2a2_estes

Andrew Estes

1/15/2022

Installing libraries, packages, and datasets

1) Consider the Insurance_A.csv data set again. This time, load the data and create a training and testing set using the following (for repeatability):

```
#creating training/test datasets
set.seed(8940)
ins <- sample(1:nrow(insurance), 0.70*floor(nrow(insurance)))
ins.train <- insurance[ins, ]
ins.test <- insurance[-ins, ]
```

A) the training set to create a regression, CART and random forest model to predict expenses using all the other variables.

B) For each model, calculate RMSE using the test set. Comment on how well each model predicts expenses.

```
#linear model
ins.lm <- lm(expenses ~., data=ins.train)
ins.train.lm.pred <- predict(ins.lm, newdata= ins.train)
RMSE(ins.train$expenses, ins.train.lm.pred)
```

```
## [1] 6034.733
```

```
ins.test.lm.predict <- predict(ins.lm, newdata = ins.test)
RMSE(ins.test$expenses, ins.test.lm.predict)
```

```
## [1] 6070.294
```

```
ins.step <- step(ins.lm, trace=0)
ins.test.step.pred <- predict(ins.step, newdata=ins.test)
RMSE(ins.test$expenses, ins.test.step.pred)
```

```
## [1] 6078.724
```

The linear model has a MRSE between 6034.733 abd 6078.724.

```
#CART model
ins.rpart <- rpart(expenses ~., data=ins.train)
ins.test.rpart.pred <- predict(ins.rpart, newdata=ins.test)
RMSE(ins.test$expenses, ins.test.rpart.pred)
```

```
## [1] 5391.953
```

The CART model has a RMSE of 5391.953.

```
#random forest model
ins.rf <- randomForest(expenses ~ ., data=ins.train)
ins.test.rf.pred <- predict(ins.rf, newdata=ins.test)
RMSE(ins.test$expenses, ins.test.rf.pred)
```

```
## [1] 4994.47
```

```
ins.rf
```

```
##
## Call:
##  randomForest(formula = expenses ~ ., data = ins.train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 22093731
##                     % Var explained: 85.07
```

The Random Forest model has a RMSE of 5018.99.

C) Remembering the residual plot from the first assignment, there were some unexplained clumps in the residuals. Briefly explain why those residuals might suggest that a tree-based predictor could be more accurate.

A clear grouping in the residual plots shows that there is a distinction between two (or more) groups that can be split by at least one variable. A tree-based method would likely show the most important variable, separate the group, and create a more accurate picture.

D) Compare RMSE on the test set with the out-of-bag RMSE for the random forest model. How close are they?

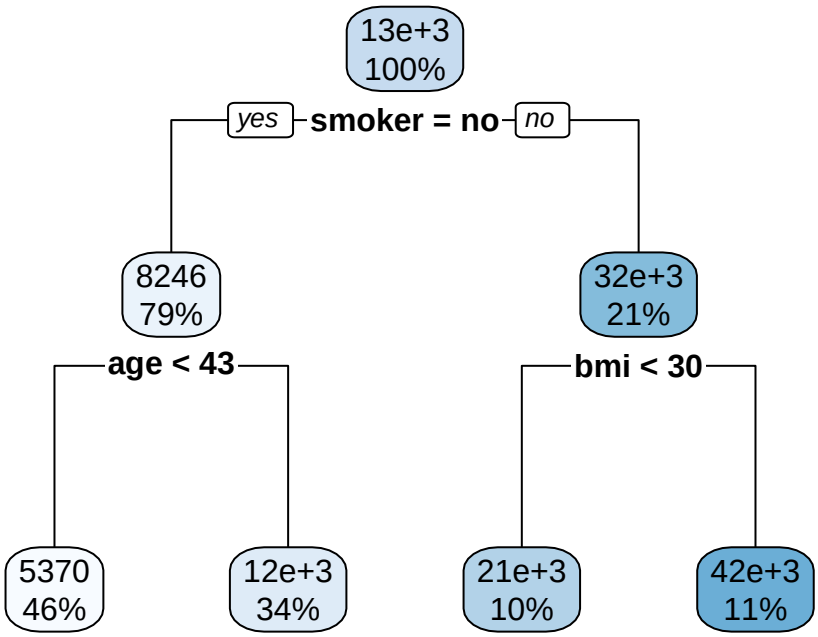
```
RMSEoob <- sqrt(22412374)
RMSEoob
```

```
## [1] 4734.171
```

The Random Forest out-of-bag RMSE is 4734.171.

