COMP540 Statistical Machine Learning

Spring 2020

HW2

Yunhao Zheng (yz157)
Ziqing Dai (zd15)

1. Gradient and Hessian of J() for logistic regression (20 points)

1.

1.1 • $g(z) = \frac{1}{1+e^{-z}}$

$$\frac{\partial g(z)}{\partial z} = -\frac{1}{(1+e^{-z})^2} \cdot (e^{-z})(-1)$$

$$= \frac{e^{-z}}{(1+e^{-z})^2}$$

$$= \left(\frac{1}{1+e^{-z}}\right) \cdot \left(\frac{e^{-z}}{1+e^{-z}}\right)$$

$$= \frac{1}{1+e^{-z}} \left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= g(z) \cdot (1-g(z))$$

1.2 •

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)})) + \frac{\lambda}{2m} \sum_{j=1}^{d} \theta_j^2$$

$$h_\theta = g(\theta^T x) = \frac{1}{1+e^{(-\theta^T x)}}$$

$$\therefore \frac{\partial J(\theta)}{\partial \theta} = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \cdot \frac{e^{(-\theta^T x)}}{1+e^{(-\theta^T x)}} - (1-y^{(i)}) \left(\frac{1}{1+e^{(-\theta^T x)}}\right) \right] x^i + \frac{\lambda}{m} \sum_{j=1}^{d} \theta_j$$

$$= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \frac{(1+e^{(-\theta^T x)})}{1+e^{-\theta^T x}} - \frac{1}{1+e^{(-\theta^T x)}} \right] x^i + \frac{\lambda}{m} \sum_{j=1}^{d} \theta_j$$

$$= -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^i)) x^{(i)} + \frac{\lambda}{m} [0, \theta_1, \theta_2 \ldots \theta_d]^T$$

1.3 $$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T \cdot (h_\theta(x) - y) + \frac{\lambda}{m} \theta^T$$

1.4. $H = \frac{1}{m}(X^T S X + \lambda I)$

$S = \text{diag}(h_\theta(x^{(1)})(1-h_\theta(x^{(1)})), \ldots, h_\theta(x^{(m)})(1-h_\theta(x^{(m)})))$

$$H = \frac{1}{m}(X^T S X) + \frac{1}{m}\lambda I$$

now, we need to proof

$z^T H z > 0$ for any non-zero z.

$$z^T H z = \frac{1}{m} z^T X^T S X z + \frac{1}{m} z^T \lambda I z$$

$$= \frac{1}{m} a^T S a + \frac{1}{m} z^T \lambda I z$$

because $\lambda > 0$, $\frac{1}{m} z^T \lambda I z > 0$.

now, we need to proof $\frac{1}{m} a^T S a > 0$

Because z is nonzero vector and X is full rank

so Xz is non-zero vector.

$$a^T S a = a_1^2 h_\theta(x^{(1)})(1-h_\theta(x^{(1)})) + \dots + a_m^2 h_\theta(x^{(m)})(1-h_\theta$$

$\because a_i^2 > 0$ for $i = 1, 2 \dots m$

since $0 < h_\theta(x^{(i)}) < 1$

$\therefore 1 > 1 - h_\theta(x^{(i)}) > 0$

$\therefore z^T H z > 0$

Therefore, H is postive definite.

1.5 • Newton's method :

$$\theta_{t+1} = \theta_t - H^{-1} \nabla_\theta J(\theta)$$

Python script

```
# newton's method
def newton(theta):
    h = 1/(1 + np.exp(-np.matmul(X, theta)))
#     J = np.sum(-y * np.log(h) - (1-y) * np.log(1-h)) / m
```

```
#     print(J)
    grad = np.zeros((dim,))
    grad[0] = np.sum(X[:, 0] * (h - y), 0) / m
    grad[1:] = (np.sum(X[:, 1:] * (h - y)[:, np.newaxis], 0) + reg * theta[1:])/ m
    S = np.zeros((4, 4))
    np.fill_diagonal(S, h*(1-h))
    Hessian = (np.matmul(np.matmul(X.T, S), X) + reg*np.identity(3))/m
    theta = theta - np.matmul(np.linalg.inv(Hessian), grad.T)
    print(theta)
    return theta

X = np.array([[1, 0, 3],
              [1, 1, 3],
              [1, 0, 1],
              [1, 1, 1]])
y = np.array([1, 1, 0, 0])
theta = np.array([0, -2, 1])
reg = 0.07
m = 4
dim = 3

theta = newton(theta)
theta = newton(theta)
```

Result

theta1 = [-3.15199171 -0.40585887 1.81504991]
theta2 = [-4.26505811 -0.29747087 2.33806757]

2. Overfitting and unregularized logistic regression

Show that for a linearly separable dataset, the maximum likelihood solution for the logistic regression model is obtained by nding a parameter vector whose decision boundary T x = 0 separates the classes and then, by taking the magnitude of to innity. What does this result physically mean? How can we avoid this singular solution?

$$\log P(D|\theta) = \sum_{i=1}^{m} y^{(i)} \log \frac{1}{1+e^{-\theta^T x^{(i)}}} + (1-y^{(i)}) \log\left(1 - \frac{1}{1+e^{-\theta^T x^{(i)}}}\right)$$

$$= -\sum_{i=1}^{m} y^{(i)} \log(1+e^{-\theta^T x^{(i)}}) + (1-y^{(i)}) \log(1+e^{\theta^T x^{(i)}})$$

$$= -\sum_{i=1}^{m} \log(1+e^{-y^{(i)}\theta^T x^{(i)}})$$

Data is linear separable $\Rightarrow$ There is θ to satisfy $y\theta^T x \geq 0$.

