Assessment_2

2023-03-15

Kernel Methods For Regression

Part 1

Question 1

The Gaussian kernel is an example of an kernel for which the model is identifiable. The model is unidentifiable for the kernel: $k(x, y) = 1$ for $x, y \in \mathbb{R}^p$. This kernel is positive semi-definite and by the Moore-Aronszajn theorem there exists a unique RKHS for which k is the reproducing kernel. In this RKHS all the functions are constant, let's pick two functions $f_1, f_2 \in H_k$ where $f_1(x) = c$, $f_2(x) = d$ for all $x \in \mathcal{X}$. Then if we let the α we use with f_2, $\alpha_2 = \alpha_1 - d + c$, where α_1 is the α we use with f_1, then these are the same model. Hence our model is unidentifiable.

Question 2

We have for some $f \in H_k$: $f = f_1 + f_2$, for some $f_1 \in \tilde{H}, f_2 \in \tilde{H}^{\perp}$. By orthogonality:

$$||f_1 + f_2||^2 = ||f_1||^2 + ||f_2||^2$$

And by the reproducing property:

$$\frac{1}{2n}\sum_{i=1}^{n} \log f(y_i; g^{-1}(\alpha + (f_1 + f_2)(x_i^0)), \phi)$$

$$= \frac{1}{2n}\sum_{i=1}^{n} \log f(y_i; g^{-1}(\alpha + f_1(x_i^0)), \phi)$$

Combining the above,

$$\frac{1}{2n}\sum_{i=1}^{n} \log f(y_i; g^{-1}(\alpha + (f_1 + f_2)(x_i^0)), \phi) - \lambda||f_1 + f_2||^2$$

$$= \frac{1}{2n}\sum_{i=1}^{n} \log f(y_i; g^{-1}(\alpha + f_1(x_i^0)), \phi) - \lambda||f_1||^2 + \lambda||f_2||^2$$

$$\geq \frac{1}{2n}\sum_{i=1}^{n} \log f(y_i; g^{-1}(\alpha + f_1(x_i^0)), \phi) - \lambda||f_1||^2$$

Hence, we have that $\hat{f}_\lambda \in \tilde{H}$ and therefore can write $\hat{f}_\lambda$ as a linear combination of $k(x_1^0, \cdot), ..., k(x_n^0, \cdot)$ therefore we can write:

$$\hat{f}_\lambda = \sum_{i=1}^{n} \hat{\beta}_{\lambda,i} k(x_i^0, \cdot)$$

Question 3

We have that,

$$\begin{aligned}
-\lambda||f||_{H_k}^2 &= -\lambda||\sum_{i=1}^{n}\beta_{\lambda,i}k(x_i^0,\cdot)||_{H_k}^2 \\
&= -\lambda < \sum_{i-1}^{n}\beta_{\lambda,i}k(x_i^0,\cdot), \sum_{j=1}^{n}\beta_{\lambda,j}k(x_j^0,\cdot) >_k \\
&= -\lambda \sum_{i-1}^{n} < \beta_{\lambda,i}k(x_i^0,\cdot), \sum_{j=1}^{n}\beta_{\lambda,j}k(x_j^0,\cdot) >_k \\
&= -\lambda \sum_{i-1}^{n}\sum_{j=1}^{n} < \beta_{\lambda,i}\beta_{\lambda,j}k(x_i^0,\cdot), k(x_j^0,\cdot) >_k \\
&= -\lambda \sum_{i-1}^{n}\sum_{j=1}^{n} \beta_{\lambda,i}\beta_{\lambda,j}k(x_i^0,x_j^0) \\
&= -\lambda \beta_\lambda^T K \beta_\lambda
\end{aligned}$$

Where $K = [k(x_i^0, x_j^0)]_{i,j}$. Hence we can rewrite (2) as:

$$\frac{1}{2n}\sum_{i=1}^{n}\log f(y_i; g^{-1}(\alpha + \beta_\lambda^T \cdot K^{(i)}), \phi) - \lambda\beta_\lambda^T K \beta_\lambda$$

Question 4

Let $m < n + 2$, consider the optimization problem posed in the previous question, the Nyonstrom method involves reducing the dimension of the gram matrix, K, hence reducing the dimensionality of the optimization problem. We will approximate the kernel, k, by $\tilde{k}^{(m)}$ such that the matrix $\tilde{K}^{(m)}$ obtained by replacing k with $\tilde{k}^{(m)}$ has rank $\leq m$. We will let,

$$\tilde{k}^{(m)}(x, x') = k_m(x)^T (K_m)^{-1} k_m(x')$$

where K_m is the first m rows and columns of K and

$$k_m(x) = (k(x_1^0, x), ..., k(x_m^0, x))$$

Let's now rewrite f using our new kernel:

$$\begin{aligned}
f_\lambda(x) &\approx \beta_\lambda^T \tilde{k}^{(m)}(x) \\
&= \beta_\lambda^T K(X_{1:m}, X)^T (K_m^0)^{-1} K(X_{1:m}, x) \\
&= \gamma (K_m^0)^{-1} K(X_{1:m}, x)
\end{aligned}$$

where $K(A, B) = [k(a_i, b_j)]_{i,j}$, X is our data matrix where $X_{1:m}$ means the data matrix including only the first m datapoints and we let $\gamma = \beta_\lambda^T K(X_{1:m}, X)^T$. Let's now rewrite the penalty:

$$\begin{aligned}
-\lambda\beta_\lambda^T K \beta_\lambda &\approx -\lambda\beta_\lambda^T \tilde{K}^{(m)} \beta_\lambda \\
&= -\lambda\beta_\lambda^T K(X_{1:m}, X)^T (K_m^0)^{-1} K(X_{1:m}, X)\beta_\lambda \\
&= -\lambda\gamma (K_m^0)^{-1} \gamma^T
\end{aligned}$$

