STOR 565 Final Project Report

An Explorational West Nile Virus Classification

Yige Chen, Jinghan Chen, Yimei Fan

Project Summary

West Nile virus (WNV) is the leading cause of mosquito-borne disease in the continental United States, according to the U.S. CDC. Currently, there is no vaccine that could avoid the spread of WNV nor is there any medication product available to cure the patients suffering from this virus. We were inspired by the current outbreak of the COVID-19 to use machine learning techniques to predict the presence of the disease in a specific region. What we decided to handle is a classification problem. The original data was obtained via Kaggle published by the Chicago Department of Public Health.

We took the creativity to combine all three datasets, train, spray, and weather, to create a cleaned training dataset for our modeling. In order to improve the model, we feature engineered two new variables, spray and Daytime in the final dataset. Our machine learning modeling process has involved techniques like Naïve Bayes, Logistic Regression, K-nearest Neighbor, Classification Tree, Random Forest, and Gradient Boosting. We evaluated each model with accuracy score, confusion matrix, and ROC Curve for their performance and cross-model comparison.

The best model we obtained turned out to be the Logistic Regression model. When choosing the model, we had to leverage between prediction accuracy and interpretability. As we will mention, the original data suffers from the unbalanced data problem, which exposes a limitation to our research at this time. We hope to explore other techniques in the future to better handle this situation so that we could improve our model, which serves our goal of creating contribution into the health industry with machine learning.

I. Project Overview

West Nile Virus is most commonly spread to humans through infected mosquitos. Around 20% of people who become infected with the virus develop symptoms ranging from a persistent fever, to serious neurological illnesses that can result in death.

In this project, we will first clean the data obtained from a competition on Kaggle website. We will visualize the trap data and spray data through Tableau to figure out how to merge the data. We will use different machine learning techniques, including naive Bayes, KNN, logistic regression, Gradient Boost, Decision Tree, and Random Forest, to predict if a mosquito caught in the City of Chicago carries West Nile Virus. We will also evaluate the performance of each model with evaluation techniques like the confusion matrix and ROC. Finally, we will state the limitations of the models and find some potential improvements.

II. Data Exploration

i. Data Cleaning

The data we obtained were from Kaggle and were essentially published by the Chicago Department of Public Health. The main part of the dataset is 'trap information'. Every year from late-May to early-October, public health workers in Chicago set up mosquito traps scattered across the city. The columns include date, address, mosquito species, id of trap, etc. There are also two additional datasets, including spray data (the GIS data for their spray efforts in 2011 and 2013) and weather data, which includes many geological information of the observation sites such as the max/min temperature, sea level pressure, moisture, daytime, and etc.

We started with the weather data, chopping off the columns specific to the address and street information that are not related to our modeling process. Second, the data types of each column were transformed into numeric type in order to make them machine learnable. The next step was handling the NA values in the weather dataset. The NA's in Tavg, WetBulb, StnPressure, SeaLevel, and AvgSpeed were implied as the mean value of each column, and the NA's in Heat, Cool, and PrecipTotal were replaced with 0, given the observations on those columns which shows that a good amount of data in those columns were 0's in order to represent a normal pattern.

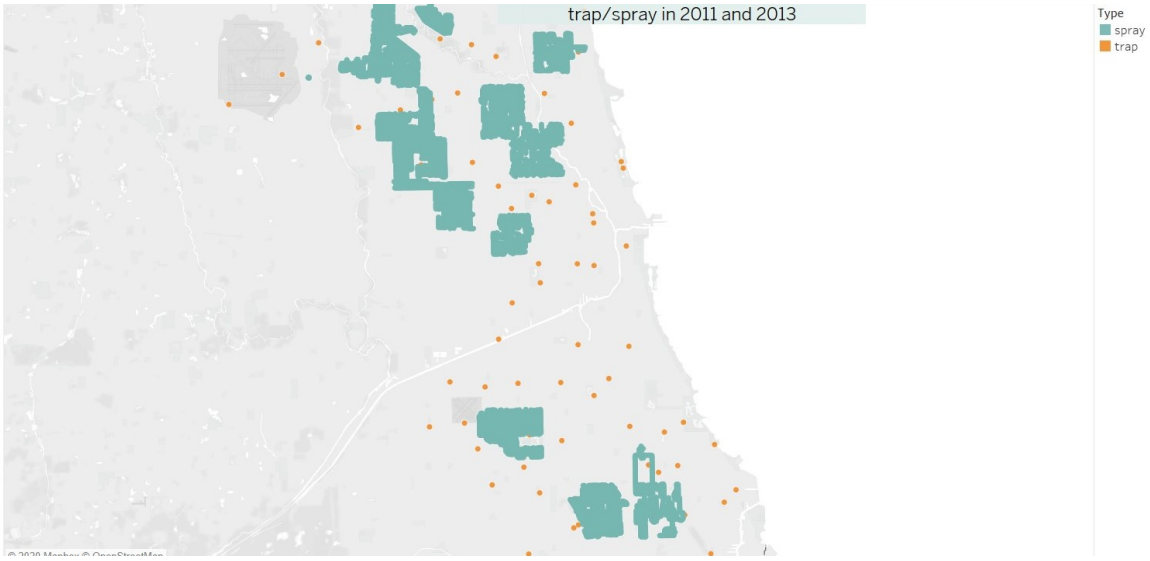
ii. Feature Engineering and Merge Datasets

The main part of our feature engineering was transforming the variables Sunrise and Sunset into the variable Daytime. Extracting the Sunrise and Sunset data that were in non-standard time format, we changed the ":" into ".'", divided the time by 100 to get the decimals, and used the function ceiling and floor to handle the decimal data. Subsequently, a new variable Daytime replaced Sunrise and Sunset, which again made the related but non-standard data processable for the machine to generate potential findings for our research.

The final step is to merge the three datasets we have, “train.csv”, “weather.csv” and “spray.csv”. Since these datasets all have column “Date”, we can merge them in order to get a dataset that concludes weather features and pest control features in the same time range. After merging the three datasets, three additional steps were taken to tackle the new problem. The column 1 to column 3 and column 5 to column 8 were dropped. Again, merging the datasets brings back the non-essential columns that contain the address, latitude, and longitude information. Second, the different species of mosquitoes were presented as text, so we replaced them with numbers from 1 to 7 to capture its categorical variable nature. Third, we transformed the response variable, WnvPresent, as a factor / categorical variable to facilitate our modeling process.

iii. Visualization

In order to understand the relations of the dataset “train.csv” and “spray.csv”, we use longitude and latitude to show the presence of mosquitoes and the track of spray.



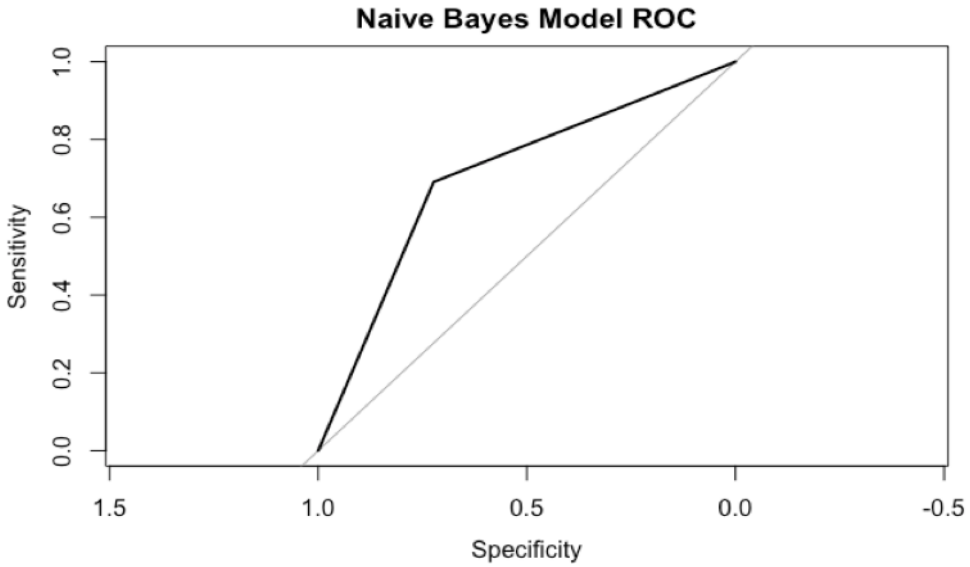
The green parts are where anti-mosquito spray is applied, and orange dots are where mosquito traps are located. From this visualization, we can figure that these two databases are correlated and can be merged together to make predictions.

III. Model Construction

i. Naive Bayes Classifier

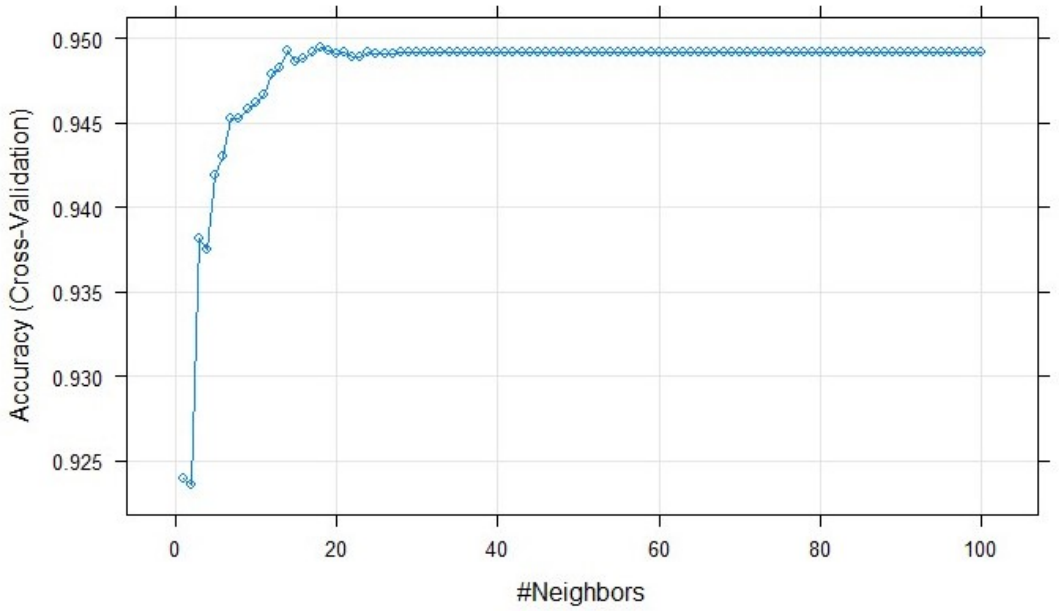
Naive Bayes Classifier combines the Prior and Posterior probabilities to minimize the risk and decreases the Bayes error. It is a basic machine learning model and we decided to use it as a potential baseline. After fitting the model, we generated the predictions made from the Naive Bayes Classifier, and subsequently output a confusion matrix and examine its ROC curve to evaluate the performance. The model reaches an accuracy of 72.14% and has a similar Area Under the Curve ratio of 70.7%, and the F1 score is 0.831. Given the unbalanced distribution

of our response variable, WnvPresent, which has a majority of the data being labeled as 0, the 72.14% test accuracy performance is not promising indeed.



