Buoy Classification

Why Classify Buoys?

There are in total 72 buoys in the dataset, and they were not classified when we first received the data. Although data of a buoy in the dataset is arranged in time order, and we are able to separate data of a buoy from that of another by noticing the time gap that exists when the data jumps to another buoy and starts from an earlier date again, ambiguity in buoy labelling raises a critical question in real life: what if data of different buoys are collected and recorded at the same time? Then how can we possibly differentiate a buoy from another? It is true that location information of buoys are contained in the dataset, and since there is usually continuity in location of a buoy, previous location data can help us infer buoy index; however, we also notice that buoys tend not to get fixed at a certain location, they float around in the ocean and sometimes, a buoy can travel 5 longitude degrees away from its original location. As a result, location data alone is not a reliable indicator to predict buoy index, and thus we are required to use more complicated tools to clearly label buoys.

Owing to insufficient data of most buoys in the dataset (more than ⅔ of all buoys have fewer than 500 data points, and if we included them in classification, we would have had difficulty dealing with an imbalanced dataset), we will use 2 buoys for classification, which have more than 2,000 data points. Models we use in this report set an example for classification of more than 2 buoys in the future. By comparing results from logistic regression, KNN classification and Classification Tree results, we will know which model is best tuned to this task.

Descriptive Analysis

The “El Nino” dataset in the A1 includes the data from 72 buoys. In A2, we chose buoy 2 since it has the most data points (2298 entries). In this assignment, we decided to include buoy 48, which has the second most data (2240 entries). Because these two buoys collected a similar amount of the data, we developed logistic regression and K-NN analysis to predict the data source, from buoy 2 or buoy 48. It is also reasonable to discover the discrepancy of data from these two buoys during the same period. As the Sea.Surface.Temp is the key measurement of the occurrence of El Nino, we plotted Zonal.Winds, Meridional Winds, Humidity and Air Temp against Sea.Surface.Temp in the scatterplot. The blue line in the scatter plot is a linear fitted model, and the red one is to show the curvilinear relationships. We also compare the data of buoy 2 to that of buoy 48 side by side.

Buoy 2 and 48 Statistics Comparison

Buoy 2

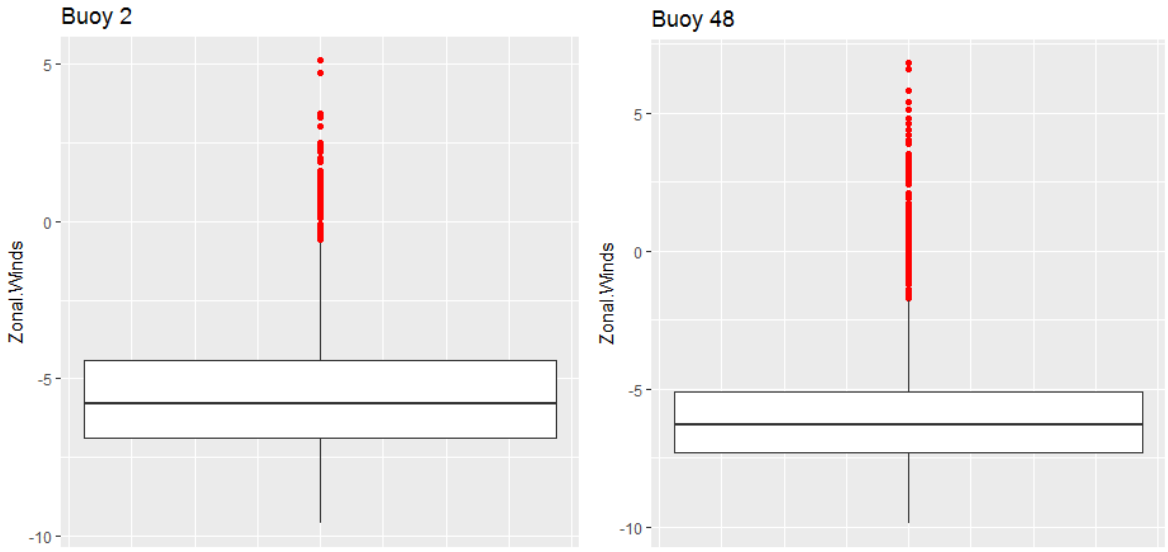
Variables	Min	Max	Mean	Median	Mode	Sd	Var
ZonalWinds	-9.60	5.10	-5.4357702	-5.80	-6.80	2.080190	4.327192
MeridionalWinds	-7.40	5.20	-0.1718016	-0.10	0.40	1.810364	3.277420
Humidity	67.80	95.80	81.6053525	81.50	79.80	3.805203	14.479566
AirTemp	22.32	29.56	26.0556745	26.03	25.96	1.255298	1.575773
SeaSurfaceTemp	22.24	30.07	26.5309443	26.38	26.05	1.468246	2.155746

Buoy 48

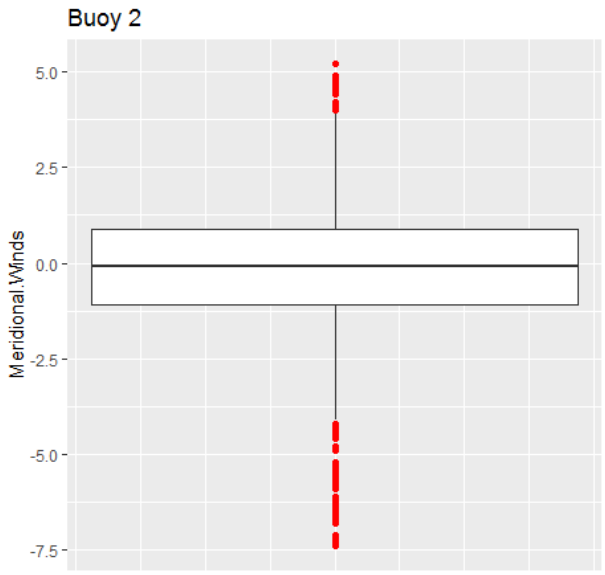
Variables48	Min	Max	Mean	Median	Mode	Sd	Var
ZonalWinds	-9.90	6.80	-5.813705	-6.30	-6.60	2.3311826	5.4344123
MeridionalWinds	-6.20	4.30	-1.000223	-1.00	-1.20	1.6569814	2.7455873
Humidity	65.10	91.80	79.190045	79.30	80.50	4.0351741	16.2826297
AirTemp	25.26	29.59	27.142607	27.09	26.76	0.8044046	0.6470667
SeaSurfaceTemp	25.20	30.70	27.642688	27.54	26.85	1.0308908	1.0627358

