Mid Term

Statistical Learning

Submitted by

Md Salman Rahman

Graduate Student

University of Texas Rio Grande Valley

Submitted to

Dr. Tamer Oraby

October 31, 2021

Contents

1 Problem 1

Question: Part a

Prove the uniqueness of the predicted value of the lasso regression. That is, show that if $\hat{\beta} \neq \hat{\alpha}$ are two lasso estimates of a multiple linear regression problem with model's matrix X then $\hat{y} = X\hat{\beta} = X\hat{\alpha}$.

Question: Part b

Prove in that case that $||\hat{\beta}||_1 = ||\hat{\alpha}||_1$

Solution

Problem 1 :

part(a) Solution:

we know, lasso function

$$\hat{\beta}^{lasso} = \underset{\beta}{\arg\min} \left[\|Y - X\beta\|_2^2 + \lambda_1 \|\beta\|_1 \right] \quad \text{------(1)}$$

Lasso consist of sum of square & sum of absoulte value ; both are convex in β. So, lasso loss function will also be convex.

Also, we know that set of solutions of convex minimization , is also convex.

Let us considerr, $\hat{\beta}$ and $\hat{\alpha}$ as two lasso estimatorr

Then we can write, also,

$$(1-c)\,\hat{\beta} + c\,\hat{\alpha} \quad \text{for} \quad c \in (0,1)$$

Also, considerr the linearr predictorr $X\hat{\alpha} \neq X\hat{\beta}$. (contradiction)

By definition from eq(1),

$$\|Y - X\hat{\beta}\|_2^2 + \lambda_1 \|\hat{\beta}\|_1 = t$$

and $$\|Y - X\hat{\alpha}\|_2^2 + \lambda_1 \|\hat{\alpha}\|_1 = t$$

} Considerr t as the minimum of lasso loss function.

As we consider $c \in (0,1)$

$$f\left((1-c)\hat{\beta} + c\hat{\alpha}\right)$$

$$= \left\| Y - X\left((1-c)\hat{\beta} + c\hat{\alpha}\right)\right\|_2^2 + \lambda_1 \left\|(1-c)\hat{\beta} + c\hat{\alpha}\right\|_1$$

$$= \|Y - X\hat{\beta} + cX\hat{\beta} - cX\hat{\alpha}\|_2^2 + \lambda_1 \|(1-c)\hat{\beta} + c\hat{\alpha}\|_1$$

$$= \left\|(1-c)\left(Y - X\hat{\beta}\right) + c\left(Y - X\hat{\beta}\right)\right\|_2^2 + \lambda_1 \left\|(1-c)\hat{\beta} + \hat{\alpha}c\right\|_1$$

$$< (1-c)\|Y - X\hat{\beta}\|_2^2 + c\|Y - X\hat{\beta}\|_2^2 + \lambda_1(1-c)\|\hat{\beta}\|_1 + c\lambda_1\|\hat{\alpha}\|_1$$

Applying convexity rule

$$= (1-c)\,t + tc \quad [t \text{ is defined earlier}]$$

$$= t$$

→ This inequality is due to the strict convexity of $f(x) = \|Y - X\beta\|_2^2$ and the convexity of $\|\beta\|_1$.

This means $(1-c)\hat{\beta} + c\hat{\alpha}$ have a lower minimum than t which is a contradiction with our inital assumption ($\hat{\beta}$ and $\hat{\alpha}$ are two lasso estimator).

So, we can say that predicted value of Lasso Regression is unique. [proved]

part(b) problem 1: proving, $\|\hat{\beta}\|_1 = \|\hat{\alpha}\|_1$

From part (a), we find that any two solutions have the same fitted values. Also, they have the same squared error loss.

From (a), mathematically, if $\hat{\beta} \neq \hat{\alpha}$ are two lasso estimates of a multiple linear regression with model's matrix X then

$$\hat{y} = X\hat{\beta} = X\hat{\alpha}$$

So, we can write,

$$\|Y - X\hat{\alpha}\|_2^2 = \|Y - X\hat{\beta}\|_2^2$$

which implies $\|\hat{\beta}\|_1 = \|\hat{\alpha}\|_1$

2 Problem 2

Question

Make a full mathematical description and derivation (as in a proof) of the L_q regularized logistic regression as a Bayesian estimation problem.

Solution

Problem 2:

Solution:

We can define generalized Gaussian distribution $GG_q(\mu, \sigma^2)$ with pdf,

$$f_q(x) = \frac{1}{2\Gamma\left(1+\frac{1}{q}\right)\sqrt{\frac{\Gamma(1/q)}{\Gamma(3/q)}}\sigma} e^{-\left(\frac{\Gamma(3/q)}{\Gamma(1/q)}\right)^{q/2}\left|\frac{x-\mu}{\sigma}\right|^q} \quad \text{for } x \in \mathbb{R} \qquad \text{---------- (2.1)}$$

with mean μ and variance σ^2.

By Bayes' rule we know,

posterior $\propto$ Likelihood $\times$ prior

Here the likelihood will be logistic

$$L(\beta) = \prod_{i=1}^{n} p(x_i)^{y_i} \left(1 - p(x_i)\right)^{1-y_i}$$

here x_i is vector of features

y_i is observed class

p is the probability of the class if $y_i = 1$

$(1-p)$ is the probability of the class if $y_i = 1-p$

Now log likelihood will be turn into sums.

$$l(\beta) = \sum_{i=1}^{n} y_i \log p(x_i) + (1-y_i) \log\left(1-p(x_i)\right)$$

$$= \sum_{i=1}^{n} y_i \log p(x_i) + \log\left(1-p(x_i)\right) - y_i \log\left(1-p(x_i)\right)$$

$$= \sum_{i=1}^{n} y_i \left[\log p(x_i) - \log(1-p(x_i))\right] + \log\left(1-p(x_i)\right)$$

$$= \sum_{i=1}^{n} y_i \log \frac{p(x_i)}{1-p(x_i)} + \sum_{i=1}^{n} \log\left(1-p(x_i)\right)$$

$$= \sum_{i=1}^{n} y_i \left(\beta_0 + x_i \cdot \beta\right) + \sum_{i=1}^{n} -\log\left(1+e^{\beta_0 + x_i \cdot \beta}\right)$$

[Because we know $\log \frac{p(x_i)}{1-p(x_i)} = \beta_0 + x_i \cdot \beta$]

we can write this as,

$$L(\beta) = e^{\sum_{i=1}^{n} y_i \left(\beta_0 + x_i \beta\right) - \sum_{i=1}^{n} \log\left(1+e^{\beta_0 + x_i \beta}\right)}$$

Now, choosing $GIG_q(0, \tau)$ to be a prior for each of coefficient $\beta_1, \ldots, \beta_p$ (with the assumption that they are independent).

