Homework 2

Ross B. Alexander (rbalexan@stanford.edu)

Problem 1 (C4E1)

Prove that

$$p(X) = \frac{\exp(\beta_0+\beta_1 X)}{1+\exp(\beta_0+\beta_1 X)} \qquad (1)$$

is equivalent to

$$\frac{p(X)}{1-p(X)} = \exp(\beta_0+\beta_1 X). \qquad (2)$$

- We have (1)

$$p(X) = \frac{\exp(\beta_0+\beta_1 X)}{1+\exp(\beta_0+\beta_1 X)}$$

- Let $\exp(\beta_0+\beta_1 X) = \alpha$, then

$$p(X) = \frac{\alpha}{1+\alpha}$$

$$\begin{aligned}
\frac{p(X)}{1-p(X)} &= \frac{1}{1-p(X)} \cdot \frac{\alpha}{1+\alpha} \\
&= \frac{1}{1-\frac{\alpha}{1+\alpha}} \cdot \frac{\alpha}{1+\alpha} \\
&= \frac{1}{\frac{(1+\alpha)-\alpha}{1+\alpha}} \cdot \frac{\alpha}{1+\alpha} \\
&= \frac{1+\alpha}{1} \cdot \frac{\alpha}{1+\alpha} \\
&= \alpha
\end{aligned}$$

$$\frac{p(X)}{1-p(X)} = \exp(\beta_0+\beta_1 X)$$

Therefore, we have shown that (1) and (2) are equivalent. ■

Problem 2 (C4E4)

(a) Generally, we use 10% of the data.

(b) Generally, we will use 1% of the data.

(c) Generally, we will use $(0.1)^{100} = 10^{-100} \rightarrow 10^{-98}\%$ of the data.

(d) We can see that as p increases, the fraction of observations "near" a given test observation decreases exponentially as 10^{-p}. As a result, KNN suffers when p is reasonably large.

(e) In $p=1$ dimensions, the length of the hypercube is

$$l^1 = \frac{1}{10}$$
$$l = \frac{1}{10}$$

In $p=2$ dimensions, the length of the hypercube is

$$l^2 = \frac{1}{10}$$
$$l = 10^{-\frac{1}{2}} \approx 0.3162$$

In $p=100$ dimensions, the length of the hypercube is

$$l^{100} = \frac{1}{10}$$
$$l = 10^{-\frac{1}{100}} \approx 0.9772$$

$l_1 = 0.1$	$10^{-\frac{1}{1}}$
$l_2 = 0.3162$	$10^{-\frac{1}{2}}$
$l_{100} = 0.9772$	$10^{-\frac{1}{100}}$

Problem 3 (CHE6)

$X_1 \equiv$ hours studied
$X_2 \equiv$ undergrad GPA
$Y \equiv$ receive an A $(Y=0$, not A; $Y=1$, A$)$

$\hat{\beta}_0 = -6$
$\hat{\beta}_1 = 0.05$
$\hat{\beta}_2 = 1$

(a) For $x_1 = 40$, $x_2 = 3.5$, $P(Y=1 \mid X=[x_1, x_2])$ is

$$P(Y=1 \mid X=[x_1, x_2]) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2)}{1+\exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2)}$$

$$= \frac{\exp(-6+0.05(40)+1(3.5))}{1+\exp(-6+0.05(40)+1(3.5))}$$

$$= \frac{\exp(-0.5)}{1+\exp(-0.5)}$$

$$\approx 0.3775$$

$$\boxed{P(Y=1 \mid X=[40, 3.5]) \approx 0.3775}$$

(b) In order to have 50% probability of getting an A, we require the linear term $(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2)$ to be zero. Given the student in (a) with an undergrad GPA of 3.5, we need to find x_1. Manipulating gives the following

$$\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0$$

$$x_1 = \frac{-\hat{\beta}_0 - \hat{\beta}_2 x_2}{\hat{\beta}_1}$$

Plugging in values gives

$$x_1 = \frac{-(-6) - (1)(3.5)}{(0.05)}$$

$$x_1 = \frac{2.5}{0.05}$$

$$x_1 = 50$$

$$\boxed{x_1 = 50 \text{ hours}}$$

Problem 4 (C4E8)

- equally-sized train & test sets
- logistic regression
  - train error 20%
  - test error 30%
- KNN ($k=1$)
  - average error 18% (average over train and test)

We should use the logistic regression model to classify new observations. For KNN with $k=1$, the training error must be 0% since the 1-nearest neighbor to any point is always itself. Given the equal split of our training and testing datasets, this implies the testing error for KNN with $k=1$ is 36%. Since the test error for logistic regression is slightly lower at 30%, we would expect predictions from the logistic regression model to be more accurate, which is preferred.

