컴퓨터 통계 방법론 HW2

20180490 이재헌

Q1. Ch4-Prob1: Using a little bit of algebra, prove that (4.2) is equivalent to (4.3). In other words, the logistic function representation and logit representation for the logistic regression model are equivalent.

Answer: $Let\ p(X) = p,\ \beta_0 + \beta_1 X = z.\quad p = \frac{e^z}{1+e^z} \rightarrow\ p + e^z p = e^z \rightarrow \frac{p}{1-p} = e^z = e^{\beta_0+\beta_1 X} = \frac{p(X)}{1-p(X)}$

Q2. Ch4-Prob4: When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

(a) Suppose that we have a set of observations, each with measurements on p = 1 feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10 % of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with X = 0.6, we will use observations in the range [0.55, 0.65]. On average, what fraction of the available observations will we use to make the prediction?

In this one dimensional setting, the length of the interval is the fraction of the available observations. If X = 0.03, we can still use the interval [0, 0.1]. Therefore, the average fraction of the available observations is 0.1 = 10%.

(b) Now suppose that we have a set of observations, each with measurements on p = 2 features, X1 and X2. We assume that (X1,X2) are uniformly distributed on [0, 1] × [0, 1]. We wish to predict a test observation's response using only observations that are within 10 % of the range of X1 and within 10 % of the range of X2 closest to that test observation. For instance, in order to predict the response for a test observation with X1 = 0.6 and X2 = 0.35, we will use observations in the range [0.55, 0.65] for X1 and in the range [0.3, 0.4] for X2. On average, what fraction of the available observations will we use to make the prediction?

For the two dimensional case, the area that we use is the fraction. The width is 0.1 and the height is 0.1. Therefore, the area is 0.01. Thus, the average fraction of available observations is 0.01 = 1%

(c) Now suppose that we have a set of observations on p = 100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10 % of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

With the same logic, the volume of the hypercube is the fraction. Therefore, the average fraction is $10^{-100} = 10^{-98}\%$.

(d) Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

(a)-(c) shows that as the dimensionality (the number of features) increases, it becomes increasingly rare to find training observations that are "near" a given test observation according to the specified criteria.

These observations demonstrate that as the number of features grows, the likelihood of finding training observations that closely match the test observation on all features simultaneously diminishes rapidly. This is a problem for KNN because it relies on finding similar training observations to make predictions. When there are very few training observations "near" the test observation in high-dimensional spaces, the accuracy and reliability of KNN predictions can suffer. This phenomenon is often referred to as the "curse of dimensionality," and it's a well-known challenge in machine learning, particularly for distance-based algorithms like KNN.

(e) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10 % of the training observations. For p = 1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.

$$\textit{Let } l \textit{ be the length of each side so that the volume of the hypercube becomes } 0.1$$

$$l^p = 0.1 = 10^{-1} \rightarrow \quad l = 10^{-\frac{1}{p}}$$

$$l \rightarrow 1 \; as \; p \rightarrow \infty$$

As the dimensionality p increases, the length of each side also increases closely to 1. If we look at only one variable's target interval, it almost includes the whole interval.

Q3. Ch4-Prob10:

10. Equation 4.32 derived an expression for $\log\left(\frac{\Pr(Y=k|X=x)}{\Pr(Y=K|X=x)}\right)$ in the setting where $p > 1$, so that the mean for the kth class, μ_k, is a p-dimensional vector, and the shared covariance $\mathbf{\Sigma}$ is a $p \times p$ matrix. However, in the setting with $p = 1$, (4.32) takes a simpler form, since the means $\mu_1, \ldots, \mu_K$ and the variance σ^2 are scalars. In this simpler setting, repeat the calculation in (4.32), and provide expressions for a_k and b_{kj} in terms of π_k, π_K, μ_k, μ_K, and σ^2.

$$\begin{aligned} &= \log\left(\frac{\pi_k}{\pi_K}\right) - \frac{1}{2}(\mu_k + \mu_K)^T \mathbf{\Sigma}^{-1}(\mu_k - \mu_K) \\ &\quad + x^T \mathbf{\Sigma}^{-1}(\mu_k - \mu_K) \\ &= a_k + \sum_{j=1}^{p} b_{kj} x_j, \qquad\qquad (4.32) \end{aligned}$$

