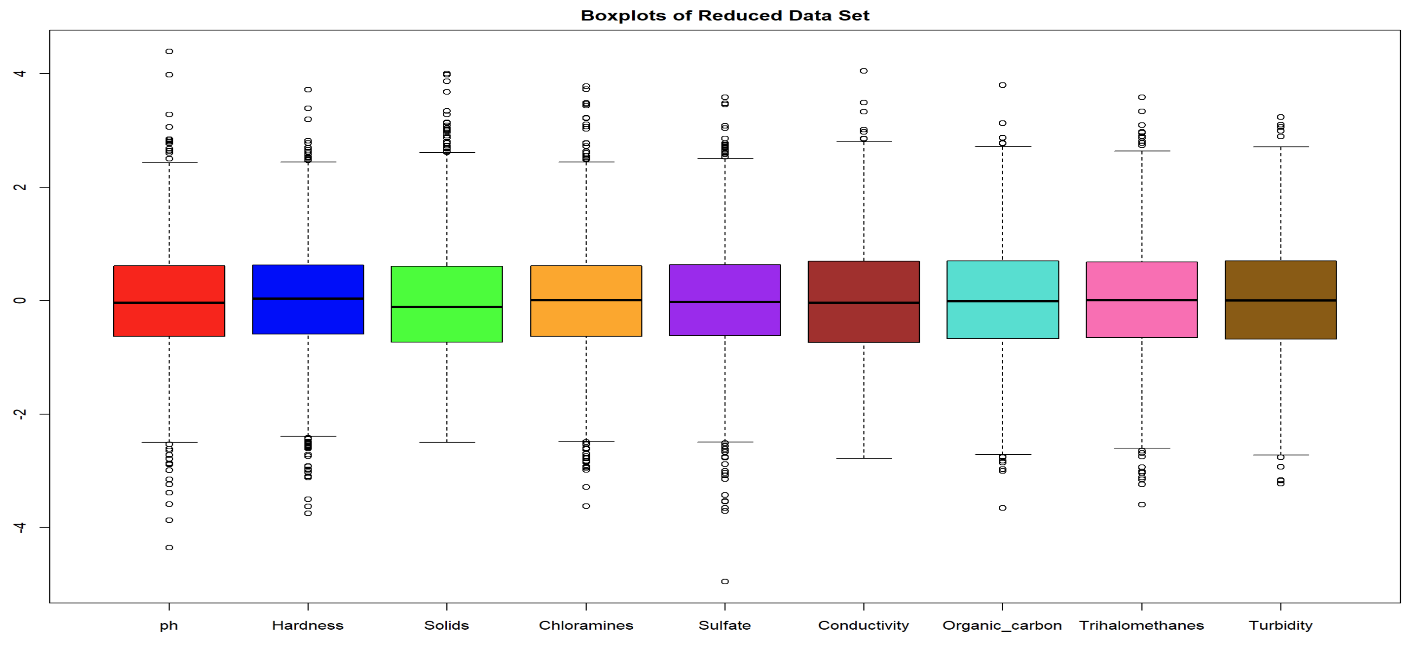
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Comparison of Variance of Predictors** | | | | |
| **Variable** | **Initial** | **Omitted** | **Imputed** | **Standardized** |
| pH | 2.542 | 2.475 | 2.161 | 1.000 |
| Hardness | 1081.079 | 1065.049 | 1081.079 | 1.000 |
| Solids | 76887834 | 74688309 | 76887834 | 1.000 |
| Chloramines | 2.506 | 2.512 | 2.506 | 1.000 |
| Sulfate | 1715.355 | 1697.866 | 1306.288 | 1.000 |
| Conductivity | 6532.529 | 6514.519 | 6532.529 | 1.000 |
| Organic Carbon | 10.944 | 11.055 | 10.944 | 1.000 |
| Trihalomethanes | 261.631 | 258.473 | 248.689 | 1.000 |
| Turbidity | 0.609 | 0.609 | 0.609 | 1.000 |

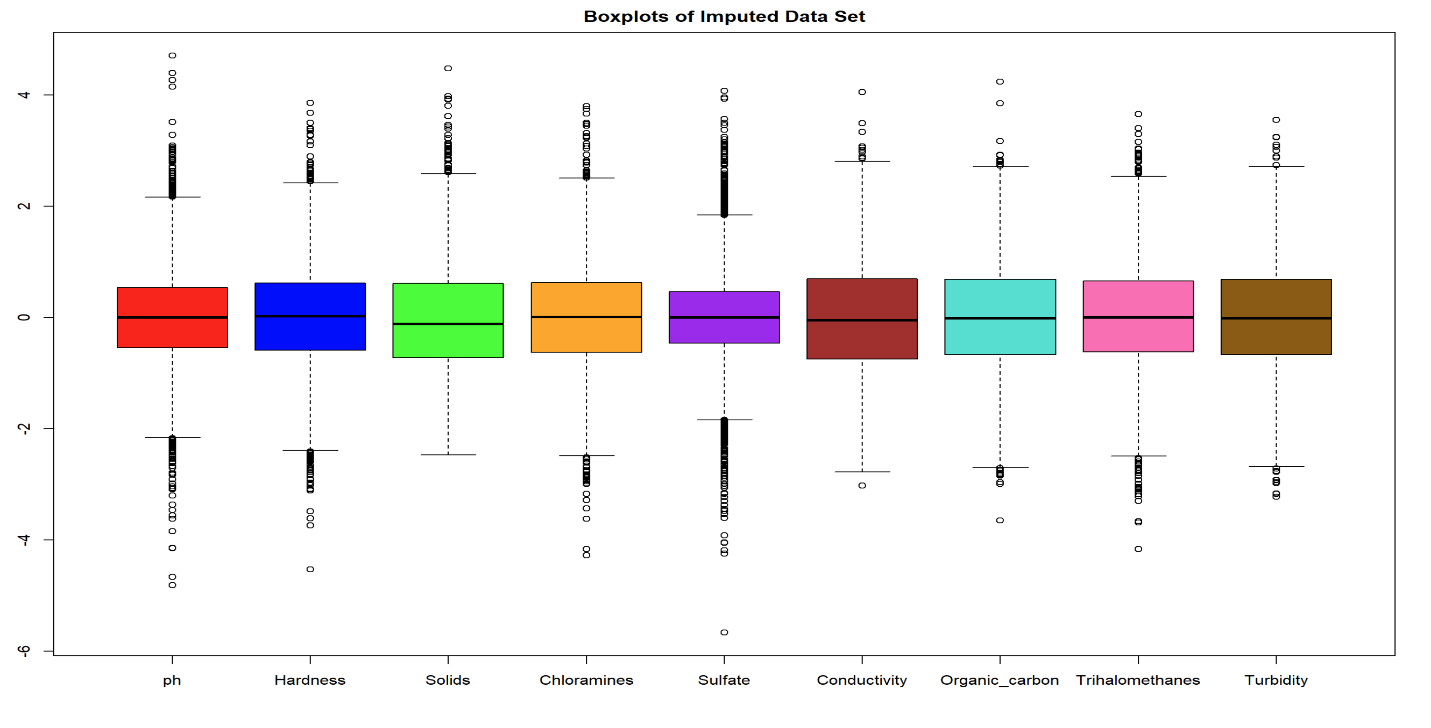
From the histograms, initial summary and especially the boxplots, we can see the scale of the variables is not similar and in some cases very large. To prevent issues due to differing magnitude later, I chose to scale and center the variables (a.k.a. standardize the variables). The summaries below show that the variables have similar magnitudes now. First is the summary for the reduced data set where observations with missing values were deleted. Second is the summary for the imputed data set where missing data was replaced with the column mean. From these summaries, we can see that the data is all on the same scale with the same mean of zero. The variance for each variable is now one.



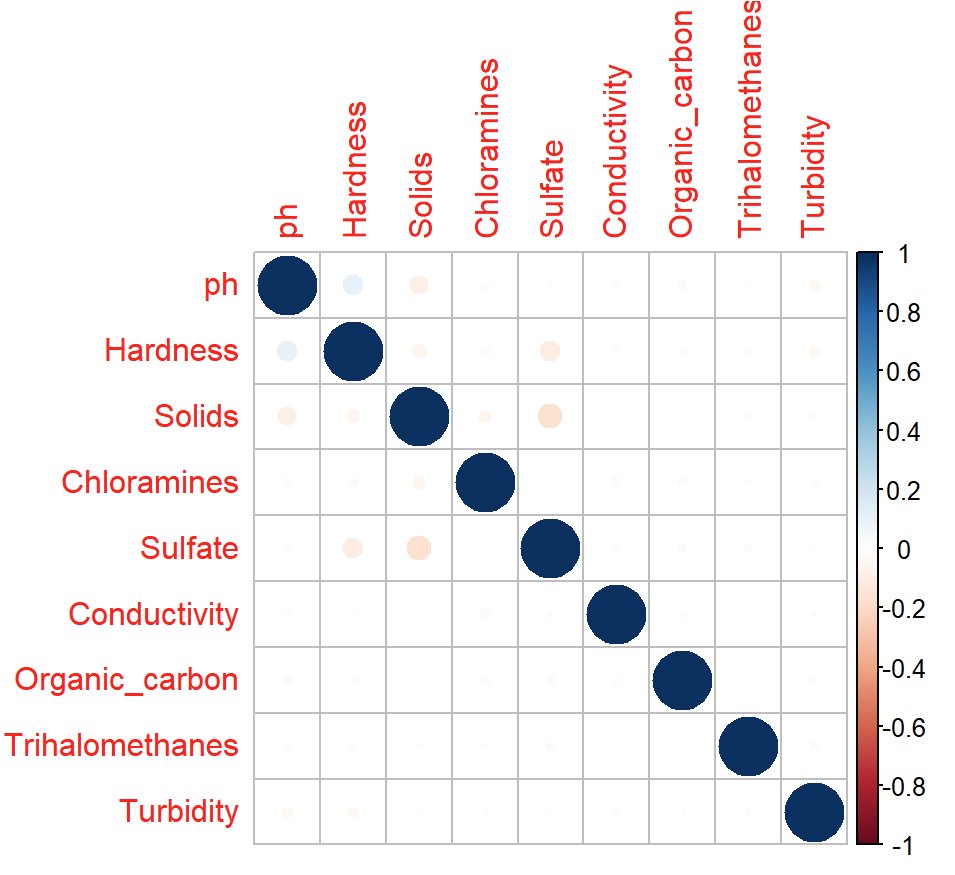
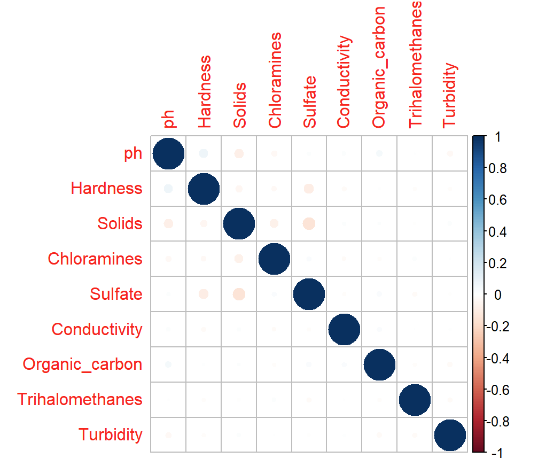


The boxplots of the unstandardized variables were not helpful for seeing the distributions since the scale of solids was so much larger than the other variables. After standardization I looked at the boxplots again. The boxplots below (one for each data set) show slightly different distributions. For example, the inner quartile range (IQR) for the sulfate and pH variables are different: they are narrower for the imputed data set. The number of outliers also differs somewhat but keeps the same general pattern for each variable. None of the outliers seem extreme enough to delete.



