Assignment 2

Group 2

2023-10-13

Task 1

Task 2

Assuming that our input data is mean-centered, enabling us to ignore the intercept term β_0. The posterior distribution is then described as follows:

$$Pr(\beta|y,X) = \frac{1}{K} PR(y|\beta,X)Pr(\beta)$$

where $K = K(y,X) = \int Pr(y|\beta,X)Pr(\beta)d\beta$. We can see that K does not depend on β. Under the specified assumptions, we can observe that

$$Pr(\beta|y,X) = \frac{1}{Z}\frac{1}{(2\pi)^{\frac{p}{2}}\sigma^2} exp\left(-\frac{(y-X\beta)^T(y-X\beta)}{2\sigma^2}\right)\frac{1}{(2\pi)^2\tau^p} exp\left(-\frac{\beta^T\beta}{2\tau^2}\right)$$

We can conclude that

$$log(Pr(\beta|y,X)) = -W - \frac{(y-X\beta)^T(y-X\beta)}{2\sigma^2} - \frac{\beta^T\beta}{2\tau^2}$$

where D does not depend on β. To maximize this expression we can define $\hat{\beta}$ as

$$\hat{\beta} = \left(X^TX + \frac{\sigma^2}{\tau^2}I\right)^{-1} X^Ty$$

When we set $\lambda = \frac{\sigma^2}{\tau^2}$, we can observe that the method mentioned above is essentially the same as ridge regression.

It is evident that the probability distribution $Pr(\beta|y,X)$ takes on a Gaussian form, with its mean and mode being in perfect alignment. We will proceed to demonstrate that the mean value is equivalent to $\hat{\beta}$. To do so, we should note that equation above for $Pr(\beta|y,X)$ implies that the covariance matrix Σ follows the relationship below

$$\Sigma^{-1} = \frac{1}{\sigma^2}\left(X^TX + \frac{\sigma^2}{\tau^2}I\right)$$

This equation leads to the conclusion that $\hat{\beta} = \frac{1}{\sigma^2}\Sigma X^Ty$. When we equate the relevant elements in the equation above for $Pr(\beta|y,X)$, it becomes apparent that this expression serves as the mean value. ## Task 3

Task 4

Task 5

Task 6

Task 7

Task 8

Task 9

Task 10

```r
suppressMessages(library("ISLR2"))
suppressMessages(library("ggplot2"))
suppressMessages(library("gridExtra"))
suppressMessages(library("dplyr"))

set.seed(7362)

data("Default", package="ISLR2")
df=Default

#First, we need to convert the 'default' and 'student' variables into the
#binary format. "Yes"=1, "No"=0.
df$default = as.integer(ifelse(df$default == "Yes", 1, 0))
df$student = as.integer(ifelse(df$student == "Yes", 1, 0))

#We create an empty dataframe to store the results.
df_results = data.frame(matrix(ncol=3,nrow=100,
                               dimnames=list(NULL,
                               c("val_set_error",
                                 "false_neg", "def_ratio"))))
# a)
glmfit_a = glm(default ~ income + balance, data = df, family =
                 binomial(link = "logit"))
print(summary(glmfit_a))
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial(link = "logit"),
##     data = df)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

```
# b)
#Train-set split (70% train-30% test)
df$id = 1:nrow(df)

train = df %>% dplyr::sample_frac(0.70)
test = dplyr::anti_join(df, train, by = 'id')

# Training the logistic regression model
glmfit_b = glm(default ~ income + balance, data = train, family =
                 binomial(link = "logit"))
print(summary(glmfit_b))
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial(link = "logit"),
##     data = train)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.169e+01  5.342e-01 -21.882  < 2e-16 ***
## income       1.787e-05  6.072e-06   2.944  0.00324 **
## balance      5.767e-03  2.795e-04  20.632  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2030.3  on 6999  degrees of freedom
## Residual deviance: 1070.9  on 6997  degrees of freedom
## AIC: 1076.9
##
## Number of Fisher Scoring iterations: 8
```

```
# Getting the predictions for the test dataset
preds_test = predict(glmfit_b,test, type = "response")

#Since we get the probabilities with the regression model, we used a 0.5
#threshold.
preds = ifelse(preds_test>0.5, 1, 0)

#To compute the validation set error and false positive ratio we use the
#confusion matrix.
temp = table(test$default, preds)
print(temp)
```

```
##    preds
##        0    1
##   0 2887   11
##   1   71   31
```

```
#Misclassification rate, false negative ratio and default ratio
#calculations are stored in the dataframe.
val_set_error = (temp[2]+temp[3])/sum(temp)
false_neg = temp[2]/sum(temp)
def_ratio = sum(test$default)/sum(temp)