This leaves us with the following optimization problem,

$$\arg \max_{\alpha\in\mathbb{R}, \phi\in(0,\inf), \gamma\in\mathbb{R}^m} \frac{1}{2n}\sum_{i=1}^{n}\log f(y_i; g^{-1}(\alpha + \gamma(K_m^0)^{-1} K(X_{1:m}, x_i^0)), \phi) - \lambda\gamma(K_m^0)^{-1}\gamma^T$$

Question 5

#TODO: change Km0 to inverse? Let's first find the spectral decomposition of K_m^0,

$$K_m^0 = S\Lambda S^{-1}$$

Then we can write,

$$\gamma^T K_m^0 \gamma = \gamma^T S^{-T} \Lambda S^{-1} \gamma \quad = ||\Lambda^{\frac{1}{2}} S^{-1} \gamma||_2^2$$

Glmnet estimates parameters that minimize the following (if using the ridge penalty):

$$-\frac{1}{n}\sum_{i=1}^{n} \log f(y_i; g^{-1}(\alpha + \gamma X_i), \phi) + \frac{\lambda}{2}||\gamma_{glm}||_2^2$$

which is the same as maximizing optimization problem as ours, except for if we instead try to find the minimum of the negative of our optimization problem, a factor of two and if we substitute for γ_{glm}. We have that,

$$\gamma_{glm} = \Lambda^{\frac{1}{2}} S\gamma$$

$$\Rightarrow \gamma = S^{-1}\Lambda^{-\frac{1}{2}}\gamma_{glm}$$

Therefore we can find our value for γ using the estimate we get from glmnet where for X_i we use $(K_m^0)^{-1}K(X_{1:m}, x_i^0)$.

Part 2

We are going to use the wesdr dataset:

```
library(gss)
data(wesdr)
head(wesdr)
```

```
##    dur  gly  bmi ret
## 1 10.3 13.7 23.8   0
## 2  9.9 13.5 23.5   0
## 3 15.6 13.8 24.8   0
## 4 26.0 13.0 21.6   1
## 5 13.8 11.1 24.6   1
## 6 31.1 11.3 24.6   1
```

Let's now split it into a testing and training set:

```
n.test <- round(0.15 * nrow(wesdr))
test_ind <- sample(seq_len(nrow(wesdr)), size = n.test)

train <- wesdr[-test_ind, ]
test <- wesdr[test_ind, ]
```

Question 6

We are going to use a binomial distribution as we are modelling a response variable that is either 0 or 1, we are going to set $\alpha = 0$ so that we are using the ridge penalty and we will use the radial basis kernel function for which the model is identifiable.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```r
library(kernlab)

gaussian_kernel <- function(x, y, sigma) {
  exp(-sum((x - y)^2) / (2*sigma^2))
}

fit_model <- function(X, y, lambda, sigma, m){
  n <- nrow(X)
  rbf <- rbfdot(sigma = sigma)
  K <- kernelMatrix(rbf, X)
  K_m_inverse <- solve(K[1:m, 1:m])
  K_mn <- matrix(0, m, n)
  X_m <- X[1:m,]
  for (i in 1:m) {
    for (j in 1:n) {
    K_mn[i,j] <- gaussian_kernel(X_m[i,], X[j,], sigma)
    }
  }
  input <- t(K_m_inverse %*% K_mn)
  results <- glmnet(input, y, family = "binomial" , alpha = 0, lambda = lambda)
  gamma_glm <- results$beta

  eig <- eigen(K_m_inverse)
  S <- eig$vectors
  L <- diag(eig$values)

  gamma <- solve(S) %*% sqrt(solve(L)) %*% gamma_glm
  list(gamma = gamma, results= results)
}

X <- as.matrix(train[,-4])
y <- train[,4]
head(fit_model(X, y, 0.1, 1, 50))
```

```
## $gamma
## 50 x 1 Matrix of class "dgeMatrix"
##                s0
##  [1,]  1.43659775
##  [2,]  1.45609816
##  [3,] -0.92103688
##  [4,] -2.83875234
##  [5,]  0.55973131
##  [6,]  0.75289553
##  [7,]  1.53109427
##  [8,]  1.49939757
##  [9,] -0.51240431
## [10,] -0.57453986
## [11,] -0.92069255
## [12,] -0.88165692
## [13,] -0.03653516
## [14,]  0.82783378
## [15,] -0.46607982
## [16,] -0.87322300
## [17,] -1.43433370
```

```
## [18,] -0.82793977
## [19,] -0.58558211
## [20,] -1.12625321
## [21,] -1.41233673
## [22,]  3.19435712
## [23,] -3.32929251
## [24,] -1.19123124
## [25,]  3.23637901
## [26,] -0.46396414
## [27,] -1.42531056
## [28,]  3.00418970
## [29,]  0.33771252
## [30,] -0.66218331
## [31,] -0.96544826
## [32,]  2.17660428
## [33,]  2.25770448
## [34,] -1.93957640
## [35,]  1.28459765
## [36,] -0.70099127
## [37,] -1.57979006
## [38,] -3.20271838
## [39,]  0.06948153
## [40,]  0.85833650
## [41,] -2.15451449
## [42,] -1.71858500
## [43,] -2.90334411
## [44,]  0.07820479
## [45,] -1.03935502
## [46,] -1.15885596
## [47,]  1.22657475
## [48,]  0.88847202
## [49,] -2.25968299
## [50,] -3.03396113
##
## $results
##
## Call:  glmnet(x = input, y = y, family = "binomial", alpha = 0, lambda = lambda)
##
##   Df  %Dev Lambda
## 1 50 13.76    0.1
```

Question 7

Let's fit our model using a grid of parameters to see for which parameters it works and doesnt't and to identify which (if any) parameters lead to errors.