ii. K-nearest Neighbor

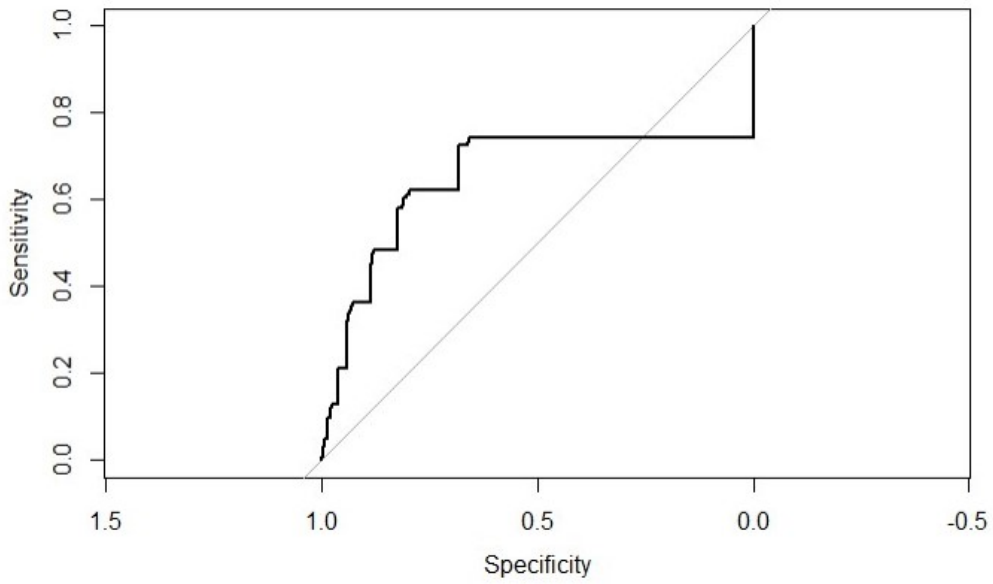
Our task is to classify whether West Nile Virus will present given some weather condition and mosquito control. Therefore, K-nearest neighbors will be a good approach to explore.
After removing some redundant variables, we utilized the "train" function from "caret" to apply cross-validation with KNN. The plot of training process is as follows:



From the result of cross-validation training, we can see that when k = 18, the training accuracy becomes highest and becomes stable, which is 94.94%. Compared to testing accuracy, which is 94.05%, we can see that this model is robust, with reasonable testing and training accuracy.

However, when we looked at the confusion matrix of this model, we found that this model works much better when it comes to predicting that the West Nile Virus will not present compared to prediction of WNV occurrence. We then took steps back and investigated the raw data and found that there are only 426 cases of WNV but 7978 cases of non-WNV. The lack of data may be the major reason.

We also investigated the ROC curve of this model as follows.

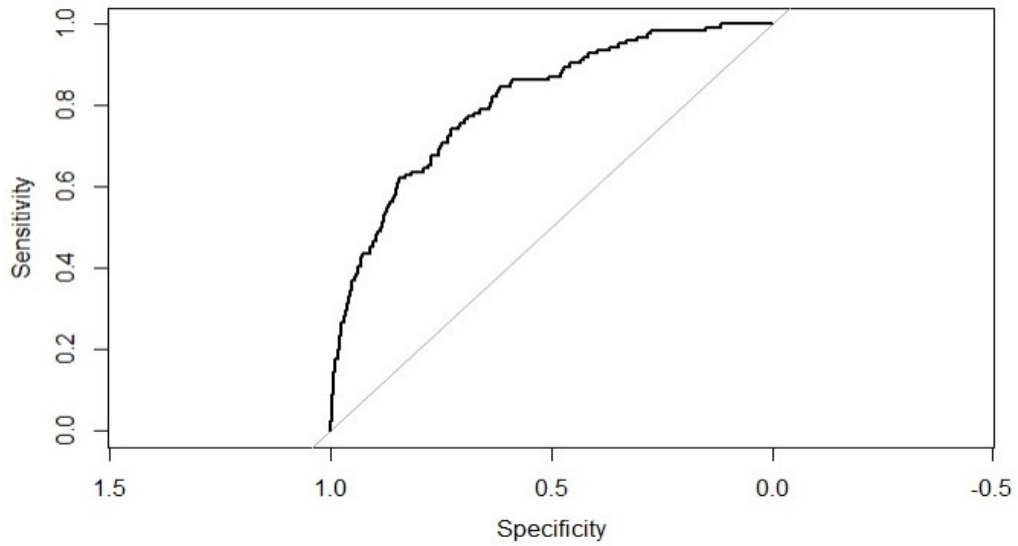


We can conclude from this plot that with increasing sensitivity rate, the specificity rate is decreasing. The high specificity implies that this model can perform well about predicting there are no WNV cases, which is the same as we can see from the confusion matrix. However, the degree of separability is not so well. In reality, this issue is related to the mosquito control policy. If we would like to reduce the presence of WNV as much as possible but don't want to waste too much spray effort, this model will perform well with high specificity.

iii. Logistic Regression

With the same data preparation process, we also chose to utilize a logistic regression method, for predicting WNV is a binary classification. After cross-validation, the training accuracy is 94.95% and the testing accuracy is 93.96%.
This model has the same problem with the K-nearest neighbor. From the result of the confusion matrix, logistic regression performs well when predicting non-WNV but when it comes to predicting WNV occurrence, the model is not the best choice.

We also investigated the ROC curve of logistic regression.

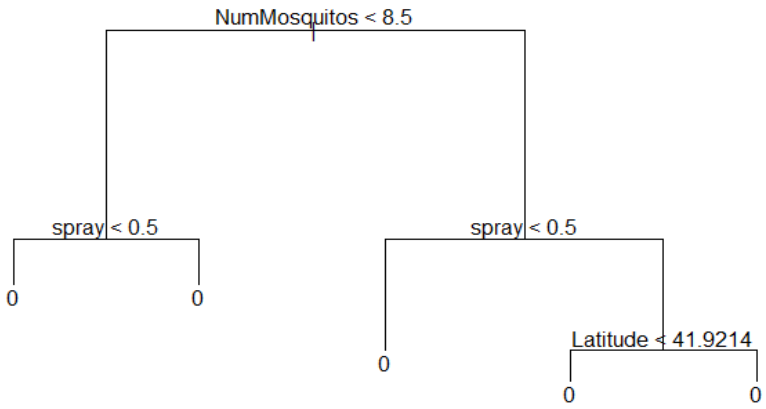


The AUC of this curve is better than that of KNN.

In regard to important features, longitude and latitude of mosquito present, temperature, wind direction, precipitation and spray or not are significant.

iv. Classification Tree

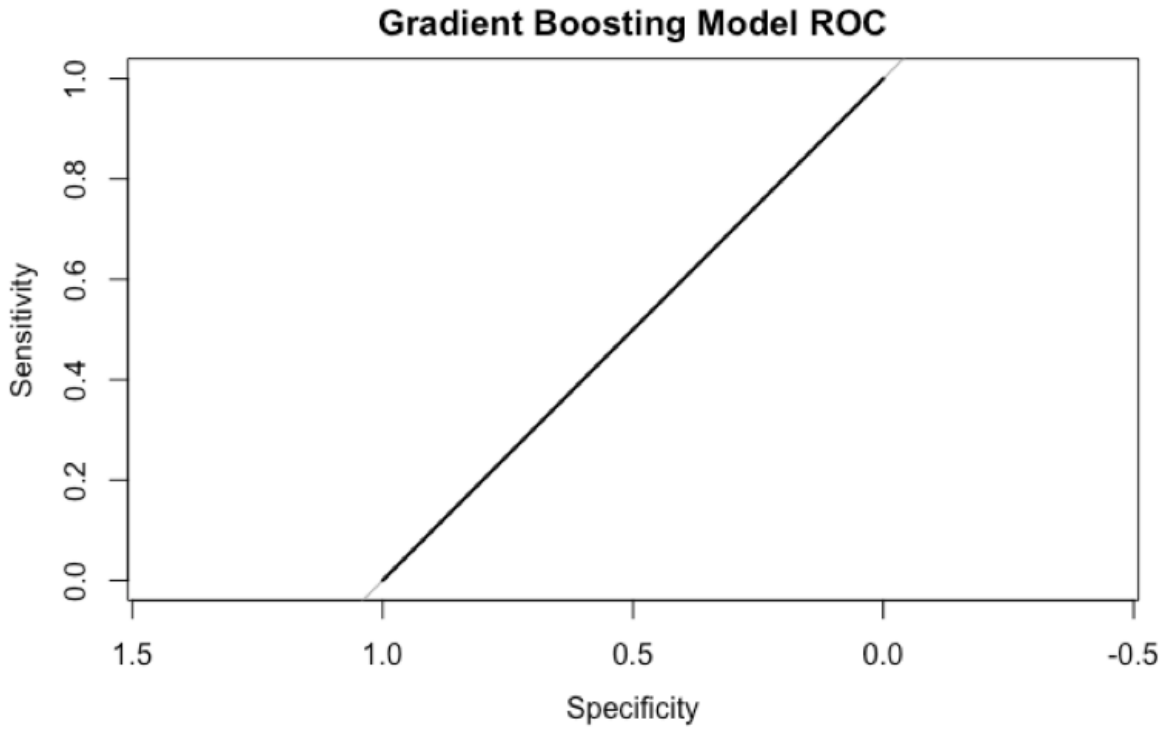
Because classification trees can be easily interpreted and explained to our audiences, we want to try this method. After making the tree, we found that the tree simply predicts every observation as 0 regardless of the variables. Therefore, this tree is useless, even though its test accuracy, 94.91%, looks good. Further analysis of the tree's performance is like what the paragraph of the gradient boosting model describes, because the two models both predict no positive result. And it is meaningless to prune such a tree. The classification tree fails to deal with this extremely unbalanced dataset. But we also want to see if random forests as an aggregation of many decision trees can have a better performance in our case.



