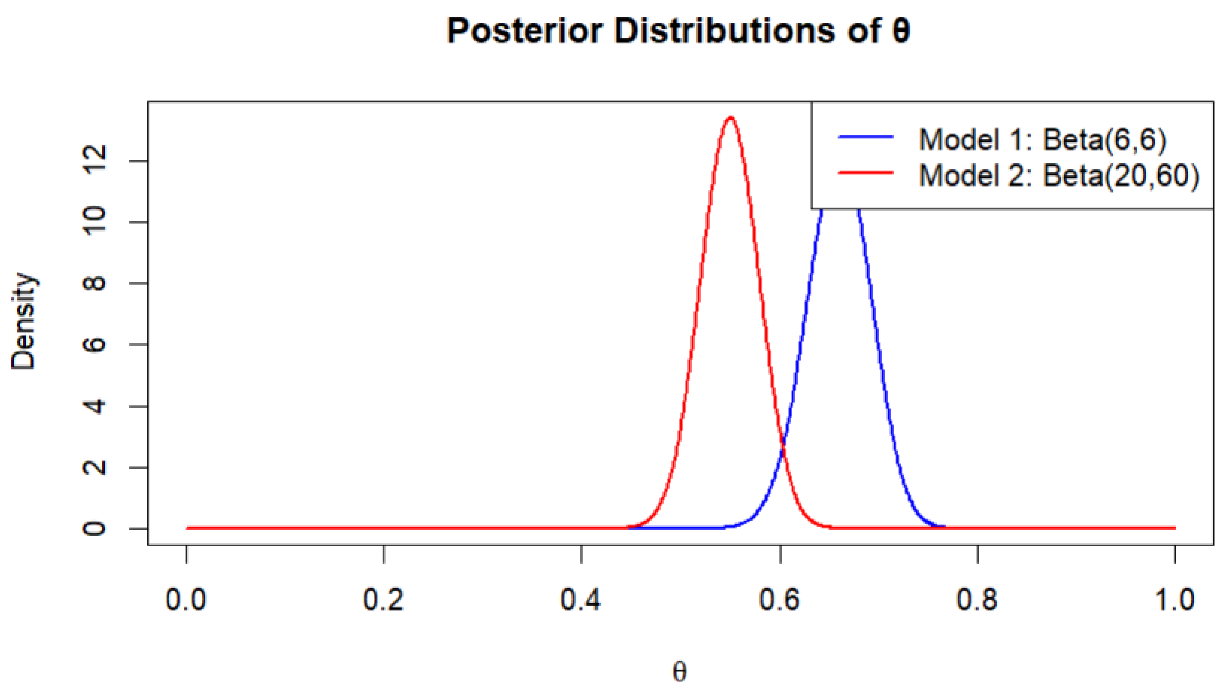
CGS698
Assignment-5

Anupam Chaudhary
(210170)

Part 1: Information-theoretic measures and cross-validation

1.1)

```
> # Given data points
> data <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
> n <- 20  # sample size of the binomial distribution
> # Summarize the data
> total_successes <- sum(data)
> total_trials <- length(data) * n
> # Prior parameters for Model 1 and Model 2
> alpha1 <- 6
> beta1 <- 6
> alpha2 <- 20
> beta2 <- 60
> # Posterior parameters for Model 1
> posterior_alpha1 <- alpha1 + total_successes
> posterior_beta1 <- beta1 + total_trials - total_successes
> # Posterior parameters for Model 2
> posterior_alpha2 <- alpha2 + total_successes
> posterior_beta2 <- beta2 + total_trials - total_successes
> # Create a sequence of theta values for plotting
> theta <- seq(0, 1, length.out = 1000)
> # Calculate the posterior distributions
> posterior1 <- dbeta(theta, posterior_alpha1, posterior_beta1)
> posterior2 <- dbeta(theta, posterior_alpha2, posterior_beta2)
> # Plot the posterior distributions
> plot(theta, posterior1, type = "l", col = "blue", lwd = 2, ylim = c(0,
max(posterior1, posterior2)),
+     xlab = expression(theta), ylab = "Density", main = "Posterior
Distributions of θ")
> lines(theta, posterior2, col = "red", lwd = 2)
> legend("topright", legend = c("Model 1: Beta(6,6)", "Model 2: Beta(20,60)"),
+       col = c("blue", "red"), lwd = 2)
```