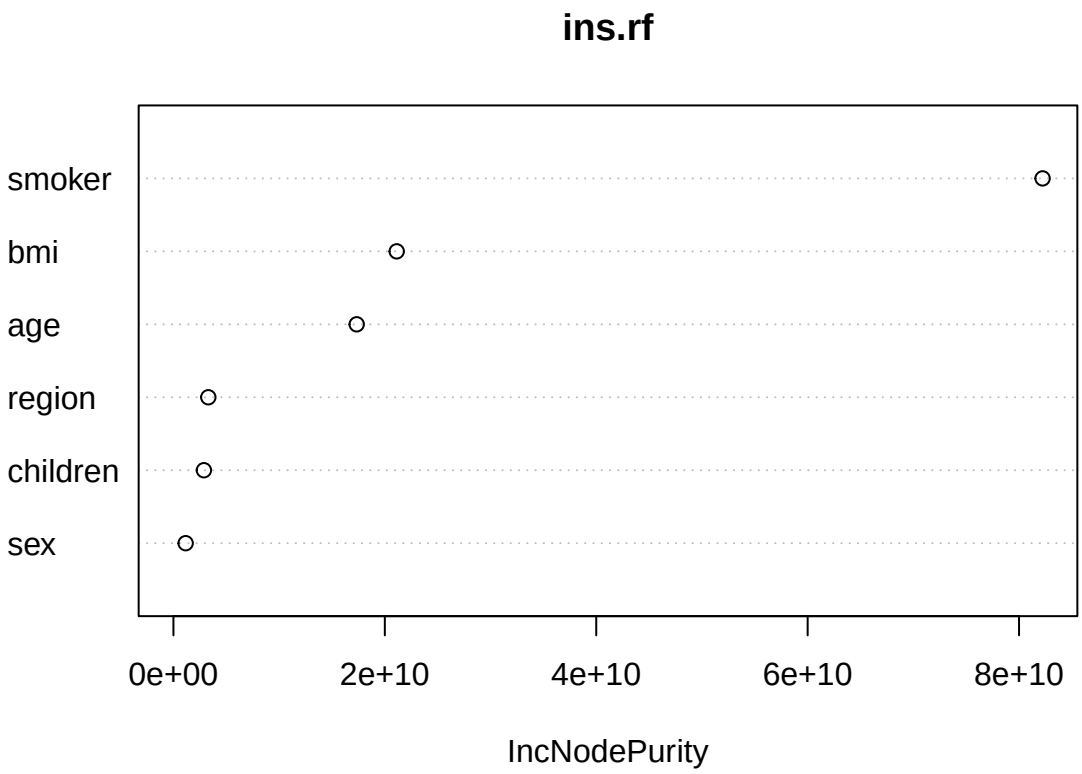
E) Plot the tree diagram for the CART model.

```
rpart.plot(rpart(expenses ~., data=ins.train))
```



F) Plot the variable importance calculated from the random forest model.

```
varImpPlot(ins.rf)
```



G) Using the results from part (e) and part (f), comment on which variables seem to be important in predicting expenses, and in which context each is important. How does this compare to what you saw in the first assignment?

The variables maintain the same level of importance in both CART and RF. Smoking status is the first predictor, followed by Age and BMI. It has a similar output to the first assignment where smoking was the crucial variable, followed by a combination of Age, BMI, Gender, Children, and Region. The CART output would be my preference based off the limited number of variables/nodes it broke the data into.

1) Download the data set dry_bean_dataset.csv. This data set comes from a recent paper by Murat Koklu and Ilker Ali Ozkan that explored using various geometric measures of dry beans to identify their variety. This classification is important in the seed industry, where providing uniform seed supplies is important. The data set consists of measurements of 13611 dry beans which come from 7 varieties. The variety of bean is identified in the Class variable, and all other variables contain geometric measures derived from imaging of the beans.

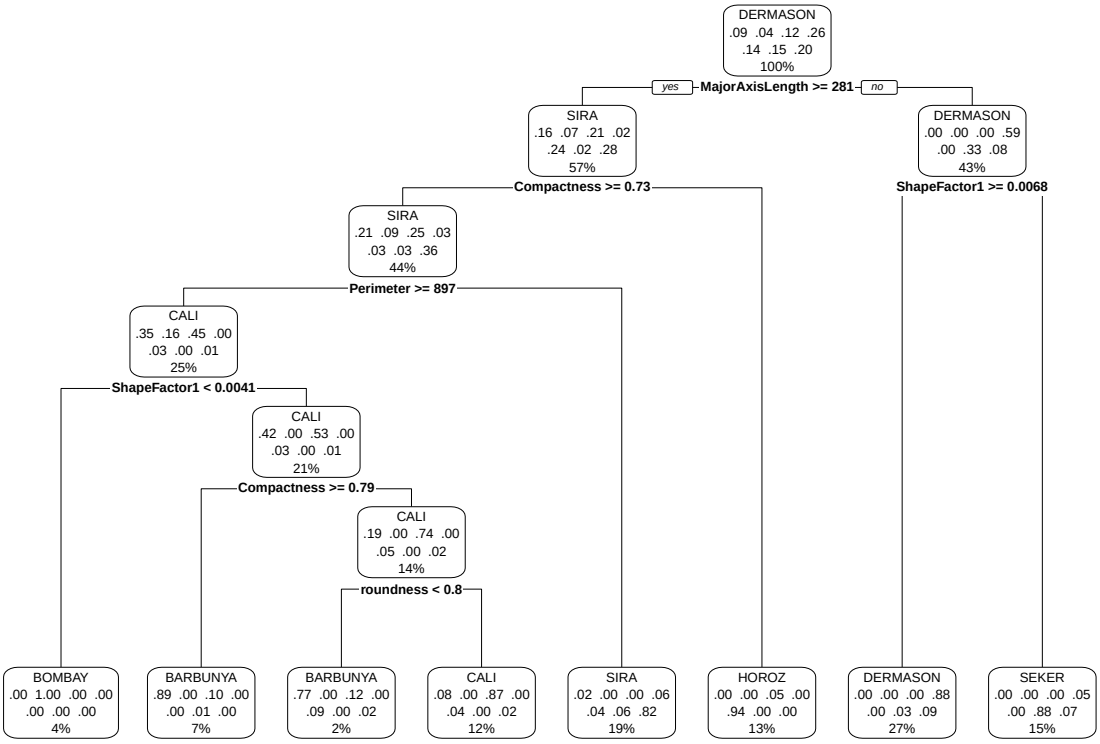
A) Load the data set, and create testing and training sets (this time, you can code it as you like).

```
drybeans <- read.csv("beans.csv")
drybeans$Class <- as.factor(drybeans$Class)

set.seed(65489)
beans <- sample(1:nrow(drybeans), 0.70*floor(nrow(drybeans)))
beans.train <- drybeans[beans, ]
beans.test <- drybeans[-beans, ]
```