v. Gradient Boosting Model

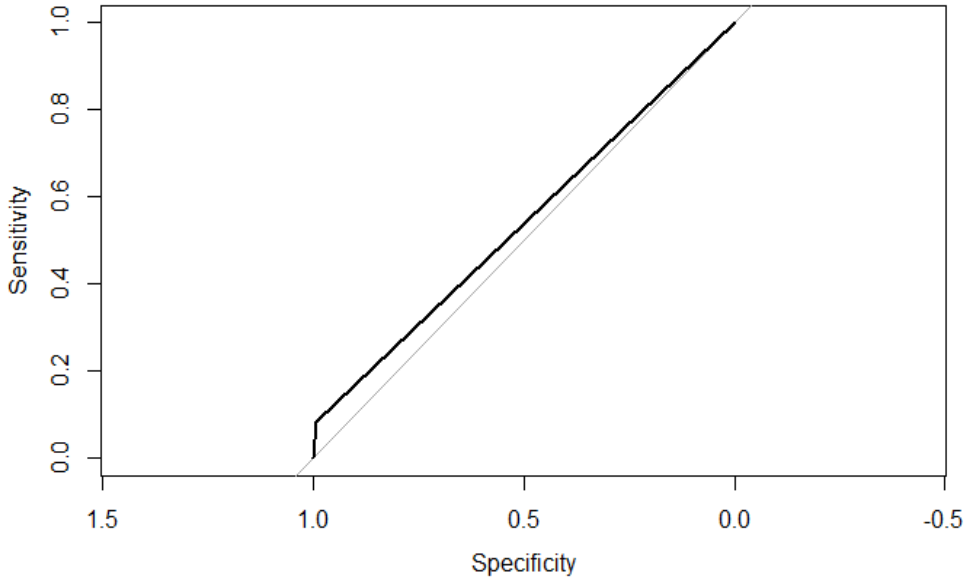
Gradient Boosting is one of the Ensemble Learning methods. The general idea is that it has a sequential correction of the predecessor's errors, and fit each predictor using its predecessor's residual errors as labels. Gradient Boosting involves an exhaustive search procedure for each CART to find the best split point for the feature. We hope that this model could help solve some of the previous models' limitations. The caret package was used to fit the model, with a training control method of repeated cross validation.

As a result, the model has a 94.76% of test accuracy, and has a F1 score of 0.973, which performs better than the Logistic Regression model and very close to the KNN model. However, examining the ROC and confusion matrix with a closer look, we found that Gradient Boosting happens to classify every test observation as 0 so that it could achieve such a high accuracy. The ROC graph coincides with this problem. On the ROC graph, the curve completely overlays with the 50% random guessing slope, meaning that the model might not be generalizable. This exposed a huge limitation for the model in the real world scenario: we cannot label every observation as 0. There exists a cost for false negative, and in our case the cost might be huge. Labeling an observation which actually has West Nile Virus as not could create a huge danger for the public health, and this is exactly the situation we hope to prevent. Even if Gradient Boosting can be robust in other modeling problems, it did not output a result that achieves our goal in our specific case.

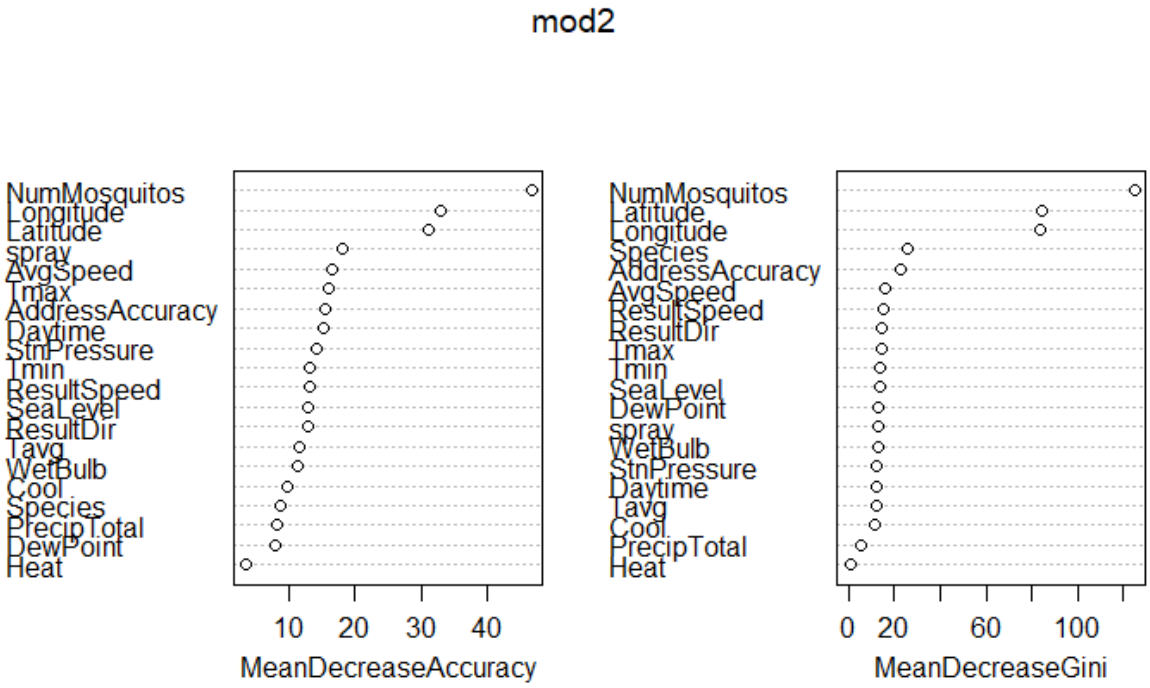


vi. Random Forest

As the method aggregating trees within the idea of bootstrap, random forest should be tried. Compared to the bagging method, random forest is preferred and expected to have a better performance because there may be a strong predictor that makes the trees correlated with each other. We consider the random forest with sqrt(p) predictors and aggregate 500 trees, which is the default number. As the cost of using many trees, we are not able to draw a simple tree and interpret it easily using random forest. The model has a 94.76% test accuracy, which is slightly smaller than predicting all observations as negative. The F1 score is 0.973. The our-of-bag estimate of the error rate is 5.35%. The model still predicts most as negative, but we at least have 98 false negatives and 12 false positives. The AUC is 0.539, which is not good. Here is the ROC curve:



The problem of the model is similar to the other models. To have an insight about the variables, it is useful to make the variable importance plot. Here the average improvements in RSS and Gini index are used to reflect the variable importance. In our case, the number of mosquitoes caught in traps, latitude, and longitude are important variables used in the model.



We also tried the random forest with p/3 predictors and the bagging with all predictors. Overall, the three models have similar performances. Compared to the first model discussed, the latter two models have slightly lower accuracies but slightly higher AUC. One big difference in the bagging model is that "spray or not" becomes the most important variable when considering RSS.

IV. Limitations

The performance of our model is fairly well, but there are still some limitations and improvements to be done.

i. Insufficient data

There are three separate datasets before merging. Unfortunately, the time range of these dataset does not have much overlap. There are only 10506 rows in our raw dataset. With limited data, the accuracy of models will be affected.

ii. Highly unbalanced data

In our cleaned dataset that were used for modeling, there are 10,506 records, out of which the vast majority has the response variable, WnvPresent, labeled as 0. This represents a great challenge for our modeling process. As shown in the previous parts, Random Forest and Gradient Boosting models both suffered from the problem of unbalanced data. The distribution of the response variable determines that those tree-based methods tend to lean heavily on the normal population group, which is 0 class. Unbalanced data are prevalent in problems that are concerned with disease detection, customer churn, and etc. We hope to better explore this problem in the future by seeking some data pre-processing tools, such as undersampling, oversampling, and synthetic Data Generation, that could potentially help assign weight to one minority class and help solve this issue.

V. Conclusions

The major problem of our dataset is unbalanced data. There are several approaches we can utilize, such as undersampling, oversampling and synthetic Data Generation, etc. In the future, we should pay more attention to preprocessing unbalanced data.

After comparing these models, we recommend the logistic regression model. There are several models which have high accuracies, such as logistic regression, K-nearest Neighbor, classification tree, gradient boosting model, random forest, but in terms of good classification methods, logistic regression has better performance. Intuitively, logistic regression is a binary classification method, which matches our project objective.

Reference

"West Nile Virus." *Centers for Disease Control and Prevention*, Centers for Disease Control and Prevention, 12 Nov. 2019, www.cdc.gov/westnile/index.html.

Julie. "Cross-Validation Essentials in R." *STHDA*, 11 Mar. 2018, www.sthda.com/english/articles/38-regression-model-validation/157-cross-validation-essentials-in-r/.