paste("Misclassification rate:",val_set_error)
```

```
## [1] "Misclassification rate: 0.0273333333333333"
```

```
paste("False negative rate:",false_neg)
```

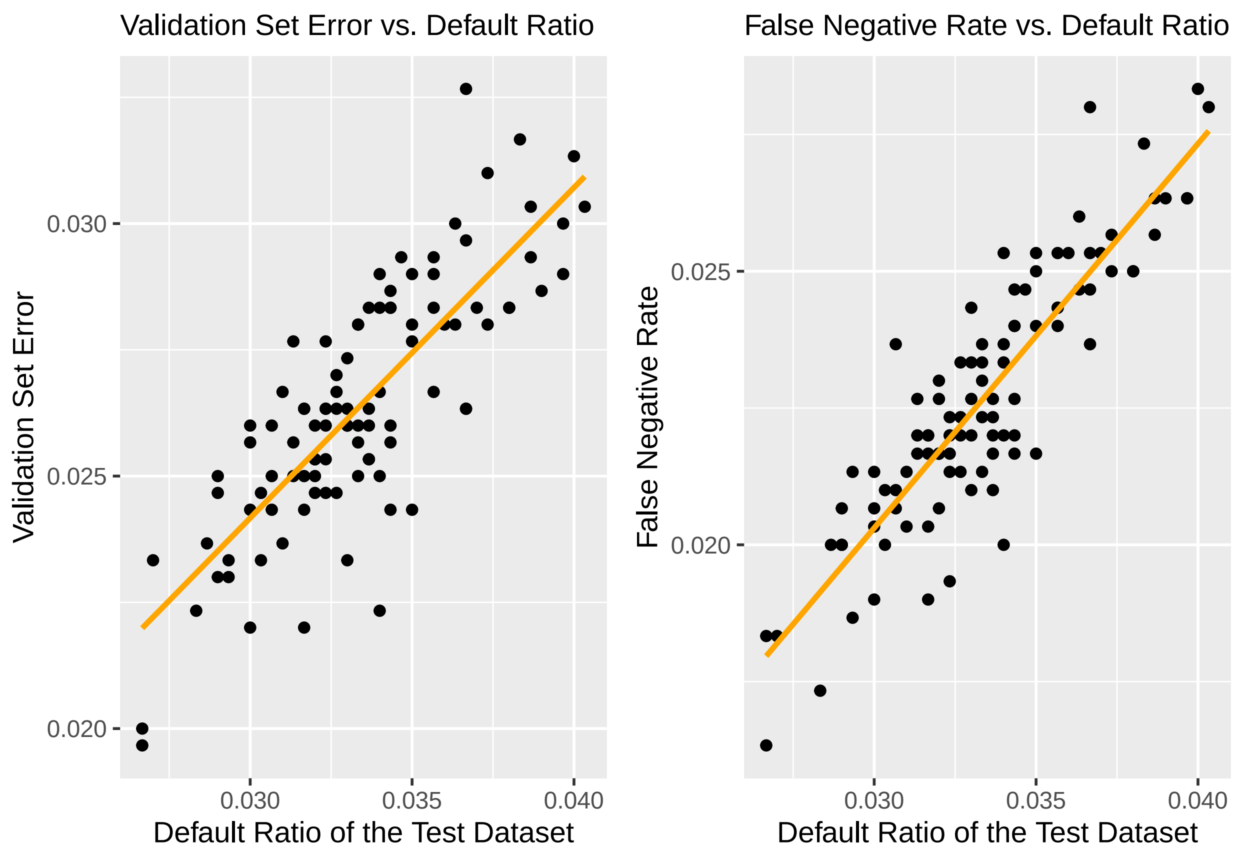
```
## [1] "False negative rate: 0.0236666666666667"
```

```
paste("Default ratio:",def_ratio)
```

```
## [1] "Default ratio: 0.034"
```

```
# c)
#We repeat the process 100 times using 100 different splits.
for(i in 1:100) {
  #Train-set split (70% train-30% test)
  df$id = 1:nrow(df)

  train = df %>% dplyr::sample_frac(0.70)
  test = dplyr::anti_join(df, train, by = 'id')

  # Training the logistic regression model
  glmfit = glm(default ~ income + balance, data = train, family =
                 binomial(link = "logit"))

  # Getting the predictions for the test dataset
  preds_test = predict(glmfit,test, type = "response")

  #Since we get the probabilities with the regression model, we used a 0.5
  #threshold.
  preds = ifelse(preds_test>0.5, 1, 0)

  #To compute the validation set error and false positive ratio we use the
  #confusion matrix.
  temp = table(test$default, preds)

  #Misclassification rate, false negative ratio and default ratio
  #calculations are stored in the dataframe.
  df_results$val_set_error[i] = (temp[2]+temp[3])/sum(temp)
  df_results$false_neg[i] = temp[2]/sum(temp)
```

```r
  df_results$def_ratio[i] = sum(test$default)/sum(temp)
}

# Creating the plots
val_set_error_plot = ggplot(df_results, aes(x=def_ratio, y=val_set_error)) +
  geom_point() +
  geom_smooth(method=lm , color="orange", se=FALSE) +
  labs(title = "Validation Set Error vs. Default Ratio",
       x = "Default Ratio of the Test Dataset",
       y = "Validation Set Error") +
  theme(plot.title = element_text(size=11))

false_neg_plot = ggplot(df_results, aes(x = def_ratio, y = false_neg)) +
  geom_point() +
  geom_smooth(method=lm , color="orange", se=FALSE) +
  labs(title = "False Negative Rate vs. Default Ratio",
       x = "Default Ratio of the Test Dataset",
       y = "False Negative Rate") +
  theme(plot.title = element_text(size=11))

grid.arrange(val_set_error_plot, false_neg_plot, nrow = 1)
```



We observe a positive relationship in both graphs stating that as the default ratio in the test dataset increases, misclassification error and the false negative rate also increase. One possible explanation for this may be the imbalanced dataset. The algorithm in an imbalanced dataset is biased toward the majority as it has more observations from that class, learning more from the observations. Therefore, in that case the model is more likely to correctly classify the majority but fail to do the same for the minority. Therefore, it's not surprising to see an increase in the false negatives as the default ratio of the test dataset increase.

Since misclassification error includes both false negatives and false positives, we again observe the same positive relationship. However, comparing both graphs we can see that for the the false negative rate vs default ratio graph we observe a stronger positive relaationship as the data points are clustered around the best fit line, on the other graph we observe a weaker positive relationship where the data points are more dispersed from the best fit line. One possible explanation for that can be the effect of the false positives.