Problem 5 (ch10)

(a) See attached code. We see a strong correlation (0.8419) between year and volume with a roughly quadratic relationship. All other variable combinations appear very weakly correlated.

(b) See attached code. The intercept and the 'Lag2' variable have p-values less than 0.05 and thus, they indicate statistical significance.

(c) See attached code. The accuracy is 56.1%. From the confusion matrix, we can see that when we observe 'Down' we make a correct prediction 11.1% of the time (bad!). When we observe 'Up' we make a correct prediction 92.1% of the time (awesome!). Thus, we tend to make far more mistakes when we observe 'Down'.

(d) See attached code. The accuracy is 62.5%. From the confusion matrix, we can see that when we observe 'Down' we make a correct prediction 20.9% of the time (bad!). When we observe 'Up' we make a correct prediction 91.8% of the time (awesome!). Thus, we tend to make far more mistakes when we observe 'Down'.

(e) See attached code. The confusion matrix (and therefore, results and conclusions) is identical to (d).

(f) See attached code. The accuracy is 41.3%. From the confusion matrix, we can see that when we observe 'Down' we make a correct prediction 100% of the time (wow!). When we observe 'Up' we make a correct prediction 0 % of the time (terrible!). Basically, QDA has simply learned to classify everything as 'Down'.

(g) See attached code. The accuracy is 49.0%. From the confusion matrix, we can see that when we observe 'Down' we make a correct prediction 48.8% of the time (bad!). When we observe 'Up' we make a correct prediction 49.2% of the time (also bad!). We tend to perform okay relatively equally on both classes.

(h) The best test accuracy we can get is using logistic regression or LDA.

Problem 6 (CSE2)

(a) The probability the first bootstrap sample is not the jth observation from the original sample is $\frac{n-1}{n}$ since there are n possible samples and $n-1$ possible samples that are not the jth one.

(b) The probability the second bootstrap sample is not the jth observation from the original sample is $\frac{n-1}{n}$ since there are n possible samples and $n-1$ possible samples that are not the jth one.

(c) Since bootstrap samples are made independently with replacement, the probability that the jth observation is not in the bootstrap sample can be expressed as

$$\prod_{i=1}^{n}\left(\frac{n-1}{n}\right) = \left(\frac{n-1}{n}\right)^n = \left(\frac{n}{n}-\frac{1}{n}\right)^n = \left(1-\frac{1}{n}\right)^n. \quad \blacksquare$$

(d) For $n=5$, we have $P(x_j \in X^b) = 1 - P(x_j \notin X^b) = 1-\left(1-\frac{1}{5}\right)^5 \simeq 0.6723$.

(e) For $n=100$, we have $P(x_j \notin X^b) = 1 - P(x_j \notin X^b) = 1-\left(1-\frac{1}{100}\right)^{100} \simeq 0.6340$.

(f) For $n=10000$, we have $P(x_j \notin X^b) = 1 - P(x_j \notin X^b) = 1-\left(1-\frac{1}{10000}\right)^{10000} \simeq 0.6321$.

(g) See attached code. We see exponentially quick convergence to 0.6321.

(h) We see that we obtain sample 4 in the bootstrap sample 6302 times out of 10000, which is quite close to the 63.40% expected for bootstrap samples of size 100.

Problem 7 (CSE5)

(a) See attached code.

(b) See attached code. We obtain a test error of 2.52%.

(c) See attached code. We obtain test errors of 2.36%, 2.76%, and 2.72%, which all appear relatively close, as expected.

(d) See attached code. We obtain a test error of 2.8%, which isn't any improvement over the model not containing the student variable.

Problem 8 (C5E6)

(a) See attached code. The estimated SEs are:

$\widehat{SE}(\hat{\beta}_0) = 0.435$

$\widehat{SE}(\hat{\beta}_{inc}) = 4.99 \times 10^{-6}$

$\widehat{SE}(\hat{\beta}_{bal}) = 2.27 \times 10^{-4}$

(b) See attached code.

(c) See attached code. The estimated SEs are:

$\widehat{SE}^b(\hat{\beta}_{inc}) = 5.29 \times 10^{-6}$

$\widehat{SE}^b(\hat{\beta}_{bal}) = 2.23 \times 10^{-4}$

(d) The standard errors from the single model and from the bootstrap method are very close!

Problem 9 (CSE8)

(a) See attached code. $n = 100$, $p = 1$. The model form is $y_i = -2x_i^2 + x_i + \epsilon_i$ $\epsilon_i \sim N(0, 1)$.