If we consider the first part of eq (2.1) it becomes and multipling for p component. So, the total prior,

$$\prod_{j=1}^{p} c\, e^{-\left(\frac{\Gamma(3/q)}{\Gamma(1/q)}\right)^{q/2} \left|\frac{\beta_j}{\tau}\right|^q}$$

$$= c^p e^{-\left(\frac{\Gamma(3/q)}{\tau^2\Gamma(1/q)}\right)^{q/2} \sum_{j=1}^{p} |\beta_j|^q}$$

So, the

$$\text{posterior} \propto e^{\sum_{i=1}^{n} y_i(\beta_0 + x_i\beta) - \sum_{i=1}^{n} \log\left(1 + e^{\beta_0 + x_i\beta}\right)} \cdot e^{-\frac{(\Gamma 3/q)}{\tau^2\Gamma(1/q)} \sum_{j=1}^{p} |\beta_j|^q}$$

$$= e^{\left[\sum_{i=1}^{n} y_i(\beta_0 + x_i\beta) - \left(\log\left(1 + e^{\beta_0 + x_i\beta}\right)\right) - \lambda \sum_{j=1}^{p} |\beta_j|^q\right]}$$

where, $\lambda = \frac{(\Gamma 3/q)}{\tau^2\Gamma(1/q)}$

So the L_q regularized logistic regression as a Bayesian estimation is:

$$\underset{\beta}{\text{argmax}} \left\{ \sum_{i=1}^{n} \left[y_i(\beta_0 + x_i\beta) - \log\left(1 + e^{\beta_0 + x_i\beta}\right) \right] - \lambda \sum_{j=1}^{p} |\beta_j|^q \right\}$$

3 Problem 3

Question

Goal: classify an iris flower based on the inputs: sepal length in cm, sepal width in cm, petal length in cm, and petal width in cm.
Split the data into 80% randomly selected data points to be the training data and the rest 20% to be the testing data. Perform the following list of methods including their cross-validation. Multi-response multiple linear regression.
1) K nearest neighbor 2) Multinomial regression 3) L_1-regularized multinomial regression 4) Elastic-net multinomial regression 5) Linear discriminant analysis 6) Quadratic discriminant analysis 7) Naive Bayes classification with Gaussian conditional distribution of inputs.
Consider also using higher power terms (e.g. x^2) and interaction terms (e.g. x_1x_2). Write down a report of the findings of each method and make a table to compare between the different methods. Use confusion matrices as well as testing errors, AIC, and BIC in the comparisons. A conclusion must be made at the end. Please submit the R codes with the report.

Solution

3.0.1 Data Description

For this problem, I use the Iris data which is a built-in data set in R. Iris data sets consists of four features called: sepal length, sepal width, petal length, and petal width of three species of Iris called setosa, virginica, and versicolor. More details descriptive statistics are given in figure 1. Figure 2 gives a details visualization of three iris species (setosa, versicolor, virginica) with repspect to sepal length and petal length.

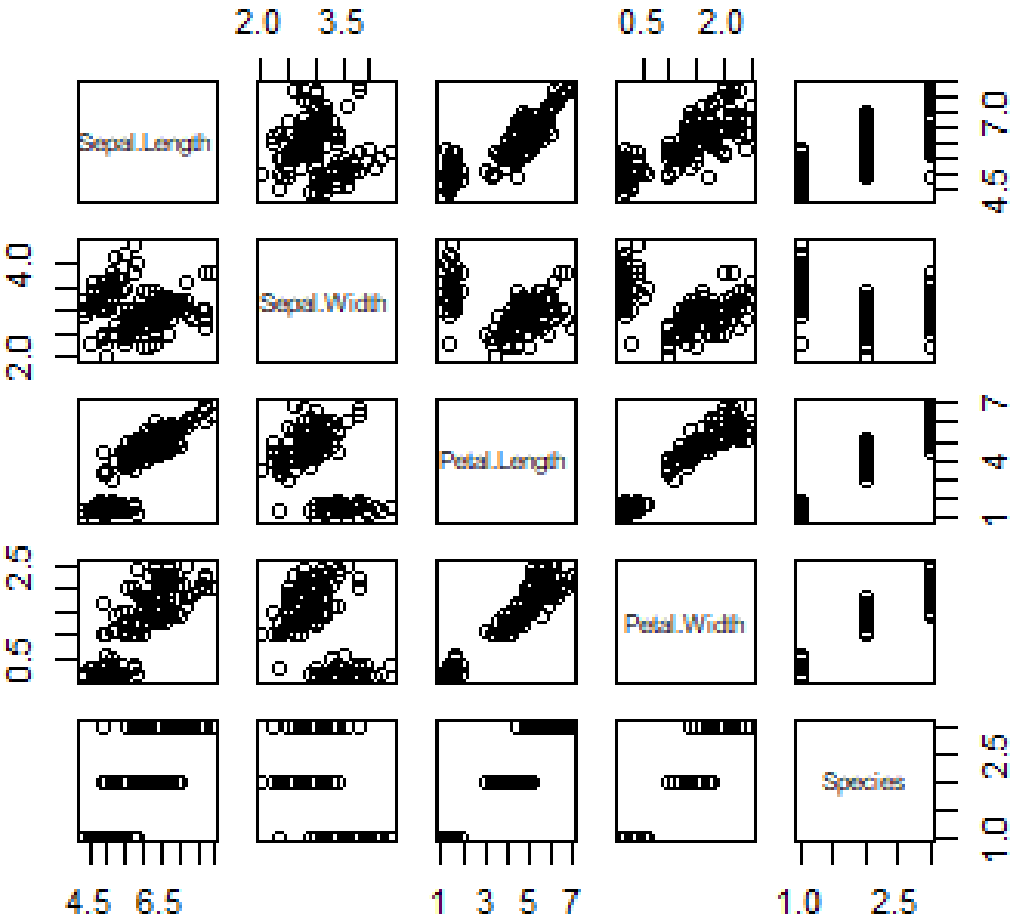


Figure 1: Iris data set contains information about length and width of sepals and petals for three iris species

