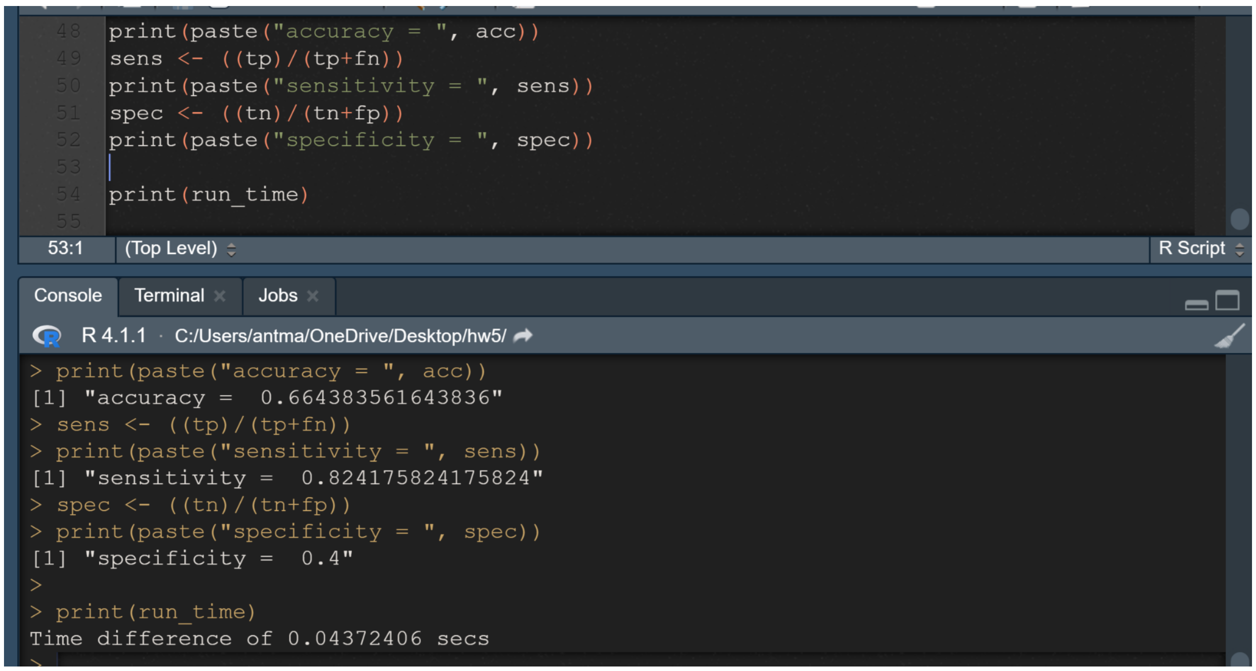
Part I: Logistic Regression

The logistic regression models got similar metrics. The accuracy with the model in R was 66% while the accuracy for the C++ mode was 67%. The sensitivity and specificity scores were very close to each other as well. However, the two models had vastly different execution times. The R script ran in .04 sec while to C++ ran in about 205 seconds. The reasoning for this slow execution time is most likely due to the 500000 iterations that must be done on nested for loops. For the R script, I calculated run time using proc.time(). I started the clock before running the model on the training data and stopped the clock immediately after. The way I computed runtime in C++ was by using the chrono library. This was very similar to running proc.time() in R. I started the clock when I created the model and stopped it before calculating/outputting any other data.

