Results with SMOTE(X, Y, K=5)

Random Forests:

* Ntree: 500
* Importance true

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | TPR | FPR | PRECISION | RECALL | F-MEASURE | MCC | Kappa |
| Class no |  |  |  |  |  |  |  |
| Class Yes |  |  |  |  |  |  |  |
| Wt. Average |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Train Results | Test Results |
|  |  |

Results with SMOTE(X, Y, K=5)

Random Forests:

* Ntree: 1000
* Importance true
* Classwt : c(1,5)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | TPR | FPR | PRECISION | RECALL | F-MEASURE | MCC | Kappa |
| Class no |  |  |  |  |  |  |  |
| Class Yes |  |  |  |  |  |  |  |
| Wt. Average |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Train Results | Test Results |
|  |  |

Results with SMOTE(X, Y, K=5)

This has given us the best performance thus far, I will add to the bottom

Random Forests:

* Ntree: 1500
* Importance true
* Classwt : c(1,10)

Tried: but did not improve or worsen the results:

proximity=TRUE

nodesize = 5 and 10

Adding now:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | TPR | FPR | PRECISION | RECALL | F-MEASURE | MCC | Kappa |
| Class no |  |  |  |  |  |  |  |
| Class Yes |  |  |  |  |  |  |  |
| Wt. Average |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Train Results | Test Results |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10)) |  |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  nodesize = 3) |  |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  nodesize = 2) |  |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  maxnodes = 100) |  |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  maxnodes = 200) |  |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  maxnodes = 300)  best accuracy thus far. |  |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  maxnodes = 400) |  |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  maxnodes = 350,  replace = FALSE) |  |
| rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  maxnodes = 350) |  |
| # this might be the best accuracy we can get in random forest  rf\_model <- randomForest(Class ~ ., data = trainData\_SMOTE[, c(selected\_features\_rfe, "Class")],  ntree = 1500,  importance = TRUE,  classwt = c(1, 10),  maxnodes = 300,  replace = FALSE) |  |

Results with SMOTE(X, Y, K=5)

Random Forests:

* Ntree: 2000
* Importance: true
* Classwt : c(1,15)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | TPR | FPR | PRECISION | RECALL | F-MEASURE | MCC | Kappa |
| Class no |  |  |  |  |  |  |  |
| Class Yes |  |  |  |  |  |  |  |
| Wt. Average |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Train Results | Test Results |
|  |  |

Results with SMOTE(X, Y, K=5)

Random Forests:

* Ntree: 1500
* Importance true
* Sampsize =c(500,500)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | TPR | FPR | PRECISION | RECALL | F-MEASURE | MCC | Kappa |
| Class no |  |  |  |  |  |  |  |
| Class Yes |  |  |  |  |  |  |  |
| Wt. Average |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Train Results | Test Results |
|  |  |