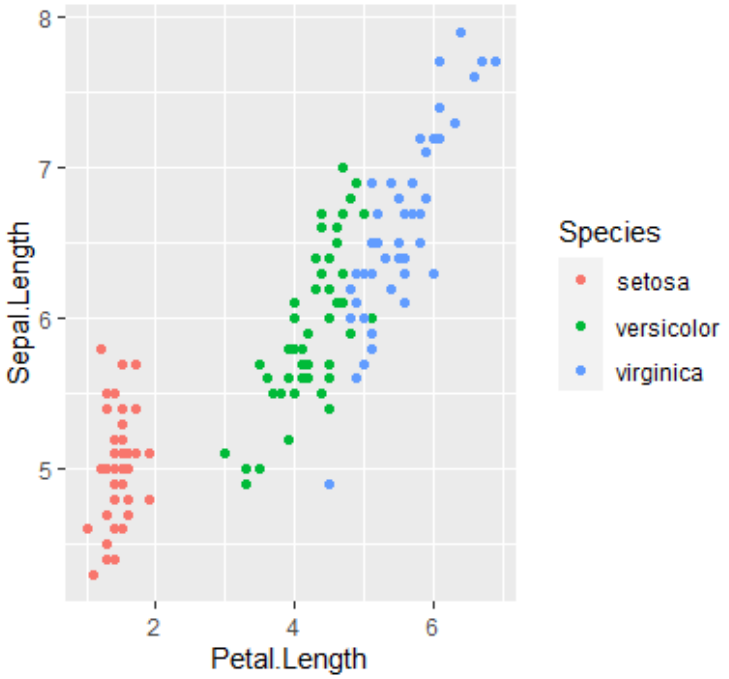


Figure 2: Visualization of Iris Species

3.0.2 Descriptive Analysis

Initially, I make some descriptive analysis of the data-set and explore the relationship between different variables, checking the missing value. I find that sepal length is highly correlated with petal length (0.87) and petal width(0.81). Also, petal length and petal width is around 96.28% correlated which is very high and I will check in the analysis whether it cause any dimensionality problem or not. Figure 3 represents a small summary statistics of the data set.

```
> summary(data)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
>
```

Figure 3: Summary

3.1 Training and Testing Data

I split the iris data sets into 80% training and 20% testing set by using sample() function of the base package and set.seed(123) is considered to make the partition reproducible. **Note:** For all the model, which I describe further, the details about every model are given in R code.

3.1.1 K Nearest Neighbor

K Nearest Neighbor is a simple and easy-to-implement supervised machine learning algorithm that can be used for both regression and classification problems. Here, I use knn function from the library class to build the KNN model. Firstly, I choose the optimal K value for our data set, and the Figure 4 represents the optimal K value graph. And, we find optimal K value = 13. With the optimal K value,

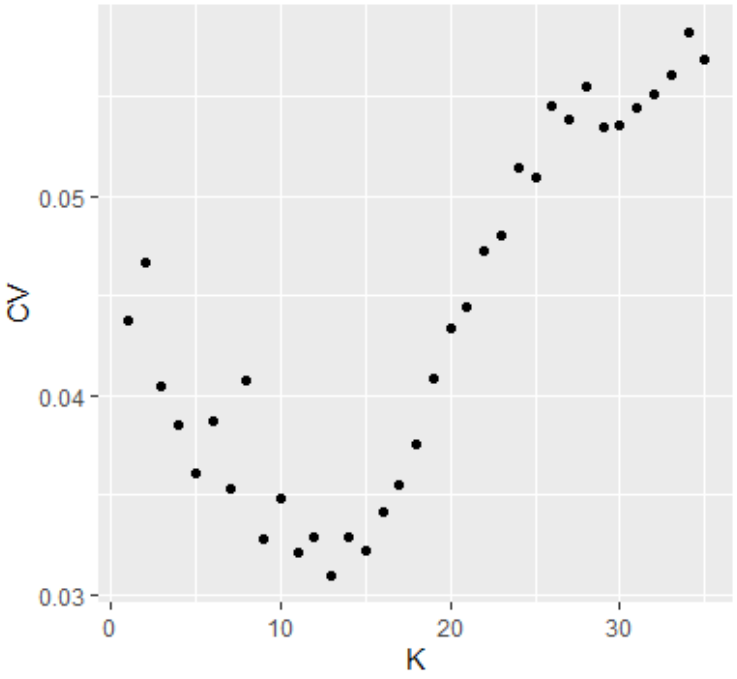
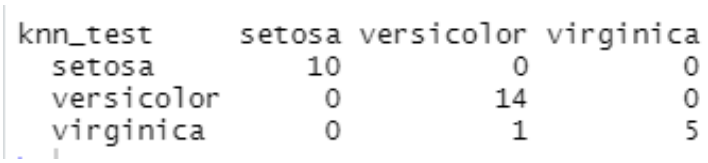


Figure 4: Optimal K Value

I trained the training data set with the KNN algorithm and test them with the

testing data set. The mis-classification error for KNN is 0.033. Also, figure 5 shows the confusion matrix for KNN. We see that there is one misclassification in the confusion matrix and its misclassify versicolor as virginica. In the summary section (3.1.9) show the specificity, sensitivity, negative prediction value, positive prediction value, prevalance, detection rate are given for each class. More details description about KNN results are available in the R code which is attach at the end of the problem.



knn_test	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	14	0
virginica	0	1	5

Figure 5: Confusion Matrix for KNN

3.1.2 Multinomial Regression

Multinomial logistic regression is a classification method that mainly generalizes logistic regression to more than two class discrete outcomes problems. For this problem, I use multinom() function from the nnet package to apply multinomial regression in the iris data set. One of the limitations of nnet is that this function does not include p-value for the regression coefficient, and I use the wald test to p values. But before running the model I specify virginica as our baseline using re level function. After setting the baseline, I trained the training data with a multinomial regression algorithm, and also generate predictions of probabilities for each species using the testing data. The summary of the multinomial regression is given in figure 6 where the AIC value is 20.349 for training data. In this process, I check interactions terms and using the ANOVA chi-square test I find there are no significant interactions with sepal length. Same is true for other interaction terms and they are not significant. Finally, I find the misclassification error for multinomial regression which is 0.033. Figure 7 shows the confusion matrix for

```
> summary(multino_model)
Call:
multinom(formula = Species ~ Sepal.Length + Sepal.Width + Petal.Length +
    Petal.Width, data = train_1)

Coefficients:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa         90.9623    118.75537    46.23126    -168.9402  -167.77440
versicolor    308.5263     40.62844    49.41762    -109.5012   -98.81164

Std. Errors:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa       0.1788943    0.8283587   0.5704804    0.3015864  0.03599308
versicolor 485.2263476  144.8274571  96.4228252  150.9922133 93.58841528

Residual Deviance: 0.3498485
AIC: 20.34985
```