```
# Define the parameter values to search over
lambda_range <- c(0.001, 0.01, 0.1, 0.5, 1.0, 2, 5, 10, 100, 1000, 10000)
sigma_range <- c(0.01, 0.1, 1.0)
m_range <- c(2, 5, 10, 20, 100, 400)

# Generate all combinations of parameters
param_combinations <- expand.grid(lambda_range, sigma_range, m_range)

# Loop through each parameter combination and calculate the score
```

```r
for (i in 1:nrow(param_combinations)) {
  lambda_val <- param_combinations[i, 1]
  sigma_val <- param_combinations[i, 2]
  m_val <- param_combinations[i, 3]
  cat("Current parameters: lambda=", lambda_val, ", sigma=", sigma_val, ", m=", m_val, "\n")
  # Call function to calculate score using the current parameters
  gamma <- fit_model(X, y, lambda_val, sigma_val, m_val)
}
```

```
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 2
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 2
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 2
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 2
## Current parameters: lambda= 1 , sigma= 0.01 , m= 2
## Current parameters: lambda= 2 , sigma= 0.01 , m= 2
## Current parameters: lambda= 5 , sigma= 0.01 , m= 2
## Current parameters: lambda= 10 , sigma= 0.01 , m= 2
## Current parameters: lambda= 100 , sigma= 0.01 , m= 2
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 2
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 2
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 2
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 2
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 2
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 2
## Current parameters: lambda= 1 , sigma= 0.1 , m= 2
## Current parameters: lambda= 2 , sigma= 0.1 , m= 2
## Current parameters: lambda= 5 , sigma= 0.1 , m= 2
## Current parameters: lambda= 10 , sigma= 0.1 , m= 2
## Current parameters: lambda= 100 , sigma= 0.1 , m= 2
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 2
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 2
## Current parameters: lambda= 0.001 , sigma= 1 , m= 2
## Current parameters: lambda= 0.01 , sigma= 1 , m= 2
## Current parameters: lambda= 0.1 , sigma= 1 , m= 2
## Current parameters: lambda= 0.5 , sigma= 1 , m= 2
## Current parameters: lambda= 1 , sigma= 1 , m= 2
## Current parameters: lambda= 2 , sigma= 1 , m= 2
## Current parameters: lambda= 5 , sigma= 1 , m= 2
## Current parameters: lambda= 10 , sigma= 1 , m= 2
## Current parameters: lambda= 100 , sigma= 1 , m= 2
## Current parameters: lambda= 1000 , sigma= 1 , m= 2
## Current parameters: lambda= 10000 , sigma= 1 , m= 2
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 5
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 5
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 5
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 5
## Current parameters: lambda= 1 , sigma= 0.01 , m= 5
## Current parameters: lambda= 2 , sigma= 0.01 , m= 5
## Current parameters: lambda= 5 , sigma= 0.01 , m= 5
## Current parameters: lambda= 10 , sigma= 0.01 , m= 5
## Current parameters: lambda= 100 , sigma= 0.01 , m= 5
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 5
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 5
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 5
```

```
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 5
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 5
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 5
## Current parameters: lambda= 1 , sigma= 0.1 , m= 5
## Current parameters: lambda= 2 , sigma= 0.1 , m= 5
## Current parameters: lambda= 5 , sigma= 0.1 , m= 5
## Current parameters: lambda= 10 , sigma= 0.1 , m= 5
## Current parameters: lambda= 100 , sigma= 0.1 , m= 5
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 5
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 5
## Current parameters: lambda= 0.001 , sigma= 1 , m= 5
## Current parameters: lambda= 0.01 , sigma= 1 , m= 5
## Current parameters: lambda= 0.1 , sigma= 1 , m= 5
## Current parameters: lambda= 0.5 , sigma= 1 , m= 5
## Current parameters: lambda= 1 , sigma= 1 , m= 5
## Current parameters: lambda= 2 , sigma= 1 , m= 5
## Current parameters: lambda= 5 , sigma= 1 , m= 5
## Current parameters: lambda= 10 , sigma= 1 , m= 5
## Current parameters: lambda= 100 , sigma= 1 , m= 5
## Current parameters: lambda= 1000 , sigma= 1 , m= 5
## Current parameters: lambda= 10000 , sigma= 1 , m= 5
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 10
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 10
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 10
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 10
## Current parameters: lambda= 1 , sigma= 0.01 , m= 10
## Current parameters: lambda= 2 , sigma= 0.01 , m= 10
## Current parameters: lambda= 5 , sigma= 0.01 , m= 10
## Current parameters: lambda= 10 , sigma= 0.01 , m= 10
## Current parameters: lambda= 100 , sigma= 0.01 , m= 10
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 10
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 10
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 10
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 10
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 10
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 10
## Current parameters: lambda= 1 , sigma= 0.1 , m= 10
## Current parameters: lambda= 2 , sigma= 0.1 , m= 10
## Current parameters: lambda= 5 , sigma= 0.1 , m= 10
## Current parameters: lambda= 10 , sigma= 0.1 , m= 10
## Current parameters: lambda= 100 , sigma= 0.1 , m= 10
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 10
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 10
## Current parameters: lambda= 0.001 , sigma= 1 , m= 10
## Current parameters: lambda= 0.01 , sigma= 1 , m= 10
## Current parameters: lambda= 0.1 , sigma= 1 , m= 10
## Current parameters: lambda= 0.5 , sigma= 1 , m= 10
## Current parameters: lambda= 1 , sigma= 1 , m= 10
## Current parameters: lambda= 2 , sigma= 1 , m= 10
## Current parameters: lambda= 5 , sigma= 1 , m= 10
## Current parameters: lambda= 10 , sigma= 1 , m= 10
## Current parameters: lambda= 100 , sigma= 1 , m= 10
## Current parameters: lambda= 1000 , sigma= 1 , m= 10
## Current parameters: lambda= 10000 , sigma= 1 , m= 10
```