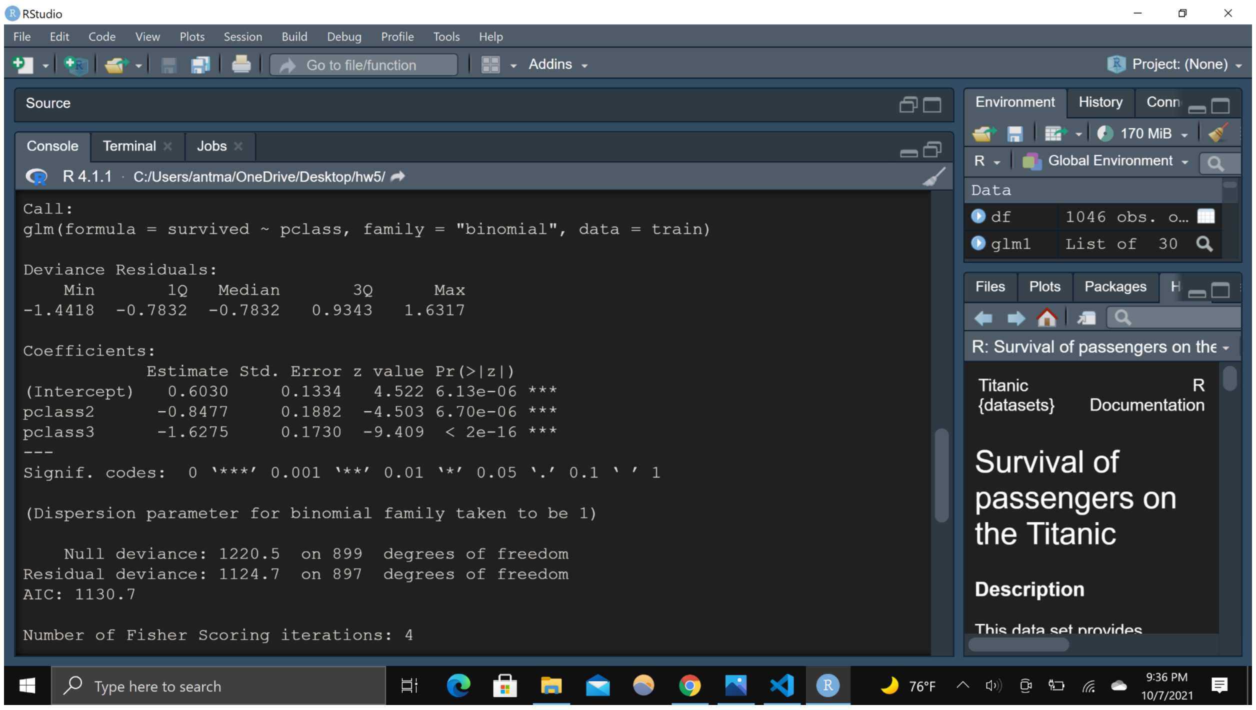
Screenshots of metrics in R.



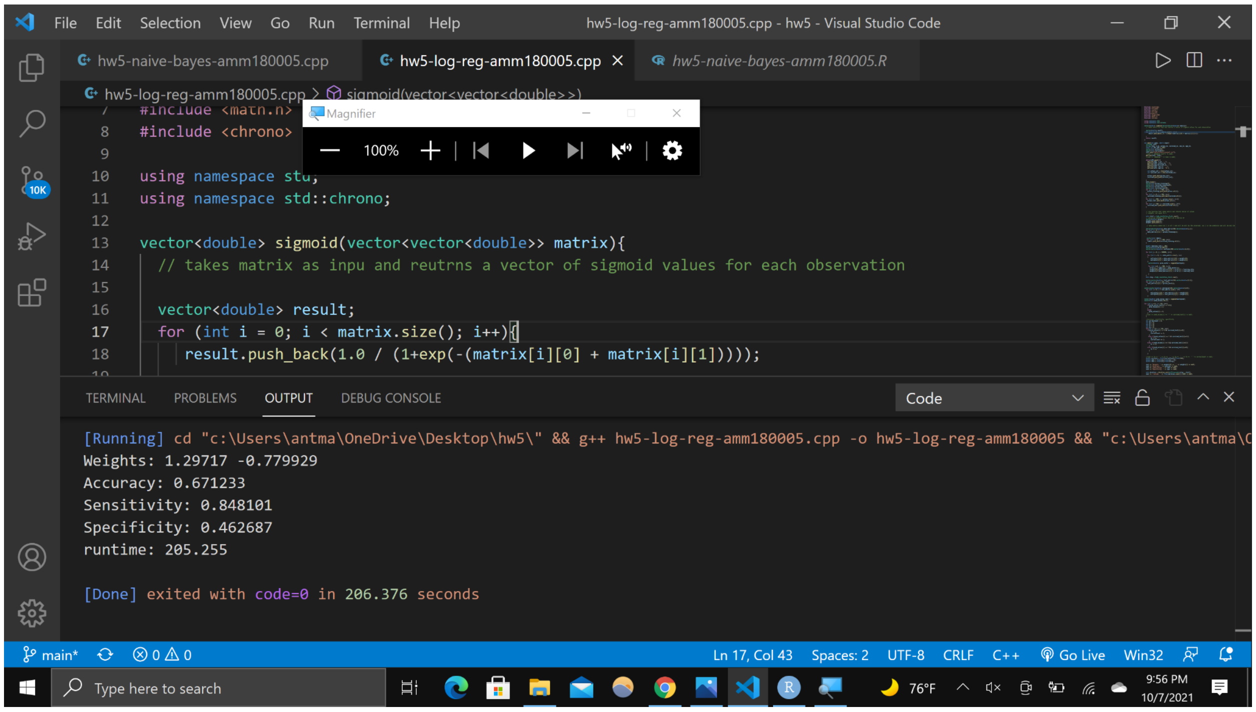
```
print(paste("accuracy = ", acc))
sens <- ((tp)/(tp+fn))
print(paste("sensitivity = ", sens))
spec <- ((tn)/(tn+fp))
print(paste("specificity = ", spec))

print(run_time)
```

```
> print(paste("accuracy = ", acc))
[1] "accuracy =  0.664383561643836"
> sens <- ((tp)/(tp+fn))
> print(paste("sensitivity = ", sens))
[1] "sensitivity =  0.824175824175824"
> spec <- ((tn)/(tn+fp))
> print(paste("specificity = ", spec))
[1] "specificity =  0.4"
>
> print(run_time)
Time difference of 0.04372406 secs
```