(b) See attached code. We observe a negative quadratic trend.

(c) See attached code. We obtain the following LOOCV errors:

degree 1: 5.89×10^0
degree 2: 1.09×10^0
degree 3: 1.10×10^0
degree 4: 1.11×10^0

(d) See attached code. We obtain the following LOOCV errors:

degree 1: 5.89×10^0
degree 2: 1.09×10^0
degree 3: 1.10×10^0
degree 4: 1.11×10^0

The results are the same as in (c) since LOOCV with least-squares regression is purely deterministic.

(e) The quadratic model had the lowest LOOCV error, as expected since the underlying function is quadratic.

(f) See attached code. In models of degree 2 or larger, the p-values of the linear and quadratic coefficients are less than 0.000, indicating strong statistical significance, which agrees with the cross-validation results.

Problem 10 (CSE9)

(a) See attached code. $\hat{\mu} = 22.533$.

(b) See attached code. $\widehat{SE}(\hat{\mu}) = 0.4089$. This means we are reasonably close to the true mean.

(c)

Problem 5 (Chapter 4, Exercise 10)

```
import pandas as pd
weekly = pd.read_csv("Weekly.csv")
```

```
weekly.info()
```

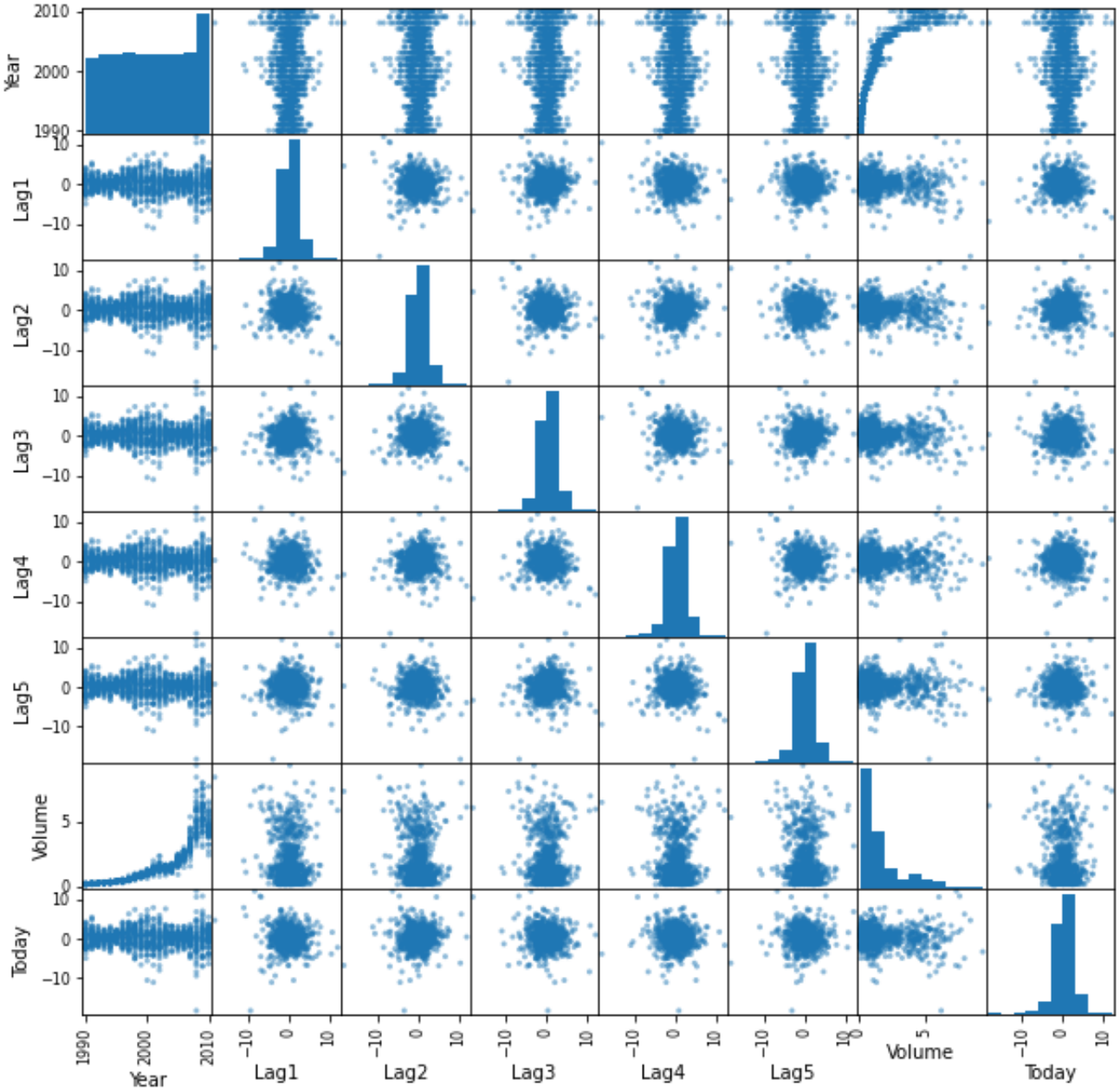
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1089 entries, 0 to 1088
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Year       1089 non-null   int64  
 1   Lag1       1089 non-null   float64
 2   Lag2       1089 non-null   float64
 3   Lag3       1089 non-null   float64
 4   Lag4       1089 non-null   float64
 5   Lag5       1089 non-null   float64
 6   Volume     1089 non-null   float64
 7   Today      1089 non-null   float64
 8   Direction  1089 non-null   object 
dtypes: float64(7), int64(1), object(1)
memory usage: 76.7+ KB
```

Problem 5(a)