1.2)

```
> # Observed data
> y <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
> N_obs <- length(y)
> # Model1: Binom(20,theta), theta ~ beta(6,6)
> # Model2: Binom(20,theta), theta ~ beta(20,60)
> # Compute log pointwise predictive density (lppd) for Model 1
> lppd_m1 <- 0
> for (i in 1:N_obs) {
+  sample_theta <- rbeta(1000, 6 + sum(y), 6 + N_obs * 20 - sum(y))
+  lpd_i <- log(mean(dbinom(y[i], 20, sample_theta)))
+  lppd_m1 <- lppd_m1 + lpd_i
+ }
> # Compute log pointwise predictive density (lppd) for Model 2
> lppd_m2 <- 0
> for (i in 1:N_obs) {
+  sample_theta <- rbeta(1000, 20 + sum(y), 60 + N_obs * 20 - sum(y))
+  lpd_i <- log(mean(dbinom(y[i], 20, sample_theta)))
+  lppd_m2 <- lppd_m2 + lpd_i
+ }
> # Print the lppd for each model

lppd for Model 1: -20.38875
lppd for Model 2: -25.86119
```

1.3)
In-sample deviance = -2*lppd

Model 1: 40.7775
Model 2: 51.72238

In-sample deviance is calculated using the log pointwise predictive density (lppd) to assess the fit of a model to the observed data. It is called "in-sample" because it evaluates the model's performance on the same data that was used to fit the model, rather than on a separate test set (which would be an out-of-sample evaluation).

1.4)
Model is better to fit data,since less deviance means better predictive accuracy.

1.5)

```
> # Given new data points
> new_data <- c(5, 6, 10, 8, 9)
> n <- 20  # sample size of the binomial distribution
> # Function to calculate log predictive density
> log_predictive_density <- function(y, alpha, beta, n) {
+  lchoose(n, y) + lbeta(y + alpha, n - y + beta) - lbeta(alpha, beta)
+ }
> # Calculate LPD for Model 1 and Model 2
> lpd_model1 <- sapply(new_data, log_predictive_density, alpha =
posterior_alpha1, beta = posterior_beta1, n = n)
> lpd_model2 <- sapply(new_data, log_predictive_density, alpha =
posterior_alpha2, beta = posterior_beta2, n = n)
> # Calculate LPPD
> lppd_model1 <- sum(lpd_model1)
> lppd_model2 <- sum(lpd_model2)
> # Calculate out-of-sample deviance
> deviance_model1 <- -2 * lppd_model1
> deviance_model2 <- -2 * lppd_model2
> # Output results

LPPD Model 1: -25.25649
LPPD Model 2: -15.78774

#out of sample deviance
Out-of-sample Deviance Model 1: 50.51298
Out-of-sample Deviance Model 2: 31.57549
```

Model 2 is better than model 1.
Since out of sample deviance for model 2 is lesser than model 1,