Figure 6: Multinomial Regression Summary

multinomial regression and it is based on testing data. We see that there is one misclassification in the confusion matrix and its misclassify versicolor as virginica. More details description about multinomial regression results are available in the R code which is attach at the end of the problem.

```
pred         setosa versicolor virginica
  virginica       0          1         5
  setosa         10          0         0
  versicolor      0         14         0
> |
```

Figure 7: Confusion Matrix for Multinomial Regression

3.1.3 L_1-Regularized Multinomial Regression

L_1-Regularized Multinomial Regression is a probability-based classification method. Firstly, I find the optimal value of λ and in figure 8, we see the mean and standard deviation for λ. Here, we notice two vertical line, one is lambda.min and another is lambda.1se and the value are 0.00197 and 0.00874. The first value is the ultimate minimum for λ and another is one standard error from the ultimate minimum. If we take the logarithm of this two, we will get the position of vertical line in the graph. Considering, all the fact, I choose ultimate minimum value as the effective lambda value and it also give the smallest error than the other one. After that, I trained the model with training data and we know that lasso is a selection method and its

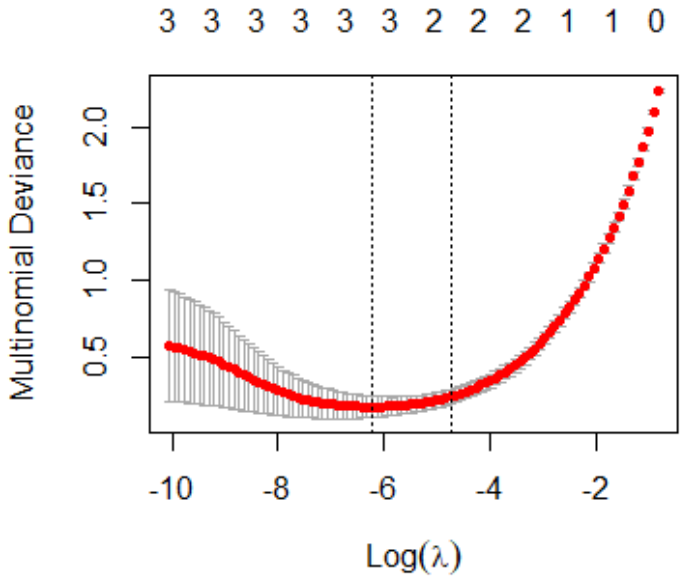


Figure 8: Optimal Value of λ

shrink some parameters towards 0 such as sepal width. Finally, I test the model with testing data and the misclassification error is 0.1. Also, I show the confusion matrix in figure 9 below. We see that there is three misclassification in the confusion matrix and its misclassify versicolor as virginica. More details description about L_1 regularized multinomial regression results are available in the R code which is attach at the end of the problem.

```
l1_test      setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         12         0
  virginica       0          3         5
> |
```

Figure 9: Confusion Matrix for L_1 Regularized Multinomial

3.1.4 Elastic-net Multinomial Regression

Elastic-net selects is a combination of ridge and lasso and its select like a lasso and shrinks like a ridge. I used glmnet to fit the elastic net model. In this model, I experiment with several alpha values between 0 and 1. And, alpha = 0.5 gives the best outcome. After that I calculate the misclassification error for testing data which is around 0.067. Also, Figure 10 shows the confusion matrix for the Elastic -net testing model. We see that there is two misclassification in the confusion matrix and its misclassify versicolor as virginica. More details description about elastic-net multinomial regression results are available in the R code which is attach at the end of the problem.

elas_test	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	13	0
virginica	0	2	5

Figure 10: Confusion Matrix for Elastic Net Multinomial Regression

3.1.5 Linear Discriminant Analysis

Linear discriminant analysis arises in the special situation when we make the assumption that the classes have a common covariance matrix. LDA is commonly used if we have more than two response classes. I will fit an LDA model using lda() function from the MASS library. The LDA output indicates that $\hat{\Pi}_1 = 0.375$, $\hat{\Pi}_2 = 0.333$, $\hat{\Pi}_3 = 0.292$. I use predict() function to make the prediction with the testing dataset and predict functions in LDA returns a list. After that, I calculate the misclassification error for testing data which is around 0.033. Also, Figure 11 shows the confusion matrix for the LDA testing model. We see that there is one misclassification in the confusion matrix and its misclassify versicolor as virginica. More details description about LDA results are available in the R code which is attach at the end of the problem.

lda_predict	setosa	versicolor	virginica
virginica	0	1	5
setosa	10	0	0
versicolor	0	14	0

Figure 11: Confusion Matrix for LDA

3.1.6 Quadratic Discriminant Analysis

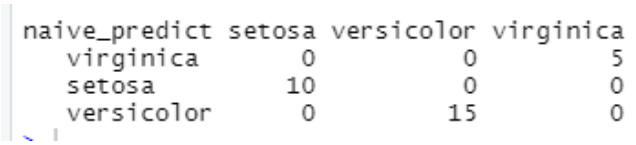
In LDA, we make the assumption that the observations within each class are drawn from a multivariate Gaussian distribution. Similarly, the QDA classifier results from assuming that the observations from each class are drawn from a Gaussian distribution, and we put estimates for the parameters into the Bayes theorem to make predictions. But, QDA assumes that each class has its own covariance matrix. In order to implement QDA I use qda() function from the library MASS. The syntax for QDA is similar to LDA but the QDA classifier involves a quadratic, rather than a linear function. After building the QDA model, I use the predict() function to make the prediction on testing data. And, I find misclassification error is 0.033. I also find the confusion matrix which is given in the figure 12 below. We see that there is one misclassification in the confusion matrix and its misclassify versicolor as virginica.

qda_predict	setosa	versicolor	virginica
virginica	0	1	5
setosa	10	0	0
versicolor	0	14	0

Figure 12: Confusion Matrix for QDA

3.1.7 Naive Bayes Classification with Gaussian Conditional Distribution of Inputs

Naive Bayes is a Bayes-based classification methods. I implement naive Bayes for iris data set in R using the naiveBayes() functions, which is a part of the e1071 library. The syntax for the naive Bayes is similar to lda() and qda(). We will use naive Bayes classifier models each quantitative feature using the Gaussian distribution. The output contains the mean and variance (estimated) for each variable in the class. For example, the mean for Sepal Length is 4.985 for setosa and the standard deviation is 0.3605. The result can also possible to verify taking the mean and sd. After building the model using naiveBayes() function, I use the predict() function to make prediction using testing data. The misclassification error is 0 for naive Bayes. Also, I attach the confusion matrix for naive Bayes classification in figure below. We see that there is no misclassification in the confusion matrix and its misclassify versicolor as virginica.



naive_predict	setosa	versicolor	virginica
virginica	0	0	5
setosa	10	0	0
versicolor	0	15	0

Figure 13: Confusion Matrix for Naive Bayes

3.1.8 Results

Here we develop seven model called: 1) K nearest neighbor 2) Multinomial regression 3) L_1-regularized multinomial regression 4) Elastic-net multinomial regression 5) Linear discriminant analysis 6) Quadratic discriminant analysis 7) Naive Bayes classification with Gaussian conditional distribution of inputs and among all tha naive Bayes gives the smallest misclassification error. And, almost all other method gives the same misclassification error 0.033 except L_1-regularized multinomial regression.