Screenshot of output in R


```
Call:
glm(formula = survived ~ pclass, family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4418  -0.7832  -0.7832   0.9343   1.6317

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.6030     0.1334   4.522 6.13e-06 ***
pclass2      -0.8477     0.1882  -4.503 6.70e-06 ***
pclass3      -1.6275     0.1730  -9.409  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1220.5  on 899  degrees of freedom
Residual deviance: 1124.7  on 897  degrees of freedom
AIC: 1130.7

Number of Fisher Scoring iterations: 4
```

Screenshot of Output and metrics in c++:

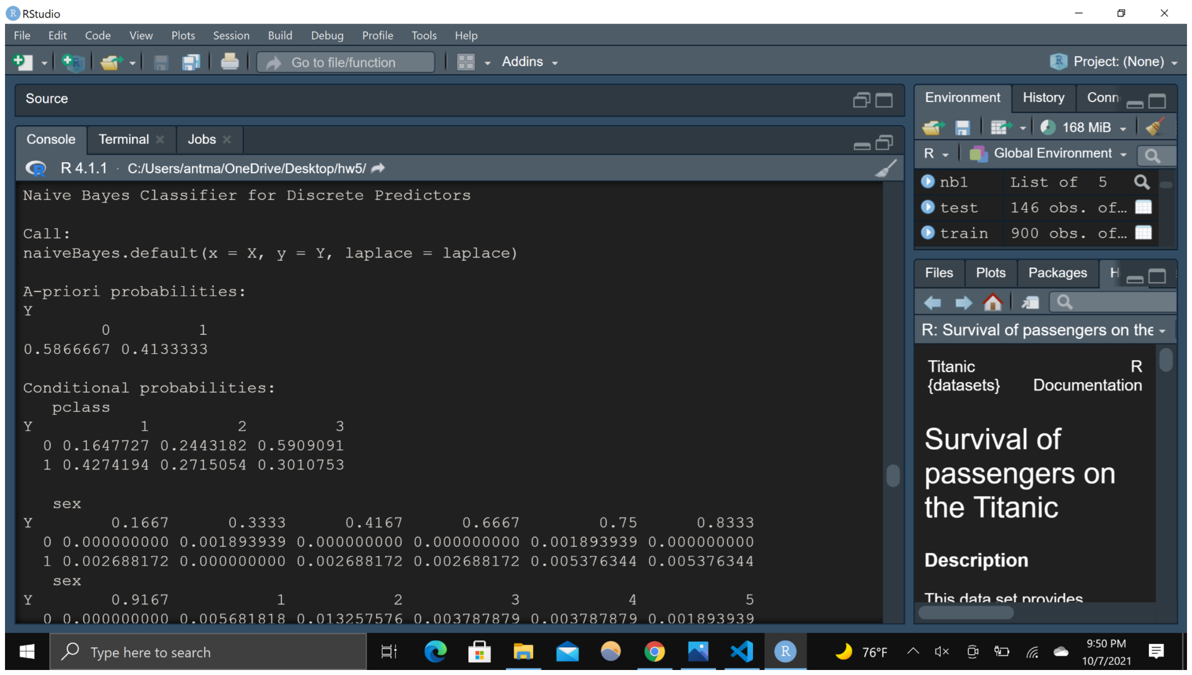

```cpp
#include <math.h>
#include <chrono>

using namespace std;
using namespace std::chrono;

vector<double> sigmoid(vector<vector<double>> matrix){
  // takes matrix as inpu and reutrns a vector of sigmoid values for each observation

  vector<double> result;
  for (int i = 0; i < matrix.size(); i++){
    result.push_back(1.0 / (1+exp(-(matrix[i][0] + matrix[i][1]))));
```

```
[Running] cd "c:\Users\antma\OneDrive\Desktop\hw5\" && g++ hw5-log-reg-amm180005.cpp -o hw5-log-reg-amm180005 && "c:\Users\antma\C
Weights: 1.29717 -0.779929
Accuracy: 0.671233
Sensitivity: 0.848101
Specificity: 0.462687
runtime: 205.255

[Done] exited with code=0 in 206.376 seconds
```