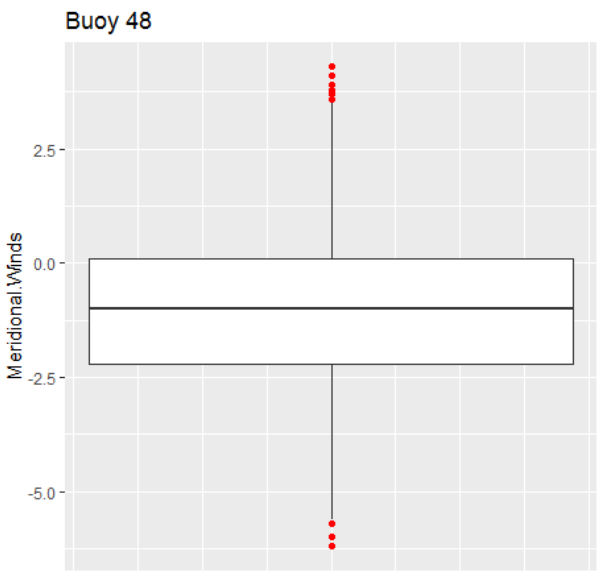
Descriptive Plots

Zonal Winds

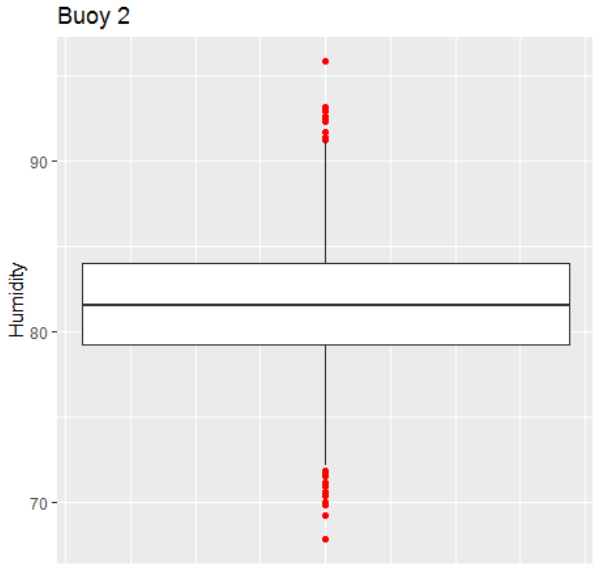


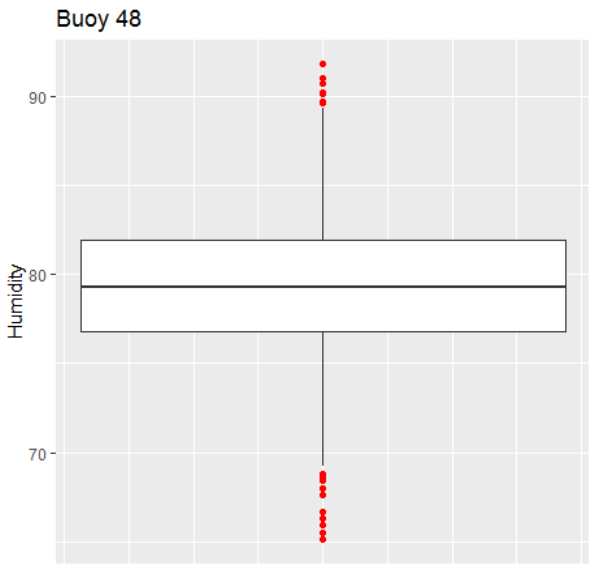
Meridional Winds



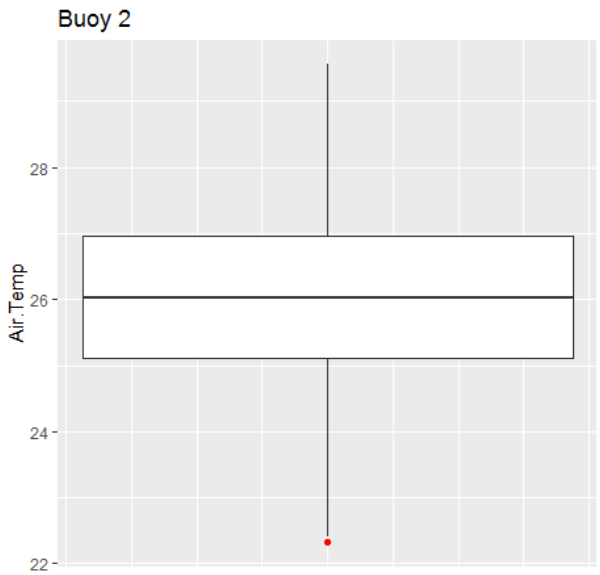


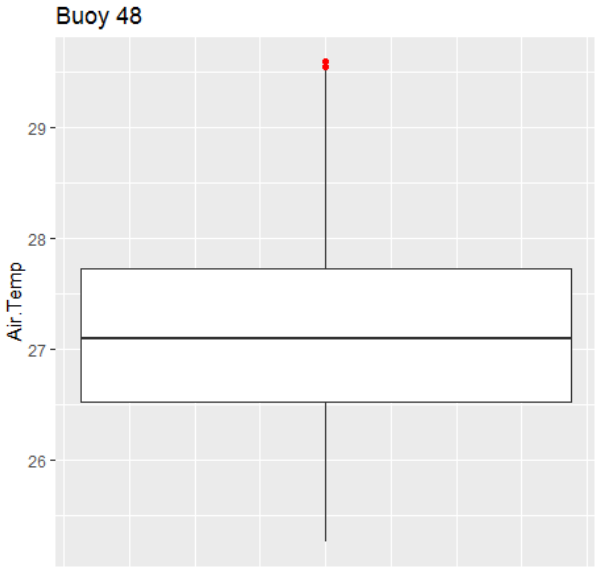
Humidity



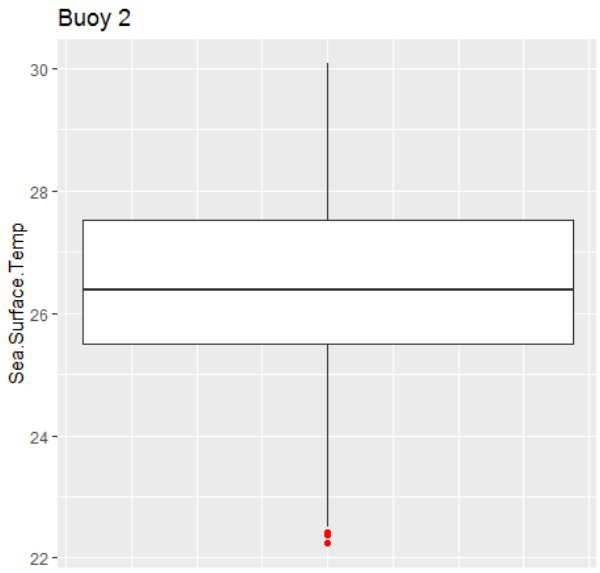


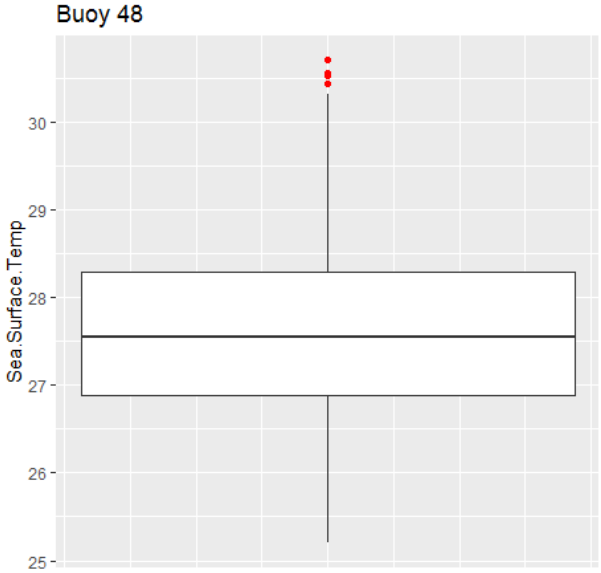
Air Temp





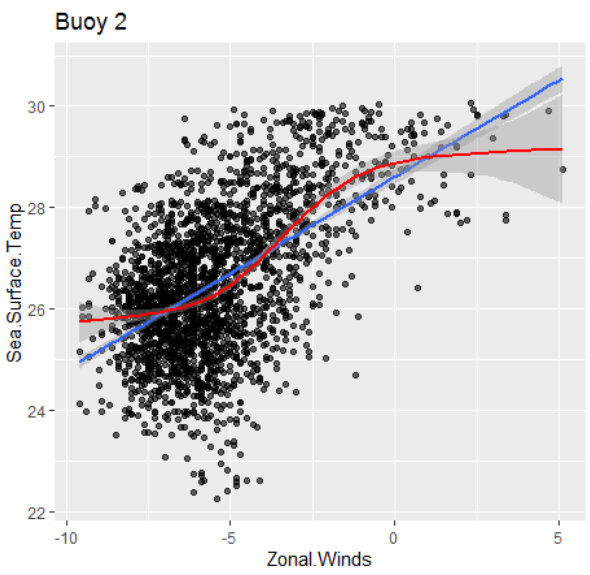
Sea Surface Temp

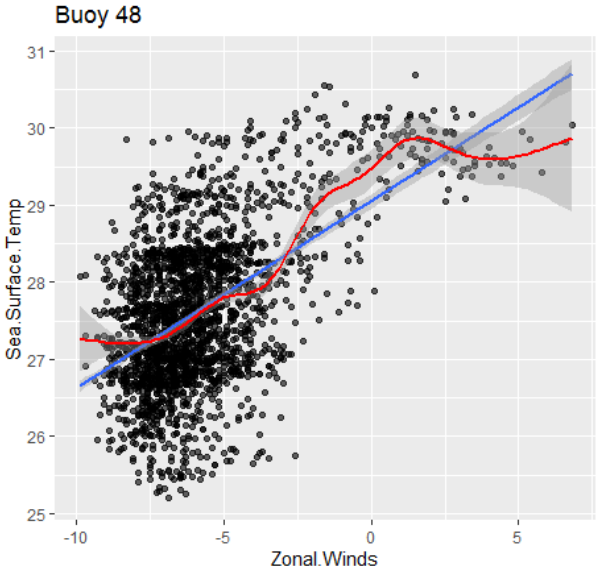




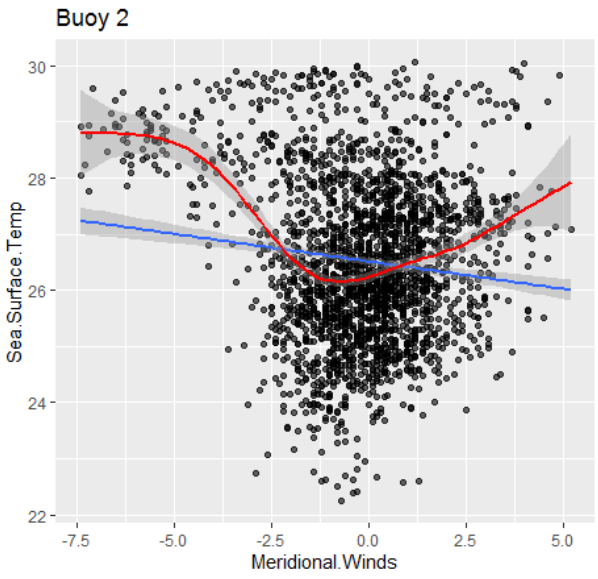
Plots

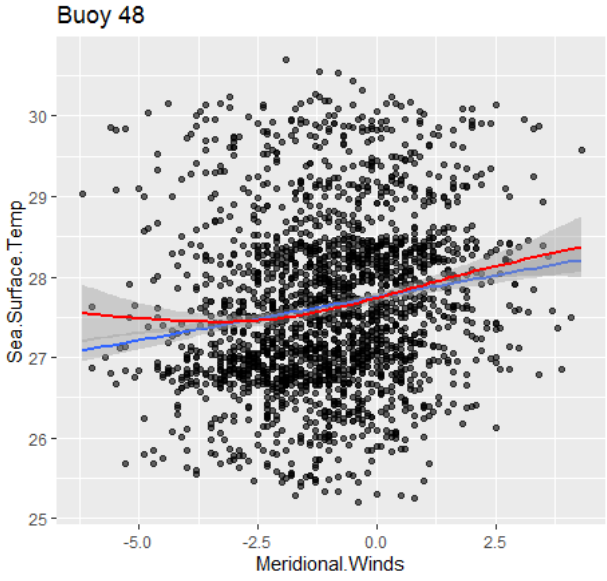
Zonal Winds



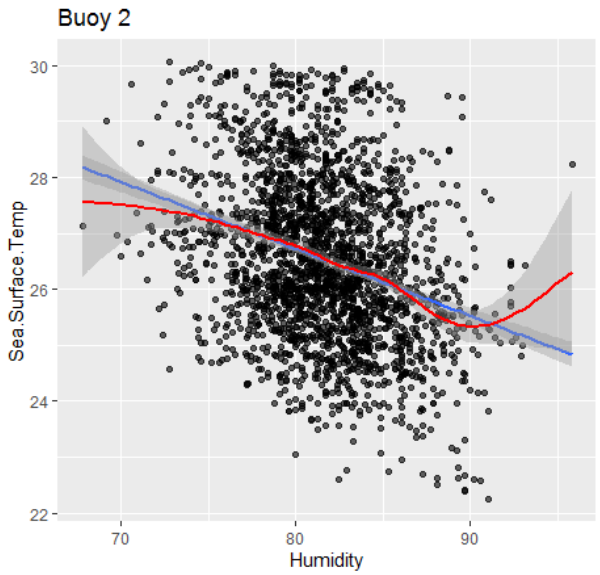


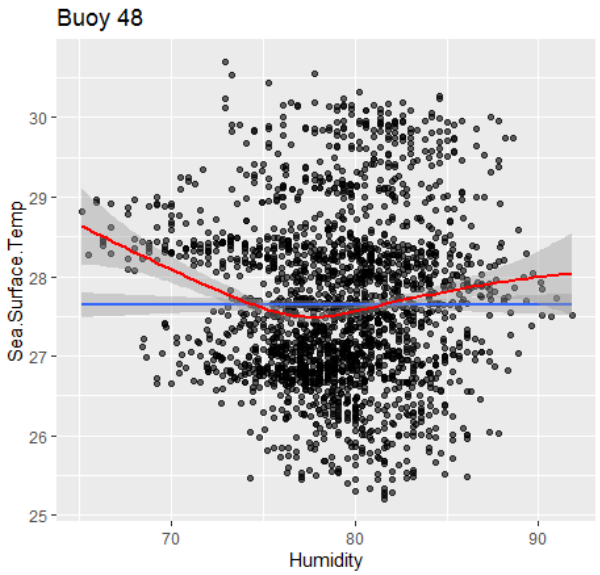
Meridional Winds



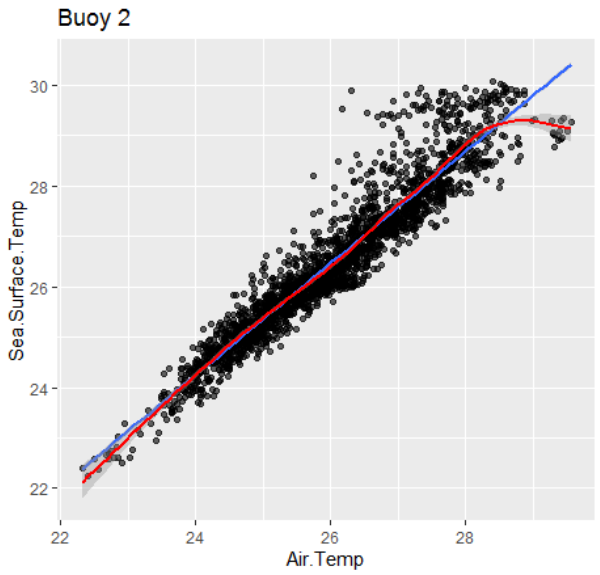


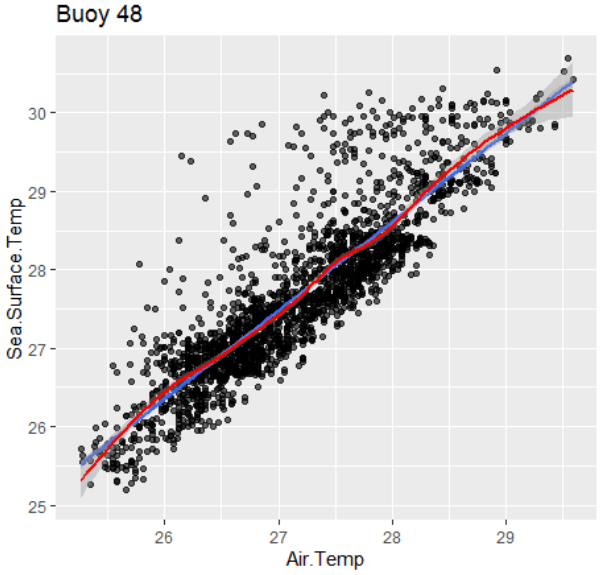
Humidity





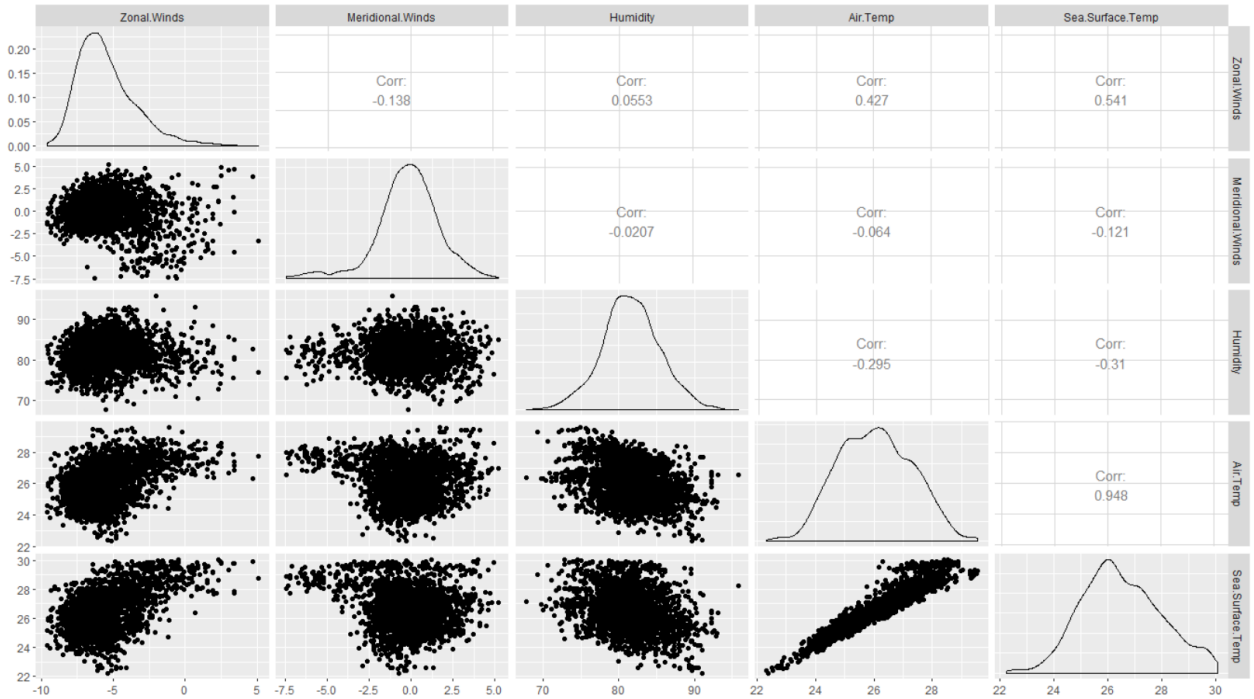
Air Temp



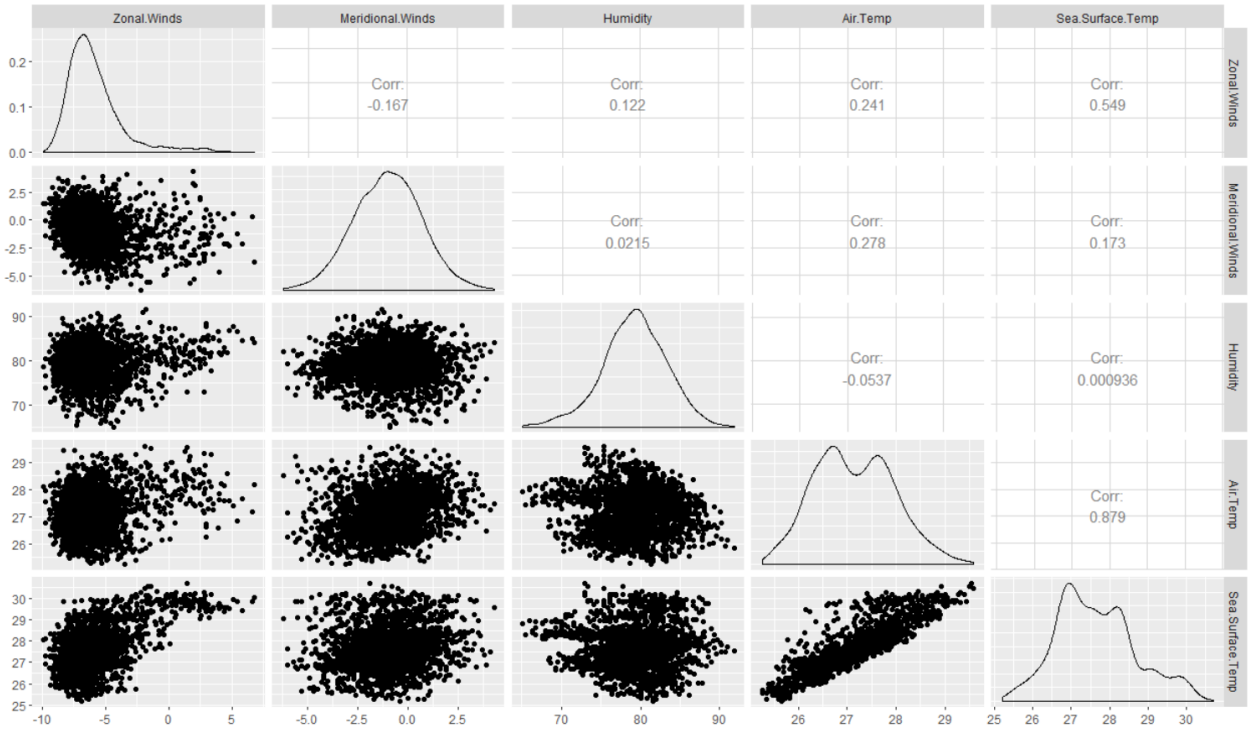


Matrix Plot

Buoy 2

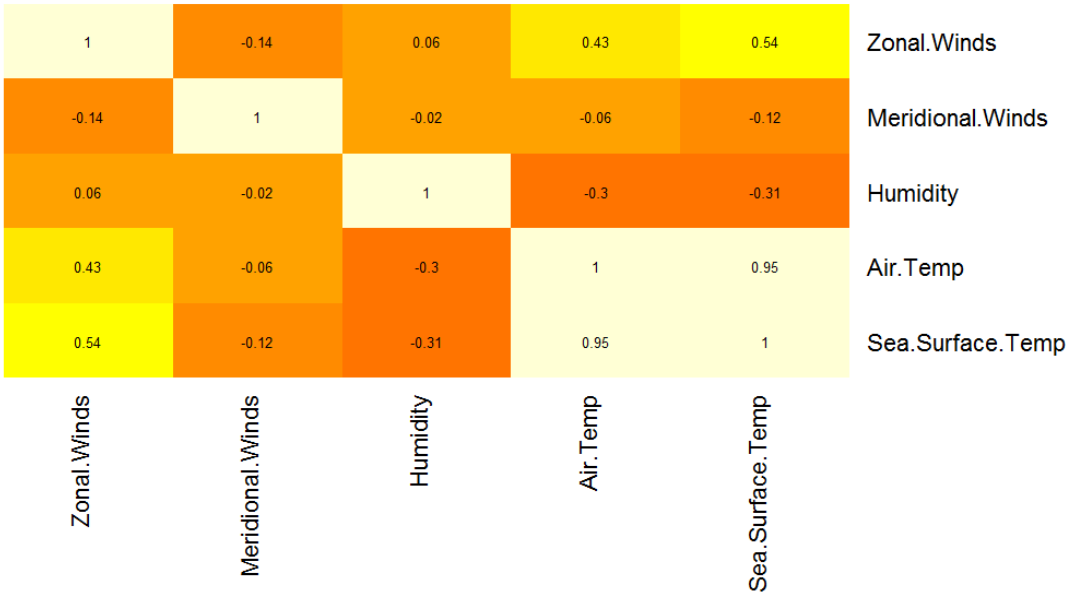

Zonal.Winds
Meridional.Winds
Humidity