3.1.9 Summary Table

From the summary table and confusion matrix summary we can conclude that all the model perform well in the iris dataset except L_1MR. And there is one common misclassification occur in all the model except naive Bayes that it misclassify versicolor as virginica. K Nearest Neighbor = KNN, Multinomial Regression = MR, L_1

Criteria	KNN	MR	L_1MR	EMR	LDA	QDA	NB
Testing Error	0.033	0.033	0.1	0.033	0.033	0.033	0

Regularized Multinomial Regression = L_1MR, Elastic-net Multinomial Regression = EMR, Linear Discriminant Analysis = LDA, Quadratic Discriminant Analysis = QDA, Naive Bayes Classification with Guassian Conditional Distribution of Inputs = NB

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	0.9333	1.0000
Specificity	1.0000	1.0000	0.9600
Pos Pred Value	1.0000	1.0000	0.8333
Neg Pred Value	1.0000	0.9375	1.0000
Prevalence	0.3333	0.5000	0.1667
Detection Rate	0.3333	0.4667	0.1667
Detection Prevalence	0.3333	0.4667	0.2000
Balanced Accuracy	1.0000	0.9667	0.9800

Figure 14: Confusion Matrix Summary for KNN

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	0.9333	1.0000
Specificity	1.0000	1.0000	0.9600
Pos Pred Value	1.0000	1.0000	0.8333
Neg Pred Value	1.0000	0.9375	1.0000
Prevalence	0.3333	0.5000	0.1667
Detection Rate	0.3333	0.4667	0.1667
Detection Prevalence	0.3333	0.4667	0.2000
Balanced Accuracy	1.0000	0.9667	0.9800

Figure 15: Confusion Matrix Summary for Multinomial

3.1.10 Conclusions

Figure 14 to 20, depicts the summary of confusion matrix for all the model where we calculate all the important metric necessary to evaluate classification model. For

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	0.8000	1.0000
Specificity	1.0000	1.0000	0.8800
Pos Pred Value	1.0000	1.0000	0.6250
Neg Pred Value	1.0000	0.8333	1.0000
Prevalence	0.3333	0.5000	0.1667
Detection Rate	0.3333	0.4000	0.1667
Detection Prevalence	0.3333	0.4000	0.2667
Balanced Accuracy	1.0000	0.9000	0.9400

Figure 16: Confusion Matrix Summary for L_1MR

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	0.8667	1.0000
Specificity	1.0000	1.0000	0.9200
Pos Pred Value	1.0000	1.0000	0.7143
Neg Pred Value	1.0000	0.8824	1.0000
Prevalence	0.3333	0.5000	0.1667
Detection Rate	0.3333	0.4333	0.1667
Detection Prevalence	0.3333	0.4333	0.2333
Balanced Accuracy	1.0000	0.9333	0.9600

Figure 17: Confusion Matrix Summary for Elastic Net

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	0.9333	1.0000
Specificity	1.0000	1.0000	0.9600
Pos Pred Value	1.0000	1.0000	0.8333
Neg Pred Value	1.0000	0.9375	1.0000
Prevalence	0.3333	0.5000	0.1667
Detection Rate	0.3333	0.4667	0.1667
Detection Prevalence	0.3333	0.4667	0.2000
Balanced Accuracy	1.0000	0.9667	0.9800

Figure 18: Confusion Matrix Summary for LDA

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	0.9333	1.0000
Specificity	1.0000	1.0000	0.9600
Pos Pred Value	1.0000	1.0000	0.8333
Neg Pred Value	1.0000	0.9375	1.0000
Prevalence	0.3333	0.5000	0.1667
Detection Rate	0.3333	0.4667	0.1667
Detection Prevalence	0.3333	0.4667	0.2000
Balanced Accuracy	1.0000	0.9667	0.9800

Figure 19: Confusion Matrix Summary for QDA

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	1.0	1.0000
Specificity	1.0000	1.0	1.0000
Pos Pred Value	1.0000	1.0	1.0000
Neg Pred Value	1.0000	1.0	1.0000
Prevalence	0.3333	0.5	0.1667
Detection Rate	0.3333	0.5	0.1667
Detection Prevalence	0.3333	0.5	0.1667
Balanced Accuracy	1.0000	1.0	1.0000

Figure 20: Confusion Matrix Summary for Naive Bayes

example, we calculate specificity which is a metric that evaluates a model's ability to predict true negative of each available category whereas sensitivity evaluate a models ability to predict true positive of each available category. From the Figure 14 to 19, we see that there sensitivity and specificity of versicolor and virginica are slightly differ from 1 as we discussed earlier all the model except naive Bayes misclassify versicolor as virginica.

All in all, after evaluating all the important metric, and misclassification error, and checking the deviation from the assumption, I can conclude that all the model perform wells but naive Bayes classifier are best among all other method.

