Austin Prince

Data Mining HW1

**Logistic Regression**

For Logistic Regression I used the scikit learn package in order to implement logistic regression

**Results**

*Logistic Regression*

*Predicted: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]*

*Accuracy Score: 0.8*

*Actual Values: [0, 1, 0, 0, 1, 0, 0, 0, 0, 0]*

**kNN**

For kNN I wrote my own code that classified based on the Euclidean distance between a sample and its k nearest neighbors. The pseudocode for this is as follows.

**Results**

*kNN n=1*

*Predicted: [0, 0, 0, 0, 1, 1, 0, 0, 0, 0]*

*Accuracy Score: 0.8*

*Actual Values: [0, 1, 0, 0, 1, 0, 0, 0, 0, 0]*

*kNN n=3*

*Predicted: [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]*

*Accuracy Score: 0.9*

*Actual Values: [0, 1, 0, 0, 1, 0, 0, 0, 0, 0]*

*kNN n=5*

*Predicted: [0, 1, 0, 0, 1, 0, 0, 0, 0, 0]*

*Accuracy Score: 1.0*

*Actual Values: [0, 1, 0, 0, 1, 0, 0, 0, 0, 0]*

**Discussion**

kNN works well as we increase k in this testing example. This is because kNN is a

non parametric model. This means that if the decision boundary is non-linear (which it may be

in) then kNN works well because its predictive accuracy simply is based on your training sample

and not assumptions on the similarity between the test sample and the training sample. In this

example because our dataset is so small there is a higher probability that the distributions of

variables in our test data will not match the distributions of variables in our training data.

Obviously if we were to increase N (sample size) then our accuracy with other models would

improve as well, but in this case it is very small so there is a higher probability that this test

sample does not accurately represent our training sample.

kNN works best when k=5 because this represents for our dataset the optimal number of

neighbors to test on. This is because kNN of 5 is