B) Create CART, random forest, and SVM classification models for bean variety, calculate and report on the accuracy of each model.

```
#CART
beans.rpart <- rpart(Class ~., data=beans.train)
beans.test.rpart.pred <- (predict(beans.rpart, newdata=beans.test))
rpart.plot(rpart(Class ~., data=beans.train))
```




```
mean(beans.test.rpart.pred)
```

```
## [1] 0.1428571
```

The mean is .143, equivalent ot a .857 accuracy method.

```
#random forest model
beans.rf <- randomForest(Class ~., data=beans.train)
beans.test.rf.pred <- predict(beans.rf, newdata=beans.test)
confusionMatrix(beans.test.rf.pred, beans.test$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
##   BARBUNYA      379      0   12        0     1     2    3
##   BOMBAY          0    141    0        0     0     0    0
##   CALI           37      0  456        0    10     0    1
##   DERMASON        0      0    0      969     5    11   63
##   HOROZ           4      0   13        1   593     0    9
##   SEKER           2      0    2       25     0   552   15
##   SIRA           14      0    3       62     6    11  682
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9236
##                  95% CI : (0.915, 0.9316)
##     No Information Rate : 0.2588
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9077
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: BARBUNYA Class: BOMBAY Class: CALI Class: DERMASON
## Sensitivity                  0.86927       1.00000      0.9383          0.9167
## Specificity                  0.99507       1.00000      0.9867          0.9739
## Pos Pred Value               0.95466       1.00000      0.9048          0.9246
## Neg Pred Value               0.98454       1.00000      0.9916          0.9710
## Prevalence                   0.10676       0.03452      0.1190          0.2588
## Detection Rate               0.09280       0.03452      0.1117          0.2373
## Detection Prevalence         0.09721       0.03452      0.1234          0.2566
## Balanced Accuracy            0.93217       1.00000      0.9625          0.9453
##                      Class: HOROZ Class: SEKER Class: SIRA
## Sensitivity                0.9642       0.9583      0.8823
## Specificity                0.9922       0.9875      0.9710
## Pos Pred Value             0.9565       0.9262      0.8766
## Neg Pred Value             0.9936       0.9931      0.9725
## Prevalence                 0.1506       0.1410      0.1893
## Detection Rate             0.1452       0.1352      0.1670
## Detection Prevalence       0.1518       0.1459      0.1905
## Balanced Accuracy          0.9782       0.9729      0.9266
```

Confusion matrix shows an overall accuracy of .923 with a 95% Confidence Interval of .914 and .931.

```
#SVM
beans.svm <- svm(Class ~., data=beans.train)
beans.svm.pred <- predict(beans.svm, newdata=beans.test)
confusionMatrix(beans.svm.pred, beans.test$Class, positive="TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
##   BARBUNYA      383      0    8        0     0     1    1
##   BOMBAY          0    141    0        0     0     0    0
##   CALI           32      0  462        0    10     0    0
##   DERMASON        0      0    0      974     5     7   61
##   HOROZ           3      0   13        1   590     0    8
##   SEKER           2      0    1       21     0   550    7
##   SIRA           16      0    2       61    10    18  696
##
## Overall Statistics
##
##                Accuracy : 0.9295
```

```
##                  95% CI : (0.9212, 0.9371)
##     No Information Rate : 0.2588
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9148
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: BARBUNYA Class: BOMBAY Class: CALI Class: DERMASON
## Sensitivity                  0.87844       1.00000      0.9506          0.9215
## Specificity                  0.99726       1.00000      0.9883          0.9759
## Pos Pred Value               0.97455       1.00000      0.9167          0.9303
## Neg Pred Value               0.98564       1.00000      0.9933          0.9727
## Prevalence                   0.10676       0.03452      0.1190          0.2588
## Detection Rate               0.09378       0.03452      0.1131          0.2385
## Detection Prevalence         0.09623       0.03452      0.1234          0.2564
## Balanced Accuracy            0.93785       1.00000      0.9695          0.9487
##                      Class: HOROZ Class: SEKER Class: SIRA
## Sensitivity                0.9593       0.9549      0.9004
## Specificity                0.9928       0.9912      0.9677
## Pos Pred Value             0.9593       0.9466      0.8667
## Neg Pred Value             0.9928       0.9926      0.9765
## Prevalence                 0.1506       0.1410      0.1893
## Detection Rate             0.1445       0.1347      0.1704
## Detection Prevalence       0.1506       0.1423      0.1966
## Balanced Accuracy          0.9761       0.9730      0.9340
```

The Confusion Matrix has an overall accuracy of .93 with a 95% Confidence Interval of .921 and .937.

C) The link above leads to the paper abstract, which contains information on the accuracy obtained by the authors for decision tree (DT) and SVM models, among others. Write a short explanation of how your (untuned) models compared to the stated accuracies in the abstract.