3.1.11 R Code

```
"
Author: Md Salman Rahman
Course: MATH 6333 Statistical Learning
Course Instructor: Dr. Tamer Oraby
Mid Term
Problem 3
"

# Solution

###################### Classification ################################

## reading the data set

library("readxl")
```

```
library(ggplot2)

data <- iris  # Iris data is built in R


# summary

summary(data)
dim(data)
pairs(data)
str(data)
cor(data[,-5])

ggplot(data,aes(x=Petal.Length,y=Sepal.Length))+
  geom_point(aes(color=Species))

# training and testing data
set.seed(123)
sample_size <- floor(0.80 * nrow(data))
train_ind <- sample(seq_len(nrow(data)), size = sample_size)
training <- data[train_ind, ]
testing <- data[-train_ind, ]
train <- as.data.frame(training)
test <- as.data.frame(testing)

attach(data)

# K nearest neighbor
```

```
library(class)
Choose_K<-function(K,M=1000){
  MSE<-0
  for(i in 1:M){
    train_knn<-sample(150,150-30) # for 5-fold CV
    KNN<-knn(data[train_knn,1:4], data[-train_knn,1:4], data[train_knn,5],
             k=K,prob=TRUE,use.all=FALSE)
    MSE<-MSE+mean(KNN != data[-train_knn,5])
  }
  CVK<-MSE/M
  return(CVK)
}


library("purrr")

LK<-1:35
vCVK<-LK %>%
  map(function(K) Choose_K(K))

vCVK<-unlist(vCVK)
vCVK<-as.data.frame(vCVK)
colnames(vCVK)<-c("CVK")

ggplot()+
  geom_point(data=vCVK,aes(x=LK,y=CVK))+
  xlab("K")+
```

```
  ylab("CV")

LK[which.min(vCVK$CVK)]

library(class)
KNN <- knn(training[,1:4],training[,1:4], training[,5],
           k=13, prob=TRUE,use.all=TRUE)
           # with appropriate k = 13


# prediction using test data

library(class)
knn_test <- knn(training[,1:4],testing[,1:4],training[,5],
                k=13, prob=TRUE,use.all = TRUE)

# error

error <- mean(knn_test != test[,5])
error

#confusion matrix

a <- table(knn_test,test[,5])

library(caret)

expected <- factor(test[,5])
```

```
predicted <- factor(knn_test)
results <- confusionMatrix(data=predicted, reference=expected)
print(results)
```

```
# Multinomial regression

# firstly making virginica as baseline

library(dplyr)
train <- train %>% mutate(Species=relevel(Species,ref="virginica"))

library(nnet)
multino_model <- multinom(Species ~
Sepal.Length + Sepal.Width +
Petal.Length + Petal.Width, data=train)

summary(multino_model)

#coefficient
coe <- summary(multino_model)
$coefficients/summary(multino_model)$standard.errors
```

```
p = (1 - pnorm(abs(coe), 0, 1)) * 2


# checking the interaction

interact_model <- multinom(Species ~
Sepal.Length + Sepal.Width +
                     Petal.Length + Petal.Width + Sepal.Length * Sepal.Width, data=

anova(multino_model,interact_model,test="Chisq")


# prediction and accuracy

library(data.table)
pred <- predict(multino_model, newdata = test, "class")
value = data.table(pred)

error_multi <- mean((pred!=test[,5]))

#confusion matrix
table(pred,test[,5])

library(caret)

expected <- factor(test[,5])
predicted <- factor(pred)
results <- confusionMatrix(data=predicted, reference=expected)
```

```
print(results)
```

```
# L1-regularized multinomial regression

library(glmnet)

x <- model.matrix(Species ~ ., data = data) [,-1]

y <- data$Species

scale.x <- scale(x)

train_glm <- model.matrix(Species ~ ., data = train) [,-1]
train_scale_x <- scale(train_glm)
train_y <-train$Species

#optimal value of lambda
set.seed(123)
l1_fit_cv <- cv.glmnet(x,y,family= "multinomial",type.multinomial="ungrouped")
plot(l1_fit_cv)



#train
```

```
l1_multi_model <- glmnet(train_scale_x, train_y, family = "multinomial", type.multi

coef_l1_multi <- predict(l1_multi_model,s=l1_fit_cv$lambda.min , exact = F, type ="


# test


l1_test <- predict(l1_multi_model, s= l1_fit_cv$lambda.min,
newx=
scale(model.matrix(Species~.,data=test)[,-1]), type = "class")

#misclassification error

mean(l1_test != test[,5])


# confusion matrix

table(l1_test,test[,5])

library(AICcmodavg)

library(caret)

expected <- factor(test[,5])
predicted <- factor(l1_test)
results <- confusionMatrix(data=predicted, reference=expected)
```

```
print(results)



# Elastic-net multinomial regression

#glmnet

scale_x_elas <- scale(train[,1:4])
test_scale_x <- scale(test[,1:4])
q <- as.matrix(train[,5])

elastic_fit <- glmnet(alpha=0.5,scale_x_elas[,1:4],train[,5],
family="multinomial")
elas_cv_fit <- cv.glmnet(alpha =0.5, scale_x_elas[,1:4],q,family="multinomial",
type.multinomial="ungrouped")



# test

elas_test <- predict(elas_cv_fit,
test_scale_x,s="lambda.min",type="class")

error_elas <- mean(elas_test !=test[,5] )


#confusion matrix
```

```
table(elas_test,test[,5])

library(caret)

expected <- factor(test[,5])
predicted <- factor(elas_test)
results <- confusionMatrix(data=predicted, reference=expected)
print(results)




# Linear Discriminant Analysis

library(MASS)

lda_fit <- lda(Species ~ Sepal.Length + Sepal.Width
               + Petal.Length + Petal.Width, data=train)
lda_fit

plot(lda_fit, col = as.integer(train$Species))

library(klaR)

partimat(Species ~ ., data=train, method="lda")
```

```
# prediction

lda_predict <- predict(lda_fit,test)$class
error_lda <- mean(lda_predict != test[,5])

# confusion matrix

table(lda_predict,test[,5])

library(caret)

expected <- factor(test[,5])
predicted <- factor(lda_predict)
results <- confusionMatrix(data=predicted, reference=expected)
print(results)

### train lda(another way)


lda_train <- predict(lda_fit)
train$lda <- lda_train$class
table(train$lda,train$Species)


## test lda(another way)

lda_test <- predict(lda_fit,test)
test$lda <- lda_test$class
```

```
table(test$lda,test$Species)


# Quadratic Discriminant Analysis

qda_fit <- qda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, t
qda_fit

# prediction

qda_predict <- predict(qda_fit,test)$class
error_qda <- mean(qda_predict != test[,5])

# confusion matrix

table(qda_predict,test[,5])

library(caret)

expected <- factor(test[,5])
predicted <- factor(qda_predict)
results <- confusionMatrix(data=predicted, reference=expected)
print(results)

#train (another way)
qda_train <- predict(qda_fit)
train$qda <- qda_train$class
table(train$qda,train$Species)
```

```
# test (another way)
qda_test <- predict(qda_fit,test)
test$qda <- qda_test$class
table(test$qda,test$Species)

# Naive Bayes Classification with Gaussian Conditional Distribution of Inputs
library(e1071)

nb_fit <- naiveBayes(Species ~ . ,  data=train)

nb_fit

# prediction

naive_predict <- predict(nb_fit,test)
error_nb <- mean(naive_predict != test[,5])

# confusion matrix

table(naive_predict,test[,5])
library(caret)

expected <- factor(test[,5])
predicted <- factor(naive_predict)
results <- confusionMatrix(data=predicted, reference=expected)
print(results)
```

4 Problem 4

Question

Consider a classification problem with general K classes. Assume that the data set is standardized within each class. The goal is to use the naive Bayes classification method with beta distribution as the conditional distribution of the inputs given the class. You must use the maximum likelihood estimators of the within class parameters.