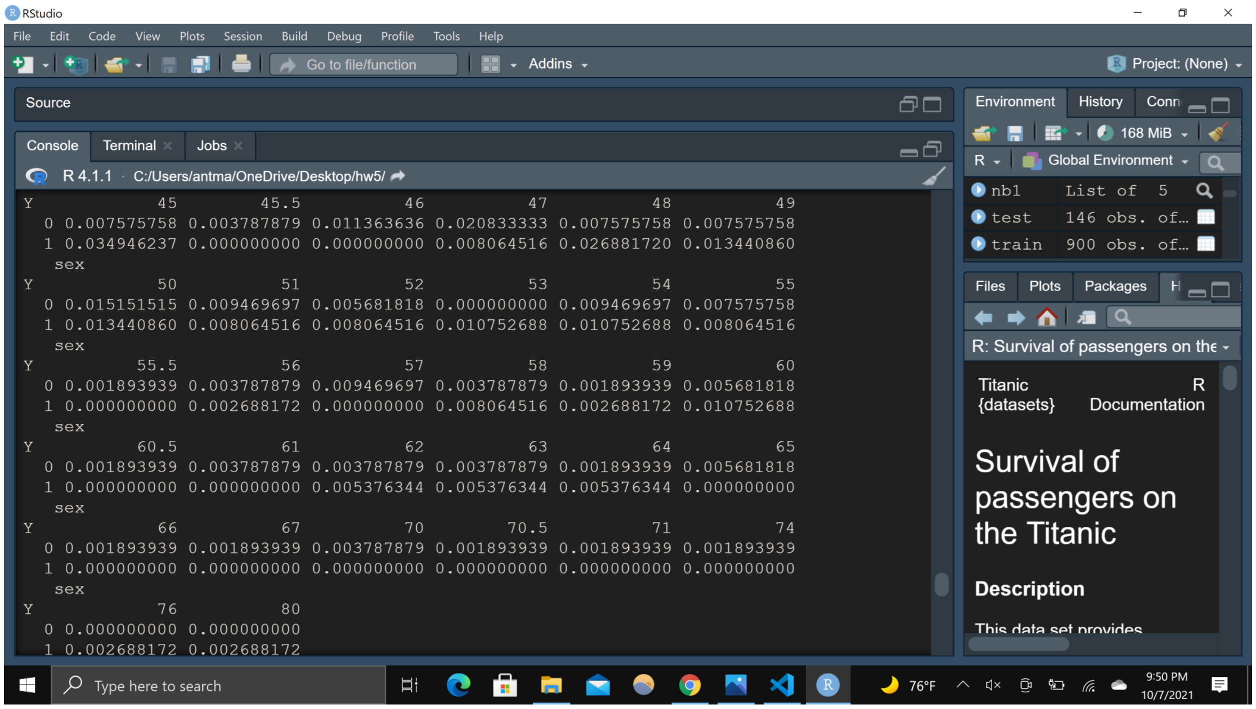
Part II : Naive Bayes

The Naives Bayes models had almost identical metrics and output. The R script had an accuracy of 62% while the C++ model had an accuracy of 58%. The specificity and sensitivity rates were also very close to each other. The more exciting metric is the runtime of these two models. The C++ model ran faster than the R.script model, .00099sec vs .0039sec. I calculated runtime the same way as I did for the logistic regression models. Using proc.time() in the R script right before and after the model executes while using the chrono library for c++ model. Making sure to only measure the run time for the model and not for outputting/calculating the results.