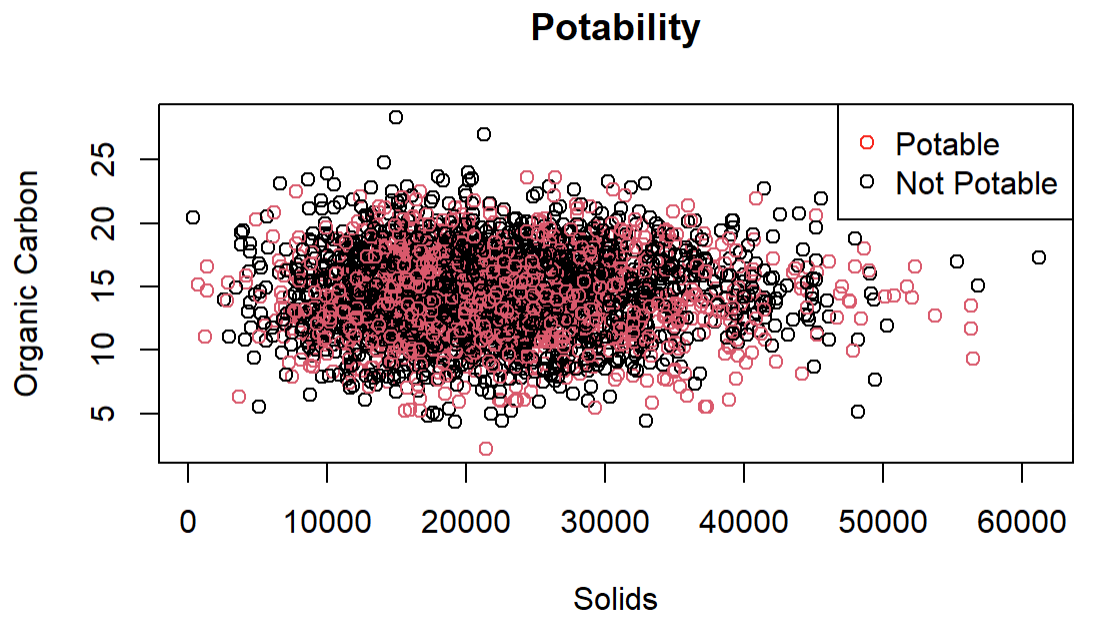
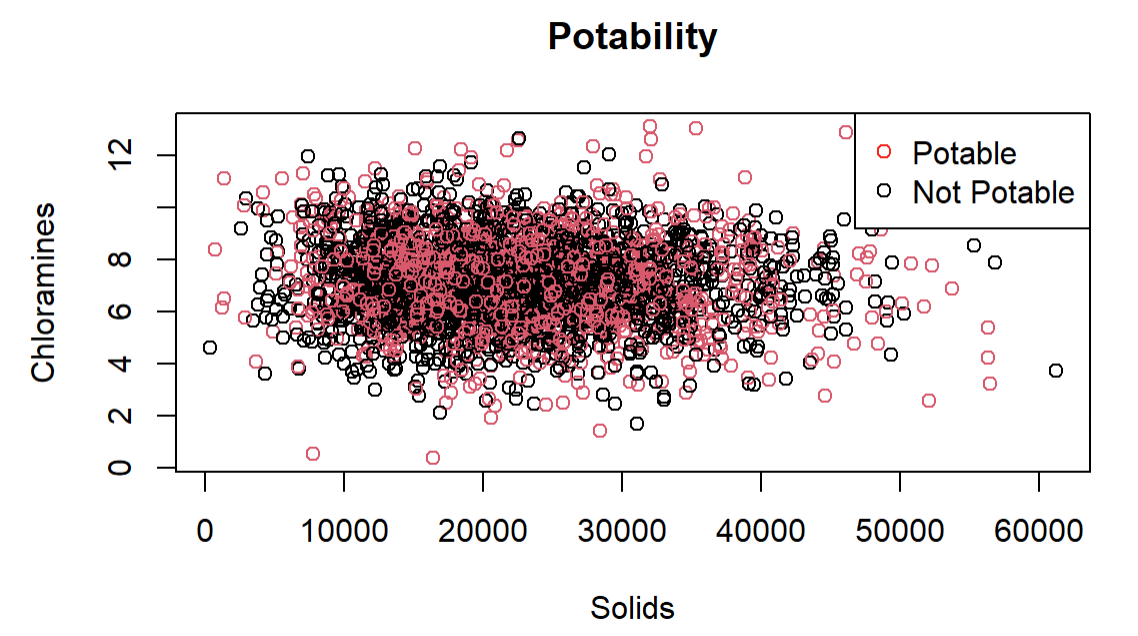


The correlation plot for the reduced data set (below) shows that there are no strong pairwise relationships between the variables. This is like the original dataset which also showed no strong pairwise relationships. This is one of the concerning aspects to this data as there should be at least a moderate correlation between pH and Hardness. Hardness acts as a buffer against acidity and is often measured as more alkaline, therefore large values of Hardness should have higher values of pH.



The correlation plot for the imputed data set (right) shows that this data set retains the same weak pairwise relationships between the variables. This is the same trend seen in the original data set.

Scatterplots were created but proved to be unhelpful due to the large number of observations in the original data set. They all appeared as circular blobs which did confirm the outcome of the correlation plots which showed no strong pairwise relationships between all the predictor variables. Pairwise plots with Potability as the color factor, like the plots below using the original data set, showed a large amount of overlap between the categories. This suggests that linear methods of separation will not perform well.

**Preparation for Model Building:**

The two data sets were first split into training and testing sets. For each one a random 80% of the data was retained for the training set and the remaining 20% for testing. The reduced data set has 1608 observations in the training set and 403 in the test set. The imputed data set has 2620 observations in the training set and 656 in the test set. Both training and test sets retain a similar proportion of 0’s and 1’s for the outcome (the imputed training and test sets have about 0.39 1’s and the reduced training and test sets have about 0.40, so not much of a difference in proportion from the original).

Several methods were applied including different approaches to variable selection, generalized linear model (using the identity link), logistic regression, linear discriminant analysis (LDA), K-nearest neighbors, different types of trees, and different types of support vector machines (SVM).

**Variable Selection**

Wrapper and embedded methods of variable selection were applied. The results of the important variables from the tree methods were compared to the outcome of the wrapper methods. SVM weights were also compared to these outcomes.

The wrapper methods, forward, backward, stepwise/both, and all-possible or best were applied to the reduced and imputed data sets. In R, the regsubsets(), StepCriterion() and olsrr() functions were applied. For the resgsubsets() function, backward, forward, sequential replacement, and exhaustive subset selection methods were used. For StepCriterion(), backward and forward subset selection methods were used. For the olsrr() function, backward, forward, and best subset selection were applied. For all of these methods, changing the default threshold values did not change the results. Interestingly, they all selected Solids as the most important variable for both data sets. The next most important variable selected by some of these methods was Chloramines for both data sets. Half the approaches differed on the second most important variable for the imputed data set with the choice of Organic Carbon over Chloramines. (Please see the table below for the comparison.)

|  |  |  |
| --- | --- | --- |
| **Comparison of Most Important 2 Variables – Wrapper Subset Selection** | | |
| **Function/Approach** | **Imputed Data Set** | **Omitted Data Set** |
| Regsubsets –All (B, F, SR, E) | Solids, Chloramines | Solids, Chloramines |
| StepCriterion – All | Solids, Organic Carbon | Solids only |
| Olsrr – BF | Solids, Organic Carbon | Solids only |
| Olsrr – Best | Solids, Chloramines | Solids, Chloramines |

The embedded methods proved to be largely unhelpful for assessing the importance of the predictors in both data sets. The Lasso method shrunk every variable’s coefficient to zero with only the intercept retained. The ridge method shrunk every variable’s coefficient to near-zero (coefficients were all on the scale of or smaller). For both of these approaches the only significant coefficient was the intercept which can be interpreted as meaning the variables were not related to the outcome in a meaningful way. This is another suspicious moment for this data set as at least one of the variables should be indicative of potability. If some of the observations were mislabeled as potable when the predictors were outside of the potable ranges, then the outcome would not be well-connected to these predictor variables. This seems to be what is making the ridge and lasso methods ineffective.

The tree methods can provide an idea of variable importance if the first variables chosen to split the data are considered the more important. For the imputed data set, the methods ranked Sulfate or Hardness as most important followed by either Hardness of pH. For the omitted data set, almost all the approaches chose Sulfate as the most important followed by pH except for the random forest with m=p-1 which reversed the important of these. It is interesting that these results had no overlap with the wrapper methods. The pruned classification tree (included in the section on the tree methods) did not use Solids at any node.

|  |  |  |
| --- | --- | --- |
| **Comparison of Most Important 2 Variables – Tree Methods** | | |
| **Tree Method** | **Imputed Data Set** | **Omitted Data Set** |
| Pruned Classification | Sulfate, Hardness | Sulfate, pH |
| Bagging | Sulfate, pH | Sulfate, pH |
| Random Forest m=p-1 | Hardness, pH | pH, Sulfate |
| Random Forest m=p/3 | Hardness, pH | Sulfate, pH |
| Random Forest m= | Hardness, pH | Sulfate, pH |

The output of the SVM methods were used to calculate and compare the weights of each variable for both data sets. The results of the SVM methods compared in the table below were quite different from each other and for the imputed data set more similar to the outcome of the wrapper methods. For the omitted data set, the results were similar in some ways to the tree methods.

|  |  |  |
| --- | --- | --- |
| **Comparison of Most Important 2 Variables – SVM** | | |
| **SVM Method** | **Imputed Data Set** | **Omitted Data Set** |
| Polynomial degree 2 | Chloramines, Solids | Sulfate, Organic Carbon |
| Polynomial degree 4 | Solids, pH | Sulfate, Conductivity |
| Radial | Solids, Chloramines | Organic Carbon, Sulfate |

**Analytical Methods**

1. **Generalized Linear Models & Logistic Regression**

The first methods I applied were using the generalized linear model function in R. First I applied this model using an identity link and then without. For both approaches I used the variable importance results of the wrapper subset selection to model Potability. For the imputed data set I used Solids and Organic Carbon and for the omitted data set I used Solids and Chloramines in the model. Only two variables were used to model Potability for each data set because my goal was interpretability. Due to the large overlap between Potable and Non-potable outcomes, prediction accuracy was not expected to be good. In fact, the error for the first approach using an identity link was 0.6356707for the imputed data set and 0.5955335 for the omitted data set (the highest of all the methods used).

The model for the imputed data set using the identity link: where the coefficient for Organic Carbon was significant at a 0.05 level but not the coefficient for Solids. Refitting the model using only Organic Carbon did not alter the prediction accuracy nor did it change the fitted model (except for the deletion of Solids). The probability that a body of water is potable is decreased by 1.96% for a 1-unit increases in Organic Carbon.

The model for the omitted data set using the identity link: P(y=1) = 0.403 + 0.019(Solids) + 0.015(Chloramines). Only the intercept was significant at a 0.10 significance level. Since neither variable was significant in this model, there is not point in interpreting the relationships modeled.

The logistic regression method (not using the identity link) resulted in a slightly different model and classification error. For the imputed data set the model is: P(y=1) = -0.419 + 0.043(Solids) – 0.081(Organic Carbon). Similarly to the previously discussed model, only the coefficient for Organic Carbon (and the intercept) was significant at a 0.05 significance level. The probability that a body of water is potable decreases by about 8% for every one-unit increase in the Organic Carbon level. The prediction accuracy for this model is 0.3643293.