Air.Temp
Sea.Surface.Temp
Corr:
-0.138
Corr:
0.0553
Corr:
0.427
Corr:
0.541
Corr:
-0.0207
Corr:
-0.064
Corr:
-0.121
Corr:
-0.295
Corr:
-0.31
Corr:
0.948

Buoy 48

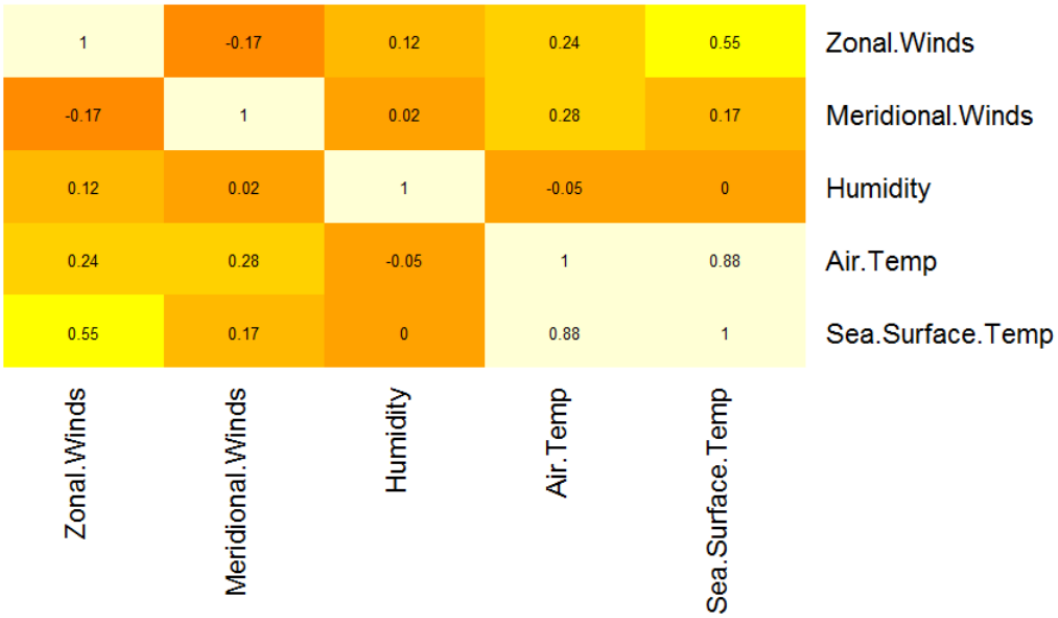

Zonal.Winds
Meridional.Winds
Humidity
Air.Temp
Sea.Surface.Temp
Corr:
-0.167
Corr:
0.122
Corr:
0.241
Corr:
0.549
Corr:
0.0215
Corr:
0.278
Corr:
0.173
Corr:
-0.0537
Corr:
0.000936
Corr:
0.879

Correlation Matrix

Buoy 2



Buoy 48



Logistic Regression

Logistic regression is used to model dichotomous outcome variables, and it calculates the odds ratio, the likelihood that the event will occur. The odds ratio represents the constant effect of

predictor variables, on the likelihood that the outcome will occur. We can only use logistic regression when we suspect that some certain data, missing the record of their buoy, belong to a certain buoy.