```
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 20
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 20
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 20
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 20
## Current parameters: lambda= 1 , sigma= 0.01 , m= 20
## Current parameters: lambda= 2 , sigma= 0.01 , m= 20
## Current parameters: lambda= 5 , sigma= 0.01 , m= 20
## Current parameters: lambda= 10 , sigma= 0.01 , m= 20
## Current parameters: lambda= 100 , sigma= 0.01 , m= 20
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 20
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 20
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 20
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 20
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 20
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 20
## Current parameters: lambda= 1 , sigma= 0.1 , m= 20
## Current parameters: lambda= 2 , sigma= 0.1 , m= 20
## Current parameters: lambda= 5 , sigma= 0.1 , m= 20
## Current parameters: lambda= 10 , sigma= 0.1 , m= 20
## Current parameters: lambda= 100 , sigma= 0.1 , m= 20
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 20
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 20
## Current parameters: lambda= 0.001 , sigma= 1 , m= 20
## Current parameters: lambda= 0.01 , sigma= 1 , m= 20
## Current parameters: lambda= 0.1 , sigma= 1 , m= 20
## Current parameters: lambda= 0.5 , sigma= 1 , m= 20
## Current parameters: lambda= 1 , sigma= 1 , m= 20
## Current parameters: lambda= 2 , sigma= 1 , m= 20
## Current parameters: lambda= 5 , sigma= 1 , m= 20
## Current parameters: lambda= 10 , sigma= 1 , m= 20
## Current parameters: lambda= 100 , sigma= 1 , m= 20
## Current parameters: lambda= 1000 , sigma= 1 , m= 20
## Current parameters: lambda= 10000 , sigma= 1 , m= 20
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 100
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 100
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 100
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 100
## Current parameters: lambda= 1 , sigma= 0.01 , m= 100
## Current parameters: lambda= 2 , sigma= 0.01 , m= 100
## Current parameters: lambda= 5 , sigma= 0.01 , m= 100
## Current parameters: lambda= 10 , sigma= 0.01 , m= 100
## Current parameters: lambda= 100 , sigma= 0.01 , m= 100
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 100
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 100
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 100
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 100
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 100
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 100
## Current parameters: lambda= 1 , sigma= 0.1 , m= 100
## Current parameters: lambda= 2 , sigma= 0.1 , m= 100
## Current parameters: lambda= 5 , sigma= 0.1 , m= 100
## Current parameters: lambda= 10 , sigma= 0.1 , m= 100
## Current parameters: lambda= 100 , sigma= 0.1 , m= 100
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 100
```

```
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 100
## Current parameters: lambda= 0.001 , sigma= 1 , m= 100
## Current parameters: lambda= 0.01 , sigma= 1 , m= 100
## Current parameters: lambda= 0.1 , sigma= 1 , m= 100
## Current parameters: lambda= 0.5 , sigma= 1 , m= 100
## Current parameters: lambda= 1 , sigma= 1 , m= 100
## Current parameters: lambda= 2 , sigma= 1 , m= 100
## Current parameters: lambda= 5 , sigma= 1 , m= 100
## Current parameters: lambda= 10 , sigma= 1 , m= 100
## Current parameters: lambda= 100 , sigma= 1 , m= 100
## Current parameters: lambda= 1000 , sigma= 1 , m= 100
## Current parameters: lambda= 10000 , sigma= 1 , m= 100
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 400
```

```
## Error in solve.default(K[1:m, 1:m]): system is computationally singular: reciprocal condition number
```

From the above we see that it works for a large range of values for lambda and sigma, however, when m get's too large we are getting an error. This is happening when we try and invert our matrix K_m, we can fix this by adding some small ϵ to the diagonal of this matrix, let's modify our function to include this:

```
fit_model <- function(X, y, lambda, sigma, m, eps=0.0001){
  n <- nrow(X)
  rbf <- rbfdot(sigma = sigma)
  K <- kernelMatrix(rbf, X)
  K_m_inverse <- solve(K[1:m, 1:m] + diag(m) * eps)
  K_mn <- matrix(0, m, n)
  X_m <- X[1:m,]
  for (i in 1:m) {
    for (j in 1:n) {
    K_mn[i,j] <- gaussian_kernel(X_m[i,], X[j,], sigma)
    }
  }
  input <- t(K_m_inverse %*% K_mn)
  results <- glmnet(input, y, family = "binomial" , alpha = 0, lambda = lambda)
  gamma_glm <- results$beta

  eig <- eigen(K_m_inverse)
  S <- eig$vectors
  L <- diag(eig$values)

  gamma <- solve(S) %*% sqrt(solve(L)) %*% gamma_glm
  list(gamma = gamma, results= results)
}
```

Let's now try our grid search to see if our function now works for large values of m:

```
# Define the parameter values to search over
lambda_range <- c(0.001, 0.01, 0.1, 0.5, 1.0, 2, 5, 10, 100, 1000, 10000)
sigma_range <- c(0.01, 0.1, 1.0)
m_range <- c(2, 5, 10, 20, 100, 400)

