Assignment 2: Introduction to Data Science and Analytics

Student Name May 2, 2025

# Introduction

This report addresses Assignment 2 for the Introduction to Data Science and Analytics course at Addis Ababa University, Department of Information Science. The assignment involves applying the naïve Bayes and k-nearest neighbors (k-NN) algorithms to the Pima Diabetes dataset from the MASS package in R. The tasks include training naïve Bayes models with different predictor sets, evaluating their performance, and performing 5-fold cross-validation for k-NN with varying *k* values. This report presents the results, including prior probabilities, confusion matrices, performance metrics (accuracy, sensi- tivity, specificity, F-score), and cross-validation errors, along with the R code used for the analysis.

# Naïve Bayes with Glucose and Pedigree Predictors

The first task involved training a naïve Bayes classifier on the Pima.tr dataset to pre- dict the type variable (Yes/No for diabetes) using glu (glucose) and ped (pedigree) as predictors. The model was evaluated on the Pima.te test set.

## Prior Probabilities

The prior probabilities for the classes are:

* + - No (non-diabetic): 0.66
    - Yes (diabetic): 0.34

## Confusion Matrix

The confusion matrix for the predictions is shown in Table 1.

Table 1: Confusion Matrix for Naïve Bayes (glu + ped)

Actual No Actual Yes

|  |  |  |
| --- | --- | --- |
| Predicted No | 203 | 51 |
| Predicted Yes | 20 | 58 |

## Performance Metrics

The performance metrics are:

* + - Accuracy: 0.786
    - Sensitivity: 0.532
    - Specificity: 0.910
    - F-score: 0.620

# Naïve Bayes with All Predictors

The second task trained a naïve Bayes classifier using all available predictors in Pima.tr

to predict type, with evaluation on Pima.te.

## Prior Probabilities

The prior probabilities remain identical due to the same training data:

* + - No: 0.66
    - Yes: 0.34

## Confusion Matrix

The confusion matrix is shown in Table 2.

Table 2: Confusion Matrix for Naïve Bayes (All Predictors)

Actual No Actual Yes

|  |  |  |
| --- | --- | --- |
| Predicted No | 185 | 43 |
| Predicted Yes | 38 | 66 |

## Performance Metrics

The performance metrics are:

* + - Accuracy: 0.756
    - Sensitivity: 0.606
    - Specificity: 0.830
    - F-score: 0.620

## Model Preference Discussion

Model 1 (using glu and ped) is simpler, with fewer predictors, which may reduce over- fitting but risks missing important information. Model 2 (using all predictors) captures more patterns but may overfit due to increased complexity. Comparing performance:

* + - Model 1: Accuracy = 0.786, F-score = 0.620
    - Model 2: Accuracy = 0.756, F-score = 0.620

Model 1 is preferred due to its higher accuracy and comparable F-score, offering a better balance of performance and simplicity.

# k-Nearest Neighbors with Cross-Validation

The third task involved training k-NN models on Pima.tr using glu and ped predictors, with 5-fold cross-validation to evaluate the effect of *k* ∈ {1*,* 3*,* 5*,* 7*,* 9}.

## Cross-Validation Errors

The error rates (1 - accuracy) for each *k* are shown in Table 3.

Table 3: 5-Fold Cross-Validation Errors for k-NN

|  |  |
| --- | --- |
| *k* | Error Rate |
| 1 | 0.415 |
| 3 | 0.315 |
| 5 | 0.275 |
| 7 | 0.270 |
| 9 | 0.265 |

## Optimal *k* and Explanation

The optimal *k* is 9, with the lowest cross-validation error of 0.265. This *k* minimizes the error by balancing bias and variance. Smaller *k* values (e.g., *k* = 1) tend to overfit, capturing noise in the data, while larger *k* values may underfit, oversimplifying the model. *k* = 9 provides the best trade-off for this dataset.

# R Code

The R code used for the analysis is provided below.

# Load required libraries

library(MASS) # For Pima.tr and Pima.te datasets library(e1071) # For naiveBayes

library(class) # For knn

# Load the Pima datasets data(Pima.tr) data(Pima.te)

# Explore the data help(Pima.tr) head(Pima.tr)

summary(Pima.tr) dim(Pima.te)

# Question 1: Naïve Bayes with glu and ped predictors # Train naïve Bayes model

nb\_model1 <- naiveBayes(type ~ glu + ped, data = Pima.tr)

# Report prior probabilities nb\_model1$apriori / sum(nb\_model1$apriori)

# Predict on test data

pred1 <- predict(nb\_model1, Pima.te)

# Compute confusion matrix

conf\_matrix1 <- table(Predicted = pred1, Actual = Pima.te$type)