```
weekly.describe(include='all')
```

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	1(
count	1089.000000	1089.000000	1089.000000	1089.000000	1089.000000	1089.000000	
unique	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	NaN	
mean	2000.048669	0.150585	0.151079	0.147205	0.145818	0.139893	
std	6.033182	2.357013	2.357254	2.360502	2.360279	2.361285	
min	1990.000000	-18.195000	-18.195000	-18.195000	-18.195000	-18.195000	
25%	1995.000000	-1.154000	-1.154000	-1.158000	-1.158000	-1.166000	
50%	2000.000000	0.241000	0.241000	0.241000	0.238000	0.234000	
75%	2005.000000	1.405000	1.409000	1.409000	1.409000	1.405000	
max	2010.000000	12.026000	12.026000	12.026000	12.026000	12.026000	

```
pd.plotting.scatter_matrix(weekly, figsize=(10,10));
```



```
weekly.corr()
```

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Tod
Year	1.000000	-0.032289	-0.033390	-0.030006	-0.031128	-0.030519	0.841942	-0.032
Lag1	-0.032289	1.000000	-0.074853	0.058636	-0.071274	-0.008183	-0.064951	-0.075
Lag2	-0.033390	-0.074853	1.000000	-0.075721	0.058382	-0.072499	-0.085513	0.059
Lag3	-0.030006	0.058636	-0.075721	1.000000	-0.075396	0.060657	-0.069288	-0.071
Lag4	-0.031128	-0.071274	0.058382	-0.075396	1.000000	-0.075675	-0.061075	-0.007
Lag5	-0.030519	-0.008183	-0.072499	0.060657	-0.075675	1.000000	-0.058517	0.011
Volume	0.841942	-0.064951	-0.085513	-0.069288	-0.061075	-0.058517	1.000000	-0.033
Today	-0.032460	-0.075032	0.059167	-0.071244	-0.007826	0.011013	-0.033078	1.000

Problem 5(b)

```
import statsmodels.api as sm

X = weekly[['Lag1', 'Lag2', 'Lag3', 'Lag4', 'Lag5', 'Volume']]
X = sm.add_constant(X)

y = weekly['Direction'].astype('category').cat.codes

log_reg = sm.Logit(y, X).fit()
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: Futu
  import pandas.util.testing as tm
Optimization terminated successfully.
         Current function value: 0.682441
         Iterations 4
```

```
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1089
Model:                          Logit   Df Residuals:                     1082
Method:                           MLE   Df Model:                            6
Date:                Fri, 16 Jul 2021   Pseudo R-squ.:                0.006580
Time:                        19:36:57   Log-Likelihood:                -743.18
converged:                       True   LL-Null:                       -748.10
Covariance Type:            nonrobust   LLR p-value:                    0.1313
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2669      0.086      3.106      0.002       0.098       0.435
Lag1          -0.0413      0.026     -1.563      0.118      -0.093       0.010
Lag2           0.0584      0.027      2.175      0.030       0.006       0.111
Lag3          -0.0161      0.027     -0.602      0.547      -0.068       0.036
Lag4          -0.0278      0.026     -1.050      0.294      -0.080       0.024
Lag5          -0.0145      0.026     -0.549      0.583      -0.066       0.037
Volume        -0.0227      0.037     -0.616      0.538      -0.095       0.050
==============================================================================
```

Problem 5(c)

