David Allen
daa200002@utdallas.edu
CS4375.004

Logistic Regression/Naive Bayes Report:

In this project I wrote implementations of Logistic Regression and Naive Bayes algorithms using library functions from R, and from scratch in C++. Following the pseudocode in the textbook, I was able to recreate both algorithms to produce identical results between my R and C++ programs.

To calculate the execution time for both models in R I used proc.time() before and after creating the model, as suggested in the homework guidelines. For getting time in C++, I used a built-in library for C++11 called chrono, which allowed me to get time elapsed through chrono::steady_clock::now() in the same manner as proc.time().

Logistic Regression:

I performed logistic regression in R using glm(), the general linear model function, setting the family to binomial. The model was trained extremely fast, from 0.00 to 0.01 seconds.

For C++, I used vectors to hold the data, as well as calculations from the data (error, sigmoid, weights, etc.). I represented matrices as vectors, using the matrix's width and height to determine their location. With some basic optimizations to the pseudocode (for example, getting the transpose of the matrix outside of the for loop), my best effort was able to train the model in about 13 seconds over 5,000 iterations, and it produced the exact same results as my R script. Any more iterations would be superfluous, and would take too long to compute. With more and more efficient code, that time might be reduced to compete with R; however, my program did not come close to R's fast runtime. It seems R is a highly-optimized language for machine learning.

Below are screenshots of output from both my R script and C++ code. They depict the coefficients of the model as well as the runtime.

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.29717    0.19678   6.592 4.34e-11 ***
pclass      -0.77993    0.08521  -9.153  < 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: 1211.4  on 899  degrees of freedom
Residual deviance: 1122.1  on 898  degrees of freedom
AIC: 1126.1

Number of Fisher Scoring iterations: 4

> end - start
   user  system elapsed
   0.01    0.00    0.01
>
```

```
LOGISTIC REGRESSION PREDICTION:


Opening file titanic_project.csv.
Reading line 1
heading: "","pclass","survived","sex","age"
new length 1046
Closing file titanic_project.csv now.
Number of records: 1046

Training data (5000 iterations)...
Training time: 13 seconds

Weights: 1.29717 -0.77993
Testing data...
```

Here are the metrics. Both implementations had the same results:

```
>
> # print results
> print(paste("Accuracy: ", acc))
[1] "Accuracy:  0.671232876712329"
> print(paste("Sensitivity: ", sensitivity))
[1] "Sensitivity:  0.720930232558139"
> print(paste("Specificity: ", specificity))
[1] "Specificity:  0.650485436893204"
>
```

```
Results
Accuracy: 0.671233
Sensitivity: 0.72093
Specificity: 0.650485

Program finished running.
```

Naive Bayes:

The Naive Bayes algorithm did a better job on the dataset than my logistic regression did, but that may be mostly due to my naive bayes program using multiple factors as predictors while my logistic regression program used only one.

The R script simply used naiveBayes() from the e1071 library. The C++ program got all the same coefficients, likelihoods, and metrics by following the pseudocode. I created separate vectors for each predictor in the data rather than keep them all in the same object. When both programs ran, R completed runtime in around 0.02 seconds, while the C++ implementation took around 5039 microseconds (0.005039 seconds) to complete. So for Naive Bayes, my C++ code was able to outspeed R by about a hundredth of a second.

Below are screenshots of output from both my R script and C++ code:

```
A-priori probabilities:
Y
  0   1
0.6 0.4

Conditional probabilities:
   pclass
Y          1         2         3
  0 0.1685185 0.2203704 0.6111111
  1 0.4166667 0.2638889 0.3194444

   sex
Y          0         1
  0 0.1592593 0.8407407
  1 0.6944444 0.3055556

   age
Y       [,1]     [,2]
  0 30.41682 14.21185
  1 28.92060 15.09074

> end - start
   user  system elapsed
   0.00    0.02    0.02
>
> # get predictions
> pred <- predict(nb1, newdata=test, type="class")
> table(pred, test$survived)

pred  0  1
   0 69 25
   1 10 42
>
> # get metrics
> acc <- mean(pred==test$survived)
> sensitivity <- sum(pred==1 & test$survived==1)/sum(pred==1)
> specificity <- sum(pred==0 & test$survived==0)/sum(pred==0)
>
> # print results
> print(paste("Accuracy: ", acc))
[1] "Accuracy:  0.76027397260274"
> print(paste("Sensitivity: ", sensitivity))
[1] "Sensitivity:  0.807692307692308"
> print(paste("Specificity: ", specificity))
[1] "Specificity:  0.734042553191489"
```

```
NAIVE BAYES PREDICTION:

Opening file titanic_project.csv.
Reading line 1
heading: "","pclass","survived","sex","age"
new length 1046
Closing file titanic_project.csv now.
Number of records: 1046
Training time: 5039 microseconds

Apriori: 0.6 0.4
Likelihoods for Passenger Class:
0.16852 0.22037 0.61111
0.41667 0.26389 0.31944
Likelihoods for Sex:
0.15926 0.84074
0.69444 0.30556
Age Mean: 30.4168 28.9206
Age Variance: 201.977 227.73

Testing data...

Predictions (first five):
0.365203 0.634797
0.892229 0.107771
0.648328 0.351672
0.891438 0.108562
0.646277 0.353723

Results
Accuracy: 0.760274
Sensitivity: 0.807692
Specificity: 0.734043

Program finished running.
```