The accuracy numbers provided by the confusion matrices in the SVM and Decision Tree (Random Forest) was comparable to the author's numbers (.93 vs .9313 and .923 vs .925).

D) The abstract also breaks down the SVM model's accuracy on each variety separately. How does your SVM model compare? You might have to figure out how to calculate your model's accuracy in predicting a single variety.

The author's classification for each bean class surprisingly varied quite a bit from my models. With the author's accuracy first, the results are as follows: Barbunya (.924 vs .878) , Bombay (1.0 vs 1.0), Cali (.95 vs .951), Dermason (.944 vs .922), Horoz (.949 vs .959), Seker (.947vs .955) and Sira (.868 vs .9).

E) The abstract claims that the author's SVM model had an accuracy of 86.84% in predicting the SIRA variety. Let's see how a logistic regression model can do at predicting whether a bean is of the SIRA variety or not.

E1) First create a response variable that indicates whether the bean is SIRA variety or not.

```
sira.train <- as.numeric(beans.train$Class == "SIRA")
beans.train$Class = sira.train

sira.test <- as.numeric(beans.test$Class == "SIRA")
beans.test$Class = sira.test
```

E2) Next, create a logistic regression model to predict this new binary variable.

```
#logistic regression
beans.log.train.pred <- glm(Class ~ ., data=beans.train, family="binomial")
summary(beans.log.train.pred)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = beans.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7496  -0.0680  -0.0048   0.0000   4.0269
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.696e+03  7.316e+02  -5.051 4.39e-07 ***
## Area            -1.342e-02  4.070e-03  -3.296 0.000979 ***
## Perimeter       -6.438e-01  5.841e-02 -11.022  < 2e-16 ***
## MajorAxisLength -1.350e+00  7.772e-01  -1.737 0.082409 .
## MinorAxisLength -4.000e+00  1.595e+00  -2.508 0.012137 *
## AspectRation     2.849e+02  7.271e+01   3.918 8.91e-05 ***
## Eccentricity    -2.367e+02  2.943e+01  -8.043 8.80e-16 ***
## ConvexArea       3.014e-05  3.056e-03   0.010 0.992130
## EquivDiameter    1.350e+01  2.497e+00   5.404 6.51e-08 ***
## Extent           4.639e+00  1.220e+00   3.803 0.000143 ***
## Solidity         8.501e+01  1.347e+02   0.631 0.527910
## roundness       -3.034e+02  2.679e+01 -11.322  < 2e-16 ***
## Compactness      7.698e+03  1.124e+03   6.847 7.56e-12 ***
## ShapeFactor1     3.644e+04  1.569e+04   2.322 0.020231 *
## ShapeFactor2     4.858e+04  2.281e+04   2.129 0.033230 *
## ShapeFactor3    -4.038e+03  5.429e+02  -7.438 1.02e-13 ***
## ShapeFactor4    -1.226e+03  3.656e+02  -3.353 0.000800 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 9415.9  on 9526  degrees of freedom
## Residual deviance: 2543.1  on 9510  degrees of freedom
## AIC: 2577.1
##
## Number of Fisher Scoring iterations: 12
```

AIC of 2577.1.

E3) Use your test set to estimate the accuracy of the logistic regression model in predicting SIRA.

```r
#logistic regression
beans.log.test <- glm(Class ~., data=beans.test, family="binomial")
summary(beans.log.test)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = beans.test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5289  -0.0395  -0.0013   0.0000   4.3637
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.306e+03  1.320e+03  -1.747 0.080564 .
## Area            -2.158e-02  8.505e-03  -2.538 0.011156 *
## Perimeter       -1.231e+00  1.364e-01  -9.028  < 2e-16 ***
## MajorAxisLength -2.804e+00  1.437e+00  -1.951 0.051028 .
## MinorAxisLength -1.011e+01  2.850e+00  -3.547 0.000390 ***
## AspectRation     2.487e+02  1.322e+02   1.881 0.060016 .
## Eccentricity    -2.349e+02  5.179e+01  -4.536 5.74e-06 ***
## ConvexArea       1.078e-02  6.386e-03   1.689 0.091253 .
## EquivDiameter    2.049e+01  4.622e+00   4.433 9.31e-06 ***
## Extent           2.702e-01  2.017e+00   0.134 0.893447
## Solidity         5.358e+02  2.925e+02   1.832 0.066992 .
## roundness       -5.620e+02  6.157e+01  -9.127  < 2e-16 ***
## Compactness      6.228e+03  1.961e+03   3.176 0.001494 **
## ShapeFactor1     3.894e+04  2.977e+04   1.308 0.190893
## ShapeFactor2    -5.459e+04  3.110e+04  -1.755 0.079200 .
## ShapeFactor3    -2.214e+03  9.234e+02  -2.397 0.016510 *
## ShapeFactor4    -2.329e+03  6.587e+02  -3.536 0.000406 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3962.87  on 4083  degrees of freedom
## Residual deviance:  968.73  on 4067  degrees of freedom
## AIC: 1002.7
##
## Number of Fisher Scoring iterations: 16
```