1) Find log $log(\frac{p(G=k|X)}{p(G=k|X}$) for $k = 1, 2, \ldots, K-1$.

2) expand the answer of (1) to be in the form of Generalized Additive Model (GAM) and identify the constants a_k, and find the functions $g_{k,i}$,

3) find the discriminant function,

4) find the decision boundary of the classification problem,

5) write down your own function in R of the naive Bayes
classification using the beta distribution for general K. That function shouldn't use any special package for naive Bayes classification.

6) Apply that function to the iris data in problem 2. Is there any improvement in the classification when compared to results of Problem 2 ?

Solution

problem 4:

Solution

(1) For this problem the goal is to use the naive Bayes classification method with beta distribution as the conditional distribution of the inputs given the class.

For naive Bayes classifier:

$$\log\left(\frac{P(G=K|X=x)}{P(G=l|X=x)}\right) = \log\left(\frac{\pi_K}{\pi_l}\right) + \sum_{i=1}^{p} \log\left(\frac{f_{K,i}(x_i)}{f_{l,i}(x_i)}\right) \quad \text{---(4.1)}$$

for any K & l.

Here the prior is $\frac{\pi_K}{\pi_K}$

And, density $f_{K,i}(x_i)$ is Beta $(\alpha_{K,i}, \beta_{K,i})$ for within class standardized data in which cases, the parameters need to be estimated based on the training data, we will use MLE of the within class parameters.

Beta distribution, $P(x|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$

and $B(\alpha,\beta) = \frac{\Gamma\alpha\,\Gamma\beta}{\Gamma(\alpha+\beta)}$

we can write the likelihood ratio as:

$$\frac{f_{K,i}(x_i)}{f_{L,i}(x_i)} = \prod_{i=1}^{p} \frac{\frac{1}{B(\alpha_K,\beta_K)} x_i^{\alpha_K-1}(1-x_i)^{\beta_K-1}}{\frac{1}{B(\alpha_L,\beta_L)} x_i^{\alpha_L-1}(1-x_i)^{\beta_L-1}}$$

$$= \prod_{i=1}^{p} \frac{B(\alpha_L,\beta_L)\, x_i^{\alpha_K-1}(1-x_i)^{\beta_K-1}}{B(\alpha_K,\beta_K)\, x_i^{\alpha_L-1}(1-x_i)^{\beta_L-1}}$$

Now taking the log on both side,

$$\log \frac{f_{K,i}(x_i)}{f_{L,i}(x_i)} = \log \left\{ \prod_{i=1}^{p} \frac{B(\alpha_L,\beta_L)\, x_i^{\alpha_K-1}(1-x_i)^{\beta_K-1}}{B(\alpha_K,\beta_K)\, x_i^{\alpha_L-1}(1-x_i)^{\beta_L-1}} \right\}$$

$$= \log \left\{ \left(\frac{B(\alpha_L,\beta_L)}{B(\alpha_K,\beta_K)} \right)^{p} \prod_{i=1}^{p} \frac{x_i^{\alpha_K-1}(1-x_i)^{\beta_K-1}}{x_i^{\alpha_L-1}(1-x_i)^{\beta_L-1}} \right\}$$

$$= \log \left[\left(\frac{B(\alpha_L,\beta_L)}{B(\alpha_K,\beta_K)} \right)^{p} \right] + \log \left[\prod_{i=1}^{p} \frac{x_i^{\alpha_K-1}(1-x_i)^{\beta_K-1}}{x_i^{\alpha_L-1}(1-x_i)^{\beta_L-1}} \right]$$

$$= \log \left[\left(\frac{B(\alpha_L,\beta_L)}{B(\alpha_K,\beta_K)} \right)^{p} \right] + \log \left[\prod_{i=1}^{p} x_i^{\alpha_K-\alpha_L} (1-x_i)^{\beta_K-\beta_L} \right]$$

$$= \log\left[\left(\frac{B(\alpha_1, \beta_1)}{B(\alpha_K, \beta_K)}\right)^p\right] + \sum_{i=1}^{p} \log\left[\left(x_i^{\alpha_K - \alpha_1}\right)\left(1-x_i\right)^{\beta_K - \beta_1}\right]$$

$$\left[\because \log(ABCD) = \log A + \log B + \log C + \log D\right]$$

$$= \log\left[\left(\frac{B(\alpha_1, \beta_1)}{B(\alpha_K, \beta_K)}\right)^p\right] + \sum_{i=1}^{p} (\alpha_K - \alpha_1) \log x_i + (\beta_K - \beta_1) \log(1-x_i)\Big]$$

$$\log\left(\frac{f_{K,i}(x_i)}{f_{1,i}(x_i)}\right) = \log\left[\left(\frac{B(\alpha_1, \beta_1)}{B(\alpha_K, \beta_K)}\right)^p\right] + \sum_{i=1}^{p} \left[(\alpha_K - \alpha_1) \log x_i + (\beta_K - \beta_1) \log(1-x_i)\right]$$

putting that in equation (4.1), we will get Naive Bayes classifier, which is

$$\log\left(\frac{P(G=K \mid X=x)}{P(G=1 \mid X=x)}\right) = \log\left(\frac{\pi_K}{\pi_1}\right) + \log\left[\left(\frac{B(\alpha_1, \beta_1)}{B(\alpha_K, \beta_K)}\right)^p\right] + \sum_{i=1}^{p} \left[(\alpha_K - \alpha_1) \log x_i + (\beta_K - \beta_1) \log(1-x_i)\right] \quad \text{---------- (4.2)}$$

4-2:

Solution:

For $l = K$, eq(4.1) becomes,

$$\log\left(\frac{p(G=K|X=x)}{p(G=K|X=x)}\right) = \log\left(\frac{\pi_K}{\pi_K}\right)$$

$$+\sum_{i=1}^{p}\log\left(\frac{f_{K,i}(x_i)}{f_{K,i}(x_i)}\right) = a_K + \sum_{i=1}^{p} g_{K,i}(x_i)$$

which is GAM.