Screenshots of output in R:



```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.5866667 0.4133333 

Conditional probabilities:
   pclass
Y           1         2         3
  0 0.1647727 0.2443182 0.5909091
  1 0.4274194 0.2715054 0.3010753

   sex
Y        0.1667      0.3333      0.4167      0.6667        0.75      0.8333
  0 0.000000000 0.001893939 0.000000000 0.000000000 0.001893939 0.000000000
  1 0.002688172 0.000000000 0.002688172 0.002688172 0.005376344 0.005376344
   sex
Y        0.9167           1           2           3           4           5
  0 0.000000000 0.005681818 0.013257576 0.003787879 0.003787879 0.001893939
```



```
Y            45         45.5           46           47           48           49
  0 0.007575758 0.003787879 0.011363636 0.020833333 0.007575758 0.007575758
  1 0.034946237 0.000000000 0.000000000 0.008064516 0.026881720 0.013440860
   sex
Y            50           51           52           53           54           55
  0 0.015151515 0.009469697 0.005681818 0.000000000 0.009469697 0.007575758
  1 0.013440860 0.008064516 0.008064516 0.010752688 0.010752688 0.008064516
   sex
Y          55.5           56           57           58           59           60
  0 0.001893939 0.003787879 0.009469697 0.003787879 0.001893939 0.005681818
  1 0.000000000 0.002688172 0.000000000 0.008064516 0.002688172 0.010752688
   sex
Y          60.5           61           62           63           64           65
  0 0.001893939 0.003787879 0.003787879 0.003787879 0.001893939 0.005681818
  1 0.000000000 0.000000000 0.005376344 0.005376344 0.005376344 0.000000000
   sex
Y            66           67           70         70.5           71           74
  0 0.001893939 0.001893939 0.003787879 0.001893939 0.001893939 0.001893939
  1 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
   sex
Y            76           80
  0 0.000000000 0.000000000
  1 0.002688172 0.002688172
```

```
nbl    List of  5
test   146 obs. of…
train  900 obs. of…
```

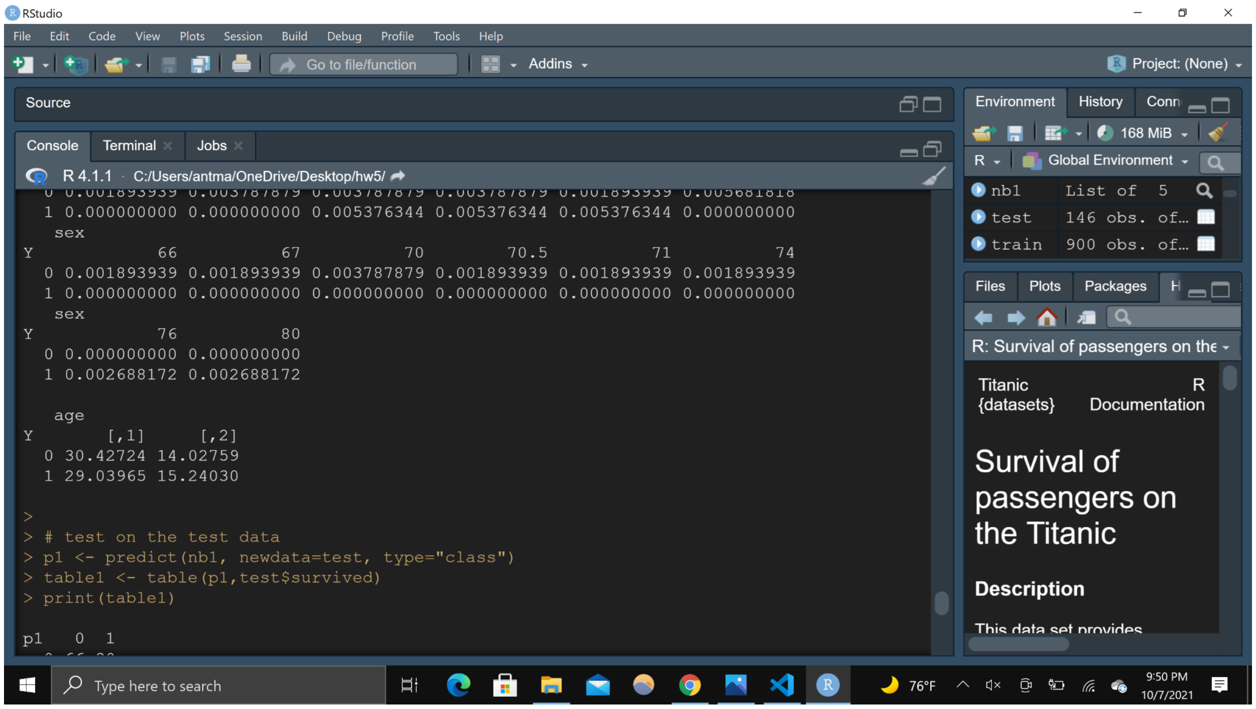
R: Survival of passengers on the Titanic