Answer: $\Sigma = \sigma^2, \;\; \Sigma^{-1} = \frac{1}{\sigma^2}$

$$(4.32) = \log\left(\frac{\pi_k}{\pi_K}\right) - \frac{\mu_k^2 - \mu_K^2}{2\sigma^2} + \frac{\mu_k - \mu_K}{\sigma^2} x$$

$$a_k = \log\left(\frac{\pi_k}{\pi_K}\right) - \frac{\mu_k^2 - \mu_K^2}{2\sigma^2}, \qquad b_k = \frac{\mu_k - \mu_K}{\sigma^2}$$

Q4. Ch4-Prob12:

12. Suppose that you wish to classify an observation $X \in \mathbb{R}$ into `apples` and `oranges`. You fit a logistic regression model and find that

$$\widehat{\Pr}(Y = \texttt{orange}|X = x) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x)}.$$

Your friend fits a logistic regression model to the same data using the *softmax* formulation in (4.13), and finds that

$$\widehat{\Pr}(Y = \texttt{orange}|X = x) = \frac{\exp(\hat{\alpha}_{\texttt{orange}0} + \hat{\alpha}_{\texttt{orange}1} x)}{\exp(\hat{\alpha}_{\texttt{orange}0} + \hat{\alpha}_{\texttt{orange}1} x) + \exp(\hat{\alpha}_{\texttt{apple}0} + \hat{\alpha}_{\texttt{apple}1} x)}.$$

(a) What is the log odds of orange versus apple in your model?

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \hat{\beta}_0 + \hat{\beta}_1 X$$

(b) What is the log odds of orange versus apple in your friend's model?

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \hat{\alpha}_{orange0} + \hat{\alpha}_{orange1}X - (\hat{\alpha}_{apple0} + \hat{\alpha}_{apple1}X)$$

(c) Suppose that in your model, $\hat{\beta}_0 = 2$ *and* $\hat{\beta}_1 = -1$. What are the coefficient estimates in your friend's model? Be as specific as possible.

$$\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1}X - \left(\hat{\alpha}_{apple0} + \hat{\alpha}_{apple1}X\right) = \hat{\beta}_0 + \hat{\beta}_1 X$$

$$\hat{\alpha}_{orange0} - \hat{\alpha}_{apple0} = \hat{\beta}_0, \qquad \hat{\alpha}_{orange1} - \hat{\alpha}_{apple1} = \hat{\beta}_1$$

$$We\ can\ set\ \ \hat{\alpha}_{apple0} = \hat{\alpha}_{apple1} = 0\ \ and\ \ \hat{\alpha}_{orange0} = \hat{\beta}_0, \qquad \hat{\alpha}_{orange1} = \hat{\beta}_1.$$

(d) Now suppose that you and your friend fit the same two models on a different data set. This time, your friend gets the coefficient estimates $\hat{\alpha}_{orange0} = 1.2, \hat{\alpha}_{orange1} = -2, \hat{\alpha}_{apple0} = 3, \hat{\alpha}_{apple1} = 0.6$. What are the coefficient estimates in your model?

$$\hat{\alpha}_{orange0} - \hat{\alpha}_{apple0} = \hat{\beta}_0 = -1.8, \qquad \hat{\alpha}_{orange1} - \hat{\alpha}_{apple1} = \hat{\beta}_1 = -2.6$$

(e) Finally, suppose you apply both models from (d) to a data set with 2,000 test observations. What fraction of the time do you expect the predicted class labels from your model to agree with those from your friend's model? Explain your answer.

$$(1)\ if\ my\ model\ classfies\ as\ an\ orange$$

$$\Pr(Y = orange \mid x) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} \geq \frac{1}{2} \;\rightarrow\; \hat{\beta}_0 + \hat{\beta}_1 X = -1.8 - 2.6X \geq 0 \;\rightarrow\; X \leq -\frac{9}{13}$$

$$(2)\ if\ my\ friend's\ model\ classfies\ as\ an\ orange$$

$$\Pr(Y = orange \mid x) = \frac{e^{\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1}X}}{e^{\hat{\alpha}_{orange0} + \hat{\alpha}_{orange1}X} + e^{\hat{\alpha}_{apple0} + \hat{\alpha}_{apple1}X}} \geq \frac{1}{2}$$

$$\rightarrow\; \hat{\alpha}_{orange0} + \hat{\alpha}_{orange1}X \;\geq\; \hat{\alpha}_{apple0} + \hat{\alpha}_{apple1}X$$

$$\rightarrow\; -1.8 - 2.6X \geq 0 \rightarrow\; X \leq -\frac{9}{13}$$

Therefore, they will agree 100%.