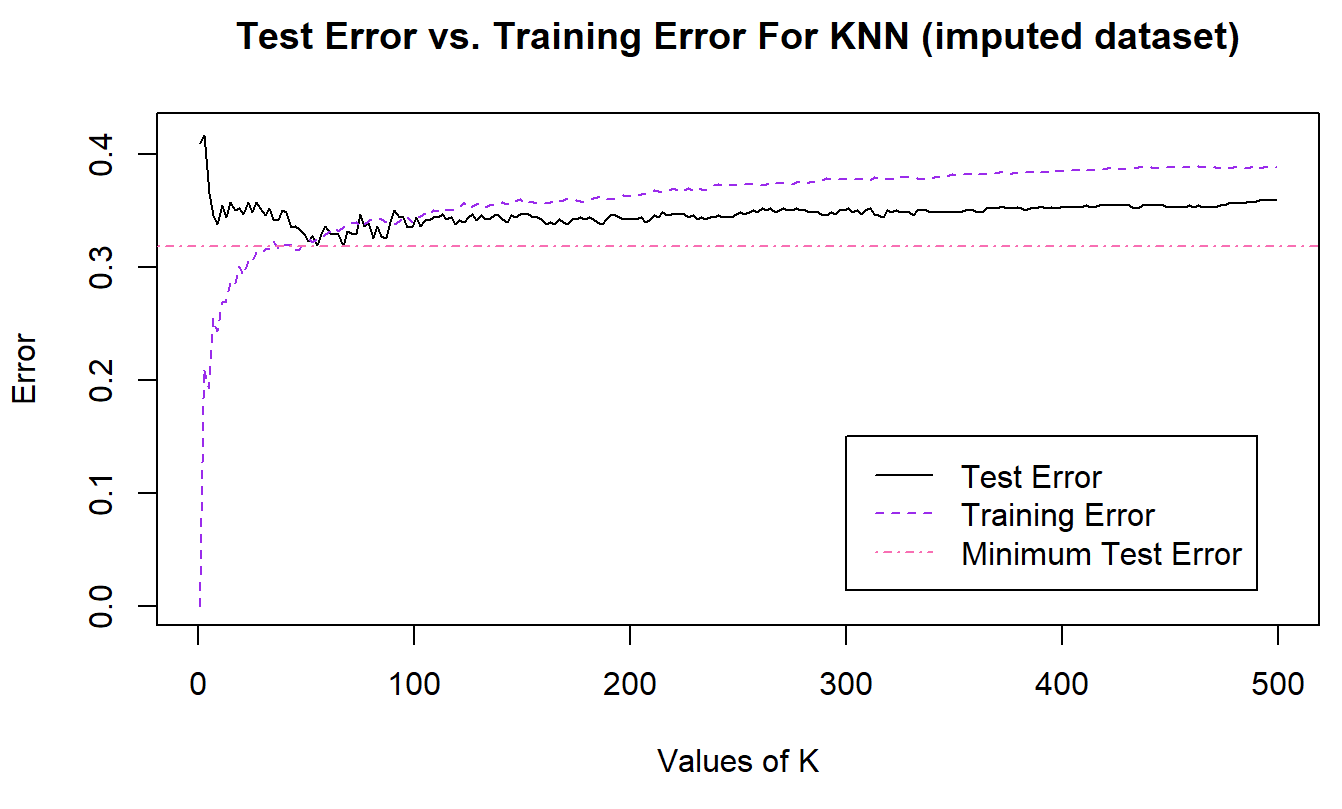
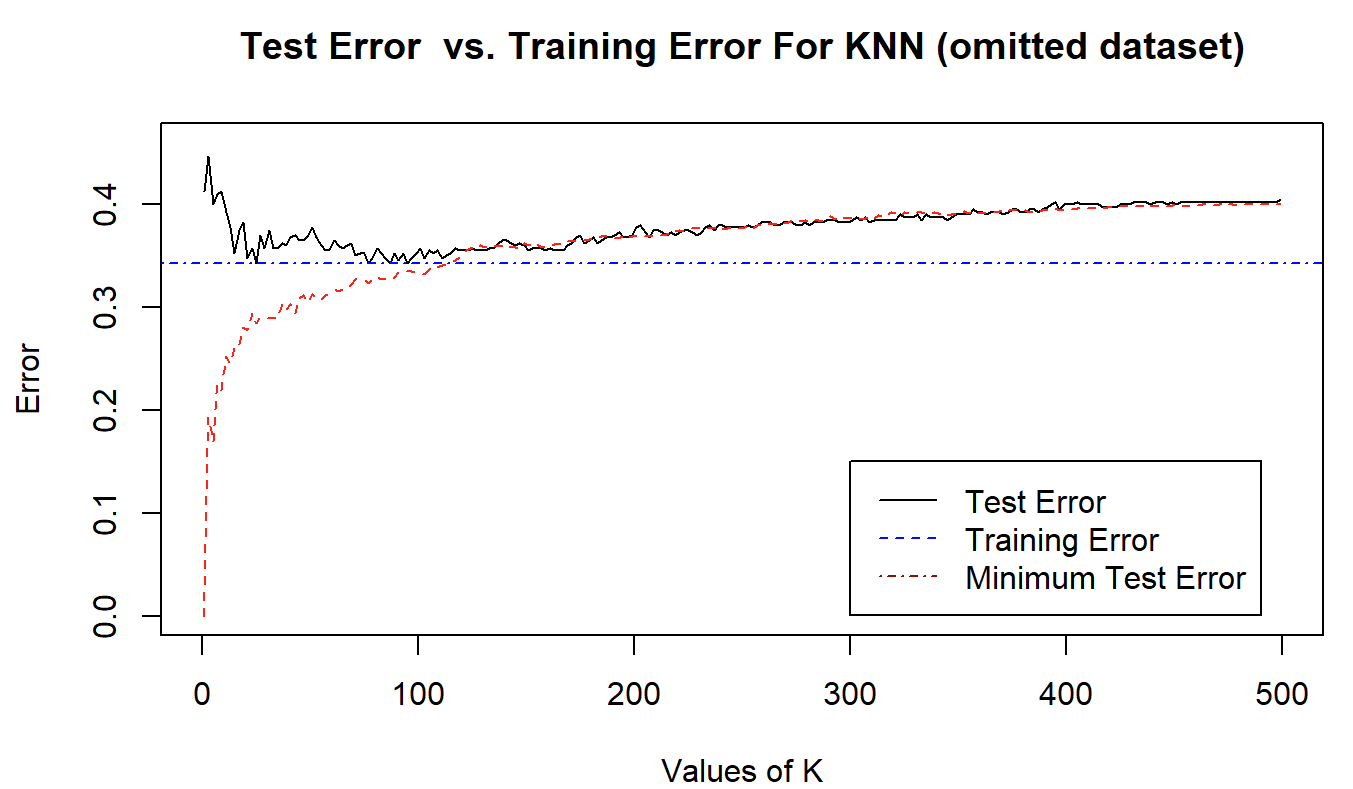
For the omitted data set the model is: P(y=1) = -0.392 + 0.077(Solids) + 0.063(Chloramines) with no coefficient (except for the intercept) significant at even a 10% significance level. The error rate for this model was 0.4044665. Fitting the models with all the variables or other combinations based on the variable selection/important results did not result in a significant model or better prediction accuracy. The results here are unsurprising since the response categories do not look linearly separable.

1. **LDA**

The next method applied to the two data sets was linear discriminant analysis (LDA). Just like the previous section, the results of the wrapper subset selection were used to model Potability; Solids and Organic Carbon were used to model for the imputed data set and Solids and Chloramines for the omitted data set. The misclassification error for the imputed data set is 0.3643293 and for the reduced data set 0.4044665 (same error as the logistic regression approach). The LDA model for the imputed data set and for the reduced data set . In general, it seems that Potability decreases with increasing Organic Carbon levels, increases with increasing Chloramine level, and increases with increased Solids. This understanding only makes sense if the previously mentioned allowable levels are kept in mind.

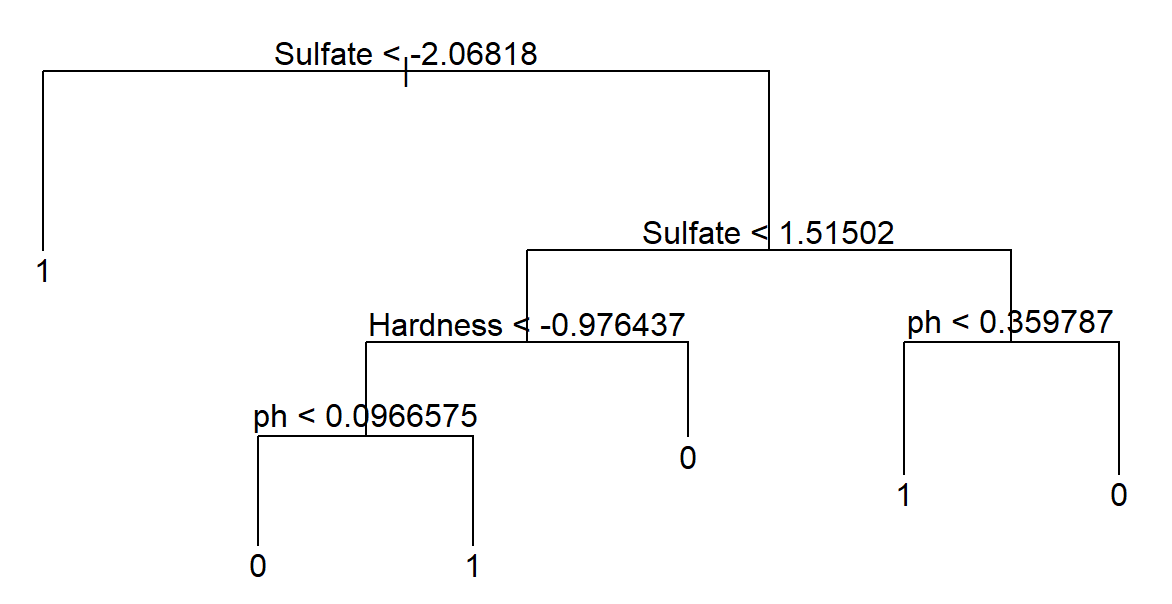
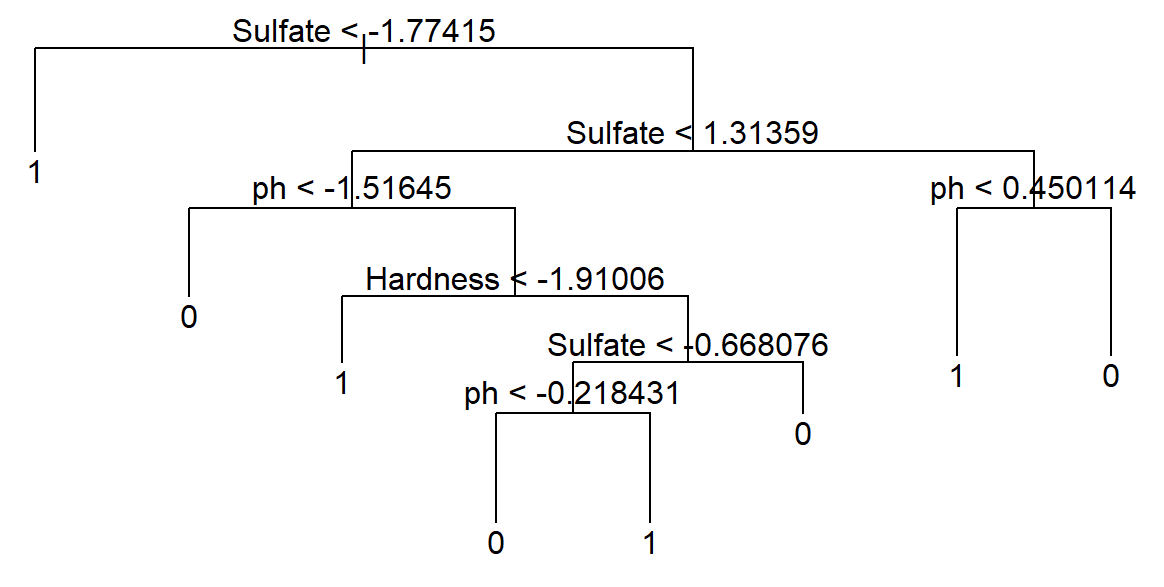
1. **KNN**

Since KNN is a non-parametric method that relies only on k-nearest neighbors to a point, I expected it to do well on this data set especially for small values of k. The outcomes (potable vs. non-potable) are not linearly separable and therefore a more flexible method should do well. For both data sets, a large range of values for k were tried and their training and testing misclassification error were compared in the plots below. The minimum test error of 0.3185976 occurred for the imputed data set at k=67. The minimum test error for the omitted data set was 0.3424318 and occurred for multiple values of k including k=25, 77, 87, and 95 which shows that this was a stable method to apply to at least the reduced data set. For the imputed data set, this method resulted in the third lowest misclassification test error. In both plots below, the error stabilized as the value of k grew. In the plot of the imputed data, as k grew the training error was larger than the test error which I believe is just due to chance from randomly splitting the data into training and testing sets.