In the following example, we selected the data measured by Buoy 2 and Buoy 48, and used logistic regression to determine whether the data belong to Buoy 2 or not. First, we combined the data from Buoy 2 and Buoy 48, randomly selected 60% of them for training and 40% for validation. For the logistic regression, we ran the regression on “buoy” against “Sea.Surface.Temp”, “Air.Temp”, “Zonal.Winds”, “Meridional.Winds”, “Humidity”, and the interaction between these variables, which were all measured by the buoy and proven to be relevant to El Nino in our last assignment.

Here is the summary of the logistic regression:

```
glm(formula = buoy ~ Sea.Surface.Temp + Meridional.Winds + Zonal.Winds:Humidity +
    Zonal.Winds:Air.Temp + Meridional.Winds:Humidity + Meridional.Winds:Air.Temp +
    Humidity:Air.Temp, family = binomial(link = "logit"), data = train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.71396  -0.76595   0.08735   0.72296   3.01477

Coefficients:
                             Estimate Std. Error z value             Pr(>|z|)
(Intercept)                16.0516517  1.0849827  14.794 < 0.0000000000000002 ***
Sea.Surface.Temp           -0.0032199  0.0009706  -3.318             0.000908 ***
Meridional.Winds           10.2932899  1.0486301   9.816 < 0.0000000000000002 ***
Zonal.Winds:Humidity       -0.0388785  0.0033862 -11.482 < 0.0000000000000002 ***
Zonal.Winds:Air.Temp        0.1286113  0.0097684  13.166 < 0.0000000000000002 ***
Meridional.Winds:Humidity  -0.0322262  0.0071631  -4.499   0.0000068294426053 ***
Meridional.Winds:Air.Temp  -0.2726314  0.0304589  -8.951 < 0.0000000000000002 ***
Humidity:Air.Temp          -0.0053468  0.0007112  -7.518   0.0000000000000555 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4402.1  on 3175  degrees of freedom
Residual deviance: 2935.4  on 3168  degrees of freedom
AIC: 2951.4

Number of Fisher Scoring iterations: 5
```

All p-values of variables are < 0.05. The regression is a fair fit to the classification so we used this regression to predict the buoy of the validation data.