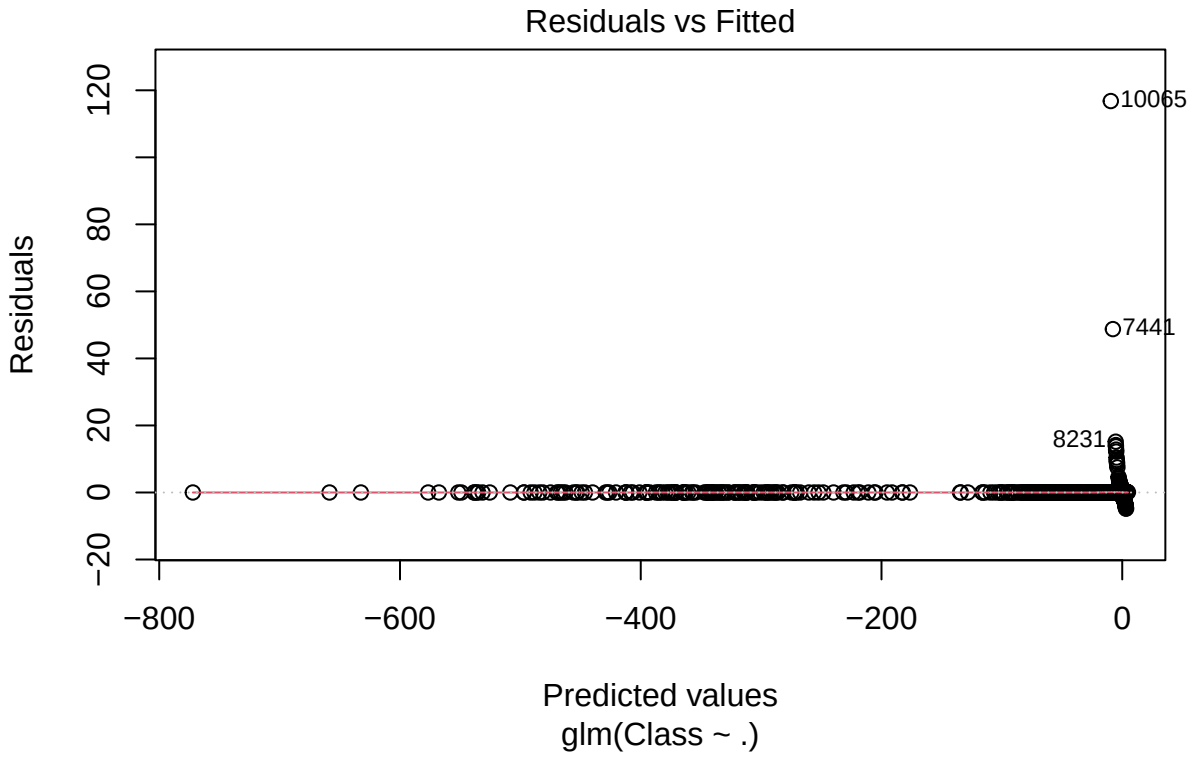
```r
odds <- predict(beans.log.train.pred, beans.test, type="response")
beans.odds <- ifelse(odds > .5, "SIRA", "NOT SIRA")
table(beans.odds)
```

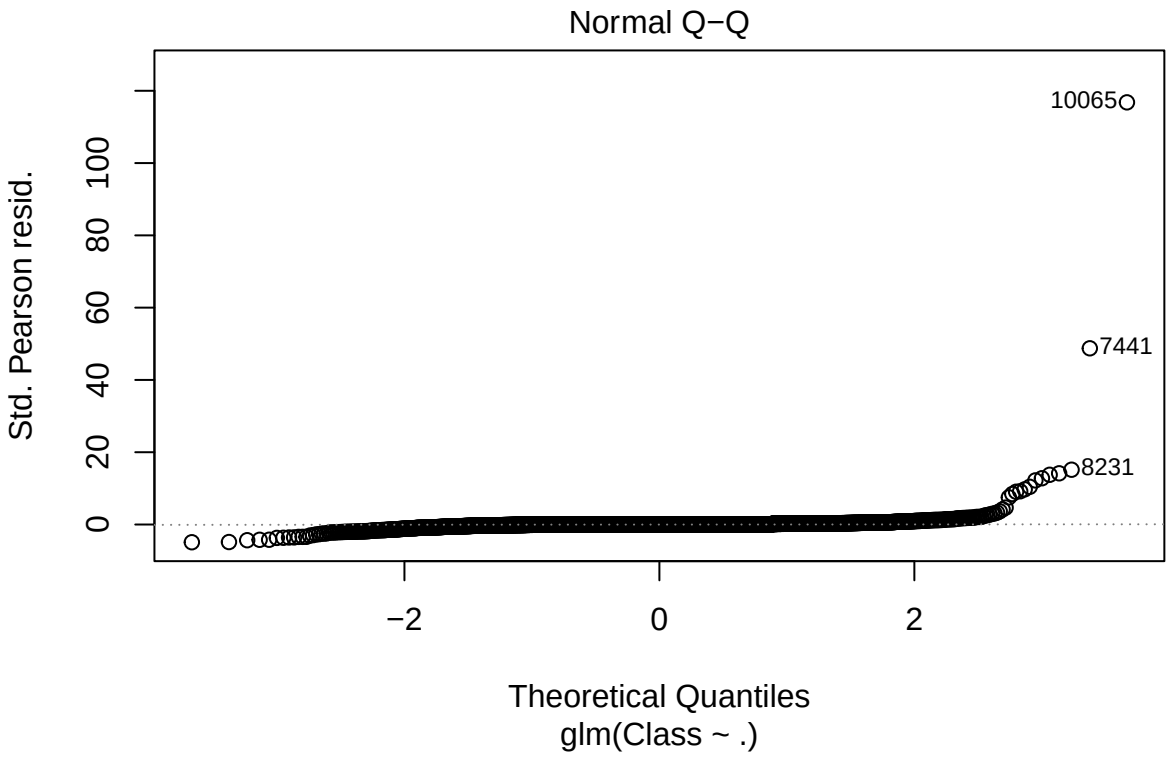
```
## beans.odds
## NOT SIRA     SIRA
##     3276      808
```