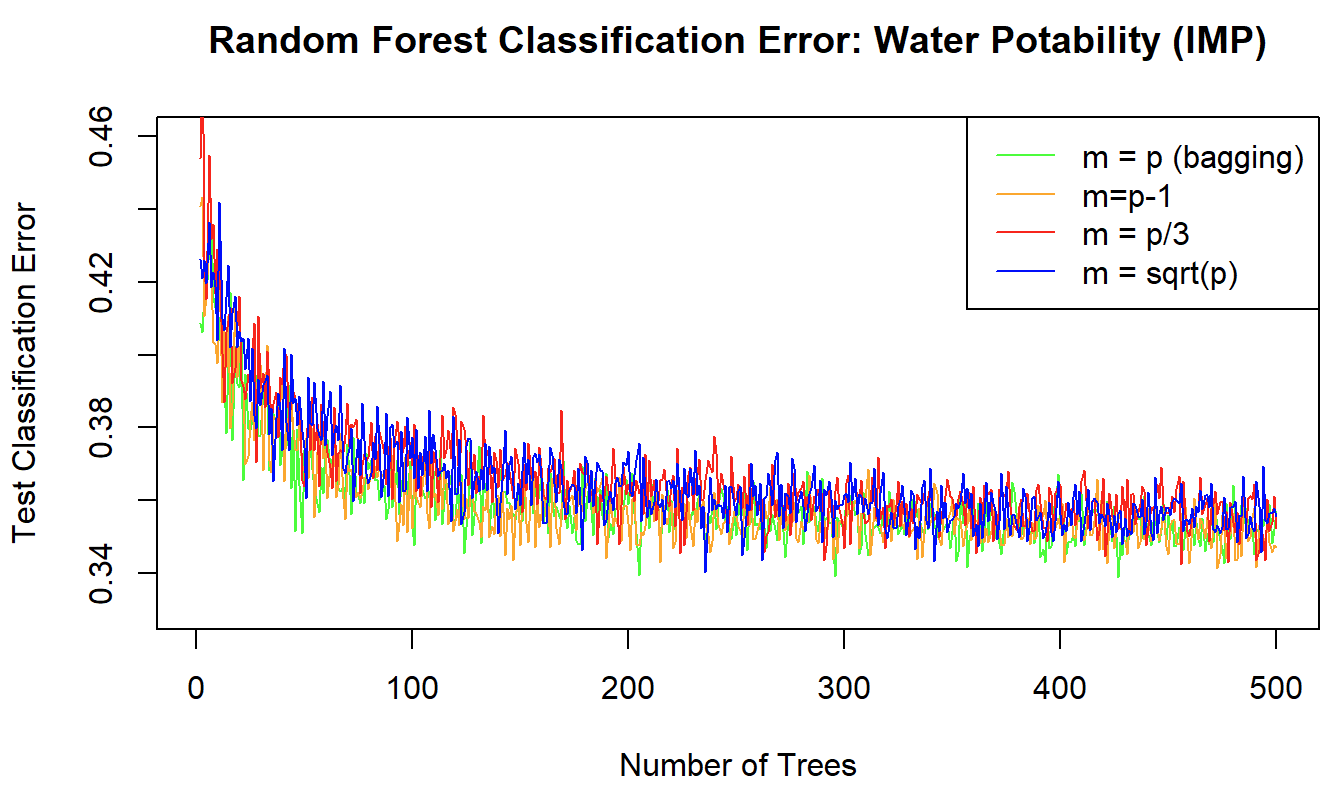
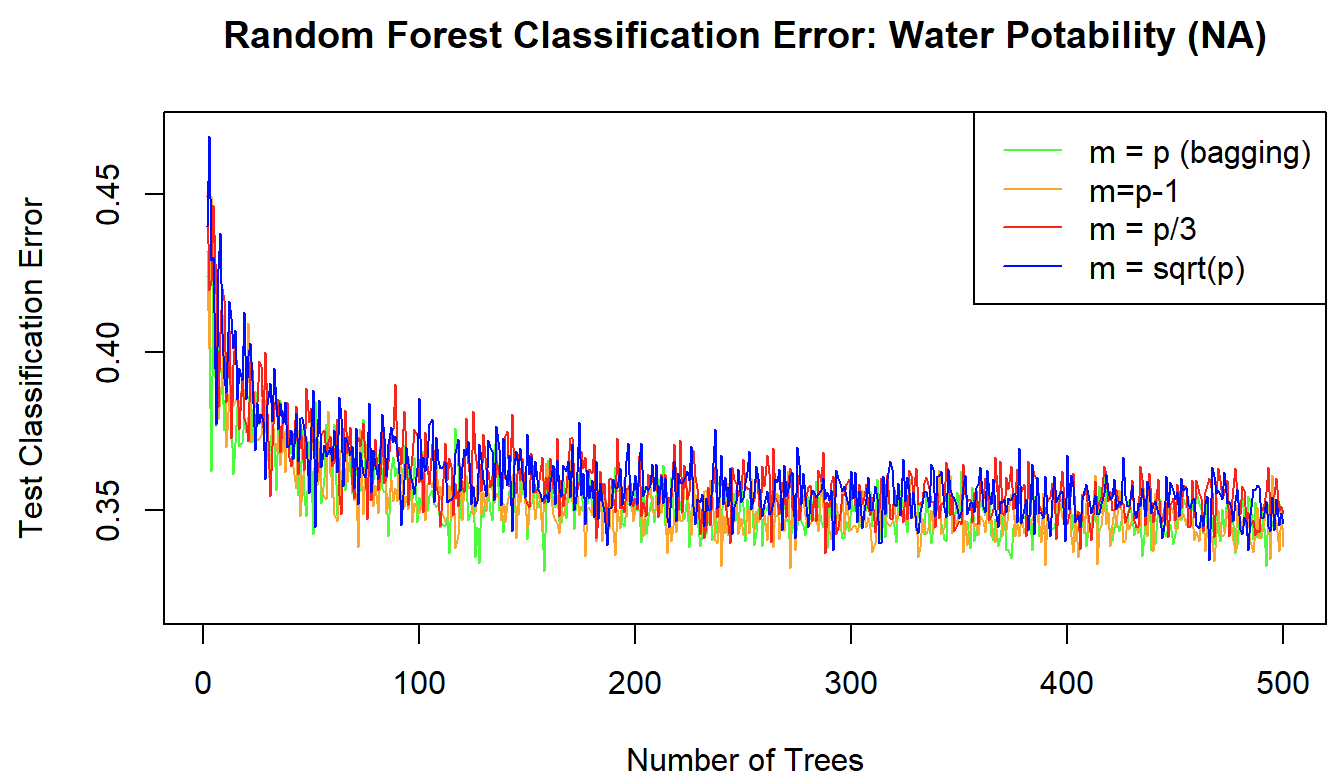
1. **Trees**

The first tree method I applied was a large classification tree followed by pruning using cross-validation. The splits in the trees were done to minimize the Gini Index rather than using the misclassification error rate. The prediction accuracy for the large classification tree for the imputed data set was 0.3765244 and for the omitted data set 0.4119107. This is slightly higher than for LDA or logistic regression. The most important variable (the first split) was Sulfate for both data sets. The pruned trees (plotted below, imputed on the left and omitted on the right) had slightly better test error suggesting that the large unpruned trees overfit the training data; 0.3353659 for the imputed data set and 0.337469 for the omitted data set. In the plots below, the trees show that Sulfate was the most important variable for the first two splits followed by pH and Hardness. No other variables were used in the pruned trees at all which suggests that Solids was not as important as the wrapper methods suggested.

Bagging was used next in the hopes of reducing the misclassification error since bootstrapping and averaging are both great ways to reduce the variance and improve the fit to the data. The default in R of 500 trees was used in this approach at first. Then between 1 and 500 trees was compared (please see the plots in the discussion of the random forest approach). For this data, there are only 9 variables to try at each node. The error rate for the imputed data set was 0.338927 and for the omitted data set was 0.3307887. This method produced the third lowest error for the omitted data set. This method ranked Sulfate followed by pH as the most important predictors for both data sets (Solids was ranked 5th).

Lastly I built random forests using different vales of m: p-1, p/3, and sqrt(p). Since there are not many variables to start with, these random forest approaches were not expected to show much if any improvement over bagging. For these approaches and the bagging method, a number of trees between 1 and 500 were applied. The comparison of the test error for each method on each data set is shown in the plots below.

The minimum test error for the imputed data set was 0.338927 which occurred for 426 trees when m=p, so for the bagging approach. The other random forest methods produced error rate for the imputed data set as follows: for m=p-1 the error was 0.3413493, for m=p/3 the error was 0.342583 and for m= the error was 0.3402585. These are very similar error rates due to the small number of predictor variables being drawn from.

The minimum test error for the imputed data set was 0.3307887 which occurred for 157 trees when m=p, again the bagging approach. The other random forest methods had comparable error rates: for m=p-1 the error was 0.3314971, for m=p/3 the error was 0.3365162 and for m= the error was 0.334296. None of these random forest approaches, even bagging, were in the top 3 methods with the lowest error rates. Again, this is most likely due to the small number of predictor variables.

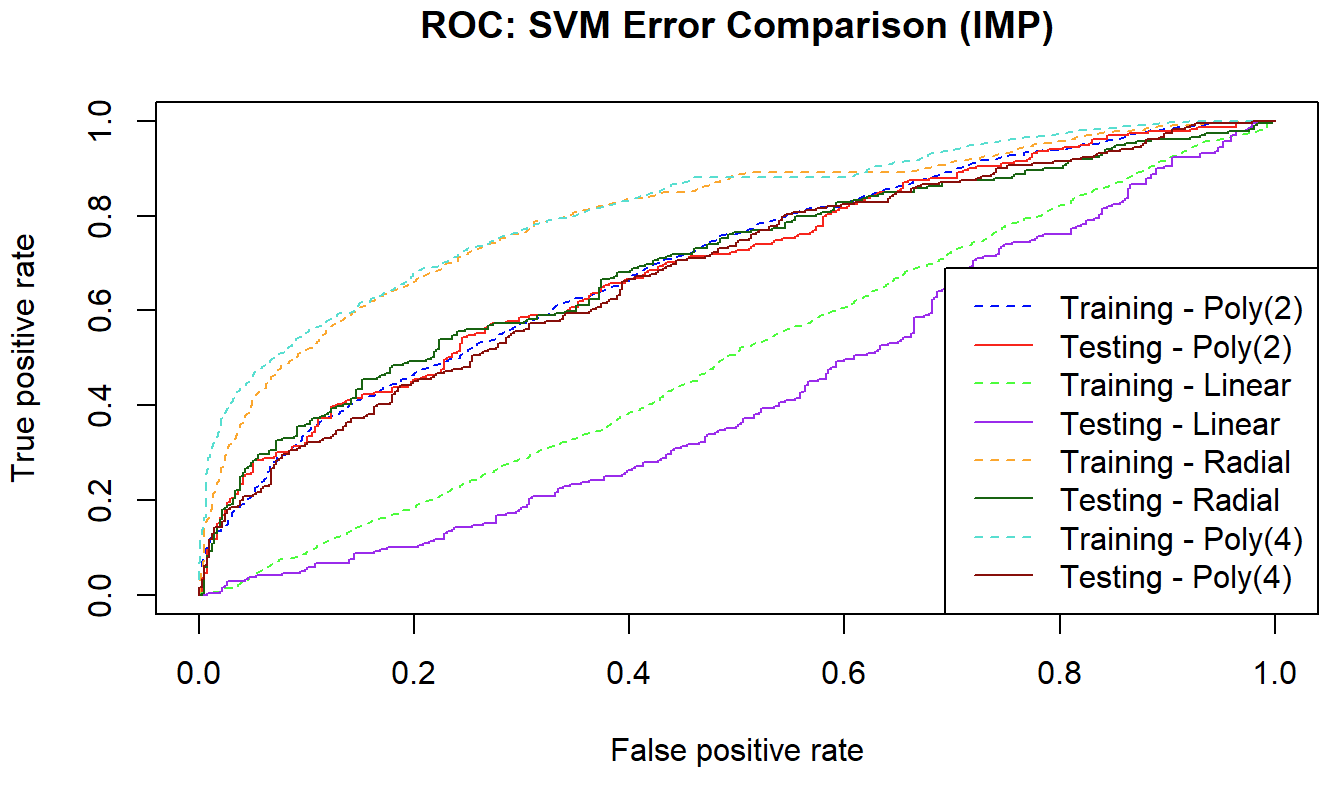
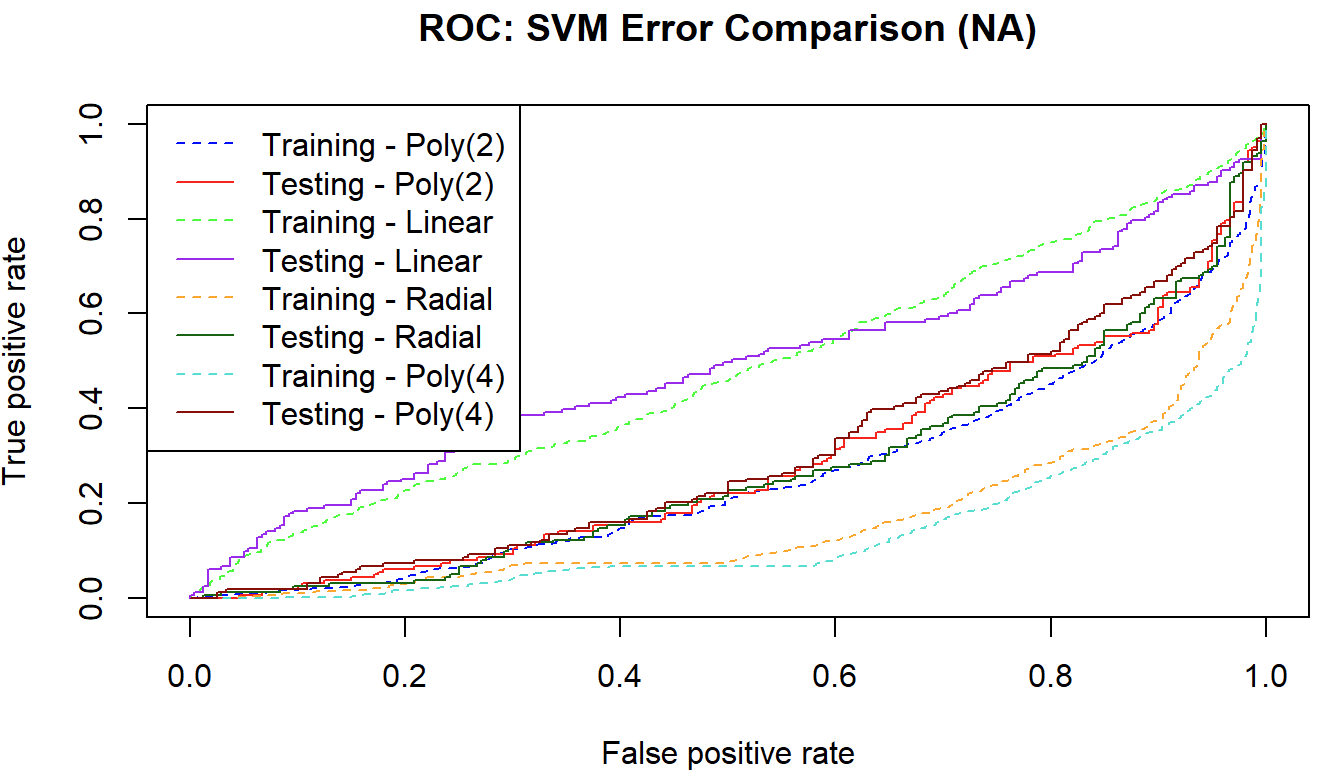
The top-ranked variables by importance for both data sets were Sulfate and pH (please see the variable selection section above for a more complete reporting).