# Calculate metrics

accuracy1 <- sum(diag(conf\_matrix1)) / sum(conf\_matrix1)

sensitivity1 <- conf\_matrix1[2, 2] / sum(conf\_matrix1[, 2]) # TP / (TP + FN) specificity1 <- conf\_matrix1[1, 1] / sum(conf\_matrix1[, 1]) # TN / (TN + FP) precision1 <- conf\_matrix1[2, 2] / sum(conf\_matrix1[2, ]) # TP / (TP + FP) f\_score1 <- 2 \* (precision1 \* sensitivity1) / (precision1 + sensitivity1)

# Print results cat("Question 1 Results:\n")

cat("Prior Probabilities:\n") print(nb\_model1$apriori / sum(nb\_model1$apriori)) cat("Confusion Matrix:\n")

print(conf\_matrix1) cat("Accuracy:", accuracy1, "\n")

cat("Sensitivity:", sensitivity1, "\n") cat("Specificity:", specificity1, "\n") cat("F-score:", f\_score1, "\n\n")

# Question 2: Naïve Bayes with all predictors # Train naïve Bayes model

nb\_model2 <- naiveBayes(type ~ ., data = Pima.tr)

# Report prior probabilities nb\_model2$apriori / sum(nb\_model2$apriori)

# Predict on test data

pred2 <- predict(nb\_model2, Pima.te)

# Compute confusion matrix

conf\_matrix2 <- table(Predicted = pred2, Actual = Pima.te$type) # Calculate metrics

accuracy2 <- sum(diag(conf\_matrix2)) / sum(conf\_matrix2) sensitivity2 <- conf\_matrix2[2, 2] / sum(conf\_matrix2[, 2]) specificity2 <- conf\_matrix2[1, 1] / sum(conf\_matrix2[, 1]) precision2 <- conf\_matrix2[2, 2] / sum(conf\_matrix2[2, ])

f\_score2 <- 2 \* (precision2 \* sensitivity2) / (precision2 + sensitivity2)

# Print results cat("Question 2 Results:\n")

cat("Prior Probabilities:\n") print(nb\_model2$apriori / sum(nb\_model2$apriori)) cat("Confusion Matrix:\n")

print(conf\_matrix2) cat("Accuracy:", accuracy2, "\n")

cat("Sensitivity:", sensitivity2, "\n") cat("Specificity:", specificity2, "\n") cat("F-score:", f\_score2, "\n\n")

# Model preference discussion cat("Model Preference Discussion:\n")

cat("Model 1 (glu + ped) has fewer predictors, which may reduce overfitting but might cat("Model 2 (all predictors) likely captures more patterns but risks overfitting.\n" cat("Comparing performance metrics:\n")

cat("Model 1 - Accuracy:", accuracy1, "F-score:", f\_score1, "\n") cat("Model 2 - Accuracy:", accuracy2, "F-score:", f\_score2, "\n")

cat("Choose Model 2 if accuracy and F-score are higher; otherwise, prefer Model 1 for

# Question 3: k-NN with 5-fold cross-validation set.seed(123) # For reproducibility

k\_values <- c(1, 3, 5, 7, 9) n <- nrow(Pima.tr)

folds <- sample(rep(1:5, length.out = n)) # Create 5 folds errors <- numeric(length(k\_values))

# Perform 5-fold cross-validation for (i in seq\_along(k\_values)) {

k <- k\_values[i] fold\_errors <- numeric(5) for (fold in 1:5) {

# Split data

train\_idx <- which(folds != fold) test\_idx <- which(folds == fold)

train\_data <- Pima.tr[train\_idx, c("glu", "ped")] test\_data <- Pima.tr[test\_idx, c("glu", "ped")] train\_labels <- Pima.tr$type[train\_idx] test\_labels <- Pima.tr$type[test\_idx]

# Train and predict with k-NN

pred\_knn <- knn(train = train\_data, test = test\_data, cl = train\_labels, k = k)

# Calculate error rate

fold\_errors[fold] <- mean(pred\_knn != test\_labels)

}

errors[i] <- mean(fold\_errors) # Average error across folds

}

# Report results cat("Question 3 Results:\n")

cat("5-Fold Cross-Validation Errors:\n") for (i in seq\_along(k\_values)) {

cat("k =", k\_values[i], "Error =", errors[i], "\n")

}

cat("Optimal k:", k\_values[which.min(errors)], "\n")

cat("Explanation: The optimal k is chosen as it minimizes the cross-validation error, cat("Small k (e.g., k=1) may overfit, while large k may underfit. k = 9 provides the

# Conclusion

The naïve Bayes model with glu and ped predictors achieved higher accuracy (0.786) compared to the model with all predictors (0.756), with similar F-scores (0.620). Thus, the simpler model is preferred for its performance and reduced risk of overfitting. For k-NN, *k* = 9 was optimal, yielding the lowest cross-validation error (0.265), indicating a good balance between bias and variance. These results demonstrate the importance of model selection and parameter tuning in classification tasks.