Barua, Sukarna et al, 'MWMOTE - Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning' (2014) 26 IEEE Transactions on Knowledge and Data Engineering 405

Stor 565 Project

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------
-------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------
-------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(readr)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
weatherdata <- read_csv("weather.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Station = col_double(),
##   Tmax = col_double(),
##   Tmin = col_double(),
##   DewPoint = col_double(),
##   ResultSpeed = col_double(),
##   ResultDir = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
# Drop out columns without meaning or with too many missing data
weather <- weatherdata[,c(2:5, 7:12, 17:22)]
```

```
summary(weather)
str(weather)
```

```
# Change into appropriate data types
weather <- transform(weather, Tavg = as.integer(Tavg),
                     WetBulb = as.integer(WetBulb),
                     Heat = as.integer(Heat),
                     Cool = as.integer(Cool),
                     Sunrise = as.numeric(Sunrise),
                     Sunset = as.numeric(Sunset),
                     PrecipTotal = as.numeric(PrecipTotal),
                     StnPressure = as.numeric(StnPressure),
                     SeaLevel = as.numeric(SeaLevel),
                     AvgSpeed = as.numeric(AvgSpeed))
str(weather)
```

```
# Check each columns while clean up the columns with NA values
anyNA(weather$Tmax)
anyNA(weather$Tmin)

# Clean the column Tavg
anyNA(weather$Tavg)
sum(is.na(weather$Tavg))
weather$Tavg[is.na(weather$Tavg)] <- round(mean(weather$Tavg, na.rm = TRUE))
anyNA(weather$Tavg)

# Clean the column WetBulb
anyNA(weather$DewPoint)
anyNA(weather$WetBulb)
sum(is.na(weather$WetBulb))
weather$WetBulb[is.na(weather$WetBulb)] <- round(mean(weather$WetBulb, na.rm = TRUE))

# Clean the columns Heat and Cool by filling NA with 0
weather$Heat[is.na(weather$Heat)] <- 0
weather$Cool[is.na(weather$Cool)] <- 0
anyNA(weather$Heat)
anyNA(weather$Cool)

# Deal with the Sunrise and Sunset times
weather <- weather[!is.na(weather$Sunrise),]
# Divide Sunrise and Sunset time by 100 to get the decimals, and then coercing into integer base
d on an adjusted round up/round down
weather$Sunrise <- weather$Sunrise/100
weather$Sunset <- weather$Sunset/100
weather$Sunrise <- ifelse(weather$Sunrise %% 1 >= 0.3, ceiling(weather$Sunrise), floor(weather$S
unrise))
weather$Sunset <- ifelse(weather$Sunset %% 1 >= 0.3, ceiling(weather$Sunset), floor(weather$Suns
et))
weather$Daytime <- weather$Sunset - weather$Sunrise
weather <- weather[, -(9:10)]
```

```
# Clean up the rest of columns
sum(is.na(weather$PrecipTotal))
weather$PrecipTotal[is.na(weather$PrecipTotal)] <- 0

sum(is.na(weather$StnPressure))
weather$StnPressure[is.na(weather$StnPressure)] <- round(mean(weather$StnPressure, na.rm = TRUE
))

sum(is.na(weather$SeaLevel))
weather$SeaLevel[is.na(weather$SeaLevel)] <- round(mean(weather$SeaLevel, na.rm = TRUE))

sum(is.na(weather$ResultSpeed))
sum(is.na(weather$ResultDir))
sum(is.na(weather$AvgSpeed))
weather$AvgSpeed[is.na(weather$AvgSpeed)] <- round(mean(weather$AvgSpeed, na.rm = TRUE))

anyNA(weather)
```

```
#merge weather, spary, and train
library(dplyr)
weather_new = read.csv("weather_new.csv")
spray = read.csv("spray.csv")
train = read.csv("train.csv")


#merge data sets using right_join
virus = right_join(weather_new,train, by  = "Date", all = T)
virus = right_join(spray,virus, by  = "Date", all = T)
```

```
write.csv(virus, file = "viruss.csv")
```

```
# load back our cleaned up training data set
train <- read_csv("virus.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Address = col_character(),
##   Species = col_character(),
##   Street = col_character(),
##   Trap = col_character(),
##   AddressNumberAndStreet = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
train <- train[, -c(1:3, 5:8)]
unique(train$Species)
train$Species[train$Species == "CULEX PIPIENS/RESTUANS"] <- 1
train$Species[train$Species == "CULEX RESTUANS"] <- 2
train$Species[train$Species == "CULEX PIPIENS"] <- 3
train$Species[train$Species == "CULEX SALINARIUS"] <- 4
train$Species[train$Species == "CULEX TERRITANS"] <- 5
train$Species[train$Species == "CULEX TARSALIS"] <- 6
train$Species[train$Species == "CULEX ERRATICUS"] <- 7
unique(train$Species)
train <- transform(train, Species = as.integer(Species),
                   WnvPresent = as.factor(WnvPresent))
```

```
str(train)
```

```
sum((train$WnvPresent) == 1)
sum((train$WnvPresent) == 0)
```

```
# Split the data into training and test
set.seed(1)
index <- createDataPartition(train$WnvPresent, p = 0.7, list = FALSE)

train1 <- train[index, ]
test1  <- train[-index, ]
```

```
# Fit the Naive Bayes Model
library(e1071)
model_nb <- naiveBayes(WnvPresent~., data = train1)
summary(model_nb)

# Evaluate the Naive Bayes classifier performance
nb_pred <- predict(model_nb, newdata = test1)
print(table(nb_pred, test1$WnvPresent))
paste("Accuracy is ", (114+2159)/(2159+114+827+51))

# Generate a ROC curve
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

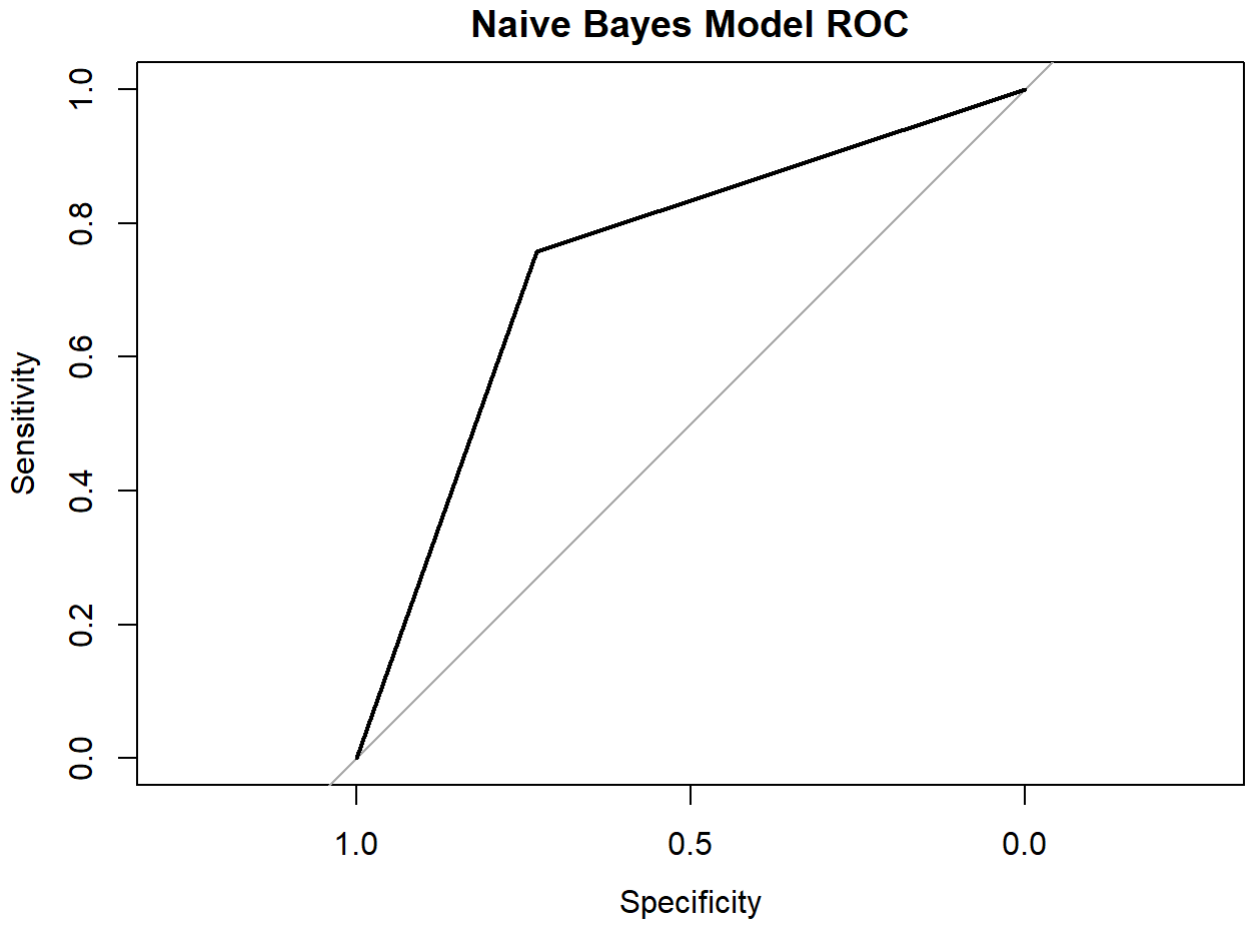
```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
roc(as.numeric(test1$WnvPresent), as.numeric(nb_pred), plot = TRUE, main = "Naive Bayes Model RO
C")
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```




```
# Calculate F1 score
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

```
F1_Score(test1$WnvPresent, nb_pred)
```