1. **SVM**

Due to the nature of the overlap in the response categories, a more flexible method such as support vector classifier or support vector machines (SVM) should work well. These soft classifiers allow misclassifications which is ideal for applying to less distinct classes. The SVM approaches applied: linear (classifier), polynomial kernel of degree 2, 3 and 4, and a radial kernel. The specific parameters for the non-linear kernels and the cost for all methods were determined through cross-validation. After some exploration, it was determined that some flexibility existed in the value of the cost with little to no impact on the misclassification error rate.

The best of these methods was the polynomial kernel with degree 2 and 4 (top two performers for both data sets) and the worst were the linear classifier and the polynomial degree 3. The radial kernel was in the middle of these extremes. The error rate for the polynomial kernel degree 2 for the imputed set was 0.2987805 (the lowest achieved) and for the omitted data set was 0.3151365. The error rate for the polynomial kernel degree 4 was 0.3094512 for the imputed data set and 0.3275434 for the omitted data set. The radial kernel produced an error rate of 0.3292683 for the imputed data set and 0.3573201 for the omitted data set. The linear and the polynomial degree 3 produced the same error rate for the omitted data set of 0.4044665. For the imputed data set the error was slightly different: 0.3643293 for linear and 0.3460366 for the polynomial degree 3 kernel. For the two best SVM approaches, there were more than 1000 support vectors for both data sets (about 1100 for the polynomial degree 2 approach and about 1200-1300 for the polynomial degree 4). The other methods required closer to 2000 support vectors.

Another way to judge the fit of these SVM models is using an ROC curve. Below are the ROC curves for all the SVM models fit except for the polynomial degree 3 model. For the imputed data set (left) the ROC curves look much more characteristic of a good fit than for the omitted data set (right). The best fits shown on the ROC curve for the imputed data is for the radial kernel (dark green) and the polynomial degree 2 (red). For the omitted data set, the best ROC curves are for the polynomial degree 4 (dark red) and 2 (red) fits.

**Comparison of Results**

|  |  |  |
| --- | --- | --- |
| **Classification Test Error Comparison** | | |
| **Method** | **Omitted Data** | **Imputed Data** |
| GLM – identity link | 0.5955335 | 0.6356707 |
| Logistic Regression | 0.4044665 | 0.3643293 |
| LDA | 0.4044665 | 0.3643293 |
| KNN | 0.3424318 | 0.3185976 \*\*\* |
| Classification Tree (unpruned) | 0.4119107 | 0.3765244 |
| Classification Tree (pruned) | 0.337469 | 0.3353659 |
| Tree - Bagging | 0.3307887 \*\*\* | 0.338927 |
| Random Forest (m=p-1) | 0.3314971 | 0.3413493 |
| Random Forest (m=p/3) | 0.3365162 | 0.342583 |
| Random Forest (m=) | 0.334296 | 0.3402585 |
| SVM – linear | 0.4044665 | 0.3643293 |
| SVM – polynomial (2) | 0.3151365 \* | 0.2987805 \* |
| SVM – polynomial (3) | 0.4044665 | 0.3460366 |
| SVM – polynomial (4) | 0.3275434 \*\* | 0.3094512 \*\* |
| SVM – radial | 0.3573201 | 0.3292683 |

**\*** represents the method with the lowest test error, \*\* represents the second lowest, \*\*\* represents the third lowest

**Summary/Discussion/Conclusion**

Overall, the two best methods to apply to these data sets was the support vector machine using a polynomial kernel with a degree of 2 followed by a degree of 4. The large overlap in the Potability classes made this a very good fit since SVM fits while allowing some misclassification. For the polynomial degree 2 fit, the cost found through cross-validation, for both data sets was 10 and for the degree of 4 fit the cost was 1. The data sets differed on which method produced the third lowest error: KNN for the imputed data and a Tree using Bagging for the omitted data.

The worst performing methods relied on a linear decision boundary which was obviously not going to perform well by looking at the scatterplots included earlier in this paper. The previously mentioned overlap in the outcomes presented a difficult separation. The identity link generalized linear model performed the worst (highest test error for both data sets). This method had an error rate similar to just classifying all observations as potable. This is obviously not desirable as just classifying all bodies of water as drinkable will certainly have bad effects on human health.

The most important variable changed depending on the method used. For the embedded methods like the Lasso model or Ridge regression found all variables to be unimportant. The wrapper methods all agreed that Solids was the most important to both data sets. The ranking of important variables by the tree methods and the ranking of the weights used in the SVM methods did not all agree with the wrapper methods. The trees ranked Sulfate as the most important and the SVM methods differed.

The most important result of this potability investigation is determining that the data set was most likely manufactured rather than reflecting the reality of testing 3276 actual bodies of water. Doubt was cast due to the perfectly normally distributed predictor variables and the ranges of the variables in the data set going far outside reasonability for seeing in nature. After determining there was reason to question the data, more specific investigation of observations labeled “potable” was carried out. An example of one concerning observation: with pH of 0.22 (pre-standardizing), solids above 39,000, sulfate above 280, and Hardness above 150 the body of water was labeled potable. A pH of that low would burn the skin of someone trying to bathe and would burn the mouth and esophagus if drunk. This water sample clearly describes a non-potable body of water.

**Future Direction**

The next analytical methods that should be applied to the current data set would be polynomial regression fits and step functions. These would be better fits to the data than linear models. Fine-tuning and comparing multiple polynomial SVM models would perhaps lower the misclassification test error. Ideally a real-world data set could be found and performing analysis using KNN, classification trees (especially random forest methods), and SVM could be more illuminating. Including both laboratory results and at-home testing results on the same water samples would be preferred. This would allow comparison and hopefully, improvement of current at-home tests. If possible, including even more predictor variables would help improve the performance of the random forest methods (m ≠ p).

**References**

1. Kaggle data set. Available from <https://www.kaggle.com/datasets/adityakadiwal/water-potability>
2. Water Education Foundation. Available from <https://www.watereducation.org/aquapedia-background/potable-water#:~:text=Potable%20water%2C%20also%20known%20as,chemicals%2C%20viruses%20and%20fecal%20matter>.
3. Important tests of water potability. Available from <https://caltestlabs.com/analytical-services/regulated-drinking-water-homeowners/wateranalyses/>
4. Potable vs. non-potable aspects and impacts. Available from <https://www.worksafe.qld.gov.au/safety-and-prevention/hazards/hazardous-exposures/non-potable-water#:~:text=What%20do%20we%20mean%20by,drinking%2C%20cooking%20and%20personal%20bathing>.
5. CA water guidelines and information. Available from <https://waterboards.ca.gov/gama/docs/wellowner_guide.pdf>
6. CDC. Available from [https://www.cdc.gov/healthywater/drinking/public/water\_disinfection.html#:~:text=What%20are%20safe%20levels%20of%20chloramine%20in%20water%3F,effects%20are%20unlikely%20to%20occur](https://www.cdc.gov/healthywater/drinking/public/water_disinfection.html)
7. EPA. Available from <https://archive.epa.gov/water/archive/web/html/sulfate.html>
8. EPA. Available from [https://www.epa.gov/ground-water-and-drinking-water/national-primary-drinking-water-regulations#Disinfectants](https://www.epa.gov/ground-water-and-drinking-water/national-primary-drinking-water-regulations)
9. James G, Witten D, Hastie T, & Tibshirani R. An Introduction to Statistical Learning with Applications in R. 2nd Ed. 2021. Available from <https://www.statlearning.com/>
10. Kaggle (data set). Available from <https://www.kaggle.com/datasets/adityakadiwal/water-potability>
11. Safewater. Available from <https://www.safewater.org/fact-sheets-1/2017/1/23/tds-and-ph>
12. USGS. Available from [https://www.usgs.gov/special-topics/water-science-school/science/hardness-water#:~:text=General%20guidelines%20for%20classification%20of,Some%20content%20may%20have%20restrictions](https://www.usgs.gov/special-topics/water-science-school/science/hardness-water)

**Code Appendix**

library(caret); library(corrplot); library(olsrr); library(glmnet); library(leaps); library(timeDate); library(Hmisc); library(DMwR2); library(e1071); library("FSelector"); library(BiocGenerics); library(MASS); library(class); library(ggplot2); library(randomForest); library(tree); library(klaR); library(gbm); library(ROCR); library(glmtoolbox); set.seed(448)

water <- read.csv("~/Grad School/Spring 23/Math 448/Project/water\_potability.csv")

dim(water); names(water) ; attach(water)

par(mfrow=c(3,3)); hist(ph, col="hotpink"); hist(Hardness, col = "red"); hist(Solids, col="blue"); hist(Chloramines, col="orange"); hist(Sulfate, col="green"); hist(Conductivity, col="purple"); hist(Organic\_carbon, col="coral"); hist(Trihalomethanes, col="turquoise"); hist(Turbidity, col="darkgreen"); par(mfrow=c(1,1))

histogram(water$Potability, freq=T, xlab="Potability")

cor(water[,-10])

ones.og = sum(Potability == 1)/nrow(water); zeroes.og = sum(Potability == 0)/nrow(water)

par(mfrow=c(2,2)); qqnorm(water$Hardness, main="Q-Q Plot: Hardness"); qqline(water$Hardness); qqnorm(water$Solids, main="Q-Q Plot: Solids"); qqline(water$Solids);mqqnorm(water$Trihalomethanes, main="Q-Q Plot: Trihalomethanes"); qqline(water$Trihalomethanes); par(mfrow=c(1,1))

plot(water[,1:9]) #too many observations for this to be helpful

#maybe boxplots would be more helpful

boxplot(water[,1:9], col = c("red", "blue", "green", "orange", "purple", "brown", "turquoise", "hotpink", "orange4"))

summary(water)

water$Potability <- as.factor(water$Potability) #Change to factor from numeric

summary(water$Potability); apply(water, 2, var,na.rm=T)

cor(water[,-10], use = "pairwise.complete.obs"); corrplot(cor(water[,-10],use = "pairwise.complete.obs"))

sum(is.na(water))

water.na <- na.omit(water) #dataset without missing values included

dim(water.na)

ones = sum(water.na$Potability == 1)/nrow(water.na)

zeroes = sum(water.na$Potability == 0)/nrow(water.na)

water.imp = water

water.imp$ph[is.na(water.imp$ph)] <- mean(water.imp$ph, na.rm=TRUE)

water.imp$Sulfate[is.na(water.imp$Sulfate)] <- mean(water.imp$Sulfate, na.rm=TRUE)

water.imp$Trihalomethanes[is.na(water.imp$Trihalomethanes)] <- mean(water.imp$Trihalomethanes, na.rm=TRUE)

sum(is.na(water.imp)) #all NA's have been replaced with column means.

apply(water.imp[,1:9], 2, var)

S.imp = cov(water.imp[,1:9])

apply(water.na[,1:9], 2, var)