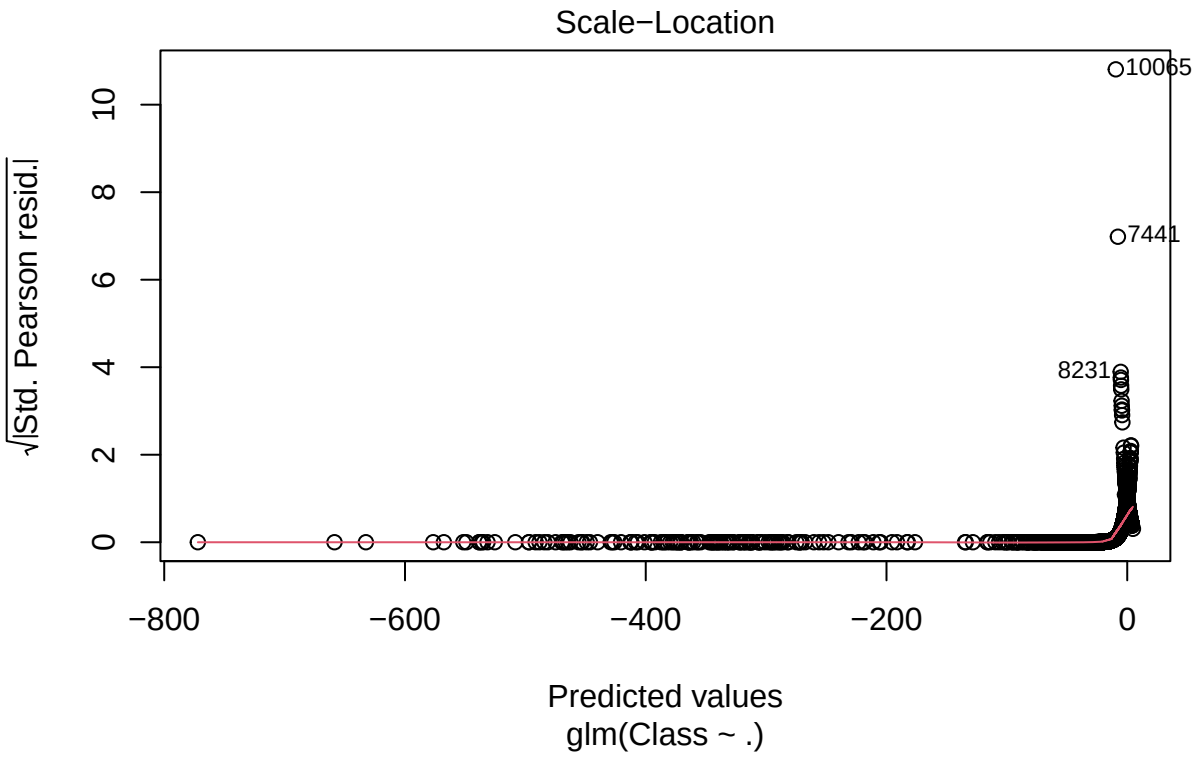
E4) Compare that accuracy to the article author's SVM model.

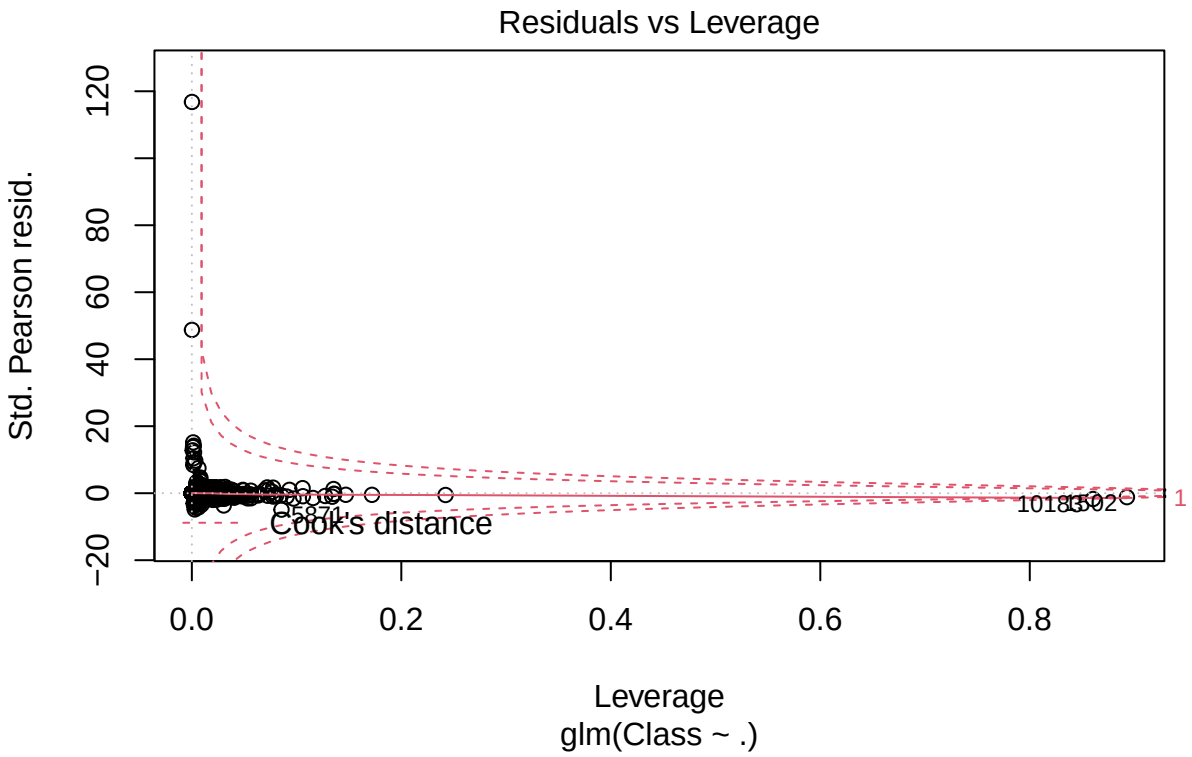
Different measures of accuracy for SVM and Logistic Regression. AIC of 410.3 Using the predict function, it predicted 19.8% of beans were SIRA. The ROC Curve provided 98.5% accuracy (something must be wrong with the way I calculated it - likely using the same dataset rather than a new one). Plotted graphs are below.

```r
plot(beans.log.test)
```