Q5. Ch4-Prob14: In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

(a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() method of the data frame. Note you may find it helpful to add a column mpg01 to the data frame by assignment. Assuming you have stored the data frame as Auto, this can be done as follows:

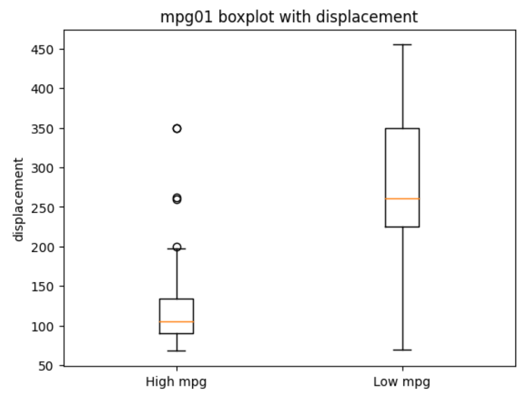
Auto['mpg01'] = mpg01

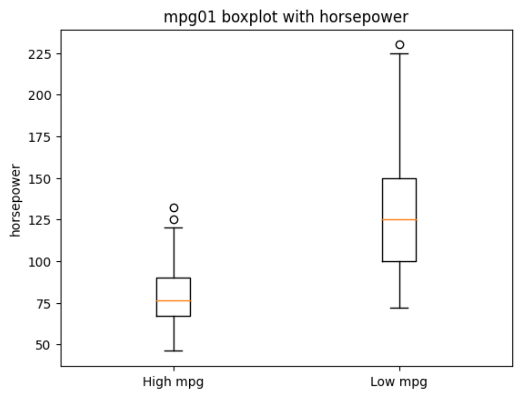
```
Auto['mpg01'] =  (Auto['mpg']>= np.median(Auto['mpg'])) * 1
```

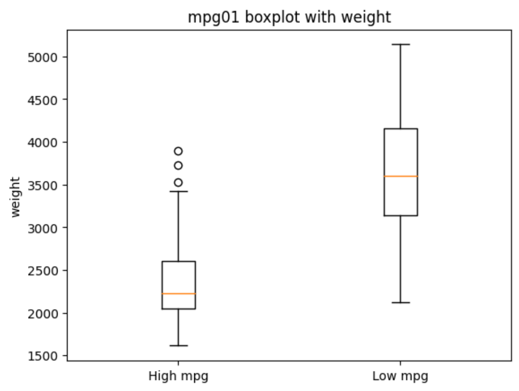
```
[ ]  Auto['mpg01']
```

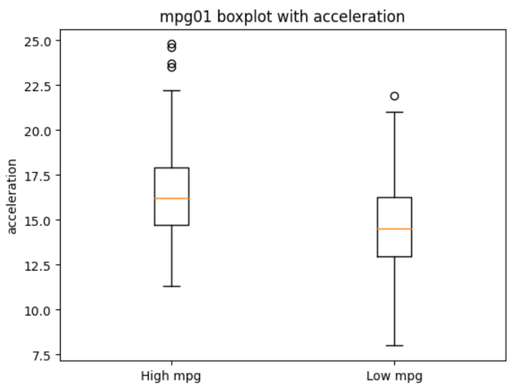
```
0      0
1      0
2      0
3      0
4      0
      ..
387    1
388    1
389    1
390    1
391    1
Name: mpg01, Length: 392, dtype: int64
```

(b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.









By investigating the boxplots for the features, I concluded that displacement, horsepower, weight and acceleration are related to mpg01.

(c) Split the data into a training set and a test set.

```
[14]  Auto_train, Auto_test = train_test_split(Auto, test_size=0.3)
```

(d) Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

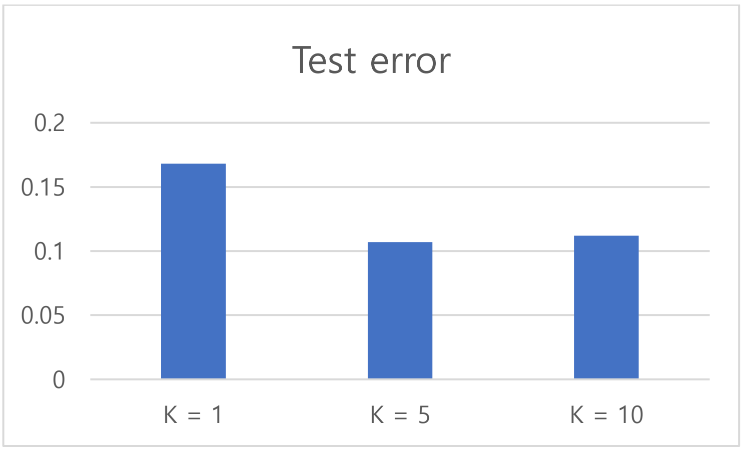
(e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