1.6)

```
> # Given data points
> data <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
> n <- 20  # sample size of the binomial distribution
> k <- length(data)
> # Function to calculate log predictive density
> log_predictive_density <- function(y, alpha, beta, n) {
+  lchoose(n, y) + lbeta(y + alpha, n - y + beta) - lbeta(alpha, beta)
+ }
> # LOO-CV for Model 1
> lpd_model1_loo <- numeric(k)
> for (i in 1:k) {
+  data_loo <- data[-i]
+  total_successes_loo <- sum(data_loo)
+  total_trials_loo <- length(data_loo) * n
+  posterior_alpha1_loo <- alpha1 + total_successes_loo
+  posterior_beta1_loo <- beta1 + total_trials_loo - total_successes_loo
+  lpd_model1_loo[i] <- log_predictive_density(data[i], posterior_alpha1_loo,
posterior_beta1_loo, n)
+ }
> # LOO-CV for Model 2
> lpd_model2_loo <- numeric(k)
> for (i in 1:k) {
+  data_loo <- data[-i]
+  total_successes_loo <- sum(data_loo)
+  total_trials_loo <- length(data_loo) * n
+  posterior_alpha2_loo <- alpha2 + total_successes_loo
+  posterior_beta2_loo <- beta2 + total_trials_loo - total_successes_loo
+  lpd_model2_loo[i] <- log_predictive_density(data[i], posterior_alpha2_loo,
posterior_beta2_loo, n)
+ }
> # Calculate LOO-CV scores
> lppd_model1_loo <- sum(lpd_model1_loo)
> lppd_model2_loo <- sum(lpd_model2_loo)
> # Calculate LOO-CV deviance
> deviance_model1_loo <- -2 * lppd_model1_loo
> deviance_model2_loo <- -2 * lppd_model2_loo
> # Output results
```

```
LOO-CV LPPD Model 1: -21.11102
LOO-CV LPPD Model 2: -27.22577
```

Part 2: Marginal likelihood and prior sensitivity

2.1)
```
> # Define the given ML_binomial function
> ML_binomial <- function(k, n, a, b) {
+  ML <- (factorial(n) / (factorial(k) * factorial(n - k))) *
+    (factorial(k + a - 1) * factorial(n - k + b - 1) / factorial(n + a + b -
1))
+  ML
+ }
> # Given values
> k <- 2
> n <- 10
> # List of priors
> priors <- list(
+  Beta_0_1_0_4 = c(0.1, 0.4),
+  Beta_1_1 = c(1, 1),
+  Beta_2_6 = c(2, 6),
+  Beta_6_2 = c(6, 2),
+  Beta_20_60 = c(20, 60),
+  Beta_60_20 = c(60, 20)
+ )
> # Calculate marginal likelihood for each prior
> results <- sapply(priors, function(prior) {
+  ML_binomial(k, n, prior[1], prior[2])
+ })
> # Output results
> results

Beta( 0.1 , 0.4 )  4.739564e-01
Beta( 1 , 1 )      9.090909e-02
Beta( 2 , 6 )      4.726891e-03
Beta( 6 , 2)       2.313863e-04
Beta(20,60)        5.079397e-21
Beta(60,20)        1.506630e-23
```

2.2)

```
> # Define the function to estimate marginal likelihood using Monte Carlo
Integration
> estimate_marginal_likelihood <- function(a, b, k, n, num_samples = 10000) {
+  # Create a data frame to store theta samples and their likelihoods
+  df_estimate <- data.frame(matrix(ncol = 2, nrow = num_samples))
+  colnames(df_estimate) <- c("theta_sample", "likelihood")
+
+  # Generate samples and compute likelihoods
+  for (i in 1:num_samples) {
+    theta_i <- rbeta(1, a, b)  # independent sample from the prior
+    likelihood <- dbinom(k, n, theta_i)
+    df_estimate[i,] <- c(theta_i, likelihood)
+  }
+
+  # Calculate the marginal likelihood
+  ML <- mean(df_estimate$likelihood)
+  return(ML)
+ }
> # Given values
> k <- 2
> n <- 10
> # Prior parameters for Beta(0.1, 0.4)
> a <- 0.1
> b <- 0.4
> # Estimate marginal likelihood for Beta(0.1, 0.4)
> ML <- estimate_marginal_likelihood(a, b, k, n)
> # Print the result
> cat("Marginal Likelihood for Beta(0.1, 0.4):", ML, "\n")


Marginal Likelihood for Beta(0.1, 0.4): 0.04046258
```