```
print(log_reg.pred_table())
```

```
[[ 54. 430.]
 [ 48. 557.]]
```

```
acc = (54 + 557) / (54 + 430 + 48 + 557)
print(acc)
```

```
0.5610651974288338
```

```
print(54/(54+430))
print(557/(557+48))
```

```
0.1115702479338843
0.9206611570247933
```

Problem 5(d)

```
weekly1990to2008 = weekly.copy()
weekly1990to2008 = weekly1990to2008[weekly1990to2008['Year'] >= 1990]
weekly1990to2008 = weekly1990to2008[weekly1990to2008['Year'] <= 2008]
X_train = weekly1990to2008['Lag2']
X_train = sm.add_constant(X_train)

y_train = weekly1990to2008['Direction'].astype('category').cat.codes

log_reg = sm.Logit(y_train, X_train).fit()
```

```
   Optimization terminated successfully.
            Current function value: 0.685555
            Iterations 4
```

```
weekly2009to2010 = weekly.copy()
weekly2009to2010 = weekly2009to2010[weekly2009to2010['Year'] >= 2009]
weekly2009to2010 = weekly2009to2010[weekly2009to2010['Year'] <= 2010]
X_test = weekly2009to2010['Lag2']
X_test = sm.add_constant(X_test)

y_test = weekly2009to2010['Direction'].astype('category').cat.codes
y_pred = (log_reg.predict(X_test) >= 0.5).astype(int)

import numpy as np
cm = np.zeros((2,2))
for i in range(2):
  for j in range(2):
    cm[i, j] = sum(yt==i and yp==j for yt, yp in zip(y_test, y_pred))

cm
```

```
   array([[ 9., 34.],
          [ 5., 56.]])
```

```
acc = (9 + 56) / (9 + 56 + 5 + 34)
print(acc)
```

```
   0.625
```

```
print(9/(9+34))
print(56/(5+56))
```

```
   0.20930232558139536
   0.9180327868852459
```

Problem 5(e)

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA()
lda.fit(X_train, y_train)
```

```
   LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
                              solver='svd', store_covariance=False, tol=0.0001)
```

```
from sklearn.metrics import confusion_matrix

y_pred = lda.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
   [[ 9 34]
    [ 5 56]]
```

```
acc = (9 + 56) / (9 + 56 + 5 + 34)
print(acc)
```

```
   0.625
```

```
print(9/(9+34))
print(56/(5+56))
```

```
   0.20930232558139536
   0.9180327868852459
```

Problem 5(f)

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA

qda = QDA()
qda.fit(X_train, y_train)
```

```
   /usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:691: U
     warnings.warn("Variables are collinear")
   QuadraticDiscriminantAnalysis(priors=None, reg_param=0.0,
                                 store_covariance=False, tol=0.0001)
```

```
y_pred = qda.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[43  0]
 [61  0]]
/usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:715: F
  X2 = np.dot(Xm, R * (S ** (-0.5)))
/usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:715: F
  X2 = np.dot(Xm, R * (S ** (-0.5)))
/usr/local/lib/python3.7/dist-packages/sklearn/discriminant_analysis.py:718: F
  u = np.asarray([np.sum(np.log(s)) for s in self.scalings_])
```

```
acc = (43) / (43+61)
print(acc)
```

```
0.41346153846153844
```

```
print(43/(43+0))
print(0/(61+0))
```

```
1.0
0.0
```

Problem 5(g)

```
from sklearn.neighbors import KNeighborsClassifier as KNN

knn = KNN(1)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

```
y_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[21 22]
 [31 30]]
```

```
acc = (21+30) / (21+30+22+31)
print(acc)
```

```
0.49038461538461536
```

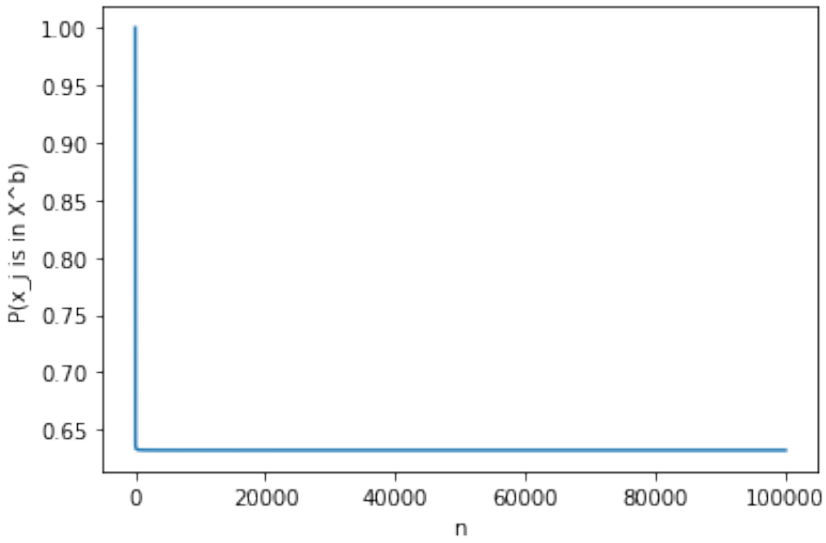
```
print(21/(21+22))
print(30/(31+30))
```

```
0.4883720930232558
0.4918032786885246
```