In 4-1 we find, Naive Bayes classifier with beta distribution as conditional distribution of the inputs given in equation 4.2. If we put $l = K$ in that equation, it becomes

$$\log\left(\frac{p(G=K|X=x)}{p(G=K|X=x)}\right) = \log\left(\frac{\pi_K}{\pi_K}\right)$$

$$+\log\left[\left(\frac{B(\alpha_K,\beta_K)}{B(\alpha_K,\beta_K)}\right)^p\right] + \sum_{i=1}^{p}\left[(\alpha_K-\alpha_K)\log x_i + (\beta_K-\beta_K)\log(1-x_i)\right]$$

$$= a_K + \sum_{i=1}^{p} g_{K,i}(x_i)$$

where, $a_K = \log\left(\frac{\pi_K}{\pi_K}\right) + \log\left[\left(\frac{B(\alpha_K,\beta_K)}{B(\alpha_K,\beta_K)}\right)^p\right]$

and $g_{K,i}(x_i) = \left[(\alpha_K-\alpha_K)\log x_i + (\beta_K-\beta_K)\log(1-x_i)\right]$

Answer

4-3: Discriminant Function:

The general discriminant function,

$$\delta_k(x) = \sum_{i=1}^{p} \log\left(f_{k,i}^{(x_i)}\right) + \log(\pi_k)$$

For our case the discriminat function will be,

$$\delta_k(x) = \log(\pi_k) + \log\left[\left(\frac{1}{B(\alpha_k, \beta_k)}\right)^p\right] + \sum_{i=1}^{p}\left[\alpha_k \log x_i + \beta_k \log(1-x_i)\right]$$

Answere

(4-4)

Solution:

Decision Boundary:

In general, The decision boundary between class k and l is that set of points for which $\hat{f}_k(x) = \hat{f}_l(x)$, that is the set

$$\left\{ x : (\hat{\beta}_{k0} - \hat{\beta}_{l0}) + (\hat{\beta}_k - \hat{\beta}_l)^T x \right\} = 0$$, an affline set or hyperplane.

So, $\hat{f}_k(x) = \hat{f}_l(x)$

$$\frac{1}{B(\alpha_k, \beta_k)} x_i^{\alpha_k - 1} (1 - x_i)^{\beta_k - 1} = \frac{1}{B(\alpha_l, \beta_l)} x_i^{\alpha_l - 1} (1 - x_i)^{\beta_l - 1}$$

$$\Rightarrow \frac{x_i^{\alpha_k - 1}}{B(\alpha_k, \beta_k)} (1 - x_i)^{\beta_k - 1} = \frac{1}{B(\alpha_l, \beta_l)} x_i^{\alpha_l - 1} (1 - x_i)^{\beta_l - 1}$$

$$\Rightarrow \frac{B(\alpha_l, \beta_l) x_i^{\alpha_k - 1} (1 - x_i)^{(\beta_k - 1)}}{B(\alpha_k, \beta_k) x_i^{\alpha_l - 1} (1 - x_i)^{(\beta_l - 1)}} = 1$$

$$\Rightarrow \frac{B(\alpha_l, \beta_l)}{B(\alpha_k, \beta_k)} x_i^{\alpha_k - \alpha_l} (1 - x_i)^{\beta_k - \beta_l} - 1 = 0$$

This is the decision boundary between class k & l.

Part 5 and Part 6

Firstly, I will write my own R function of the naive Bayes classification using the beta distribution for general K. Then, I apply the function for the classification of iris data. In problem 3, part 7 I did Naive Bayes classification with Gaussian conditional distribution of inputs. I will also compare both result.
I write the R function using five-step.

- **Step 1:** I separate the training and testing data using the R sample function.
- **Step 2:** After that I summarize the iris data sets. In this step I normalize the training data and get positive class index. Also, find the means and standard deviation of the scaled data
- **Step 3:** Then I calculate beta probability density function. I dbeta function of R is used to incorporate beta distribution as a conditional distribution.
- **Step 4:** After that I make the prediciton for testing data set.
- **Step 5:** Finally calculate the misclassification error.

Here, I got a misclassification error of 0.067 using beta as a conditional distribution. Earlier in problem 3 part 7 we get zero misclassification error so that I can say that there is no improvement using this beta as a conditional distribution of the inputs given the class.

R Code

```
data=iris


# converting data into matrix


data=as.matrix()
```

```
pred_train=rep(1,100)
pred_test=rep(1,100)

# index of y
y_index=ncol(data)

# for loop to iterate over 50

for(i in c(1:50))
{

  set.seed(i)
  sample_size <- floor(0.80 * nrow(data))
  train_ind <- sample(seq_len(nrow(data)), size = sample_size)
  training_data <- data[train_ind, ]
  testing_data <- data[-train_ind, ]
  training_x=training_data[,-y_index]
  training_y=training_data[,y_index]

  #mean and standard deviation for the 4 parameter


  mean_tra <- apply(training_x,2, mean)
  sd_tra   <- apply(training_x,2, sd)


  tr_offsets <- t(t(training_x) - mean_tra)
```

```
 tr_scaled_data  <- t(t(tr_offsets) / sd_tra)

 positive_idx = which(training_data[,y_index] == 1)


 p_data= tr_scaled_data[positive_idx,]
n_data = tr_scaled_data[-positive_idx,]


 pos_means=apply(p_data,2,mean)
 pos_sd=apply(p_data,2,sd)

 neg_means=apply(n_data,2,mean)
 neg_sd=apply(n_data,2,sd)

 test_x=testing_data[,1:y_index-1]

 predict_func=function(test_x_row){

   target=0;

   #Using dbeta() function for beta distribution
   p_pos=sum(log(dbeta(test_x_row,pos_means,pos_sd)))+log(length(positive_idx)/len
   p_neg=sum(log(dbeta(test_x_row,neg_means,neg_sd)))+log( 1 - (length(positive_id

   if(p_pos>p_neg){
     target=1
   }
```

```
    else{
      target=0
    }
  }



  pred_test[i]=length(which((y_pred==target)==TRUE))/length(target)
  test_off <- t(t(test_x) - mean_tra)
  tst_scaled_data  <- t(t(test_off) / sd_tra)
  y_pred=apply(tst_scaled_data,1,predict_func)
  target=testing_data[,y_index]
  y_pred_train=apply(tr_scaled_data,1,predict_func)

  pred_train[i]=length(which((y_pred_train==training_y)==TRUE))/length(training_y)

}

# now getting the accuracy for the prediction train


mean(pred_train)*100


# now getting the accuracy for the prediction test
```

```
q <- mean(pred_test)*100
error = 100 - q
```