(f) Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

(g) Perform naive Bayes on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

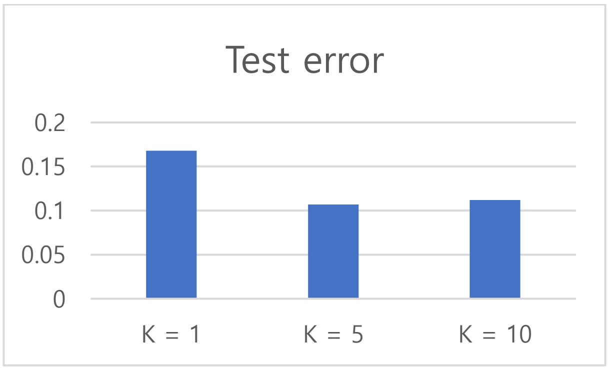
Answer for (d) – (g):

	LDA	QDA	Log. Reg.	Naive Bayes
Test error	0.102	0.097	0.097	0.092



(h) Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

	K = 1	K = 5	K = 10
Test error	0.168	0.107	0.112



K = 5 seems to be the best for this test data set.

Q6. Ch5-Prob1: Using basic statistical properties of the variance, as well as single variable calculus, derive (5.6). In other words, prove that) given by (5.6) does indeed minimize $Var(\alpha X + (1-\alpha)Y)$.

$$Var(\alpha X + (1-\alpha)Y) = \alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)Cov(X,Y)$$
$$= \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2\alpha(1-\alpha)\sigma_{XY} =: f(\alpha)$$

$$\frac{df}{d\alpha} = 2\sigma_X^2\alpha - 2\sigma_Y^2(1-\alpha) + 2\sigma_{XY}(1-2\alpha) = 2(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}) - 2(\sigma_Y^2 - \sigma_{XY}) = 0$$

$$\therefore \alpha = \frac{(\sigma_Y^2 - \sigma_{XY})}{(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY})}$$

Q7. Ch5-Prob2: We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.

$$Let\ n\ be\ the\ number\ of\ original\ samples.$$

$$\Pr(1st \neq j^{th}) = (1 - \frac{1}{n})$$

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

$$Since\ we\ sample\ with\ replacement, \Pr(2nd \neq j^{th}) = (1 - \frac{1}{n})$$

(c) Argue that the probability that the jth observation is not in the bootstrap sample is $\left(1 - \frac{1}{n}\right)^n$.

Since each sampling is independent,

$$\Pr(j^{th}\ is\ not\ in\ the\ bootstrap) = \prod_{i=1}^{n} \Pr(i\ th\ observation \neq j^{th}) = \prod_{i=1}^{n}\left(1 - \frac{1}{n}\right) = \left(1 - \frac{1}{n}\right)^n$$

(d) When n = 5, what is the probability that the jth observation is in the bootstrap sample?

$$1 - \left(1 - \frac{1}{5}\right)^5 = 1 - 0.3277 = 0.6723$$

(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

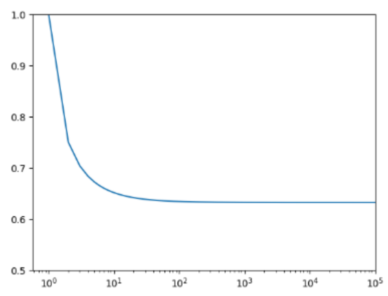
$$1 - \left(1 - \frac{1}{100}\right)^{100} = 1 - 0.3660 = 0.6340$$

(f) When n = 10,000, what is the probability that the jth observation is in the bootstrap sample?

$$1 - \left(1 - \frac{1}{10000}\right)^{10000} = 1 - 0.3679 = 1 - 0.6321$$

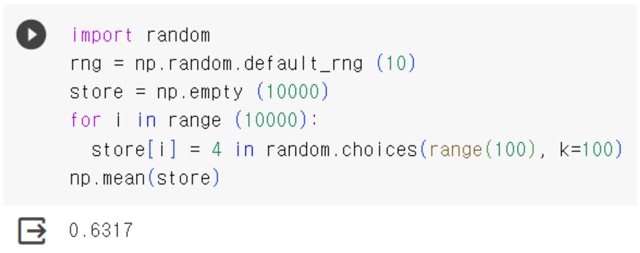
(g) Create a plot that displays, for each integer value of n from 1 to 100,000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

From the graph below, we can see that the sequence converges to a number near 0.3679 and it decreases monotonically. In fact, it converges to $1 - e^{-1} = 0.6321$.



