CS 4710: Artificial Intelligence
Homework 4: Classification
Allison Moyer
Scott Mallory

Initially, the classifier was created using the logistic regression method. After processing and storing the data from the initialization file (census.names) and the training data (census.train), the classifier computed initial values for the weights using a matrix of all of the values from the training data (X), another matrix of all of the outputs from the training data (Y) and the equation $(X^TX)^{-1}(X^TY)$. From there the classifier modifies its weight values (θ) by $\alpha\frac{\delta}{\delta_\theta}J(\theta)$ where the derivative function is the one we derived in class and, after experimentation, the alpha (learning rate) value was chosen to be 0.1. However, this implementation ran into difficulties when it came to halting the modification function. In theory, the modification function for the theta values should halt when the cost is sufficiently minimized or the changes in the theta values become small, but this classifier did not exhibit these behaviors. Instead, the classifier ran the modification function for 2000 iterations, which produced somewhat decent results (below).

| **Logistic Regression** | | |
|---|---|---|
| Size of training data | Size of prediction data | Accuracy |
| 1000 | 1500 | 0.74 |
| 1000 | 500 | 0.72 |
| 1500 | 1000 | 0.59 |
| 1500 | 500 | 0.62 |

In an attempt to make the classifier better, the array of theta values was expanded to hold specific values for each discrete feature value, instead of just one theta value per feature. In order to change the logic of the weight from, for example, “is education a good indication of salary” to “is having a PhD a good indication of salary”. The weights for the continuous features remained unchanged, however, the iterations were changed to 1000 after experimentation. Results of this change are shown below.

| **Modified Logistic Regression** | | |
|---|---|---|
| Size of training data | Size of prediction data | Accuracy |
| 1000 | 1500 | 0.74 |
| 1000 | 500 | 0.73 |
| 1500 | 1000 | 0.82 |
| 1500 | 500 | 0.84 |

As shown, this modification significantly improved the accuracy of this classifier. However, the classifier took almost 20 seconds to compute and the outcome was still largely variable due to the facts that the cost function did not converge and the theta values did not become regularized (instead the classifier halts after given iterations). Therefore, the next implementation attempted was the Naive Bayes Classifier. To accomplish this, the probabilities of the discrete feature values and the mean and variance of the continuous features were calculated from the training data, and predictions were made based on the resulting overall probability. The results can be seen below.

| **Naive Bayes** | | |
|---|---|---|
| Size of training data | Size of prediction data | Accuracy |
| 1000 | 1500 | 0.71 |
| 1000 | 500 | 0.71 |
| 1500 | 1000 | 0.73 |
| 1500 | 500 | 0.73 |

To improve the results of this classifier, a learning aspect was added. After computing the probabilities, the classifier then goes through a subset of the training data and if the predicted output doesn't match, alters each probability by 0.001 (decided with experimentation). So the correct output's probability is multiplied by 1.001 and the incorrect output's probability is multiplied by 0.999. The results can be seen below.

| **Naive Bayes With Learning** | | |
|---|---|---|
| Size of training data | Size of prediction data | Accuracy |
| 1000 | 1500 | 0.77 |
| 1000 | 500 | 0.76 |
| 1500 | 1000 | 0.78 |
| 1500 | 500 | 0.78 |

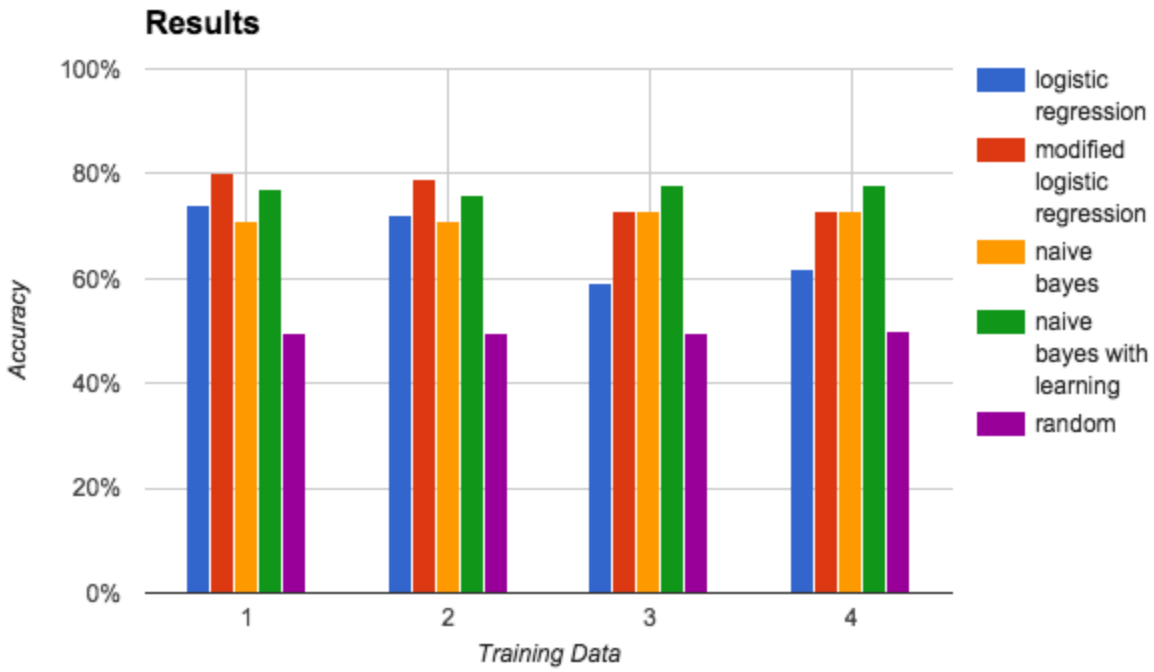
The added learning feature clearly increased the accuracy of the naive bayes classifier.

To compare all of these results, a random classifier was implemented using Math.random() and if the result is < 0.5, the classifier predicts the first output (>50k), and otherwise it predicts the second (<=50k). This classifier ran 100 times for each test and the accuracy was averaged and can be seen on the next page.

| Random Classifier | | |
|---|---|---|
| Size of training data | Size of prediction data | Accuracy |
| 1000 | 1500 | 0.498 |
| 1000 | 500 | 0.496 |
| 1500 | 1000 | 0.496 |
| 1500 | 500 | 0.501 |

As expected, the random classifier's accuracy hovers around 50%.

All of the data is presented together below where the training data number refers to the row number in each table with different training size combinations, respectively.




Although the classifier using modified logistic regression achieved the highest accuracy, the classifier using naive bayes with added learning scores well and is more consistent. Therefore, in conclusion, the naive bayes with learning was the best classifier created and should be used in the competition.