S.na = cov(water.na[,1:9])

water.imp[,1:9] <- scale(water.imp[,1:9], center=TRUE, scale=TRUE)

water.na[,1:9] <- scale(water.na[,1:9], center=TRUE, scale=TRUE)

summary(water.imp[,-10]); summary(water.na[,-10])

sum(water$Potability == 1)/nrow(water)

boxplot(water.imp[,1:9], col = c("red", "blue", "green", "orange", "purple", "brown", "turquoise", "hotpink", "orange4"), main="Boxplots of Imputed Data Set")

boxplot(water.na[,1:9], col = c("red", "blue", "green", "orange", "purple", "brown", "turquoise", "hotpink", "orange4"), main="Boxplots of Reduced Data Set")

apply(water.na[,1:9], 2, var)

s.std.na = cov(water.na[,1:9])

apply(water.imp[,1:9], 2, var)

s.std.imp = cov(water.imp[,1:9])

cors.na <- cor(water.na[,1:9]); corrplot(cors.na)

cors.imp <- cor(water.imp[,1:9]); corrplot(cors.imp)

set.seed(448)

tr <- sample(1:nrow(water.na), 0.8\*nrow(water.na))

na.train <- water.na[tr,]

na.test <- water.na[-tr,]

train <- sample(1:nrow(water.imp), 0.8\*nrow(water.imp))

imp.train <- water.imp[train,]

imp.test <- water.imp[-train,]

print("The number of 0's and 1's in the NA-omitted training data"); summary(na.train$Potability)

print("The number of 0's and 1's in the NA-imputed training data"); summary(imp.train$Potability)

print("The proportion of 1's in the Na-omitted training data");sum(na.train$Potability == 1)/nrow(na.train)

print("The proportion of 1's in the Na-omitted test data");sum(na.test$Potability == 1)/nrow(na.test)

print("The proportion of 1's in the Na-imputed training data");sum(imp.train$Potability == 1)/nrow(imp.train)

print("The proportion of 1's in the Na-imputed test data");sum(imp.test$Potability == 1)/nrow(imp.test)

fit.na = glm(Potability ~ ph + Hardness + Solids + Chloramines + Sulfate + Conductivity + Organic\_carbon + Trihalomethanes + Turbidity, data = water.na, family = "binomial")

fit.imp = glm(Potability ~ ph + Hardness + Solids + Chloramines + Sulfate + Conductivity + Organic\_carbon + Trihalomethanes + Turbidity, data = water.imp, family = "binomial")

summary(fit.na); summary(fit.imp)

stepCriterion(fit.na, criterion = "p-value", direction = "forward", levels = c(0.10,0.10))

stepCriterion(fit.na, criterion = "p-value",direction = "backward", levels = c(0.10,0.10))

stepCriterion(fit.imp, criterion = "p-value", direction = "forward", levels = c(0.10,0.10))

stepCriterion(fit.imp, criterion = "p-value", direction = "backward", levels = c(0.10,0.10))

mod.na <- lm(as.numeric(Potability ) ~ ph + Hardness + Solids + Chloramines + Sulfate + Conductivity + Organic\_carbon + Trihalomethanes + Turbidity, data = water.na)

mod.imp <- lm(as.numeric(Potability) ~ ph + Hardness + Solids + Chloramines + Sulfate + Conductivity + Organic\_carbon + Trihalomethanes + Turbidity, data = water.imp)

summary(mod.na); summary(mod.imp)

ols\_step\_forward\_p(mod.na, penter=0.10); ols\_step\_forward\_p(mod.imp, 0.1)

ols\_step\_backward\_p(mod.na, prem=0.1); ols\_step\_backward\_p(mod.imp, prem=0.1)

ols\_step\_both\_p(mod.na,penter=0.3, prem=0.3); ols\_step\_both\_p(mod.imp, penter=0.3, prem=0.3)

k<-ols\_step\_all\_possible(mod.na)

which.max(k$adjr); k$predictors[10]

l <- ols\_step\_all\_possible(mod.imp)

which.max(l$adjr); l$predictors[46]

b.na <- regsubsets(water.na[,1:9], water.na[,10], nvmax=9, method="backward")

f.na <- regsubsets(water.na[,1:9], water.na[,10], nvmax=9, method="forward")

s.na <- regsubsets(water.na[,1:9], water.na[,10], nvmax=9, method="seqrep")

e.na <- regsubsets(water.na[,1:9], water.na[,10], nvmax=9, method="exhaustive")

summary(b.na); which.max(summary(b.na)$adjr2)

summary(f.na); which.max(summary(f.na)$adjr2)

summary(s.na); which.max(summary(s.na)$adjr2)

summary(e.na); which.max(summary(e.na)$adjr2)

b.imp <- regsubsets(water.imp[,1:9], water.imp[,10], nvmax=9, method="backward")

f.imp <- regsubsets(water.imp[,1:9], water.imp[,10], nvmax=9, method="forward")

s.imp <- regsubsets(water.imp[,1:9], water.imp[,10], nvmax=9, method="seqrep")

e.imp <- regsubsets(water.imp[,1:9], water.imp[,10], nvmax=9, method="exhaustive")

summary(b.imp); which.max(summary(b.imp)$adjr2)

summary(f.imp); which.max(summary(f.imp)$adjr2)

summary(s.imp); which.max(summary(s.imp)$adjr2)

summary(e.imp); which.max(summary(e.imp)$adjr2)

x.natrain <- as.matrix(na.train[,1:9]); y.natrain <- na.train[,10]

x.natest <- as.matrix(na.test[,1:9]) ; y.natest <- na.test[,10]

x.imptrain <- as.matrix(imp.train[,1:9]); y.imptrain <- imp.train[,10]

x.imptest <- as.matrix(imp.test[,1:9]); y.imptest <- imp.test[,10]

grid=10^seq(10,-5,length=100)

out.na = glmnet(x.natrain,y.natrain,alpha=0, lambda=grid, family="binomial", thresh=1e-12)

cv.outna=cv.glmnet(x.natrain,y.natrain,alpha=0, lambda=grid, family="binomial", thresh=1e-12)

plot(cv.outna)#CV error plot

bestlam.na=cv.outna$lambda.min

ridge.namod=glmnet(x.natrain,y.natrain,alpha=0,lambda=bestlam.na, family="binomial")

ridge.predna=predict(ridge.namod,s=bestlam.na,newx=x.natest)

out.imp = glmnet(x.imptrain,y.imptrain,alpha=0, lambda=grid, family="binomial", thresh=1e-12)

cv.outimp=cv.glmnet(x.imptrain,y.imptrain,alpha=0, lambda=grid, family="binomial", thresh=1e-12)

plot(cv.outimp)#CV error plot

bestlam.imp=cv.outimp$lambda.min

ridge.impmod=glmnet(x.imptrain,y.imptrain,alpha=0,lambda=bestlam.imp, family="binomial")

ridge.predimp=predict(ridge.impmod,s=bestlam.imp,newx=x.imptest, type="coefficients")

coef(ridge.namod); coef(ridge.impmod)

plot(out.na, label=T); plot(out.imp, label=T)

plot(ridge.impmod, label=T); plot(ridge.namod, label=T)

lass.namod=glmnet(x.natrain,y.natrain,alpha=1,lambda=grid, family="binomial")

lass.impmod=glmnet(x.imptrain,y.imptrain,alpha=1,lambda=grid, family="binomial")

plot(lass.namod, label=T); plot(lass.impmod, label=T)

lass.nacv = cv.glmnet(x.natrain,y.natrain,alpha=1,lambda=grid, family="binomial")

lass.impcv = cv.glmnet(x.imptrain,y.imptrain,alpha=1,lambda=grid, family="binomial")

na.bestlam=lass.nacv$lambda.min

imp.bestlam=lass.impcv$lambda.min

lasso.nacoef=predict(lass.namod,type="coefficients",s=na.bestlam)

lasso.impcoef=predict(lass.impmod,type="coefficients",s=imp.bestlam)

plot(water.imp$Solids, water.imp$Organic\_carbon, col=as.factor(water.imp$Potability), xlab="Solids", ylab="Organic Carbon", main="Potability (Imputed)")

legend("topright", legend=c("Potable", "Non-Potable"), col=c("red", "black"), pch=1)

plot(water.imp$Solids, water.imp$Chloramines, col=as.factor(water.imp$Potability), xlab="Solids", ylab="Chloramines", main="Potability (Imputed)")

legend("topright", legend=c("Potable", "Non-Potable"), col=c("red", "black"), pch=1)

plot(water.na$Solids, water.na$Chloramines, col=as.factor(water.imp$Potability), xlab="Solids", ylab="Chloramines", main="Potability (Omitted)")

legend("topright", legend=c("Potable", "Non-Potable"), col=c("red", "black"), pch=1)

plot(water$Solids, water$Chloramines, col=water$Potability,xlab="Solids", ylab="Chloramines", main="Potability")

legend("topright", legend=c("Potable", "Not Potable"), col=c("red", "black"), pch=1)

plot(water$Solids, water$Organic\_carbon, col=water$Potability,xlab="Solids", ylab="Organic Carbon", main="Potability")

legend("topright", legend=c("Potable", "Not Potable"), col=c("red", "black"), pch=1)

plot(water.imp$Sulfate, water.imp$Hardness, col=as.factor(water.imp$Potability), xlab="Sulfate", ylab="Hardness", main="Potability (Imputed)")

legend("bottomleft", legend=c("Potable", "Non-Potable"), col=c("red", "black"), pch=1)

plot(water.imp$Hardness, water.imp$ph, col=as.factor(water.imp$Potability), xlab="Hardness", ylab="pH", main="Potability (Imputed)")

legend("bottomleft", legend=c("Potable", "Non-Potable"), col=c("red", "black"), pch=1)

plot(water.na$Sulfate, water.na$ph, col=as.factor(water.imp$Potability), xlab="Sulfate", ylab="pH", main="Potability (Omitted)")

legend("bottomleft", legend=c("Potable", "Non-Potable"), col=c("red", "black"), pch=1)

glm.impid = glm(Potability~Solids+Organic\_carbon, data=imp.train, family=binomial(link="identity"))

pred.glmimp = predict(glm.impid, imp.test)

val.imp = ifelse(pred.glmimp<0.5, 1, 0)

mean(val.imp != y.imptest)#0.6356707

summary(glm.imp)

glm.impre = glm(Potability~Organic\_carbon, data=imp.train, family=binomial(link="identity"))

pred.impre = predict(glm.impre, imp.test)

val.impre = ifelse(pred.impre<0.5, 1, 0)

mean(val.impre != y.imptest)#0.6356707

summary(glm.impre)

imp.allglm = glm(Potability~., data=imp.train, family=binomial(link="identity"))

pred.allimp = predict(imp.allglm, imp.test)

values.allimp = ifelse(pred.glmimp<0.5, 1, 0)

mean(values.allimp != y.imptest)#0.6356707

glm.naid = glm(Potability~Solids+Chloramines, data=na.train, family=binomial(link="identity"))

pred.glmna = predict(glm.naid, na.test)

val.na = ifelse(pred.glmna<0.5, 1, 0)

mean(val.na != y.natest)#0.5955335

summary(glm.na)

na.allglm = glm(Potability~., data=na.train, family=binomial(link="identity"))

pred.allna = predict(na.allglm, na.test)

values.allna = ifelse(pred.allna<0.5, 1, 0)

mean(values.allna != y.natest)#0.5930521

summary(na.allglm)

glm.imp = glm(Potability~ Solids+Organic\_carbon, data=imp.train, family="binomial")

summary(glm.imp)

glm.imp2 = glm(Potability~ Solids+Organic\_carbon+Chloramines, data=imp.train, family="binomial")

summary(glm.imp2) #no improvement

glmimp.probs <- predict(glm.imp, type="response")

glmimp.pred <- rep(0,length(imp.train$Potability))

glmimp.pred[glmimp.probs>0.5]=1

glmimp.trtab <- table(glmimp.pred, imp.train$Potability)

train.err.imp <- mean(glmimp.pred != imp.train$Potability)

glmimp.probste <- predict(glm.imp, imp.test, type="response")

glm.imptepred <- rep(0, length(imp.test$Potability))

glm.imptepred[glmimp.probste>0.5]=1

glmimp.tetab <- table(glm.imptepred, imp.test$Potability)

test.err.imp <- mean(glm.imptepred != imp.test$Potability)

glm.na = glm(Potability~Solids+Chloramines, data=na.train, family="binomial")

summary(glm.na)

glmna.probs <- predict(glm.na, type="response")

glmna.pred <- rep(0,length(na.train$Potability))

glmna.pred[glmna.probs>0.5]=1

glmna.trtab <- table(glmna.pred, na.train$Potability)

train.err.na <- mean(glmna.pred != na.train$Potability)

glmna.probste <- predict(glm.na, na.test, type="response")

glm.natepred <- rep(0, length(na.test$Potability))

glm.natepred[glmna.probste>0.5]=1

glmna.tetab <- table(glm.natepred, na.test$Potability)

test.err.na <- mean(glm.natepred != na.test$Potability)

lda.fitna <- lda(Potability~Solids+Chloramines, na.train)

ldatrain.napred <- predict(lda.fitna, na.train)

ldatrain.napredqual <- ldatrain.napred$class

lda.tr.natab <- table(ldatrain.napredqual, na.train$Potability)

lda.te.napred <- predict(lda.fitna, na.test)

ldatest.napredqual <- lda.te.napred$class

lda.te.natab <- table(ldatest.napredqual, na.test$Potability)

ldana.trerr <- lda.tr.natab[1,2]/sum(lda.tr.natab[1,])

ldana.teerr <- lda.te.natab[1,2]/sum(lda.te.natab[1,])

lda.fitna

lda.fitimp <- lda(Potability~Solids+Organic\_carbon, imp.train)

ldatrain.imppred <- predict(lda.fitimp, imp.train)

ldatrain.imppredqual <- ldatrain.imppred$class

lda.tr.imptab <- table(ldatrain.imppredqual, imp.train$Potability)

lda.te.imppred <- predict(lda.fitimp, imp.test)

ldatest.imppredqual <- lda.te.imppred$class

lda.te.imptab <- table(ldatest.imppredqual, imp.test$Potability)

ldaimp.trerr <- lda.tr.imptab[1,2]/sum(lda.tr.imptab[1,])

ldaimp.teerr <- lda.te.imptab[1,2]/sum(lda.te.imptab[1,])

lda.fitimp

k.na <- seq(from=1, to=500, by=2)

knntest.errs.na <- numeric(length(k.na)); knntr.errs.na <- numeric(length(k.na))

for(i in 1:length(k.na)){

knn.pred <- knn(x.natrain, x.natest, y.natrain, k=i)

knntest.errs.na[i] <- mean(knn.pred != y.natest)

knn.predtr <- knn(x.natrain, x.natrain, y.natrain, k=i)

knntr.errs.na[i] <- mean(knn.predtr != y.natrain)