# Generate all combinations of parameters
param_combinations <- expand.grid(lambda_range, sigma_range, m_range)

# Loop through each parameter combination and calculate the score
for (i in 1:nrow(param_combinations)) {
```

```
  lambda_val <- param_combinations[i, 1]
  sigma_val <- param_combinations[i, 2]
  m_val <- param_combinations[i, 3]
  cat("Current parameters: lambda=", lambda_val, ", sigma=", sigma_val, ", m=", m_val, "\n")
  # Call function to calculate score using the current parameters
  gamma <- fit_model(X, y, lambda_val, sigma_val, m_val)
}
```

```
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 2
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 2
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 2
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 2
## Current parameters: lambda= 1 , sigma= 0.01 , m= 2
## Current parameters: lambda= 2 , sigma= 0.01 , m= 2
## Current parameters: lambda= 5 , sigma= 0.01 , m= 2
## Current parameters: lambda= 10 , sigma= 0.01 , m= 2
## Current parameters: lambda= 100 , sigma= 0.01 , m= 2
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 2
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 2
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 2
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 2
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 2
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 2
## Current parameters: lambda= 1 , sigma= 0.1 , m= 2
## Current parameters: lambda= 2 , sigma= 0.1 , m= 2
## Current parameters: lambda= 5 , sigma= 0.1 , m= 2
## Current parameters: lambda= 10 , sigma= 0.1 , m= 2
## Current parameters: lambda= 100 , sigma= 0.1 , m= 2
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 2
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 2
## Current parameters: lambda= 0.001 , sigma= 1 , m= 2
## Current parameters: lambda= 0.01 , sigma= 1 , m= 2
## Current parameters: lambda= 0.1 , sigma= 1 , m= 2
## Current parameters: lambda= 0.5 , sigma= 1 , m= 2
## Current parameters: lambda= 1 , sigma= 1 , m= 2
## Current parameters: lambda= 2 , sigma= 1 , m= 2
## Current parameters: lambda= 5 , sigma= 1 , m= 2
## Current parameters: lambda= 10 , sigma= 1 , m= 2
## Current parameters: lambda= 100 , sigma= 1 , m= 2
## Current parameters: lambda= 1000 , sigma= 1 , m= 2
## Current parameters: lambda= 10000 , sigma= 1 , m= 2
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 5
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 5
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 5
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 5
## Current parameters: lambda= 1 , sigma= 0.01 , m= 5
## Current parameters: lambda= 2 , sigma= 0.01 , m= 5
## Current parameters: lambda= 5 , sigma= 0.01 , m= 5
## Current parameters: lambda= 10 , sigma= 0.01 , m= 5
## Current parameters: lambda= 100 , sigma= 0.01 , m= 5
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 5
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 5
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 5
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 5
```

```
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 5
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 5
## Current parameters: lambda= 1 , sigma= 0.1 , m= 5
## Current parameters: lambda= 2 , sigma= 0.1 , m= 5
## Current parameters: lambda= 5 , sigma= 0.1 , m= 5
## Current parameters: lambda= 10 , sigma= 0.1 , m= 5
## Current parameters: lambda= 100 , sigma= 0.1 , m= 5
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 5
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 5
## Current parameters: lambda= 0.001 , sigma= 1 , m= 5
## Current parameters: lambda= 0.01 , sigma= 1 , m= 5
## Current parameters: lambda= 0.1 , sigma= 1 , m= 5
## Current parameters: lambda= 0.5 , sigma= 1 , m= 5
## Current parameters: lambda= 1 , sigma= 1 , m= 5
## Current parameters: lambda= 2 , sigma= 1 , m= 5
## Current parameters: lambda= 5 , sigma= 1 , m= 5
## Current parameters: lambda= 10 , sigma= 1 , m= 5
## Current parameters: lambda= 100 , sigma= 1 , m= 5
## Current parameters: lambda= 1000 , sigma= 1 , m= 5
## Current parameters: lambda= 10000 , sigma= 1 , m= 5
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 10
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 10
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 10
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 10
## Current parameters: lambda= 1 , sigma= 0.01 , m= 10
## Current parameters: lambda= 2 , sigma= 0.01 , m= 10
## Current parameters: lambda= 5 , sigma= 0.01 , m= 10
## Current parameters: lambda= 10 , sigma= 0.01 , m= 10
## Current parameters: lambda= 100 , sigma= 0.01 , m= 10
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 10
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 10
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 10
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 10
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 10
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 10
## Current parameters: lambda= 1 , sigma= 0.1 , m= 10
## Current parameters: lambda= 2 , sigma= 0.1 , m= 10
## Current parameters: lambda= 5 , sigma= 0.1 , m= 10
## Current parameters: lambda= 10 , sigma= 0.1 , m= 10
## Current parameters: lambda= 100 , sigma= 0.1 , m= 10
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 10
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 10
## Current parameters: lambda= 0.001 , sigma= 1 , m= 10
## Current parameters: lambda= 0.01 , sigma= 1 , m= 10
## Current parameters: lambda= 0.1 , sigma= 1 , m= 10
## Current parameters: lambda= 0.5 , sigma= 1 , m= 10
## Current parameters: lambda= 1 , sigma= 1 , m= 10
## Current parameters: lambda= 2 , sigma= 1 , m= 10
## Current parameters: lambda= 5 , sigma= 1 , m= 10
## Current parameters: lambda= 10 , sigma= 1 , m= 10
## Current parameters: lambda= 100 , sigma= 1 , m= 10
## Current parameters: lambda= 1000 , sigma= 1 , m= 10
## Current parameters: lambda= 10000 , sigma= 1 , m= 10
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 20
```