$(x\ axis = n\ \&\ y\ axis = 1 - \left(1 - \frac{1}{n}\right)^n)$.

(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We first create an array store with values that will subsequently be overwritten using the function np.empty(). We then repeatedly create bootstrap samples, and each time we record whether or not the fifth observation is contained in the bootstrap sample. Comment on the results obtained.



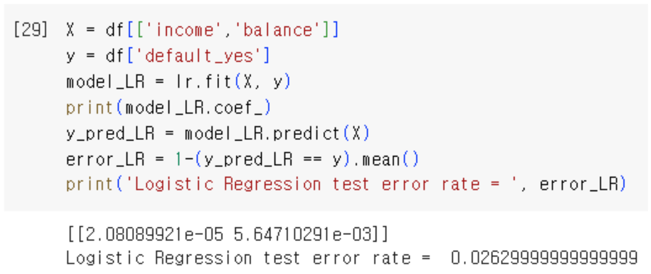
```
import random
rng = np.random.default_rng (10)
store = np.empty (10000)
for i in range (10000):
  store[i] = 4 in random.choices(range(100), k=100)
np.mean(store)
```

```
0.6317
```

100 bootstrap samples contain 4th observation with probability 0.6317, which is very close to the theoretical value 0.6321. Therefore, we verified the theory.

Q8. Ch5-Prob5: In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(a) Fit a logistic regression model that uses income and balance to predict default.



```
[29] X = df[['income','balance']]
     y = df['default_yes']
     model_LR = lr.fit(X, y)
     print(model_LR.coef_)
     y_pred_LR = model_LR.predict(X)
     error_LR = 1-(y_pred_LR == y).mean()
     print('Logistic Regression test error rate = ', error_LR)
```

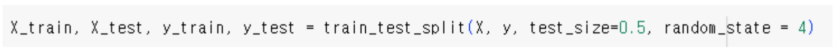
```
[[2.08089921e-05 5.64710291e-03]]
Logistic Regression test error rate =  0.02629999999999999
```

Coefficient = [0.000021, 0.00565]

Test error rate = 2.6%

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i) Split the sample set into a training set and a validation set.

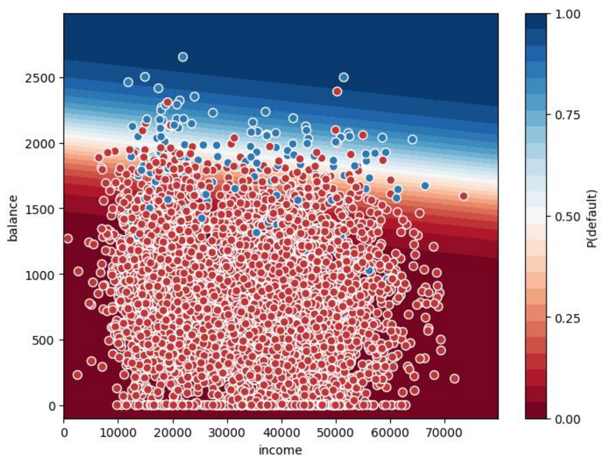


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state = 4)
```

ii) Fit a multiple logistic regression model using only the training observations.

```
[31] mod = lr.fit(X_train, y_train)
     mod.coef_

     array([[2.47383350e-05, 5.49074745e-03]])
```

We can visualize the data with the logistic regression line as follows.



iii) Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.

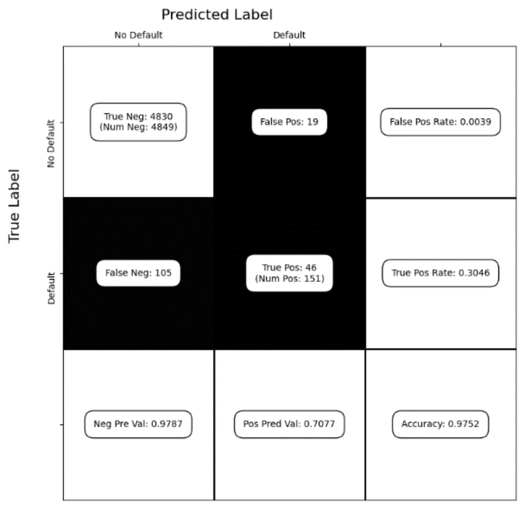
iv) Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
[34] y_pred = mod.predict(X_test)
     1-(y_pred == y_test).mean()

     0.023800000000000043
```