Titanic {datasets} R Documentation

Survival of passengers on the Titanic

Description

This data set provides



```
  0 0.001893939 0.003787879 0.003787879 0.003787879 0.001893939 0.005681818
  1 0.000000000 0.000000000 0.005376344 0.005376344 0.005376344 0.000000000
   sex
Y            66           67           70         70.5           71           74
  0 0.001893939 0.001893939 0.003787879 0.001893939 0.001893939 0.001893939
  1 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
   sex
Y            76           80
  0 0.000000000 0.000000000
  1 0.002688172 0.002688172

   age
Y       [,1]     [,2]
  0 30.42724 14.02759
  1 29.03965 15.24030

>
> # test on the test data
> p1 <- predict(nbl, newdata=test, type="class")
> table1 <- table(p1,test$survived)
> print(table1)

p1   0   1
```

```
nbl    List of  5
test   146 obs. of…
train  900 obs. of…
```

R: Survival of passengers on the Titanic

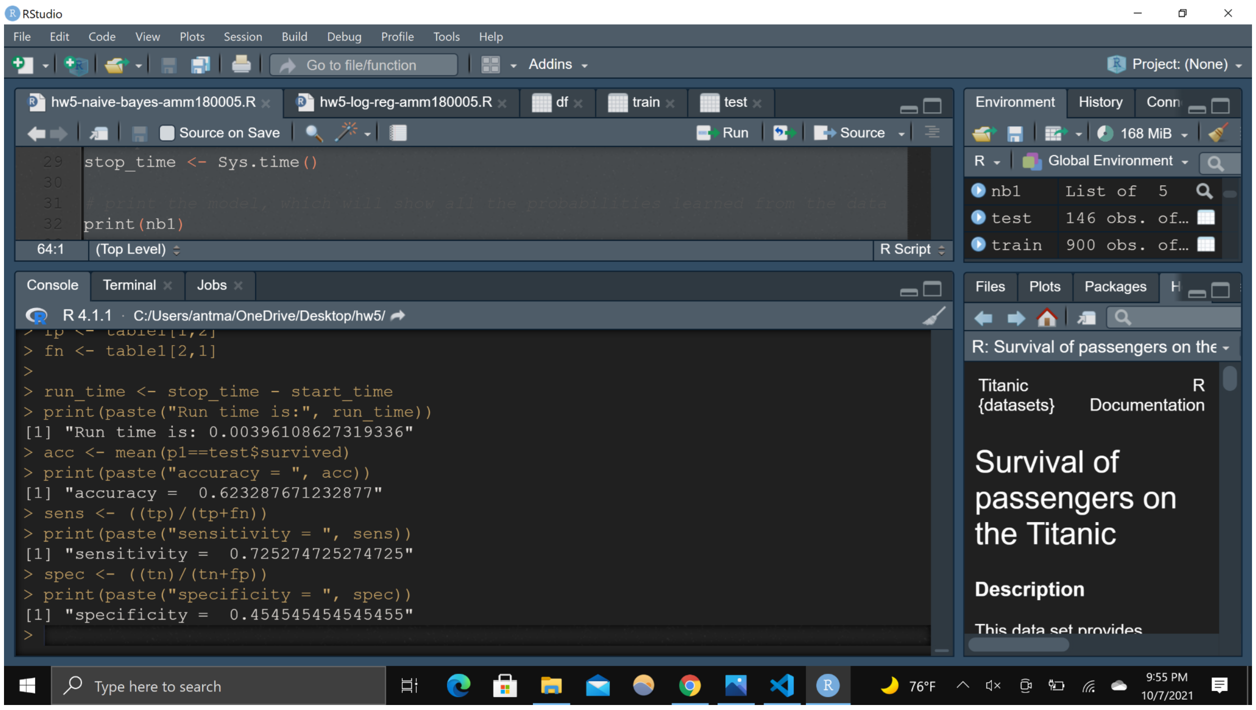
Titanic {datasets} R Documentation

Survival of passengers on the Titanic

Description

This data set provides

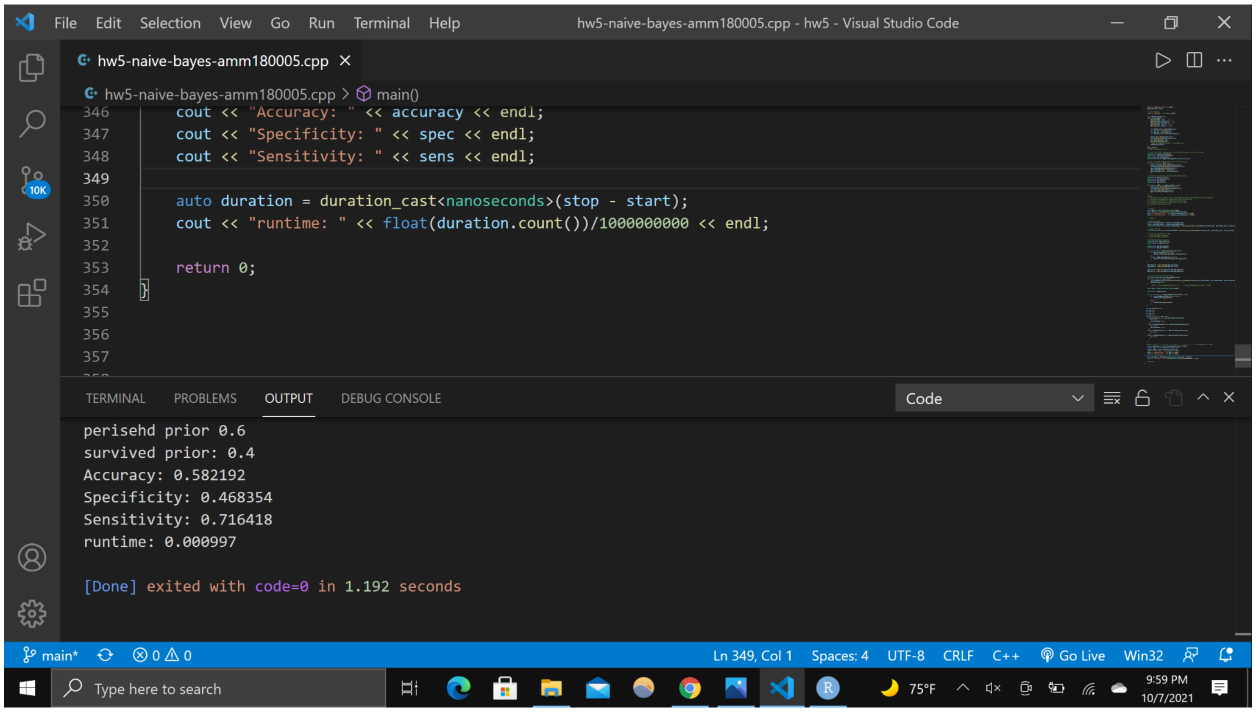
Screenshot of Metrics in R:



```
stop_time <- Sys.time()

# print the model, which will show all the probabilities learned from the data
print(nb1)
```

```
> fp <- table1[1,2]
> fn <- table1[2,1]
>
> run_time <- stop_time - start_time
> print(paste("Run time is:", run_time))
[1] "Run time is: 0.00396108627319336"
> acc <- mean(p1==test$survived)
> print(paste("accuracy = ", acc))
[1] "accuracy =  0.623287671232877"
> sens <- ((tp)/(tp+fn))
> print(paste("sensitivity = ", sens))
[1] "sensitivity =  0.725274725274725"
> spec <- ((tn)/(tn+fp))
> print(paste("specificity = ", spec))
[1] "specificity =  0.454545454545455"
>
```