```
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 20
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 20
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 20
## Current parameters: lambda= 1 , sigma= 0.01 , m= 20
## Current parameters: lambda= 2 , sigma= 0.01 , m= 20
## Current parameters: lambda= 5 , sigma= 0.01 , m= 20
## Current parameters: lambda= 10 , sigma= 0.01 , m= 20
## Current parameters: lambda= 100 , sigma= 0.01 , m= 20
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 20
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 20
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 20
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 20
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 20
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 20
## Current parameters: lambda= 1 , sigma= 0.1 , m= 20
## Current parameters: lambda= 2 , sigma= 0.1 , m= 20
## Current parameters: lambda= 5 , sigma= 0.1 , m= 20
## Current parameters: lambda= 10 , sigma= 0.1 , m= 20
## Current parameters: lambda= 100 , sigma= 0.1 , m= 20
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 20
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 20
## Current parameters: lambda= 0.001 , sigma= 1 , m= 20
## Current parameters: lambda= 0.01 , sigma= 1 , m= 20
## Current parameters: lambda= 0.1 , sigma= 1 , m= 20
## Current parameters: lambda= 0.5 , sigma= 1 , m= 20
## Current parameters: lambda= 1 , sigma= 1 , m= 20
## Current parameters: lambda= 2 , sigma= 1 , m= 20
## Current parameters: lambda= 5 , sigma= 1 , m= 20
## Current parameters: lambda= 10 , sigma= 1 , m= 20
## Current parameters: lambda= 100 , sigma= 1 , m= 20
## Current parameters: lambda= 1000 , sigma= 1 , m= 20
## Current parameters: lambda= 10000 , sigma= 1 , m= 20
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 100
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 100
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 100
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 100
## Current parameters: lambda= 1 , sigma= 0.01 , m= 100
## Current parameters: lambda= 2 , sigma= 0.01 , m= 100
## Current parameters: lambda= 5 , sigma= 0.01 , m= 100
## Current parameters: lambda= 10 , sigma= 0.01 , m= 100
## Current parameters: lambda= 100 , sigma= 0.01 , m= 100
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 100
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 100
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 100
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 100
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 100
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 100
## Current parameters: lambda= 1 , sigma= 0.1 , m= 100
## Current parameters: lambda= 2 , sigma= 0.1 , m= 100
## Current parameters: lambda= 5 , sigma= 0.1 , m= 100
## Current parameters: lambda= 10 , sigma= 0.1 , m= 100
## Current parameters: lambda= 100 , sigma= 0.1 , m= 100
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 100
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 100
```

```
## Current parameters: lambda= 0.001 , sigma= 1 , m= 100
## Current parameters: lambda= 0.01 , sigma= 1 , m= 100
## Current parameters: lambda= 0.1 , sigma= 1 , m= 100
## Current parameters: lambda= 0.5 , sigma= 1 , m= 100
## Current parameters: lambda= 1 , sigma= 1 , m= 100
## Current parameters: lambda= 2 , sigma= 1 , m= 100
## Current parameters: lambda= 5 , sigma= 1 , m= 100
## Current parameters: lambda= 10 , sigma= 1 , m= 100
## Current parameters: lambda= 100 , sigma= 1 , m= 100
## Current parameters: lambda= 1000 , sigma= 1 , m= 100
## Current parameters: lambda= 10000 , sigma= 1 , m= 100
## Current parameters: lambda= 0.001 , sigma= 0.01 , m= 400
## Current parameters: lambda= 0.01 , sigma= 0.01 , m= 400
## Current parameters: lambda= 0.1 , sigma= 0.01 , m= 400
## Current parameters: lambda= 0.5 , sigma= 0.01 , m= 400
## Current parameters: lambda= 1 , sigma= 0.01 , m= 400
## Current parameters: lambda= 2 , sigma= 0.01 , m= 400
## Current parameters: lambda= 5 , sigma= 0.01 , m= 400
## Current parameters: lambda= 10 , sigma= 0.01 , m= 400
## Current parameters: lambda= 100 , sigma= 0.01 , m= 400
## Current parameters: lambda= 1000 , sigma= 0.01 , m= 400
## Current parameters: lambda= 10000 , sigma= 0.01 , m= 400
## Current parameters: lambda= 0.001 , sigma= 0.1 , m= 400
## Current parameters: lambda= 0.01 , sigma= 0.1 , m= 400
## Current parameters: lambda= 0.1 , sigma= 0.1 , m= 400
## Current parameters: lambda= 0.5 , sigma= 0.1 , m= 400
## Current parameters: lambda= 1 , sigma= 0.1 , m= 400
## Current parameters: lambda= 2 , sigma= 0.1 , m= 400
## Current parameters: lambda= 5 , sigma= 0.1 , m= 400
## Current parameters: lambda= 10 , sigma= 0.1 , m= 400
## Current parameters: lambda= 100 , sigma= 0.1 , m= 400
## Current parameters: lambda= 1000 , sigma= 0.1 , m= 400
## Current parameters: lambda= 10000 , sigma= 0.1 , m= 400
## Current parameters: lambda= 0.001 , sigma= 1 , m= 400
## Current parameters: lambda= 0.01 , sigma= 1 , m= 400
## Current parameters: lambda= 0.1 , sigma= 1 , m= 400
## Current parameters: lambda= 0.5 , sigma= 1 , m= 400
## Current parameters: lambda= 1 , sigma= 1 , m= 400
## Current parameters: lambda= 2 , sigma= 1 , m= 400
## Current parameters: lambda= 5 , sigma= 1 , m= 400
## Current parameters: lambda= 10 , sigma= 1 , m= 400
## Current parameters: lambda= 100 , sigma= 1 , m= 400
## Current parameters: lambda= 1000 , sigma= 1 , m= 400
## Current parameters: lambda= 10000 , sigma= 1 , m= 400
```

It does! we have solved our problem.

Question 8

This function will compute the approximate solution to (2) for any $c \in C$ and integer $m \leq n + 2$ where lambda is chosen using 10-fold cross validation using the missclassification error, here is the function and it ran on an example so we can check it works:

```
fit_model_cv_l <- function(X, y, sigma, m, eps=0.0001){
  n <- nrow(X)
```

```
  rbf <- rbfdot(sigma = sigma)
  K <- kernelMatrix(rbf, X)
  K_m_inverse <- solve(K[1:m, 1:m] + diag(m) * eps)
  K_mn <- matrix(0, m, n)
  X_m <- X[1:m,]
  for (i in 1:m) {
    for (j in 1:n) {
    K_mn[i,j] <- gaussian_kernel(X_m[i,], X[j,], sigma)
    }
  }
  input <- t(K_m_inverse %*% K_mn)
  cv <- cv.glmnet(input, y, family = "binomial" , alpha = 0)
  lambda <- cv$lambda.min
  fit_model(X, y, lambda, sigma, m)
}

X <- as.matrix(train[,-4])
y <- train[,4]
head(fit_model_cv_l(X, y, 1, 50))
```

```
## $gamma
## 50 x 1 Matrix of class "dgeMatrix"
##                s0
##  [1,]  0.65645607
##  [2,]  0.68160129
##  [3,] -0.50743271
##  [4,] -1.22837711
##  [5,]  0.31072940
##  [6,]  0.49733992
##  [7,]  0.78307911
##  [8,] -0.59573578
##  [9,] -0.19855383
## [10,] -0.22282967
## [11,] -0.44417850
## [12,] -0.43310506
## [13,] -0.01739477
## [14,] -0.61182629
## [15,]  0.17587337
## [16,] -0.46536274
## [17,]  0.74145635
## [18,] -0.45580510
## [19,]  0.32417823
## [20,]  0.63052165
## [21,] -0.74413164
## [22,]  1.57950390
## [23,] -1.76148661
## [24,]  0.67230522
## [25,]  1.87386007
## [26,] -0.25030847
## [27,]  0.66277274
## [28,] -0.92765512
## [29,]  1.19986664
## [30,]  0.10856541
## [31,] -0.33432751
```

```
## [32,] -0.83144776
## [33,]  1.07155423
## [34,] -1.00539546
## [35,]  0.68147054
## [36,] -0.33737044
## [37,] -0.87675913
## [38,] -1.75643034
## [39,]  0.04047828
## [40,]  0.52743931
## [41,]  1.22061893
## [42,] -1.04919397
## [43,] -1.61660969
## [44,]  0.09292736
## [45,]  0.53928117
## [46,] -0.72922809
## [47,]  0.60192452
## [48,]  0.38193206
## [49,] -1.32221001
## [50,] -1.66136450
##
## $results
##
## Call:  glmnet(x = input, y = y, family = "binomial", alpha = 0, lambda = lambda)
##
##   Df %Dev Lambda
## 1 50 9.32 0.3513
```

Question 9

Let's now make it so that sigma is also chosen by cross-validation, here is the function:

```
predict_our_model <- function(fit, X, sigma, m){
  n <- nrow(X)
  rbf <- rbfdot(sigma = sigma)
  K <- kernelMatrix(rbf, X)
  K_m_inverse <- solve(K[1:m, 1:m])
  K_mn <- matrix(0, m, n)
  X_m <- X[1:m,]
  for (i in 1:m) {
    for (j in 1:n) {
    K_mn[i,j] <- gaussian_kernel(X_m[i,], X[j,], sigma)
    }
  }
  input <- t(K_m_inverse %*% K_mn)

  predict(fit$results, newx=input)
}

fit_model_cv_l_sigma <- function(X, y, m, sigma_list=c(0.01, 0.1, 1.0)) {

  # initialize variables to store the best parameter and misclassification error
  best_param <- NULL
  best_error <- Inf
  best_fit <- NULL
```

```
  # loop over the parameter values
  for (i in 1:length(sigma_list)) {

    # set the parameter value
    sigma <- sigma_list[i]

    # carry out cross-validation using the misclassification error
    folds <- cut(seq(1,nrow(X)), breaks=10, labels=FALSE)
    misclass_error <- rep(NA, 10)
    for (j in 1:10) {
      test_idx <- which(folds == j)
      train_idx <- which(folds != j)
      train_data <- X[train_idx, ]
      test_data <- X[test_idx, ]
      train_y <- y[train_idx]
      test_y <- y[test_idx]
      fit <- fit_model_cv_l(train_data, train_y, sigma, m)
      pred <- predict_our_model(fit, test_data, sigma, m)
      misclass_error[j] <- mean(pred != test_y)
    }
    mean_misclass_error <- mean(misclass_error)

    # check if the current parameter gives a lower mean misclassification error than the previous best
    if (mean_misclass_error < best_error) {
      best_param <- sigma_list[i]
      best_error <- mean_misclass_error
      best_fit <- fit
    }
  }

  # return the best parameter value and misclassification error as a list
  best_fit[["sigma"]] = best_param
  best_fit[["MisclassError"]] = best_error
  return(best_fit)
}

X <- as.matrix(train[,-4])
y <- train[,4]
head(fit_model_cv_l_sigma(X, y, 50))
```