}

plot(k.na, knntest.errs.na, main="Test Error For KNN (omitted dataset)", xlab="Values of K", ylab="Test Error")

abline(h=0.3424318)

k.na[which.min(knntest.errs.na)]; knntest.errs.na[which.min(knntest.errs.na)]; knntr.errs.na[which.min(knntest.errs.na)]; min(knntest.errs.na); which.min(knntest.errs.na); k.na[13]; k.na[39]; k.na[44]; k.na[48]

plot(k.na, knntest.errs.na, type="l", main="Test Error vs. Training Error For KNN (omitted dataset)", xlab="Values of K", ylab="Error", col="black", ylim=c(0, 0.46), lty=1) lines(k.na,knntr.errs.na, col="red", lty=2)

abline(h=0.3424318, col="blue", lty=4)

legend(300,0.15, legend=c("Test Error", "Training Error", "Minimum Test Error"), lty=c(1,2,4) , col=c("black", "blue", "darkred"))

plot(k.na[1:50], knntest.errs.na[1:50])

k.na1[which.min(knntest.errs.na1)]; knntest.errs.na1[which.min(knntest.errs.na1)]

k.imp <- seq(from=1, to=500, by=2)

knntest.errs.imp <- numeric(length(k.imp)); knntr.errs.imp <- numeric(length(k.imp))

for(i in 1:(length(k.imp))){

knn.pred <- knn(x.imptrain, x.imptest, y.imptrain, k=i)

knntest.errs.imp[i] <- mean(knn.pred != y.imptest)

knn.predtr <- knn(x.imptrain, x.imptrain, y.imptrain, k=i)

knntr.errs.imp[i] <- mean(knn.predtr != y.imptrain)

}

plot(k.na, knntest.errs.imp, main="Test Error For KNN (imputed dataset)", xlab="Values of K", ylab="Test Error")

abline(h=0.3185976)

min(knntest.errs.imp); which.min(knntest.errs.imp); k.na[34]

plot(k.na, knntest.errs.imp, type="l",main="Test Error vs. Training Error For KNN (imputed dataset)", xlab="Values of K", ylab="Error", col="black", lty=1, ylim=c(0, 0.42)) lines(k.imp,knntr.errs.imp, col="purple", lty=2)

abline(h=0.3185976, col="hotpink", lty=4)

legend(300,0.15, legend=c("Test Error", "Training Error", "Minimum Test Error"), lty=c(1,2,4), col=c("black", "purple", "hotpink"))

min(knntest.errs.imp); k.imp[which.min(knntest.errs.imp)]; knntr.errs.imp[which.min(knntest.errs.imp)]

na.tree = tree(Potability~., na.train, split="gini")

summary(na.tree)

plot(na.tree); text(na.tree, pretty=0)

imp.tree = tree(Potability~., imp.train, split="gini")

summary(imp.tree)

plot(imp.tree); text(imp.tree, pretty=0)

na.predtree = predict(na.tree, na.test, type="class")

table(na.predtree, y.natest)

(81+85)/(81+85+155+82)

imp.predtree = predict(imp.tree, imp.test, type="class")

table(imp.predtree, y.imptest); (113+134)/(113+134+283+126)

cv.natree = cv.tree(na.tree, FUN=prune.misclass)

summary(cv.natree)

par(mfrow=c(1,2))

plot(cv.natree$size, cv.natree$dev, type="b", xlab="Numer of Terminal Nodes", ylab="CV Errors", main="Error Rate vs Tree Size")

plot(cv.natree$k, cv.natree$dev, type="b", xlab="Cost-Complexity Parameter", ylab="CV Errors", main="Error Rate vs Cost-Complexity")

par(mfrow=c(1,1))

min(cv.natree$dev); cv.natree$size[which.min(cv.natree$dev)]

na.prunetree = prune.misclass(na.tree, best=cv.natree$size[which.min(cv.natree$dev)])

plot(na.prunetree); text(na.prunetree, pretty=0)

summary(na.prunetree)

naprune.pred= predict(na.prunetree, na.test, type="class")

table(naprune.pred, y.natest); (81+84)/(156+81+84+82)

na.prunetree2 = prune.misclass(na.tree, best=4)

plot(na.prunetree2); text(na.prunetree2, pretty=0)

summary(na.prunetree2)

naprune.pred2= predict(na.prunetree2, na.test, type="class")

table(naprune.pred2, y.natest); (132+13)/(237+31+132+13)

cv.imptree = cv.tree(imp.tree, FUN=prune.misclass)

summary(cv.imptree)

par(mfrow=c(1,2))

plot(cv.imptree$size, cv.imptree$dev, type="b", xlab="Numer of Terminal Nodes", ylab="CV Errors", main="Error Rate vs Tree Size")

plot(cv.imptree$k, cv.imptree$dev, type="b", xlab="Cost-Complexity Parameter", ylab="CV Errors", main="Error Rate vs Cost-Complexity")

par(mfrow=c(1,1))

imp.prunetree = prune.misclass(imp.tree, best=185)

plot(imp.prunetree); text(imp.prunetree, pretty=0)

summary(imp.prunetree)

imp.prunepred = predict(imp.prunetree, imp.test, type="class")

table(imp.prunepred, y.imptest); (108+141)/(276+108+141+131)

imp.prunetree2 = prune.misclass(imp.tree, best=4)

plot(imp.prunetree2); text(imp.prunetree2, pretty=0)

summary(imp.prunetree2)

imp.prunepred2 = predict(imp.prunetree2, imp.test, type="class")

table(imp.prunepred2, y.imptest); (205+16)/(401+205+16+34)

best\_seq <- seq(from=2, to=150, by=2)

treena.err <- numeric(length(best\_seq))

treeimp.err <- numeric(length(best\_seq))

for(i in 1:length(best\_seq)){

prune.na <- prune.misclass(na.tree, best=best\_seq[i])

prunena.pred <- predict(prune.na, na.test, type="class")

treena.err[i] <- mean(prunena.pred != y.natest)

prune.imp <- prune.misclass(imp.tree, best=best\_seq[i])

pruneimp.pred <- predict(prune.imp, imp.test, type="class")

treeimp.err[i] <- mean(pruneimp.pred != y.imptest)

}

plot(best\_seq, treena.err, main="Pruned Classification Trees (NA set)", xlab="Number of Terminal Nodes", ylab="Test Classification Error",type="b")

plot(best\_seq, treeimp.err, main="Pruned Classification Trees (IMP set)", xlab="Number of Terminal Nodes", ylab="Test Classification Error",type="b")

min(treena.err); which.min(treena.err); min(treeimp.err); which.min(treeimp.err)

na.prunetree3 = prune.misclass(na.tree, best=7)

plot(na.prunetree3); text(na.prunetree3, pretty=0)

summary(na.prunetree3)

imp.prunetree3 = prune.misclass(imp.tree, best=5)

plot(imp.prunetree3); text(imp.prunetree3, pretty=0)

summary(imp.prunetree3)

na.bag = randomForest(Potability~., na.train, mtry=dim(x.natrain)[2], importance=T)

na.yhatbag = predict(na.bag, na.test, type="class")

mean(na.yhatbag != y.natest); table(na.yhatbag, y.natest)

importance(na.bag); varImpPlot(na.bag)

imp.bag = randomForest(Potability~., imp.train, mtry=dim(x.imptrain)[2], importance=T)

imp.yhatbag = predict(imp.bag, imp.test, type="class")

mean(imp.yhatbag != y.imptest); table(imp.yhatbag, y.imptest)

varImpPlot(imp.bag)

trees <- seq(from=2, to=500, by=1)

bagna.testerr = rep(NA, length(trees)); bagimp.testerr = rep(NA, length(trees)); rfna.testerr = rep(NA, length(trees)); rfimp.testerr = rep(NA, length(trees))

for(i in 1: length(trees)){

bagna.p <- randomForest(x.natrain, y = y.natrain, xtest = x.natest, ytest = y.natest, mtry = ncol(x.natrain), ntree = trees[i])

bagimp.p <- randomForest(x.imptrain, y = y.imptrain, xtest = x.imptest, ytest = y.imptest, mtry = ncol(x.imptrain), ntree = trees[i])

bagna.testerr[i] = mean(bagna.p$test$err.rate)

bagimp.testerr[i] = mean(bagimp.p$test$err.rate)

rfna.p <- randomForest(x.natrain, y = y.natrain, xtest = x.natest, ytest = y.natest, mtry = ncol(x.natrain) - 1, ntree = trees[i])

rfimp.p <- randomForest(x.imptrain, y = y.imptrain, xtest = x.imptest, ytest = y.imptest, mtry = ncol(x.imptrain) - 1, ntree = trees[i])

rfna.testerr[i] <- mean(rfna.p$test$err.rate)

rfimp.testerr[i] <- mean(rfimp.p$test$err.rate)

}

plot(trees, rfna.testerr, type="l", xlab="Tree Size", ylab="Error", main="RF (m=p-1) Tree Size vs. Error (NA set)")

plot(trees,rfimp.testerr, type="l", xlab="Tree Size", ylab="Error", main="RF (m=p-1) Tree Size vs. Error (IMP set)")

rf2na.testerr = rep(NA, length(trees)); rf2imp.testerr = rep(NA, length(trees)); rf3na.testerr = rep(NA, length(trees)); rf3imp.testerr = rep(NA, length(trees))

for(i in 1:length(trees)){

rfna.p2 <- randomForest(x.natrain, y = y.natrain, xtest = x.natest, ytest = y.natest, mtry = ncol(x.natrain)/3, ntree = trees[i])

rfimp.p2 <- randomForest(x.imptrain, y = y.imptrain, xtest = x.imptest, ytest = y.imptest, mtry = ncol(x.imptrain)/3, ntree = trees[i])

rf2na.testerr[i] = mean(rfna.p2$test$err.rate)

rf2imp.testerr[i] = mean(rfimp.p2$test$err.rate)

rfna.p3 <- randomForest(x.natrain, y = y.natrain, xtest = x.natest, ytest = y.natest, mtry = sqrt(ncol(x.natrain)), ntree = trees[i])

rfimp.p3 <- randomForest(x.imptrain, y = y.imptrain, xtest = x.imptest, ytest = y.imptest, mtry = sqrt(ncol(x.imptrain)), ntree = trees[i])

rf3na.testerr[i] = mean(rfna.p3$test$err.rate)

rf3imp.testerr[i] = mean(rfimp.p3$test$err.rate)