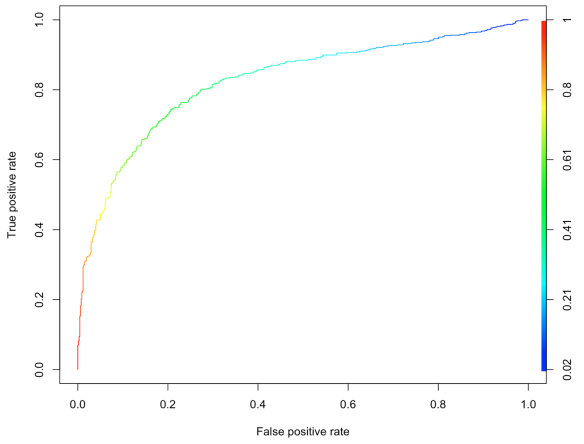
Here is a sample of the outcome of the prediction. “1” in Column Actual means that the buoy is Buoy 2, and “0” (though not in this table) for not Buoy 2. In Column Predicted, the results are

predicted as whether the data are from Buoy 2. If the predicted probability > 0.5, the buoy is predicted to be Buoy 2.

actual	predicted
1	0
1	1
1	1
1	0
1	1

Here are the confusion matrix of the prediction and ROC curve:

```
          Reference
Prediction   0   1
         0 545 184
         1 132 501

               Accuracy : 0.768
                 95% CI : (0.7446, 0.7902)
    No Information Rate : 0.5029
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5362

 Mcnemar's Test P-Value : 0.004118

            Sensitivity : 0.8050
            Specificity : 0.7314
         Pos Pred Value : 0.7476
         Neg Pred Value : 0.7915
             Prevalence : 0.4971
         Detection Rate : 0.4001
   Detection Prevalence : 0.5352
      Balanced Accuracy : 0.7682

       'Positive' Class : 0
```



The confusion matrix describes the performance of the logistic regression. We are specifically concerned with True Positive on the left up grid, the true prediction of positive cases; and True Negative on the right down grid, the true prediction of negative cases. By Observing the confusion matrix, we can calculate accuracy, the proportion of true prediction among all predictions; sensitivity, the proportion of observed positives that were predicted to be positive; and specificity, the proportion of observed negatives that were predicted to be negatives. Other two grids are both false predictions. ROC curve plots sensitivity, (True positive rate), against 1-specificity (False positive rate)

From the results we notice that the following:

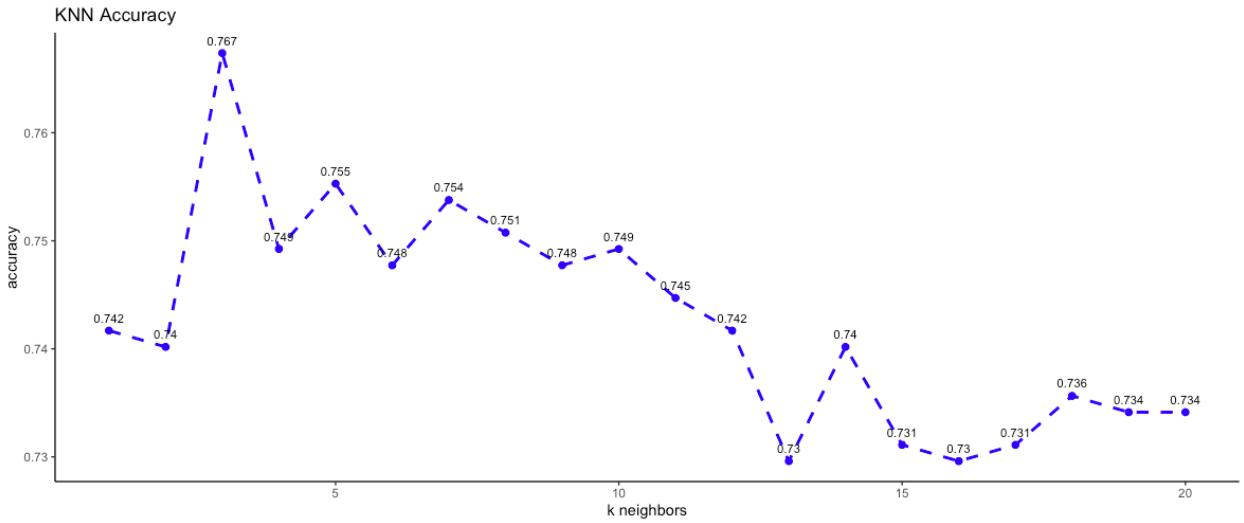
- Accuracy = 0.768
- Sensitivity = 0.805
- Specificity = 0.7314

Since the Accuracy, Sensitivity, and Specificity are high, the ROC curve is close to the left up corner, the logistic regression is effective in classifying buoys. We can also use this model to classify other buoys.

KNN

We also used the K-Nearest Neighbor (KNN) algorithm for the classification of Buoy 2 or Bouy 48. KNN is a Supervised machine learning algorithm that stores all the available cases and classifies the new data or case based on feature similarity. It is mostly used to classify a data point based on how its neighbours are classified. Since there is no structured method to find the best value for *"K",* we need to find out various values by trial and error and assuming that training data is unknown.

In this analysis, we increased the complexity of the model by increasing the number of predictors to include the initial six predictors, their squared and interaction terms leading to 21 predictors. We split the data into 70% training data, 15% validating data and 15% testing data. We then calculated the distance between two data points using the Euclidean distance method. Inorder to choose the best *"K"*, we run the KNN algorithm on the validating data for different value of *"K*" from 1 to 20. Below is a graph showing the accuracy (true positives and true negatives) derived from the confusion matrix for different values of *"K".* The graph clearly shows that a *"K"* of **3** yields the highest accuracy of 76.73%.