Problem 6 (Chapter 5, Exercise 2)

Problem 6(g)

```
import matplotlib.pyplot as plt
plt.plot(np.arange(1, 100000), [1-(1-1/n)**n for n in np.arange(1, 100000)])
plt.xlabel("n")
plt.ylabel("P(x_j is in X^b)");
```



Problem 6(h)

```
np.random.seed(1)

ctr = 0
for i in range(10000):
  a = np.random.choice(range(100), size=100, replace=True)
  if 4 in a:
    ctr += 1

print(ctr)
```

```
   6302
```

Problem 7 (Chapter 5, Exercise 5)

```
default = pd.read_csv("Default.csv")
```

```
np.random.seed(1)
```

Problem 7(a)

```
X = default[['income', 'balance']]
X = sm.add_constant(X)

y = default['default'].astype('category').cat.codes

log_reg = sm.Logit(y, X).fit()
```

```
   Optimization terminated successfully.
            Current function value: 0.078948
            Iterations 10
```

Problem 7(b)

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

log_reg = sm.Logit(y_train, X_train).fit()

y_pred = (log_reg.predict(X_test) >= 0.5).astype(int)

error = np.mean([yt != yp for yt, yp in zip(y_test, y_pred)])
print(error)
```

```
   Optimization terminated successfully.
            Current function value: 0.078638
            Iterations 10
   0.0252
```

Problem 7(c)

```
# trial 1
X_train, X_test, y_train, y_test = train_test_split(X, y)

log_reg = sm.Logit(y_train, X_train).fit()

y_pred = (log_reg.predict(X_test) >= 0.5).astype(int)

error = np.mean([yt != yp for yt, yp in zip(y_test, y_pred)])
print(error)

# trial 2
X_train, X_test, y_train, y_test = train_test_split(X, y)

log_reg = sm.Logit(y_train, X_train).fit()

y_pred = (log_reg.predict(X_test) >= 0.5).astype(int)

error = np.mean([yt != yp for yt, yp in zip(y_test, y_pred)])
print(error)

# trial 3
X_train, X_test, y_train, y_test = train_test_split(X, y)

log_reg = sm.Logit(y_train, X_train).fit()

y_pred = (log_reg.predict(X_test) >= 0.5).astype(int)

error = np.mean([yt != yp for yt, yp in zip(y_test, y_pred)])
print(error)
```

```
Optimization terminated successfully.
         Current function value: 0.079030
         Iterations 10
0.0236
Optimization terminated successfully.
         Current function value: 0.076866
         Iterations 10
0.0276
Optimization terminated successfully.
         Current function value: 0.079056
         Iterations 10
0.0272
```

Problem 7(d)

```
default['student'] = default['student'].astype('category').cat.codes

X = default[['income', 'balance', 'student']]
X = sm.add_constant(X)

X_train, X_test, y_train, y_test = train_test_split(X, y)

log_reg = sm.Logit(y_train, X_train).fit()

y_pred = (log_reg.predict(X_test) >= 0.5).astype(int)

error = np.mean([yt != yp for yt, yp in zip(y_test, y_pred)])
print(error)
```

```
   Optimization terminated successfully.
            Current function value: 0.076317
            Iterations 10
   0.028
```

Problem 8 (Chapter 5, Exercise 6)

```
default = pd.read_csv("Default.csv")


np.random.seed(1)
```

Problem 8(a)

```
X = default[['income', 'balance']]
X = sm.add_constant(X)

y = default['default'].astype('category').cat.codes

log_reg = sm.Logit(y, X).fit()
```

```
   Optimization terminated successfully.
            Current function value: 0.078948
            Iterations 10
```

```
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9997
Method:                           MLE   Df Model:                            2
Date:                Fri, 16 Jul 2021   Pseudo R-squ.:                  0.4594
Time:                        19:36:59   Log-Likelihood:                -789.48
converged:                       True   LL-Null:                       -1460.3
Covariance Type:            nonrobust   LLR p-value:                4.541e-292
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -11.5405      0.435    -26.544      0.000     -12.393     -10.688
income      2.081e-05   4.99e-06      4.174      0.000     1.1e-05    3.06e-05
balance        0.0056      0.000     24.835      0.000       0.005       0.006
==============================================================================