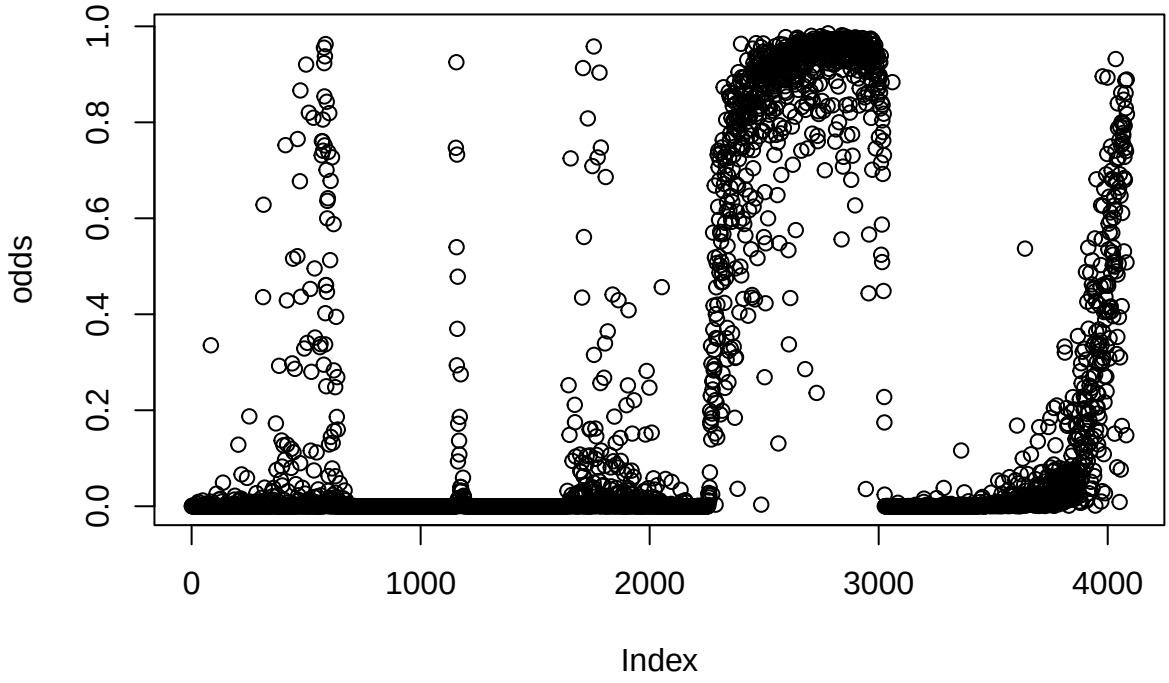



Normal Q−Q
Std. Pearson resid.
0
20
40
60
80
100
10065
7441
8231
−2
0
2
Theoretical Quantiles
glm(Class ~ .)