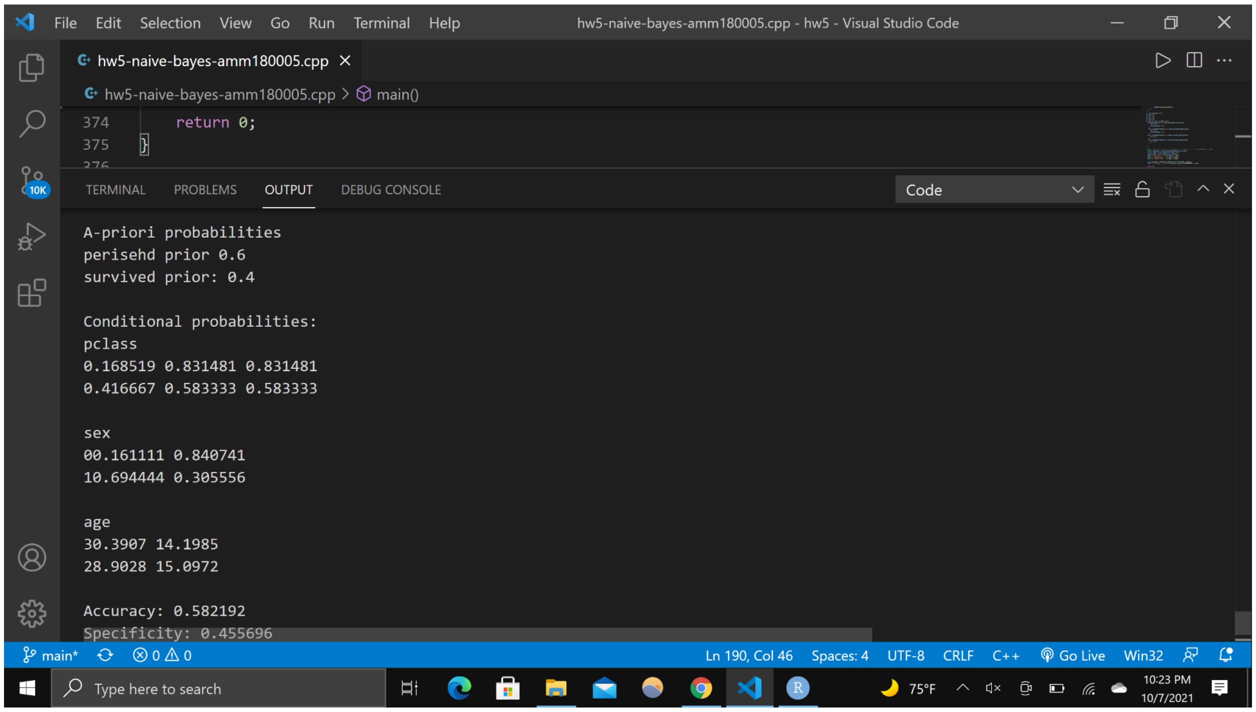
Screenshots of metrics in R:



```
    cout << "Accuracy: " << accuracy << endl;
    cout << "Specificity: " << spec << endl;
    cout << "Sensitivity: " << sens << endl;

    auto duration = duration_cast<nanoseconds>(stop - start);
    cout << "runtime: " << float(duration.count())/1000000000 << endl;

    return 0;
}
```

```
perisehd prior 0.6
survived prior: 0.4
Accuracy: 0.582192
Specificity: 0.468354
Sensitivity: 0.716418
runtime: 0.000997

[Done] exited with code=0 in 1.192 seconds
```

Screenshots of output in C++:


```
A-priori probabilities
perisehd prior 0.6
survived prior: 0.4

Conditional probabilities:
pclass
0.168519 0.831481 0.831481
0.416667 0.583333 0.583333

sex
00.161111 0.840741
10.694444 0.305556

age
30.3907 14.1985
28.9028 15.0972

Accuracy: 0.582192
Specificity: 0.455696
```