Using a *"K"* = 3, we run the KNN algorithm on the testing data and this yields a marginally higher accuracy of 77.17%. With this result, overfitting is slightly minimized since the difference between validating and training data accuracy is rather marginal. This model neither underfits nor overfits. Below is a snippet showing the confusion matrix from this model.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 285  98
         1  60 249

               Accuracy : 0.7717
                 95% CI : (0.7386, 0.8025)
    No Information Rate : 0.5014
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5435

 Mcnemar's Test P-Value : 0.003245

            Sensitivity : 0.8261
            Specificity : 0.7176
         Pos Pred Value : 0.7441
         Neg Pred Value : 0.8058
             Prevalence : 0.4986
         Detection Rate : 0.4118
   Detection Prevalence : 0.5535
      Balanced Accuracy : 0.7718

       'Positive' Class : 0
```

From the results we notice that the following results:

- Accuracy = 0.7717
- Specificity = 0.7176
- Sensitivity = 0.8261

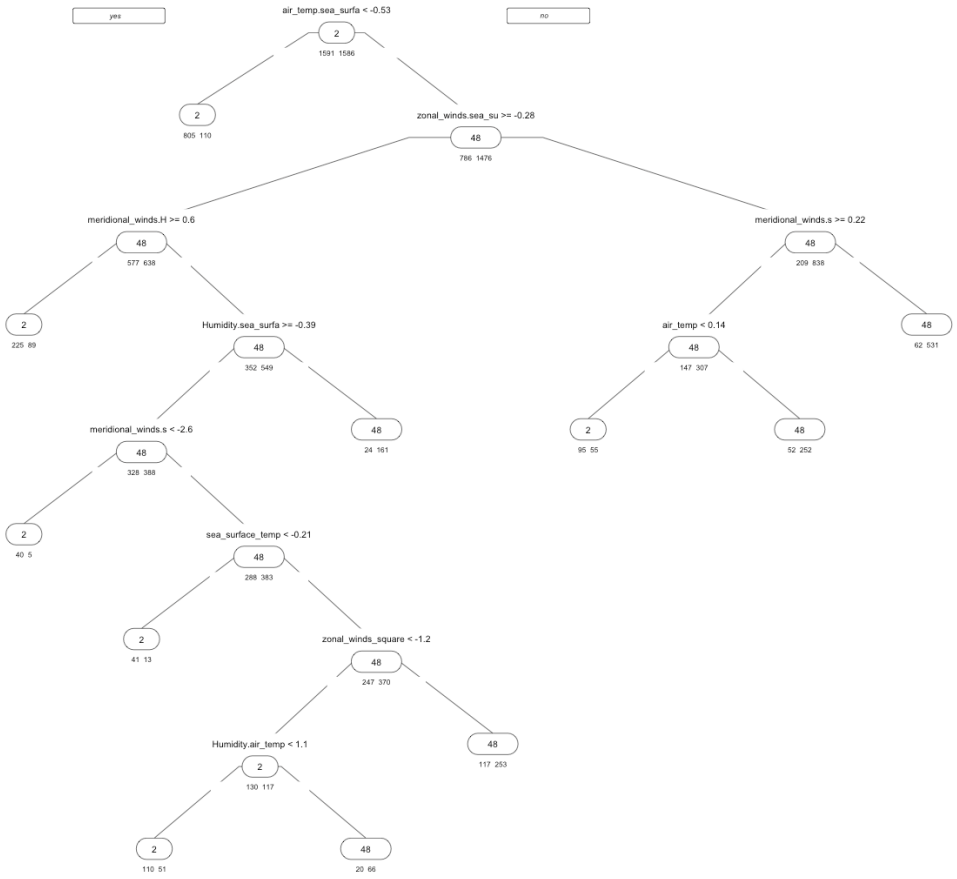
With KNN, the Accuracy is slightly higher than in logistic regression. Based on these results, we can conclude that KNN is as effective in classifying the buoys.

Classification Tree

In this part, we used a number of classification tree methods to determine a certain data point belongs to buoy 2 or 48. In addition, on top of the 6 variables given in the dataset, we created interaction and squared variables. Moreover, we also created a variable called 'month' reflecting

the month in which the data is recorded to capture possible seasonal effect of the time series data. We partitioned train and test data by 70% threshold.

We first used a simple tree to classify the two buoys. Many parameters in the rpart() function are set to default. The tree plot and confusion are shown below.



```
Confusion Matrix and Statistics

          Reference
Prediction   2  48
         2  546 150
         48 161 504

               Accuracy : 0.7715
                 95% CI : (0.7482, 0.7936)
    No Information Rate : 0.5195
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5426

 Mcnemar's Test P-Value : 0.5707

            Sensitivity : 0.7723
            Specificity : 0.7706
         Pos Pred Value : 0.7845
         Neg Pred Value : 0.7579
             Prevalence : 0.5195
         Detection Rate : 0.4012
   Detection Prevalence : 0.5114
      Balanced Accuracy : 0.7715

       'Positive' Class : 2
```

The tree looks simple, however, the result is not bad. The accuracy on test data of this model is 77.15%, while sensitivity and specificity are on a similar level. We would continue the experiment to more complex trees. In this case, we set cp = 0 and minsplit = 1 in the rpart() function and significantly increase the complexity of a single tree. The confusion matrix is shown below.

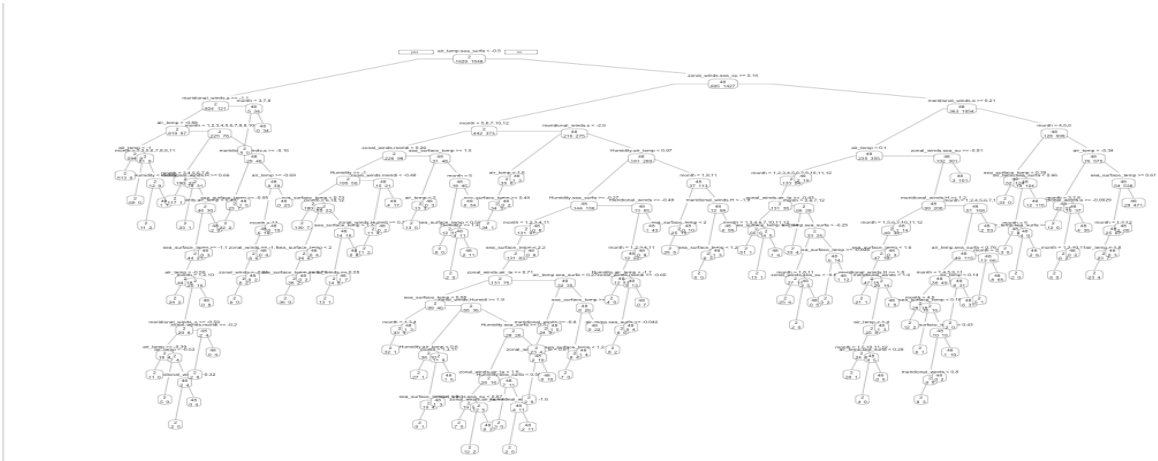
```
Confusion Matrix and Statistics

          Reference
Prediction   2  48
         2  563 120
         48 144 534

               Accuracy : 0.806
                 95% CI : (0.784, 0.8267)
    No Information Rate : 0.5195
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.612

 Mcnemar's Test P-Value : 0.1569

            Sensitivity : 0.7963
            Specificity : 0.8165
         Pos Pred Value : 0.8243
         Neg Pred Value : 0.7876
             Prevalence : 0.5195
         Detection Rate : 0.4137
   Detection Prevalence : 0.5018
      Balanced Accuracy : 0.8064

       'Positive' Class : 2
```

The accuracy increases to 80% consequently.

We then used the built-in cross validation method to find the optimal complexity and nsplit parameters for rpart() function. The table for errors is shown below.