Possibly complete quasi-separation: A fraction 0.14 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

Problem 8(b)

```
def boot_fn(default, indices):

  X = default[['income', 'balance']]
  X = X.iloc[indices, :]
  X = sm.add_constant(X)

  y = default['default'].astype('category').cat.codes
  y = y.iloc[indices]

  log_reg = sm.Logit(y, X).fit()

  return log_reg.params[1:]
```

Problem 8(c)

```
beta_is, beta_bs = [], []
B = 100

for i in range(B):
```

```
beta_i, beta_b = boot_fn(default, np.random.choice(range(10000), size=10000));
beta_is.append(beta_i)
beta_bs.append(beta_b)
```

```
         Current function value: 0.070837
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.075469
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.074056
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.080399
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.079418
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.075494
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.076585
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.078997
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.077621
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.077644
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.078250
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.076597
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.082268
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.079406
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.080869
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.075792
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.079341
         Iterations 10
```

```
Optimization terminated successfully.
         Current function value: 0.076661
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.086022
         Iterations 10
Optimization terminated successfully.
         Current function value: 0.083204
         Iterations 10
```

```
beta_i_mean = np.mean(beta_is)
beta_i_se   = np.sqrt(1/(B-1)*np.sum( [(beta_i - beta_i_mean)**2 for beta_i in be

beta_b_mean = np.mean(beta_bs)
beta_b_se   = np.sqrt(1/(B-1)*np.sum( [(beta_b - beta_b_mean)**2 for beta_b in be

print("beta_income  (mean & SE)", beta_i_mean, beta_i_se)
print("beta_balance (mean & SE)", beta_b_mean, beta_b_se)
```

```
beta_income  (mean & SE) 2.016973494321125e-05 5.4383662396369285e-06
beta_balance (mean & SE) 0.005677686072213888 0.00022069873134705593
```

Problem 9 (Chapter 5, Exercise 8)

Problem 9(a)

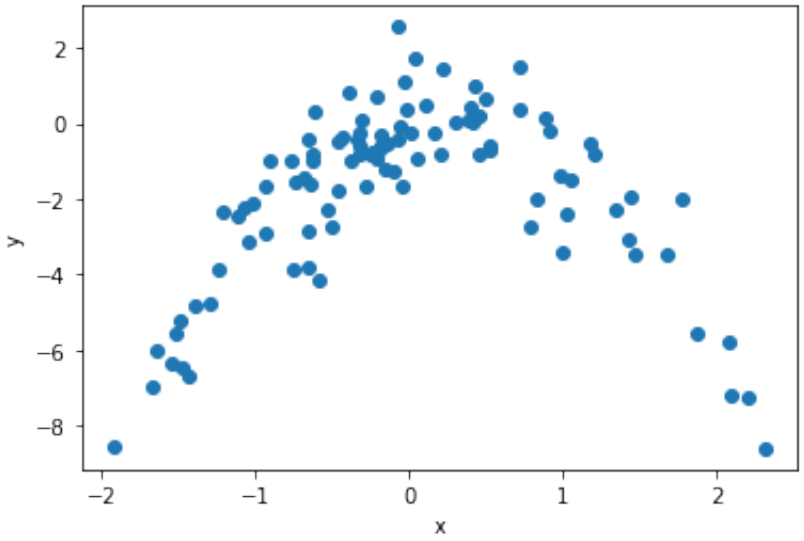
```
data = pd.read_csv("ch5_ex8.csv")
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   x       100 non-null    float64
 1   y       100 non-null    float64
dtypes: float64(2)
memory usage: 1.7 KB
```

Problem 9(b)

```
plt.scatter(data.x, data.y)
plt.xlabel("x")
plt.ylabel("y");
```



Problem 9(c)

```
np.random.seed(1)
```

```
from sklearn.preprocessing import PolynomialFeatures

X = np.array(data['x']).reshape(-1, 1)
y = np.array(data['y']).reshape(-1, 1)

for k in [1, 2, 3, 4]:

  loocv_errors = []

  for i in range(data.shape[0]):

    X_loo  = np.delete(X, i).reshape(-1, 1)
    y_loo  = np.delete(y, i).reshape(-1, 1)

    pf     = PolynomialFeatures(degree=k)
    Xp_loo = pf.fit_transform(X_loo)

    lin_reg = sm.OLS(y_loo, Xp_loo).fit()
    y_pred = lin_reg.predict(pf.transform(X[i].reshape(1,1)))
    loocv_error = (y_pred - y[i])**2

    loocv_errors.append(loocv_error)

  loocv_errors = np.array(loocv_errors)
  print("degree %d LOOCV error:" %k, "%+3.2e" %loocv_errors.mean())
```

```
 degree 1 LOOCV error: +5.89e+00
 degree 2 LOOCV error: +1.09e+00
 degree 3 LOOCV error: +1.10e+00
 degree 4 LOOCV error: +1.11e+00
```