Test error rate = 2.38%

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

With different training & test data sets, I got 2.64%, and 2.48% error rates. On average, the test error rate is 2.50%. The confusion matrix is like below.



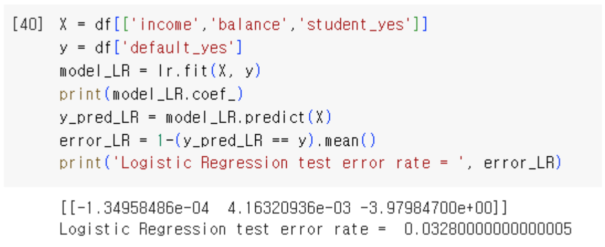
Predicted Label

True Label

	No Default	Default	
No Default	True Neg: 4830 (Num Neg: 4849)	False Pos: 19	False Pos Rate: 0.0039
Default	False Neg: 105	True Pos: 46 (Num Pos: 151)	True Pos Rate: 0.3046
	Neg Pre Val: 0.9787	Pos Pred Val: 0.7077	Accuracy: 0.9752

False negative rate = 2.13%

(d) Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.



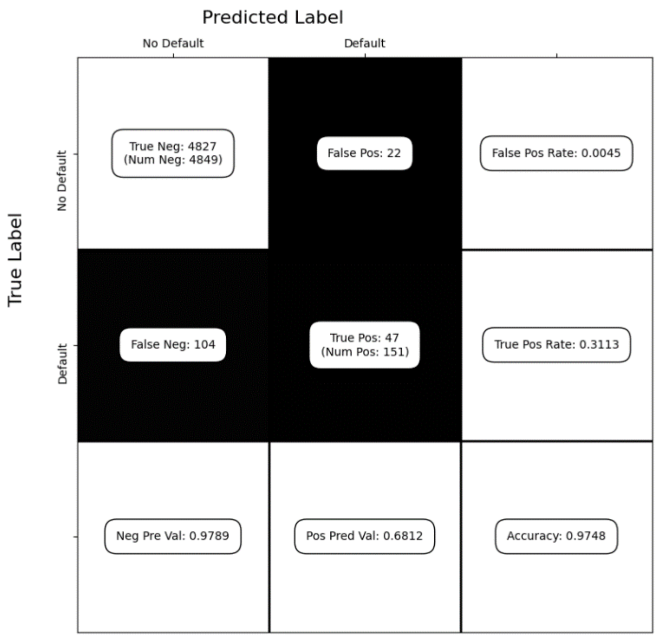
```
[40] X = df[['income','balance','student_yes']]
     y = df['default_yes']
     model_LR = lr.fit(X, y)
     print(model_LR.coef_)
     y_pred_LR = model_LR.predict(X)
     error_LR = 1-(y_pred_LR == y).mean()
     print('Logistic Regression test error rate = ', error_LR)

     [[-1.34958486e-04  4.16320936e-03 -3.97984700e+00]]
     Logistic Regression test error rate =  0.03280000000000005
```

The coefficient for student is negative as it was in the textbook.

Test error rate = 3.28%, which is greater than the one without the student dummy variable.

Adding the dummy variable doesn't necessarily decrease the test error rate.

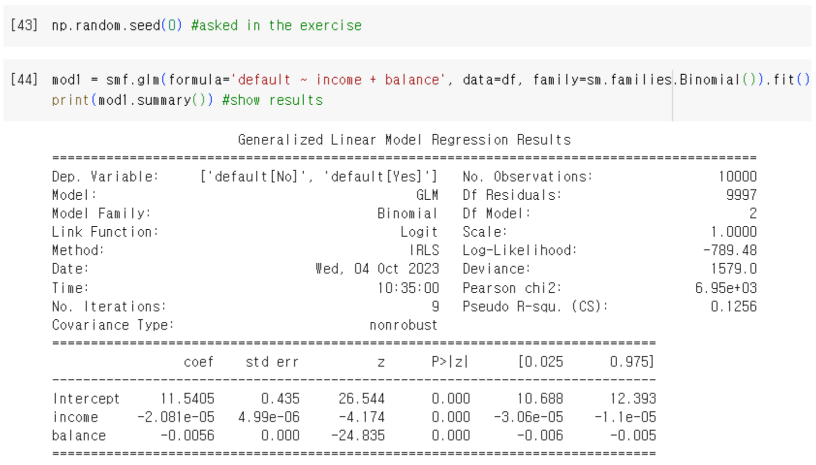
The confusion matrix is below.