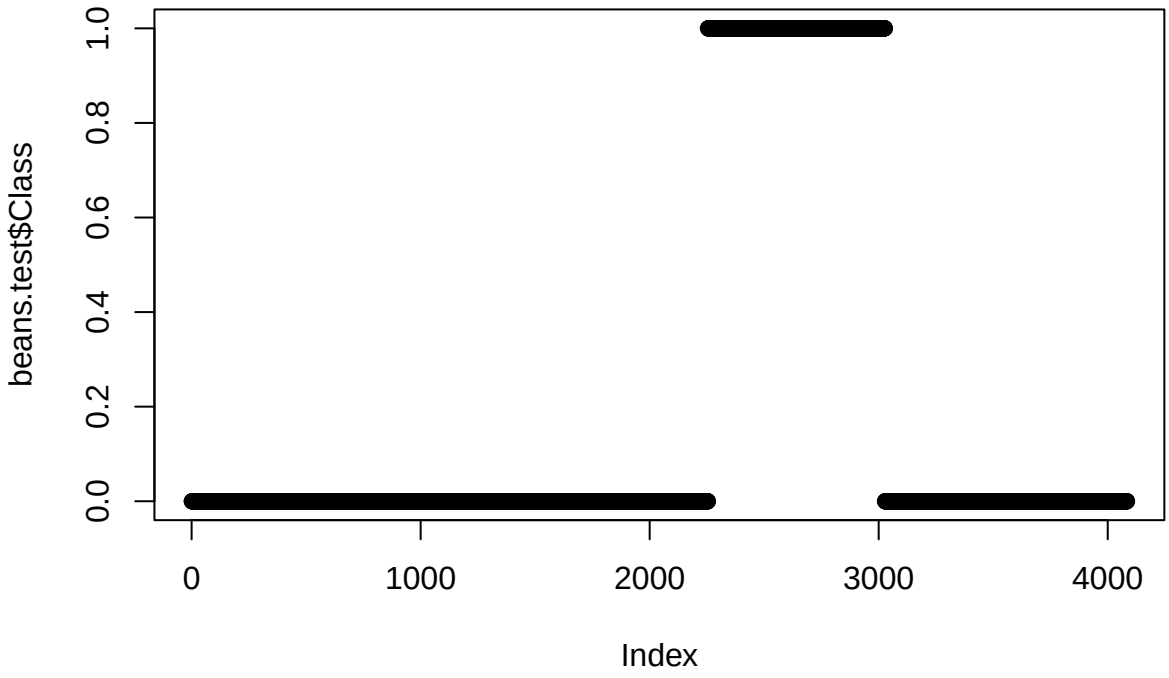

Scale−Location
10065
7441
8231
√|Std. Pearson resid.|
0
2
4
6
8
10
−800
−600
−400
−200
0
Predicted values
glm(Class ~ .)


Residuals vs Leverage
Std. Pearson resid.
120
80
60
40
20
0
−20
0.0
0.2
0.4
0.6
0.8
Cook's distance
1
Leverage
glm(Class ~ .)

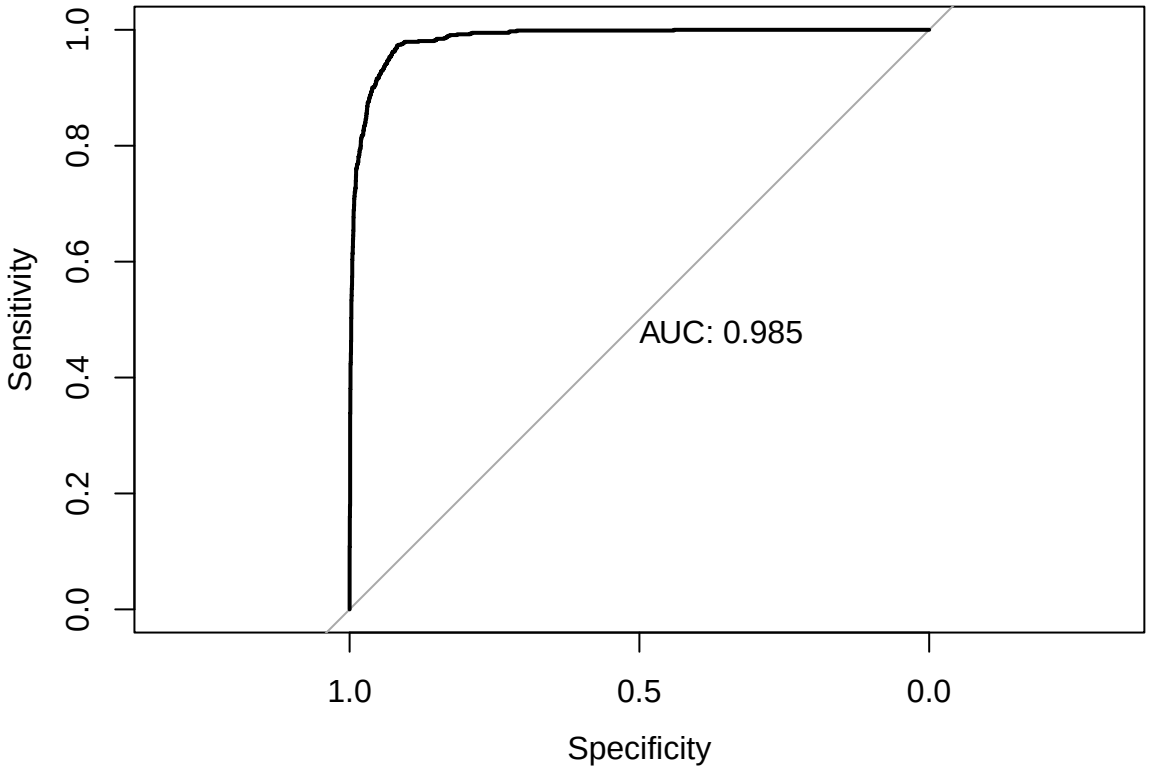
```
plot(odds)
```




```
plot(beans.test$Class)
```




```
test_prob <- predict(beans.log.train.pred, beans.test, type="response")
test_roc <- roc(beans.test$Class ~ test_prob, plot = TRUE, print.auc = TRUE)
```


Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
AUC: 0.985
1.0
0.5
0.0
Specificity