Problem 9(d)

```
np.random.seed(2)
```

```
for k in [1, 2, 3, 4]:

  loocv_errors = []

  for i in range(data.shape[0]):

    X_loo  = np.delete(X, i).reshape(-1, 1)
    y_loo  = np.delete(y, i).reshape(-1, 1)

    pf     = PolynomialFeatures(degree=k)
    Xp_loo = pf.fit_transform(X_loo)

    lin_reg = sm.OLS(y_loo, Xp_loo).fit()
    y_pred = lin_reg.predict(pf.transform(X[i].reshape(1,1)))
    loocv_error = (y_pred - y[i])**2

    loocv_errors.append(loocv_error)

  loocv_errors = np.array(loocv_errors)
  print("degree %d LOOCV error:" %k, "%+3.2e" %loocv_errors.mean())
```

```
degree 1 LOOCV error: +5.89e+00
degree 2 LOOCV error: +1.09e+00
degree 3 LOOCV error: +1.10e+00
degree 4 LOOCV error: +1.11e+00
```

Problem 9(f)

```
X = np.array(data['x']).reshape(-1, 1)
y = np.array(data['y']).reshape(-1, 1)

for k in [1, 2, 3, 4]:

  pf = PolynomialFeatures(degree=k)
  Xp = pf.fit_transform(X)

  lin_reg = sm.OLS(y, Xp).fit()
  print(lin_reg.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.010
Model:                            OLS   Adj. R-squared:                 -0.000
Method:                 Least Squares   F-statistic:                    0.9610
Date:                Fri, 16 Jul 2021   Prob (F-statistic):              0.329
Time:                        19:37:03   Log-Likelihood:                -226.84
No. Observations:                 100   AIC:                             457.7
```

```
No. Observations:                  100   AIC:                             457.
Df Residuals:                       98   BIC:                             462.
Df Model:                            1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.8185      0.236     -7.692      0.000      -2.288      -1.349
x1             0.2430      0.248      0.981      0.329      -0.249       0.735
==============================================================================
Omnibus:                       17.572   Durbin-Watson:                   2.198
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               20.491
Skew:                          -1.051   Prob(JB):                     3.55e-05
Kurtosis:                       3.704   Cond. No.                         1.06
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correct
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.813
Model:                            OLS   Adj. R-squared:                  0.809
Method:                 Least Squares   F-statistic:                     210.6
Date:                Fri, 16 Jul 2021   Prob (F-statistic):           5.10e-36
Time:                        19:37:03   Log-Likelihood:                -143.55
No. Observations:                 100   AIC:                             293.1
Df Residuals:                      97   BIC:                             300.9
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0954      0.133     -0.715      0.476      -0.360       0.169
x1             0.8996      0.113      7.961      0.000       0.675       1.124
x2            -1.8666      0.092    -20.399      0.000      -2.048      -1.685
==============================================================================
Omnibus:                        1.794   Durbin-Watson:                   2.236
Prob(Omnibus):                  0.408   Jarque-Bera (JB):                1.225
Skew:                          -0.183   Prob(JB):                        0.542
Kurtosis:                       3.399   Cond. No.                         2.47
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correct
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.813
Model:                            OLS   Adj. R-squared:                  0.807
Method:                 Least Squares   F-statistic:                     139.1
Date:                Fri, 16 Jul 2021   Prob (F-statistic):           8.04e-35
Time:                        19:37:03   Log-Likelihood:                -143.51
No. Observations:                 100   AIC:                             295.0
Df Residuals:                      96   BIC:                             305.4
```

Problem 10 (Chapter 5, Exercise 9)

```
boston = pd.read_csv("Boston.csv")
```

Problem 10(a)

```
mu_hat = boston['medv'].mean()
print(mu_hat)
```

```
22.532806324110698
```

Problem 10(b)

```
SE_mu_hat = boston['medv'].std()/np.sqrt(boston.shape[0])
print(SE_mu_hat)
```

```
0.4088611474975351
```

Problem 10(c)

Problem 10(d)

Problem 10(e)

Problem 10(f)

Problem 10(g)

Problem 10(h)