However, it slightly decreased the false negative rate from 2.13% to 2.11%.

Q9. Ch5-Prob6: We continue to consider the use of a logistic regression model to predict the probability of default using income and balance on the Default data set. In particular, we will now compute estimates for the standard errors of the income and balance logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the sm.GLM() function. Do not forget to set a random seed before beginning your analysis.

(a) Using the summarize() and sm.GLM() functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predictors.

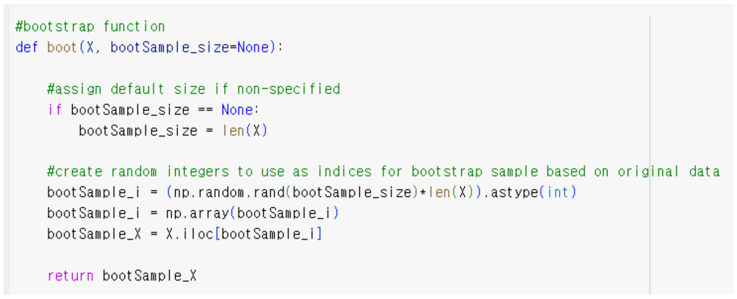


```
[43] np.random.seed(0) #asked in the exercise
```

```
[44] mod1 = smf.glm(formula='default ~ income + balance', data=df, family=sm.families.Binomial()).fit()
     print(mod1.summary()) #show results
```

```
                          Generalized Linear Model Regression Results
==============================================================================================
Dep. Variable:     ['default[No]', 'default[Yes]']   No. Observations:               10000
Model:                                         GLM   Df Residuals:                    9997
Model Family:                             Binomial   Df Model:                           2
Link Function:                               Logit   Scale:                         1.0000
Method:                                       IRLS   Log-Likelihood:               -789.48
Date:                             Wed, 04 Oct 2023   Deviance:                      1579.0
Time:                                     10:35:00   Pearson chi2:                6.95e+03
No. Iterations:                                  9   Pseudo R-squ. (CS):            0.1256
Covariance Type:                         nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     11.5405      0.435     26.544      0.000      10.688      12.393
income     -2.081e-05   4.99e-06     -4.174      0.000   -3.06e-05    -1.1e-05
balance       -0.0056      0.000    -24.835      0.000      -0.006      -0.005
==============================================================================
```

(b) Write a function, boot_fn(), that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.



```
#bootstrap function
def boot(X, bootSample_size=None):

    #assign default size if non-specified
    if bootSample_size == None:
        bootSample_size = len(X)

    #create random integers to use as indices for bootstrap sample based on original data
    bootSample_i = (np.random.rand(bootSample_size)*len(X)).astype(int)
    bootSample_i = np.array(bootSample_i)
    bootSample_X = X.iloc[bootSample_i]

    return bootSample_X
```

Coefficient (b): [-0.000021, -0.005647] = [-2.08089755293e-05, -0.0056471029503]

(c) Following the bootstrap example in the lab, use your boot_fn() function to estimate the standard errors of the logistic regression coefficients for income and balance.

Results (c): [-0.000021, -0.005672]

(d) Comment on the estimated standard errors obtained using the sm.GLM() function and using the bootstrap.

	Coeff_income	Coeff_balance
Logistic regression	-0.00002081	-0.005600
sm.glm()	-0.00002081	-0.005647
Bootstrap	-0.00002100	-0.005672

$SE_B(\hat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\theta}_b - \bar{\theta}\right)^2}$. Since the coefficients are too small, the standard errors are too small to decide the significant digits. Therefore, I only compared the coefficients.