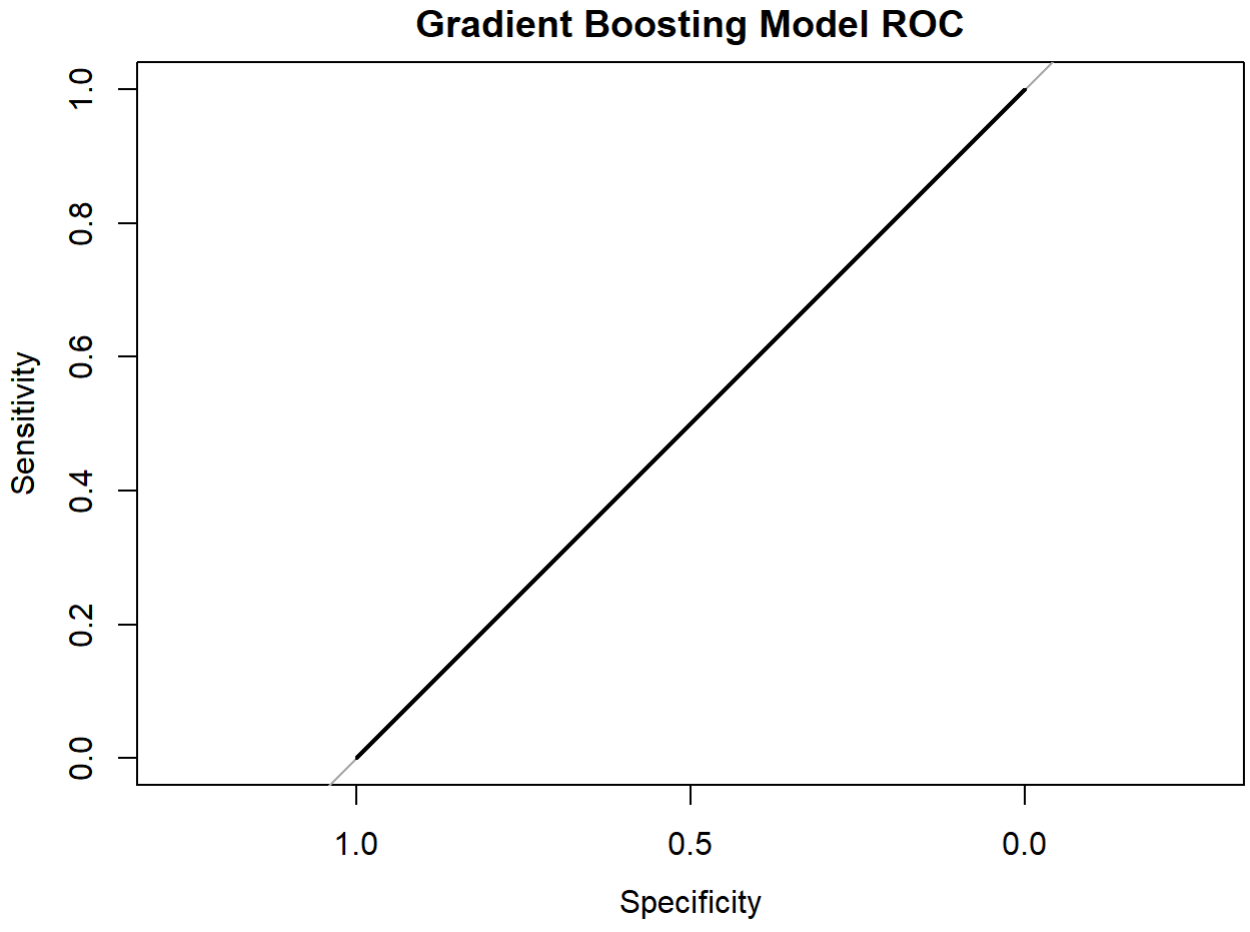
```
# Create a Gradient Boosting model
model_gbm <- caret::train(WnvPresent ~ ., data = train1, method = "gbm", trControl = trainContro
l(method = "repeatedcv", number = 5, repeats = 3, verboseIter = FALSE), verbose = 0)
print(model_gbm)

# Output confusion matrix
gbm_pred <- predict(model_gbm, newdata = test1)
print(caret::confusionMatrix(
  data = predict(model_gbm, test1),
  reference = test1$WnvPresent))

# Generate a ROC curve
roc(as.numeric(test1$WnvPresent), as.numeric(gbm_pred), plot = TRUE, main = "Gradient Boosting M
odel ROC")
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```




```
# Calculate F1 score
F1_Score(test1$WnvPresent, gbm_pred)
```

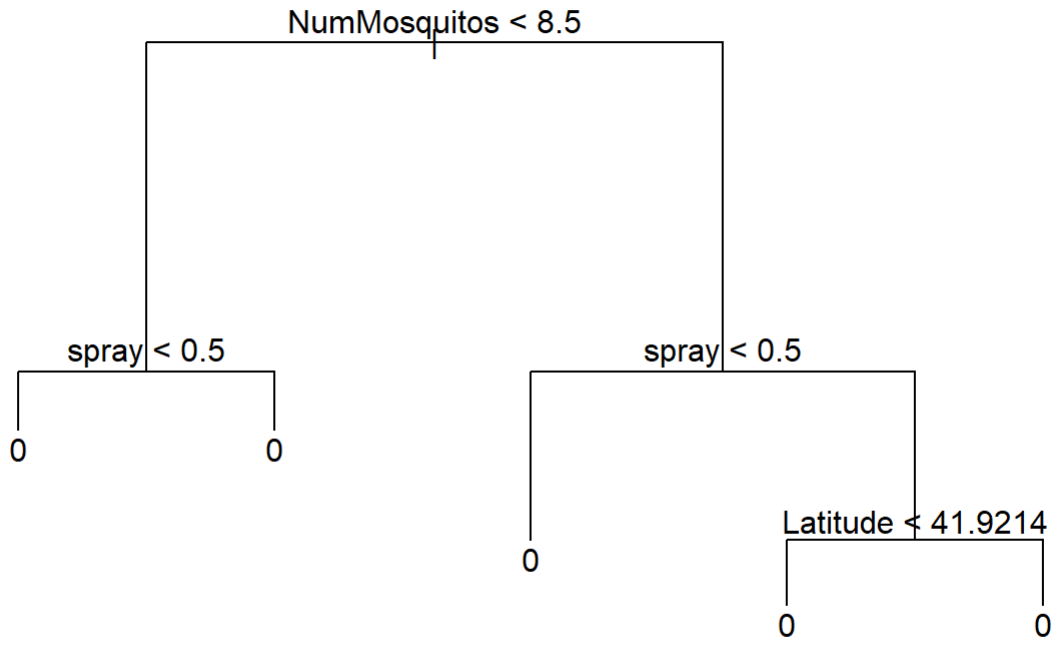
```
# Fit the classification tree
require(tree)
```

```
## Loading required package: tree
```

```
## Warning: package 'tree' was built under R version 3.6.3
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
virus_tree = tree(WnvPresent ~., data = train1)
summary(virus_tree)
plot(virus_tree)
text(virus_tree, pretty = 0)
```

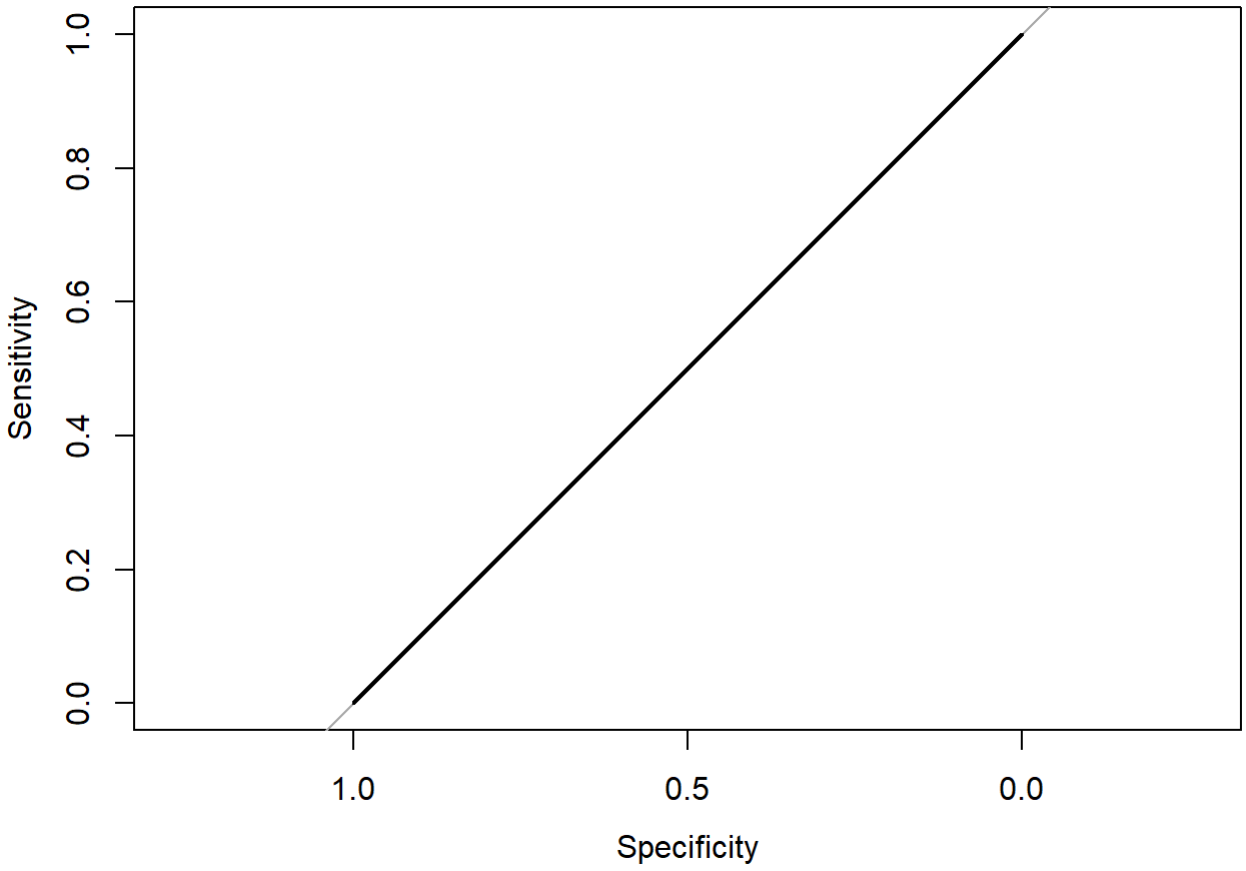