}

plot(trees,bagna.testerr, col = "green", type = "l", xlab = "Number of Trees", ylab = "Test Classification Error", main = "Random Forest Classification Error: Water Potability (NA)", ylim=c(0.32, 0.47) )

lines(trees,rfna.testerr, col="orange", type="l" )

lines(trees, rf2na.testerr, col = "red", type = "l")

lines(trees, rf3na.testerr, col = "blue", type = "l")

legend("topright", c("m = p (bagging)", "m=p-1", "m = p/3", "m = sqrt(p)"), col = c("green", "orange", "red", "blue"), cex =1, lty = 1)

plot(trees,bagimp.testerr, col = "green", type = "l", xlab = "Number of Trees", ylab = "Test Classification Error", main = "Random Forest Classification Error: Water Potability (IMP)", ylim=c(0.33, 0.46) )

lines(trees, rfimp.testerr, col="orange", type="l")

lines(trees, rf2imp.testerr, col = "red", type = "l")

lines(trees, rf3imp.testerr, col = "blue", type = "l")

legend("topright", c("m = p (bagging)","m=p-1", "m = p/3", "m = sqrt(p)"), col = c("green", "orange","red", "blue"), cex =1, lty = 1)

min(bagna.testerr);which.min(bagna.testerr) ;min(rfna.testerr); which.min(rfna.testerr); trees[which.min(rfna.testerr)]; min(rf2na.testerr); which.min(rf2na.testerr); trees[which.min(rf2na.testerr)]; min(rf3na.testerr); which.min(rf3na.testerr); trees[which.min(rf3na.testerr)]; varImpPlot(na.bag); varImpPlot(rfna.p); varImpPlot(rfna.p2); varImpPlot(rfna.p3)

min(bagimp.testerr); which.min(bagimp.testerr);min(rfimp.testerr); which.min(rfimp.testerr); trees[which.min(rfimp.testerr)]; min(rf2imp.testerr); which.min(rf2imp.testerr); trees[which.min(rf2imp.testerr)]; min(rf3imp.testerr); which.min(rf3imp.testerr); trees[which.min(rf3imp.testerr)]; varImp(imp.bag); varImpPlot(rfimp.p); varImpPlot(rfimp.p2); varImpPlot(rfimp.p3)

natune.out1=tune(svm,Potability~.,data=na.train,kernel="linear",ranges=list(cost=c(0.000005, 0.000001, 0.00005, 0.00001,0.0005, 0.0001, 0.001, 0.01, 0.1, 1, 5, 10)), scale=FALSE)

summary(natune.out1)

bestmod.na1=natune.out1$best.model

summary(bestmod.na1)

imptune.out1=tune(svm,Potability~.,data=imp.train,kernel="linear",ranges=list(cost=c(0.000005, 0.000001, 0.00005, 0.00001,0.0005, 0.0001, 0.001, 0.01, 0.1, 1, 5, 10)), scale=FALSE)

summary(imptune.out1)

bestmod.imp1=imptune.out1$best.model

summary(bestmod.imp1)

na.svmfit = svm(Potability~., data=na.train, kernel="linear", cost=0.000005,scale=FALSE, decision.values=T)

na.yhat1 = predict(na.svmfit, x.natest)

mean(na.yhat1 != y.natest)

imp.svmfit = svm(Potability~., data=imp.train, kernel="linear", cost=0.000005, scale=FALSE, decision.values=T)

imp.yhat1 = predict(imp.svmfit, x.imptest)

mean(imp.yhat1 != y.imptest)

na.svmfitb = svm(Potability~., data=na.train, kernel="linear", cost=0.001,scale=FALSE)

na.yhat1b = predict(na.svmfitb, x.natest)

mean(na.yhat1b != y.natest) #0.4044665, same

imp.svmfitb = svm(Potability~., data=imp.train, kernel="linear", cost=0.001, scale=FALSE)

imp.yhat1b = predict(imp.svmfitb, x.imptest)

mean(imp.yhat1b != y.imptest)

natune.out2 = tune(svm,Potability~.,data=na.train,kernel="polynomial", degree=2, ranges = list(cost=c( 0.0001, 0.01, 0.1, 1,5,10)))

summary(natune.out2)

bestmod.na2 = natune.out2$best.model

summary(bestmod.na2)

na.svmfit2 = svm(Potability~., data=na.train, kernel="polynomial", degree=2, cost=10, scale=FALSE, decision.values=T)

na.yhat2 = predict(na.svmfit2, x.natest)

mean(na.yhat2 != y.natest)

imptune.out2 = tune(svm,Potability~.,data=imp.train,kernel="polynomial", degree=2, ranges=list(cost=c(0.0001, 0.01, 0.1, 1,5,10)))

summary(imptune.out2)

bestmod.imp2 = imptune.out2$best.model

summary(bestmod.imp2)

imp.svmfit2 = svm(Potability~., data=imp.train, kernel="polynomial", degree=2, cost=10, scale=FALSE, decision.values=T)

imp.yhat2 = predict(imp.svmfit2, x.imptest)

mean(imp.yhat2 != y.imptest)

natune.out3 = tune(svm,Potability~.,data=na.train,kernel="polynomial", degree=3, ranges = list(cost=c( 0.0001, 0.01, 0.1, 1,5,10)))

summary(natune.out3)

bestmod.na3 = natune.out3$best.model

summary(bestmod.na3)

na.svmfit3 = svm(Potability~., data=na.train, kernel="polynomial", degree=3, cost=0.1, scale=FALSE)

na.yhat3 = predict(na.svmfit3, x.natest)

mean(na.yhat3 != y.natest)

imptune.out3 = tune(svm,Potability~.,data=imp.train,kernel="polynomial", degree=3, ranges = list(cost=c( 0.0001, 0.01, 0.1, 1,5,10)))

summary(imptune.out3)

bestmod.imp3 = imptune.out3$best.model

summary(bestmod.imp3)

imp.svmfit3 = svm(Potability~., data=imp.train, kernel="polynomial", degree=3, cost=1, scale=FALSE)

imp.yhat3 = predict(imp.svmfit3, x.imptest)

mean(imp.yhat3 != y.imptest) #0.3460366

natune.out4 = tune(svm,Potability~.,data=na.train,kernel="polynomial", degree=4, ranges = list(cost=c( 0.0001, 0.01, 0.1, 1,5,10)))

summary(natune.out4)

bestmod.na4 = natune.out4$best.model

summary(bestmod.na4)

na.svmfit4 = svm(Potability~., data=na.train, kernel="polynomial", degree=4, cost=1, scale= FALSE, decision.values=T)

na.yhat4 = predict(na.svmfit4, x.natest)

mean(na.yhat4 != y.natest)

imptune.out4 = tune(svm,Potability~.,data=imp.train,kernel="polynomial", degree=4, ranges = list(cost=c( 0.0001, 0.01, 0.1, 1,5,10)))

summary(imptune.out4)

bestmod.imp4 = imptune.out4$best.model

summary(bestmod.imp4)

imp.svmfit4 = svm(Potability~., data=imp.train, kernel="polynomial", degree=4, cost=1, scale= FALSE, decision.values=T)

imp.yhat4 = predict(imp.svmfit4, x.imptest)

mean(imp.yhat4 != y.imptest)

natune.outr=tune(svm, Potability~., data=na.train, kernel="radial", ranges = list(cost=c(0.01,0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4)))

summary(natune.outr)

summary(natune.outr$best.model)

na.svmfitr = svm(Potability~., data=na.train, kernel="radial", cost=1, scale= FALSE, decision.values = T)

imptune.outr=tune(svm, Potability~., data=imp.train, kernel="radial", ranges = list(cost=c(0.01,0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4)))

summary(imptune.outr)

summary(imptune.outr$best.model)

imp.svmfitr = svm(Potability~., data=imp.train, kernel="radial", cost=1, scale= FALSE, decision.values = T)

mean(predict(natune.outr$best.model,newdata=x.natest) != y.natest)

mean(predict(imptune.outr$best.model,newdata=x.imptest) != y.imptest) #0.3292683

na.svmfitb; imp.svmfitb; na.svmfit2; imp.svmfit; na.svmfit3; imp.svmfit3; na.svmfit4; imp.svmfit4; na.svmfitr; imp.svmfitr

na.svmfitb.weights = t(na.svmfitb$SV) %\*% na.svmfitb$coefs

imp.svmfitb.weights = t(imp.svmfitb$SV) %\*% imp.svmfitb$coefs

imp.svmfit2.weights = t(imp.svmfit2$SV) %\*% imp.svmfit2$coefs

na.svmfit2.weights = t(na.svmfit2$SV) %\*% na.svmfit2$coefs

na.svmfit3.weights = t(na.svmfit3$SV) %\*% na.svmfit3$coefs

imp.svmfit3.weights = t(imp.svmfit3$SV) %\*% imp.svmfit3$coefs

na.svmfit4.weights = t(na.svmfit4$SV) %\*% na.svmfit4$coefs

imp.svmfit4.weights = t(imp.svmfit4$SV) %\*% imp.svmfit4$coefs

na.svmfitr.weights = t(na.svmfitr$SV) %\*% na.svmfitr$coefs

imp.svmfitr.weights = t(imp.svmfitr$SV) %\*% imp.svmfitr$coefs

rocplot=function(pred, truth, ...){

predob = prediction(pred, truth)

perf = performance(predob, "tpr", "fpr")

plot(perf,...)}

na.fitlin=attributes(predict(na.svmfitb, na.train, decision.values = TRUE))$decision.values

na.fitlintest = attributes(predict(na.svmfitb, na.test, decision.values = TRUE))$decision.values

na.fitted=attributes(predict(na.svmfit2, na.train, decision.values = TRUE))$decision.values

natest.fitted = attributes(predict(na.svmfit2, na.test, decision.values = TRUE))$decision.values

na.fitr = attributes(predict(na.svmfitr, na.train, decision.values = TRUE))$decision.values

na.fitrtest = attributes(predict(na.svmfitr, na.test, decision.values = TRUE))$decision.values

nafitpoly = attributes(predict(na.svmfit4, na.train, decision.values = TRUE))$decision.values

nafitpolytest= attributes(predict(na.svmfit4, na.test, decision.values = TRUE))$decision.values

rocplot(na.fitted,y.natrain, main="ROC: SVM Error Comparison (NA)", col="blue", lty=2); rocplot(natest.fitted,y.natest, add=T, col="red", lty=1); rocplot(na.fitlin, y.natrain, add=T, col="green", lty=2); rocplot(na.fitlintest, y.natest, add=T, col="purple", lty=1); rocplot(na.fitr, y.natrain, add=T, col="orange", lty=2); rocplot(na.fitrtest, y.natest, add=T, col="darkgreen", lty=1); rocplot(nafitpoly, y.natrain, add=T, col="turquoise", lty=2); rocplot(nafitpolytest, y.natest, add=T, col="red4", lty=1)

legend("topleft", legend=c("Training - Poly(2)", "Testing - Poly(2)", "Training - Linear", "Testing - Linear", "Training - Radial", "Testing - Radial","Training - Poly(4)", "Testing - Poly(4)"), lty=c(2,1,2,1,2,1, 2, 1), col=c("blue", "red", "green", "purple", "orange", "darkgreen", "turquoise", "red4"))

imp.fitlin=attributes(predict(imp.svmfitb, imp.train, decision.values = TRUE))$decision.values

imp.fitlintest = attributes(predict(imp.svmfitb, imp.test, decision.values = TRUE))$decision.values

imp.fitted=attributes(predict(imp.svmfit2, imp.train, decision.values = TRUE))$decision.values

imptest.fitted = attributes(predict(imp.svmfit2, imp.test, decision.values = TRUE))$decision.values

imp.fitr = attributes(predict(imp.svmfitr, imp.train, decision.values = TRUE))$decision.values

imp.fitrtest = attributes(predict(imp.svmfitr, imp.test, decision.values = TRUE))$decision.values

impfitpoly =attributes(predict(imp.svmfit4, imp.train, decision.values = TRUE))$decision.values

impfitpolytest = attributes(predict(imp.svmfit4, imp.test, decision.values = TRUE))$decision.values

rocplot(imp.fitted,y.imptrain, main="ROC: SVM Error Comparison (IMP)", col="blue", lty=2); rocplot(imptest.fitted,y.imptest, add=T, col="red", lty=1); rocplot(imp.fitlin, y.imptrain, add=T, col="green", lty=2); rocplot(imp.fitlintest, y.imptest, add=T, col="purple", lty=1); rocplot(imp.fitr, y.imptrain, add=T, col="orange", lty=2); rocplot(imp.fitrtest, y.imptest, add=T, col="darkgreen", lty=1); rocplot(impfitpoly, y.imptrain, add=T, col="turquoise", lty=2); rocplot(impfitpolytest, y.imptest, add=T, col="red4", lty=1)

legend("bottomright", legend=c("Training - Poly(2)", "Testing - Poly(2)", "Training - Linear", "Testing - Linear", "Training - Radial", "Testing - Radial","Training - Poly(4)", "Testing - Poly(4)"), lty=c(2,1,2,1,2,1), col=c("blue", "red", "green", "purple", "orange", "darkgreen", "turquoise", "red4"))