```
## $gamma
## 50 x 1 Matrix of class "dgeMatrix"
##                  s0
##  [1,]  1.591237e-07
##  [2,]  2.704329e-06
##  [3,] -2.091028e-06
##  [4,] -1.577493e-06
##  [5,]  8.016742e-07
##  [6,] -7.568510e-06
##  [7,]  7.549181e-06
##  [8,] -8.930431e-06
##  [9,]  1.499799e-06
## [10,] -3.429631e-06
## [11,] -2.371041e-06
```

```
## [12,] -1.784042e-06
## [13,] -5.815656e-06
## [14,] -6.992650e-06
## [15,] -5.463789e-06
## [16,] -1.015994e-05
## [17,] -4.930184e-07
## [18,]  8.026546e-06
## [19,] -2.761578e-05
## [20,] -5.139168e-08
## [21,]  7.584227e-06
## [22,]  3.691202e-05
## [23,] -1.223511e-05
## [24,]  3.255395e-05
## [25,] -1.315822e-05
## [26,]  1.687449e-05
## [27,] -3.461956e-05
## [28,]  2.048075e-06
## [29,] -7.075795e-05
## [30,]  8.161573e-06
## [31,]  3.550068e-05
## [32,] -1.019103e-04
## [33,]  5.603571e-05
## [34,] -4.318271e-05
## [35,] -7.207591e-05
## [36,]  1.035485e-05
## [37,] -1.142273e-05
## [38,]  1.207901e-04
## [39,]  2.618963e-05
## [40,] -1.780179e-04
## [41,] -6.847141e-05
## [42,] -3.575141e-05
## [43,] -6.037151e-05
## [44,]  6.173354e-05
## [45,] -6.926709e-05
## [46,]  1.051831e-04
## [47,] -6.757526e-05
## [48,]  9.467387e-06
## [49,]  3.629619e-05
## [50,] -2.677580e-05
##
## $results
##
## Call:  glmnet(x = input, y = y, family = "binomial", alpha = 0, lambda = lambda)
##
##   Df %Dev Lambda
## 1 50 0.06  44.74
##
## $sigma
## [1] 0.01
##
## $MisclassError
## [1] 1
```

Again we run it on an example to check it works.

Question 10

Let's now fit our model for different values of m and calculate the error on the test set:

```
X_test <- as.matrix(test[,-4])
y_test <- test[,4]

m_list <- c(10,20,40,50)
misclass_error <- rep(NA, length(m_list))
for(i in 1:length(m_list)){
  m <- m_list[i]
  results <- fit_model_cv_l_sigma(X, y, m)
  pred <- predict_our_model(results, X_test, results$sigma, m)
  misclass_error[i] <- mean(pred != y_test)
}
misclass_error
```

```
## [1] 1 1 1 1
```

For some reason the fit of the model I am getting is very bad, but I couldn't find what was wrong before hand in time! #TODO: check misclassification error being computed correctly, also maybe change how error calculated? cross entropy? #TODO: actually, think I just need to round outputs to 0 or 1 ### Question 11 Let's now find the GAM estimate of model (1), where we will choose the penalty parameters, $\{\lambda_j\}_{j=1}^{p}$, using Generalized Cross Validation (GCV) which is the method gam uses by default:

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
fit <- gam(ret~s(dur)+s(gly)+s(bmi), data = train)
preds <- predict(fit, newdata = test[,-4])
preds
```

```
##         660         594         400          15         352         265
##  0.84557203  0.89987895  0.51826874  0.47185790  0.25171539  0.84035581
##         617           7         277          92         130         113
##  0.18735172  0.49790462  0.37773306  0.62088208  0.43452359  0.45967168
##         608         573         443         554         553         288
##  0.52518342  0.63518518  0.62884116  0.39549360  0.47248689  0.53083731
##         506         193         474         620         235         220
##  0.43923342  0.54527300  0.67021078  0.54577900  0.55168150  0.84633996
##         374           2         569         223         527         633
##  0.62218385  0.62208485  0.18858137  0.37036637  0.12876051  0.33817046
##         418          59         485         225         460         110
##  0.56701730  0.24589565  0.37529626  0.38226361  0.22932672  0.68535825
##         162         169         232         175         236         187
##  0.86855899  0.11010091  0.38058503  0.37500173  0.16362101  0.26717942
##         337         470         401         246         119         256
##  0.06450848  0.47597721  0.58911223  0.41495047  0.61214736  0.50954571
##         177         259         204         372         616         322
##  0.42059140  0.57588881  0.44932311  0.36178209  0.43718046  0.68563424
##         466         522         449         545         599         442
##  0.38417351 -0.11001244  0.89716298  0.11707847  0.50335096  0.27260351
##           1         578          43         646         316         341
##  0.64584009  0.30391500  0.01648494  0.60954475  0.41528791  0.06076037
##         483         317         144         181         622          19
```

```
##  0.68588905  0.66876502  0.33671924  0.61183965  0.45338530  0.26146188
##         568          58          48         298         402         406
##  0.17468929  0.54796214 -0.03405138  0.50692253  0.28907921  0.34967889
##         254         191           3         493         508         612
##  0.25054383  0.86600597  0.64506544  0.52389705  0.30997653  0.27300856
##         212         104         127          97         134         419
##  0.01110583  0.13993147  0.26100169  0.78527810  0.94354354  0.21542257
##         199         240         518         136          74         472
##  0.74014219  0.22953552  0.25656138  0.52560448  0.35342443  0.60543104
##         488         392          75          38
##  0.63622267  0.54609077  0.29537715  0.32865677
```

```
test[,4]
```

```
##   [1] 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 0 0 0 0
##  [75] 0 0 1 1 0 0 0 0 0 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 0 0
```

```
misclass_error <- mean(pred != test[,4])
misclass_error
```

```
## [1] 1
```

#TODO: comment on this result after fiddling with error calc.