```
# Evaluate the performance of the tree model
tree_pred = predict(virus_tree, test1, type = "class")
ytab1 = table(tree_pred, test1$WnvPresent)
confusionMatrix(ytab1)

roc(as.numeric(test1$WnvPresent), as.numeric(tree_pred), plot = TRUE)
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

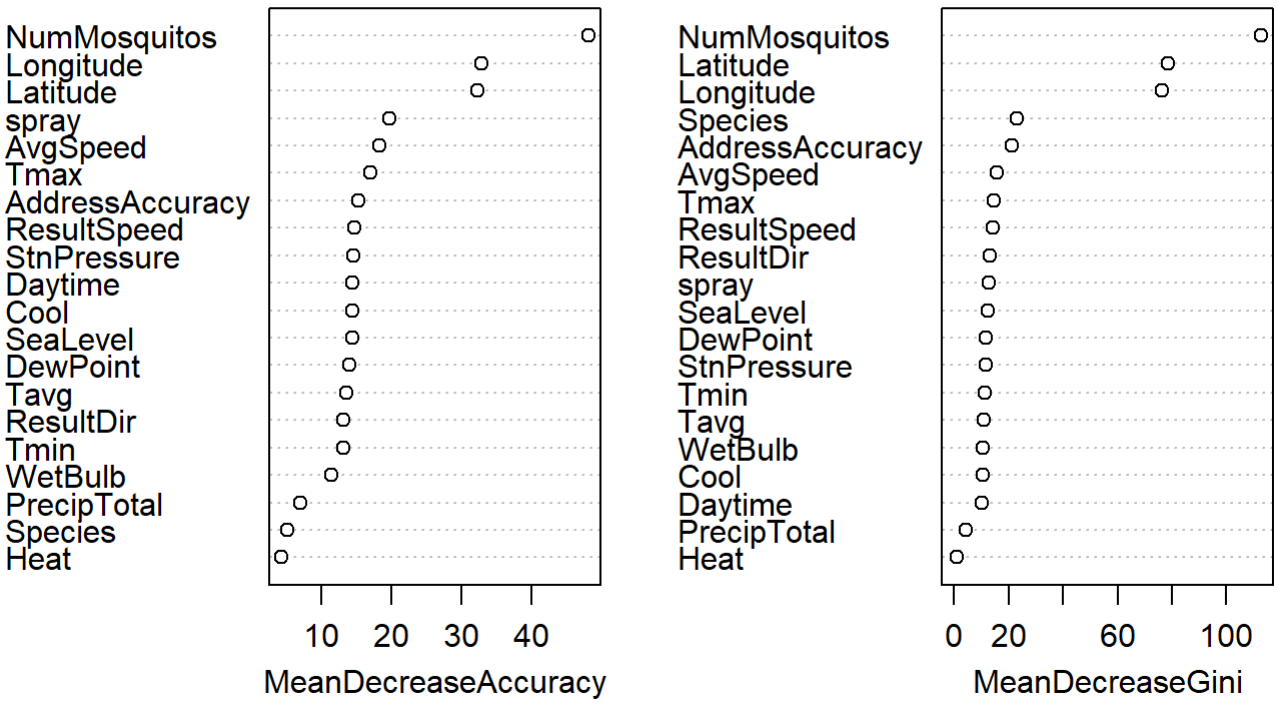



```
F1_Score(test1$WnvPresent, tree_pred)
```

```
# Create the random forest model
library(randomForest)
mod2 = randomForest( WnvPresent~., data=train1, mtry=sqrt(ncol(train1)-1),importance =TRUE)
mod2
rf_pred = predict(mod2, newdata = test1)
ytab2 = table(rf_pred, test1$WnvPresent)
confusionMatrix(ytab2)
```

```
varImpPlot(mod2)
```

mod2

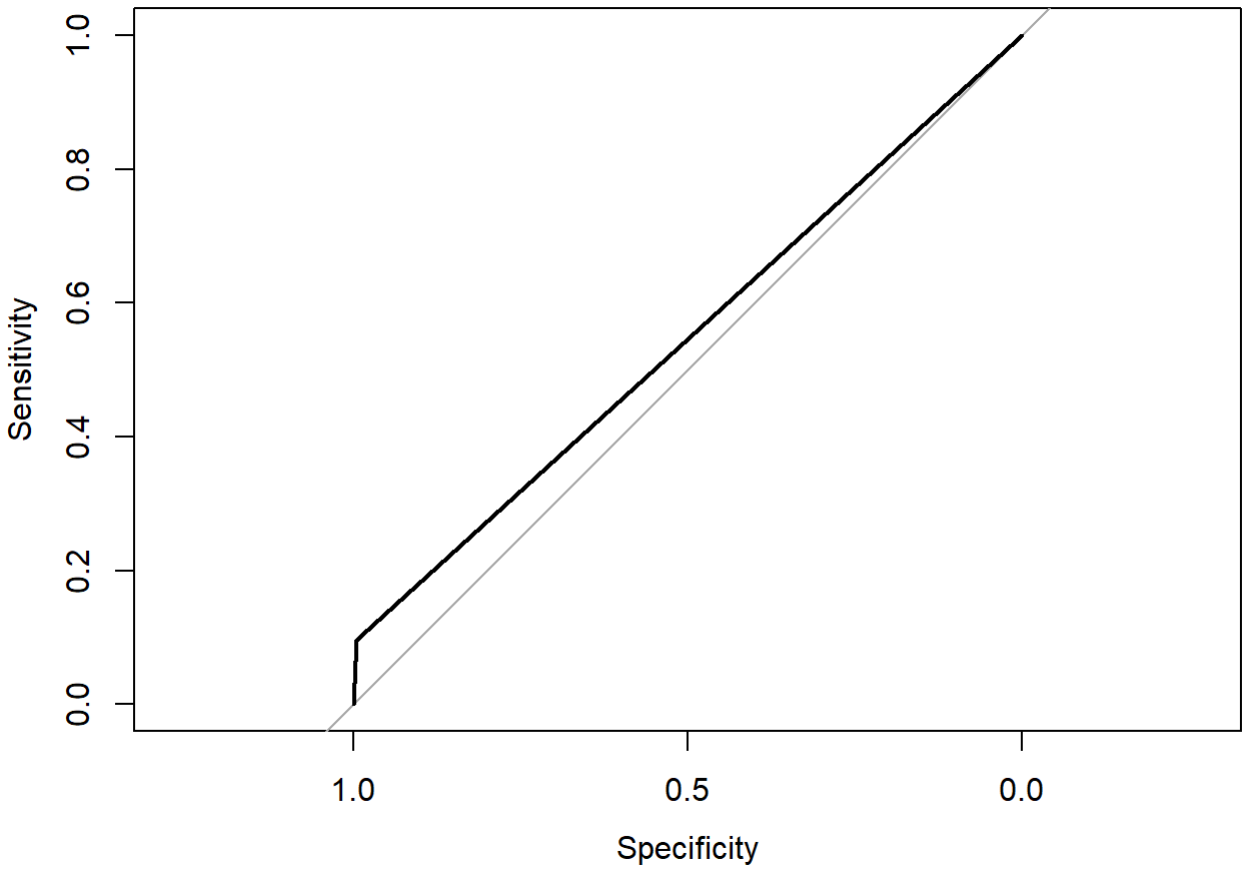



```
importance(mod2)[,3:4]

roc(as.numeric(test1$WnvPresent), as.numeric(rf_pred), plot = TRUE)
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```



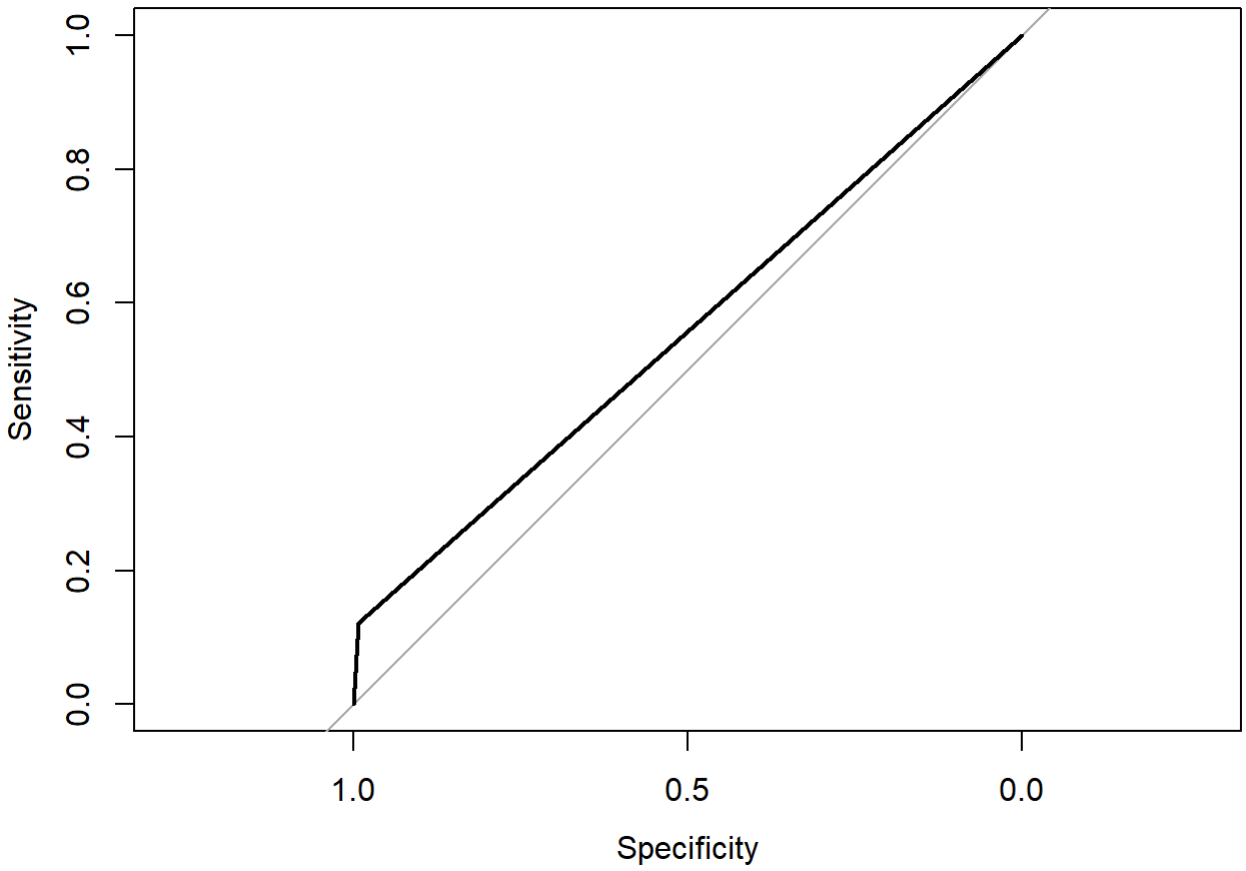