```
           CP nsplit rel error  xerror     xstd
1  0.43505675      0  1.000000 1.06179 0.017737
2  0.04287516      1  0.564943 0.57755 0.016099
3  0.01261034      3  0.479193 0.49748 0.015355
4  0.01103405      5  0.453972 0.45965 0.014944
5  0.00882724     10  0.377049 0.42308 0.014506
6  0.00788146     13  0.350567 0.40227 0.014237
7  0.00504414     15  0.334805 0.38840 0.014050
8  0.00441362     20  0.309584 0.36129 0.013664
9  0.00409836     22  0.300757 0.35939 0.013636
10 0.00315259     25  0.287516 0.34363 0.013398
11 0.00283733     28  0.277427 0.32976 0.013179
12 0.00252207     30  0.271753 0.33354 0.013240
13 0.00220681     34  0.261665 0.33607 0.013280
14 0.00189155     38  0.252837 0.33291 0.013230
15 0.00163934     45  0.239596 0.32850 0.013159
16 0.00157629     50  0.231400 0.32850 0.013159
17 0.00147121     66  0.206179 0.32156 0.013046
18 0.00126103     70  0.199243 0.32030 0.013025
19 0.00094578    116  0.140605 0.32850 0.013159
20 0.00088272    133  0.123581 0.33733 0.013299
21 0.00084069    138  0.119168 0.33733 0.013299
22 0.00078815    141  0.116646 0.33733 0.013299
23 0.00063052    146  0.112232 0.33985 0.013339
24 0.00054044    189  0.085120 0.33922 0.013329
25 0.00052543    196  0.081337 0.33922 0.013329
26 0.00047289    204  0.076923 0.34174 0.013368
27 0.00042034    211  0.073140 0.34174 0.013368
28 0.00037831    219  0.069357 0.34174 0.013368
29 0.00031526    224  0.067465 0.34931 0.013485
30 0.00001000    236  0.063682 0.35183 0.013523
```

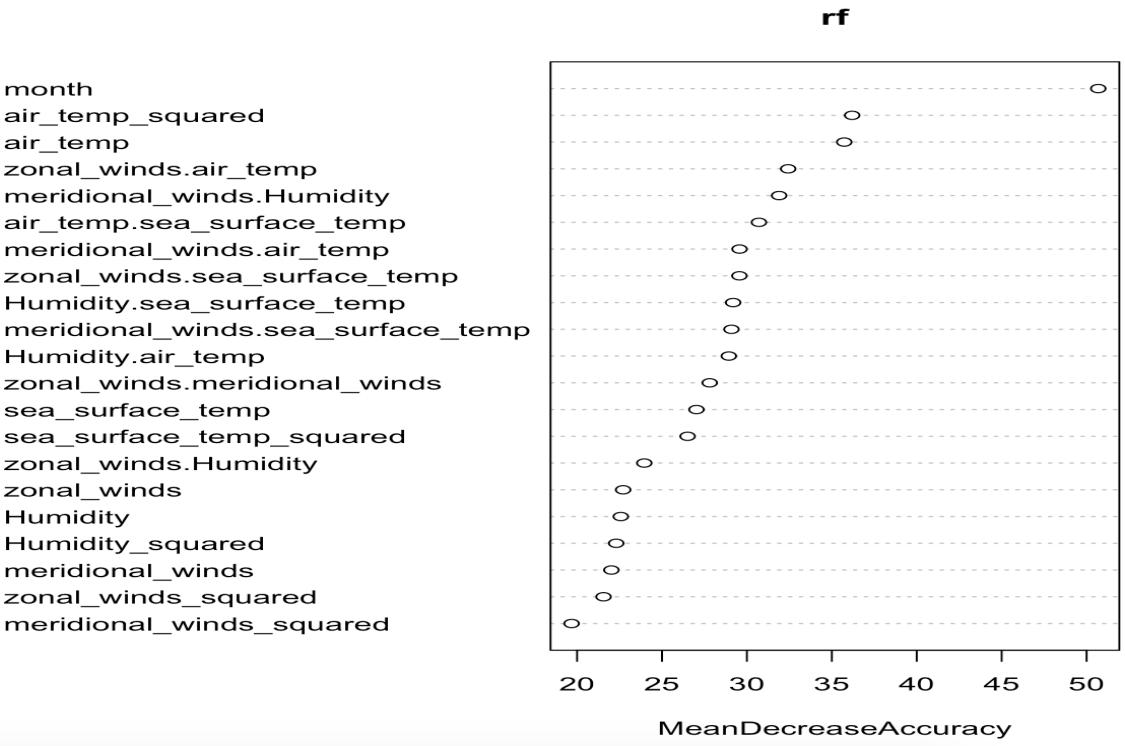
As shown in the table, the minimum xerror is at CP = 0.0013 and nsplit = 70. The R-Squared fitting score for this model is 1-0.20 ~ 80%. However, the tree plot looks a lot more complicated than the previous tree plot.



When we compare the results of the above three tree methods, we can see that in general, classification tree yields an accuracy of around 80%, and increasing complexity of trees does not significantly enhance the accuracy of the model.

Then we turned our sights to random forest and boosting tree methods. There are two tables shown below: the first table displays confusion matrix of random forest model's predicting capability on test dataset; the second table shows features' importance by mean decrease accuracy. Month turns out to be more significant than any other variables to predict the index of buoys. One interpretation is that time series data usually poses great seasonal effect, and capturing the seasonal effect usually enhances accuracy of models. The accuracy of the random forest method reaches 84.80%, better than that of a single tree.

```
Confusion Matrix and Statistics

          Reference
Prediction   2  48
        2  590  90
        48 117 564

               Accuracy : 0.8479
                 95% CI : (0.8277, 0.8666)
    No Information Rate : 0.5195
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6958

 Mcnemar's Test P-Value : 0.07074

            Sensitivity : 0.8345
            Specificity : 0.8624
         Pos Pred Value : 0.8676
         Neg Pred Value : 0.8282
             Prevalence : 0.5195
         Detection Rate : 0.4335
   Detection Prevalence : 0.4996
      Balanced Accuracy : 0.8484

       'Positive' Class : 2
```



Finally, we generated a boosting tree method to classify the two buoys. The accuracy is further enhanced to 90.3%.

```
Confusion Matrix and Statistics

          Reference
Prediction   2  48
        2  595  61
        48  71 634

               Accuracy : 0.903
                 95% CI : (0.886, 0.9182)
    No Information Rate : 0.5107
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8059

 Mcnemar's Test P-Value : 0.4334

            Sensitivity : 0.8934
            Specificity : 0.9122
         Pos Pred Value : 0.9070
         Neg Pred Value : 0.8993
             Prevalence : 0.4893
         Detection Rate : 0.4372
   Detection Prevalence : 0.4820
      Balanced Accuracy : 0.9028

       'Positive' Class : 2
```

Conclusion

In this exercise, we applied logistic regression, KNN, and classification tree to classify two buoys. For each classification method, we used similar features, including variables provided in the dataset, squared terms and interactions among these variables; except in the decision tree part, we created and included a month variable. All these classification methods yield good enough results (accuracy > 75%) and set examples for real life using cases. In all of them, boosting tree method appears to have the best accuracy of more than 90%. However, this method is really computationally costly. It takes more than 1 minute for the model to be trained on the trained dataset containing only 3,000 data points. If we expand the model to include multiple buoys (more than 2) and more data points, we will need considerably more time to consolidate a boosting tree model. By that time, maybe a regular tree with enough leaves or other classification methods suffice to complete the task. Otherwise, if we already suspect that the data belongs to a certain buoy, we can choose to use logistic regression to investigate whether the data is really measured by that certain buoy or not.