Q10. Ch5-Prob7: In Sections 5.1.2 and 5.1.3, we saw that the cross_validate() function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just sm.GLM() and the predict() method of the fitted model within a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the Weekly data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

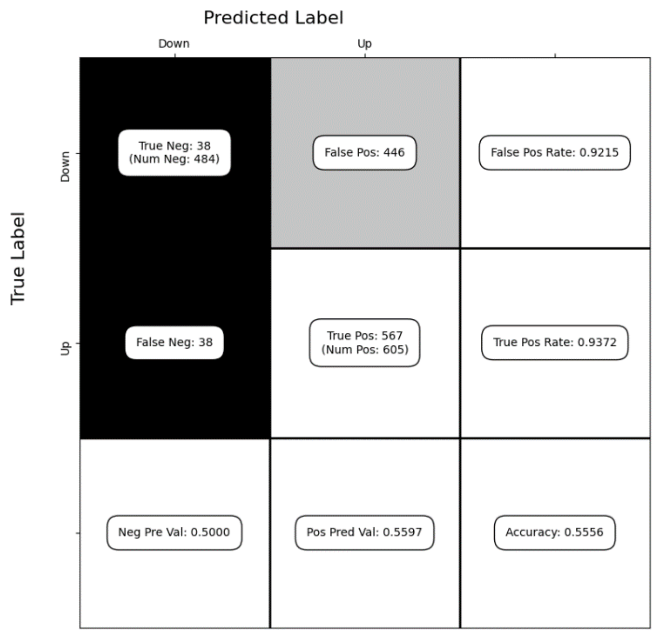
(a) Fit a logistic regression model that predicts Direction using Lag1 and Lag2.

```
X = Weekly[['Lag1','Lag2']]
y = Weekly['Is_up']
model_LR = lr.fit(X, y)
print(model_LR.coef_)
y_pred_LR = model_LR.predict(X)
error_LR = 1-(y_pred_LR == y).mean()
print('Logistic Regression test error rate = ', error_LR)
```

```
[[-0.03872222  0.0602483 ]]
Logistic Regression test error rate =  0.4444444444444444
```

Test error rate = 44.4%

The confusion matrix is like below.



(b) Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.

```
mod.fit(X, y)
print(mod.intercept_, mod.coef_, (mod.predict(X) == y).mean())  # accuracy
mod.fit(X.iloc[1:], y.iloc[1:])
print(mod.intercept_, mod.coef_, (mod.predict(X) == y).mean())
```

```
[0.22122405] [[-0.03872222  0.0602483 ]] 0.5555555555555556
[0.22324305] [[-0.03843317  0.06084763]] 0.5564738292011019
```

The accuracy slightly increased meaning the first observation would be misclassified with this model.

(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if P(Direction = "Up"|Lag1, Lag2) > 0.5. Was this observation correctly classified?

```
mod.predict([X.iloc[0].values]) == y[0]
```

```
array([False])
```

No, it is misclassified.

(d) Write a for loop from i = 1 to i = n, where n is the number of observations in the data set, that performs each of the following steps:

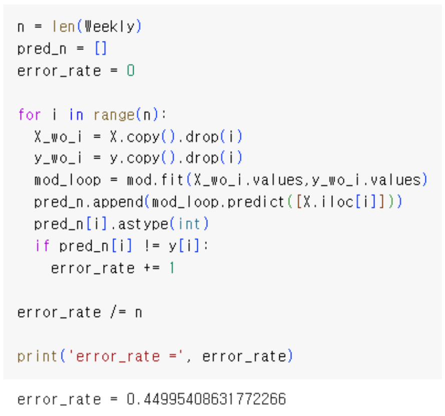
i) Fit a logistic regression model using all but the ith observation to predict Direction using Lag1 and Lag2.

ii) Compute the posterior probability of the market moving up for the ith observation.

iii) Use the posterior probability for the ith observation in order to predict whether or not the market moves up.

iv) Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

(e) Take the average of the n numbers obtained in (d) iv) in order to obtain the LOOCV estimate for the test error. Comment on the results.



```
n = len(Weekly)
pred_n = []
error_rate = 0

for i in range(n):
  X_wo_i = X.copy().drop(i)
  y_wo_i = y.copy().drop(i)
  mod_loop = mod.fit(X_wo_i.values,y_wo_i.values)
  pred_n.append(mod_loop.predict([X.iloc[i]]))
  pred_n[i].astype(int)
  if pred_n[i] != y[i]:
    error_rate += 1

error_rate /= n

print('error_rate =', error_rate)
```

```
error_rate = 0.44995408631772266
```

The test error rate of LOOCV = 45.00% > 44.44%. But the test error rate is quite similar.