```
F1_Score(test1$WnvPresent, rf_pred)
```

```
mod3 = randomForest( WnvPresent~., data=train1, mtry= (ncol(train1)-1)/3,importance =TRUE)
mod3
rf_pred2 = predict(mod3, newdata = test1)
ytab3 = table(rf_pred2, test1$WnvPresent)
confusionMatrix(ytab3)
```

```
# Generate a ROC curve
roc(as.numeric(test1$WnvPresent), as.numeric(rf_pred2), plot = TRUE)
```

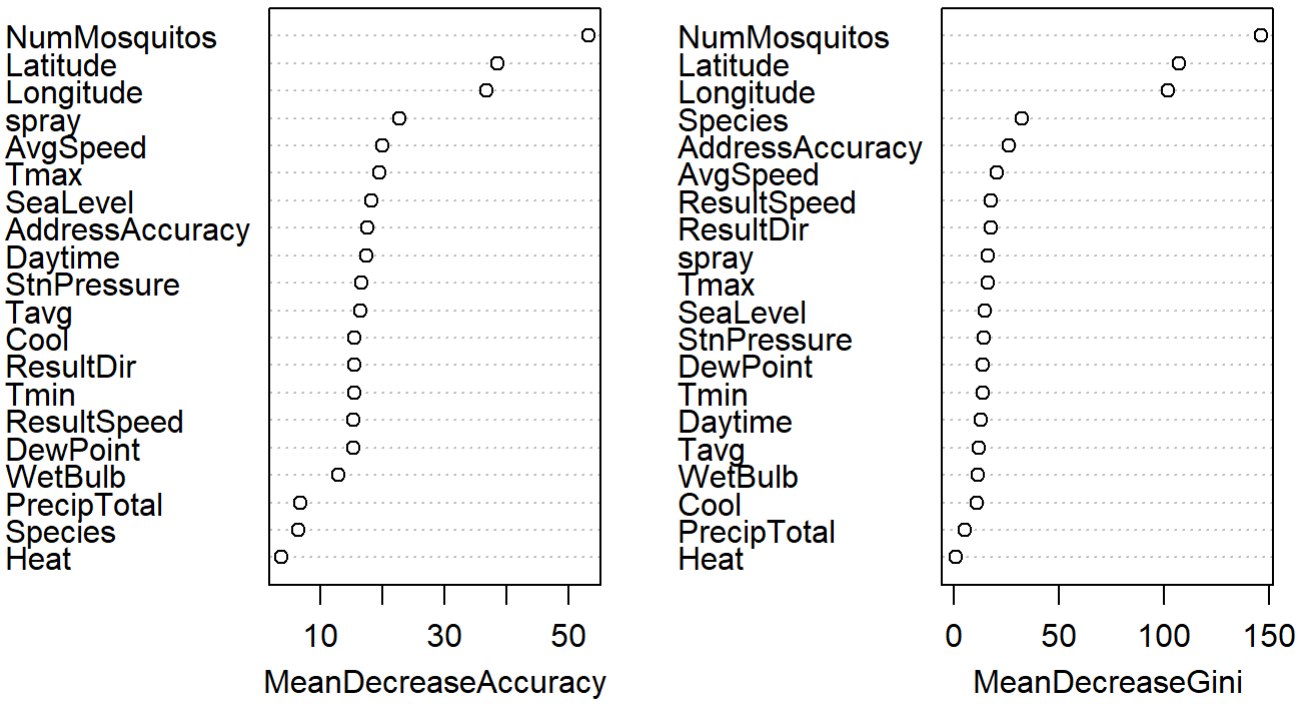
```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```




```
# Calculate F1 score
F1_Score(test1$WnvPresent, rf_pred2)
```

```
varImpPlot(mod3)
```



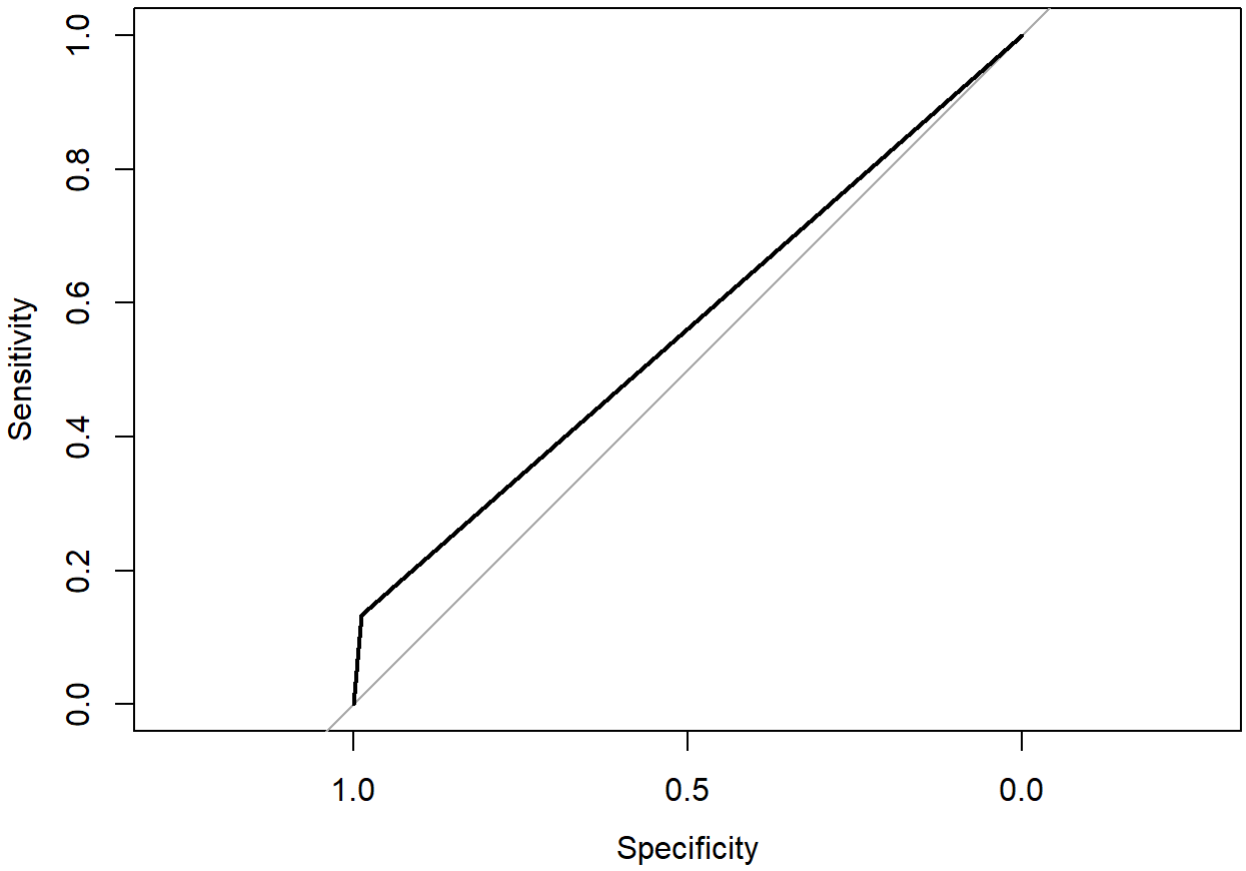

```
importance(mod3)[,3:4]
```

```
mod4 = randomForest( WnvPresent~., data=train1, mtry= ncol(train1)-1,importance =TRUE)
mod4
rf_pred3 = predict(mod4, newdata = test1)
ytab4 = table(rf_pred3, test1$WnvPresent)
confusionMatrix(ytab4)
```

```
roc(as.numeric(test1$WnvPresent), as.numeric(rf_pred3), plot = TRUE)
```

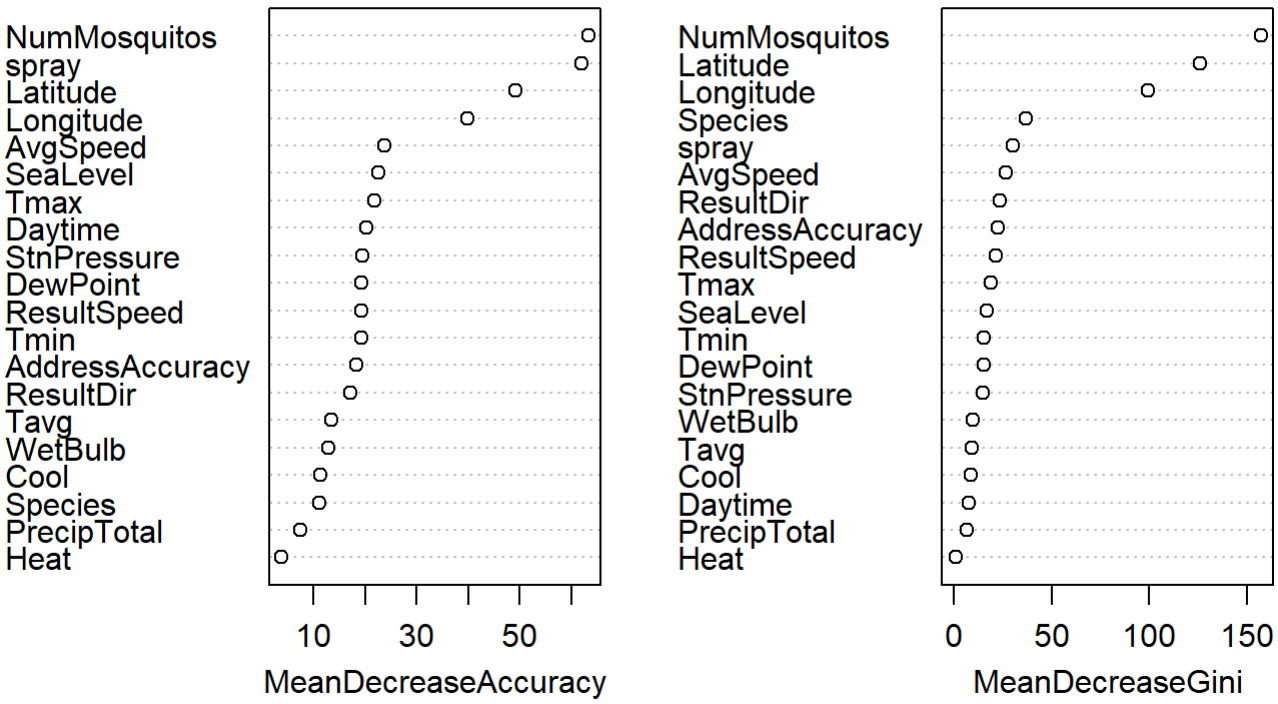
```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```




```
# Calculate F1 score
F1_Score(test1$WnvPresent, rf_pred3)
```

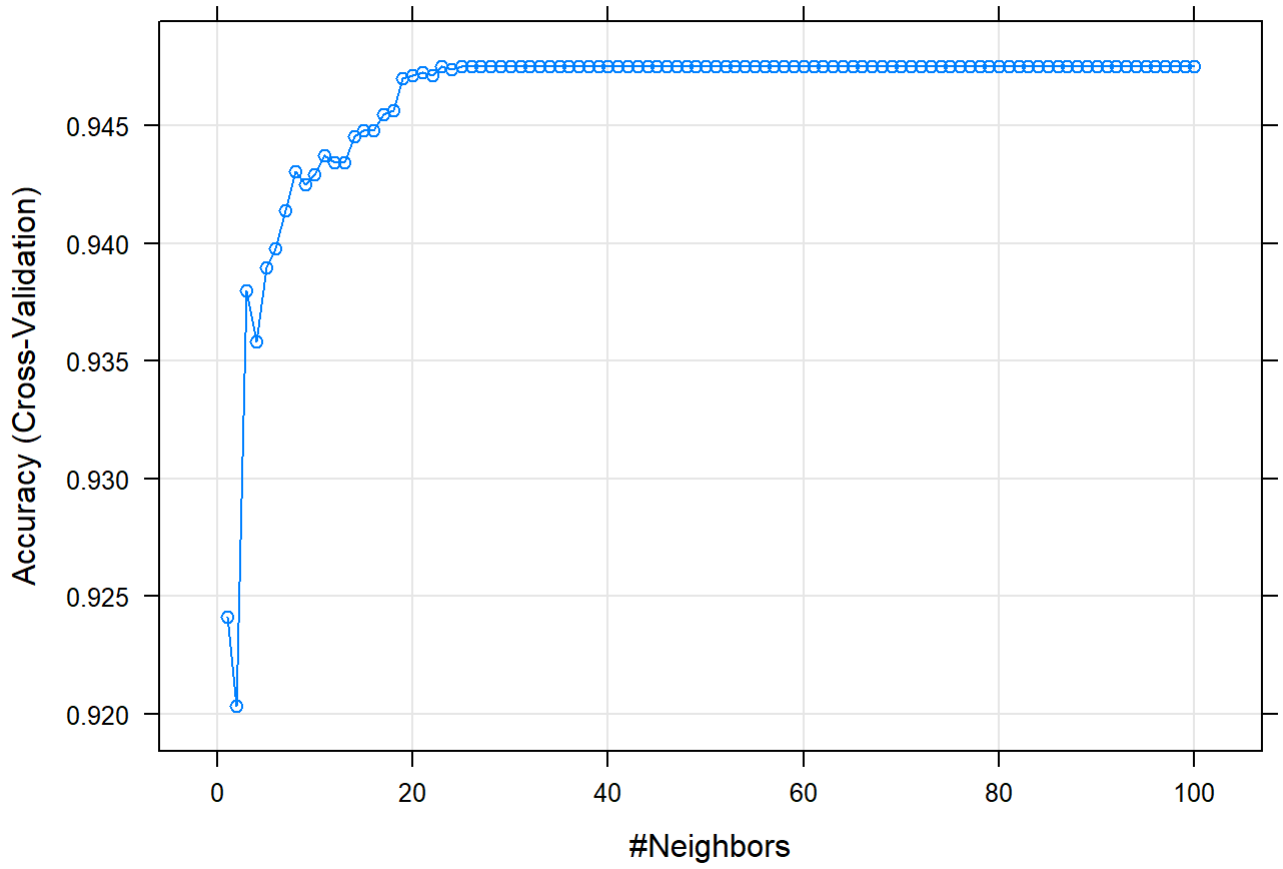
```
varImpPlot(mod4)
```




```
importance(mod4)[,3:4]
```

```
# Create the KNN model
set.seed(100)
trControl = trainControl(method = "cv", number = 5)
knn = train(as.factor(WnvPresent)~.,
            method = "knn",
            tuneGrid = expand.grid(k = 1:100),
            trControl = trControl,
            metric = "Accuracy",
            data = train1)

plot(knn)
```



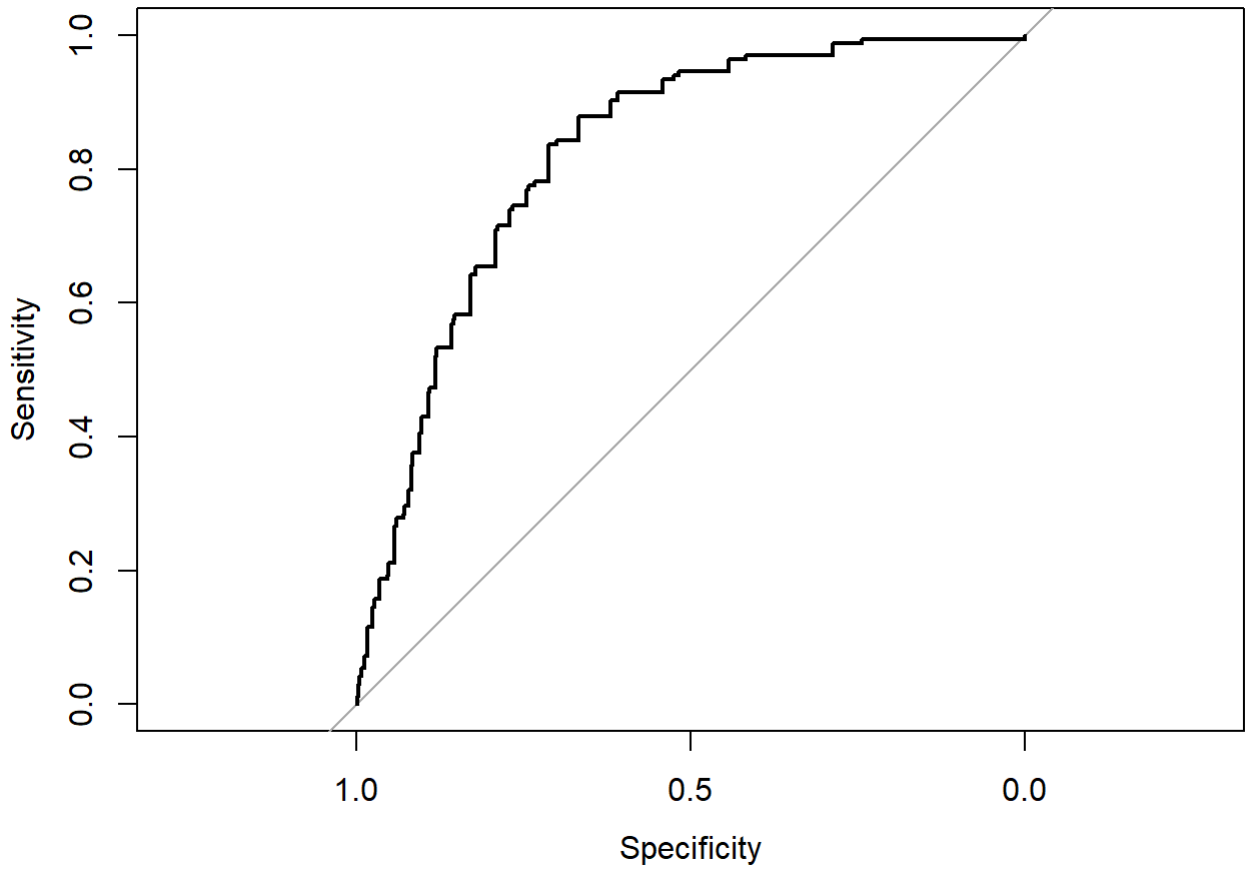

```
knn_pred = predict(knn,test1)
confusionMatrix(knn_pred,as.factor(test1$WnvPresent))
```

```
knn_predi = predict(knn,test1,type = "prob")
knnROC = roc(as.factor(test1$WnvPresent), knn_predi[,1])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
plot(knnROC, type = "S")
```




```
# Create the logistic regression model
set.seed(100)
trControl = trainControl(method = "repeatedcv", number = 10,savePredictions = TRUE)
logis = train(as.factor(WnvPresent)~.,
            method = "glm",
            family = "binomial",
            trControl = trControl,
            tuneLength = 5,
            data = train1)

pred = predict(logis,test1)
```

```
confusionMatrix(pred,as.factor(test1$WnvPresent))
logis_predi = predict(logis,test1,type = "prob")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
logisROC = roc(as.factor(test1$WnvPresent), logis_predi[,1])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
plot(logisROC, type = "S")
```