To achieve the maximum of log likelihood, we need to achieve the minimum of $1+e^{-y\theta^T x}$, which is the maximum of $y\theta^T x$.

$y\theta^T x = \|y\| \cdot \|\theta\| \|x\| \cos\langle\theta, x\rangle$.

y, x are data. The angle between θ, x determines the hyperplane we found.

$\Rightarrow$ We can take the magnitude of θ to infinity to achieve maximum log likelihood.
It means that after we found the hyperplane to separate data, the ~~lost~~ loss will keep decreasing but we just keep expanding $\|\theta\|$
To ~~avoid~~ avoid this situation: add regularization to our model to keep $\|\theta\|$ from going to infinity.

3. Implementing a k-nearest-neighbor classifier

Problem 3.1 Distance matrix computation with two loops (5 points)

See code in k nearest neighbor.py and knn.ipynb

Problem 3.2 Compute majority label (5 points)

When k=1, accuracy= 0.274000
When k=5, accuracy=0.278000

Problem 3.3 Distance matrix computation with one loop (5 points)

See code in compute distances one loop in k nearest neighbor.py

Problem 3.4 Distance matrix computation with no loops (5 points)

Two loop version took 18.779556 seconds
One loop version took 25.988581 seconds
No loop version took 0.118588 seconds

Problem 3.5 Choosing k by cross validation (5 points)

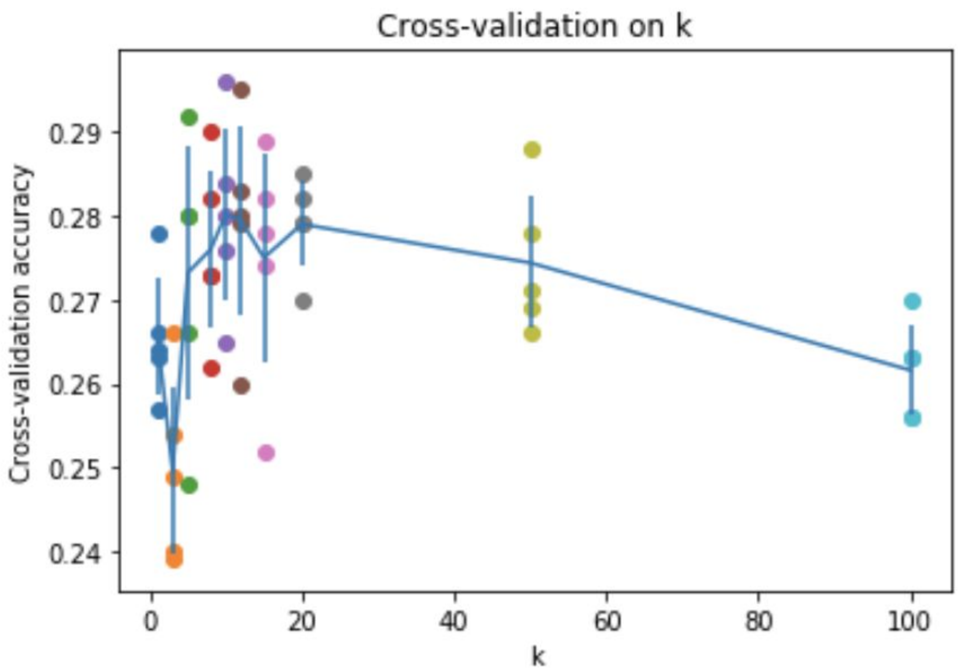
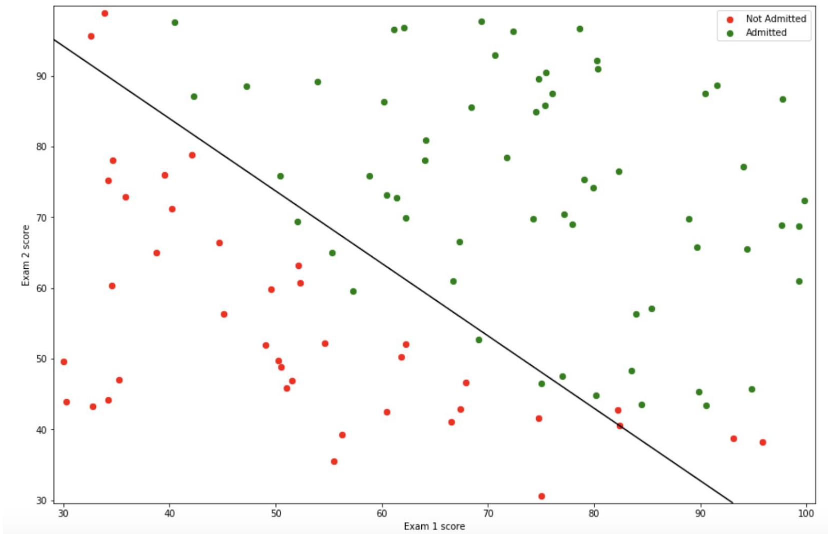


Figure 1. Cross validation on k
Best result K=10,
Got 141 / 500 correct => accuracy: 0.282000

4 Implementing logistic regression (45 points)

Problem 4A1: Implementing logistic regression: the sigmoid function (5 points)

Problem 4A2: Cost function and gradient of logistic regression (5 points)

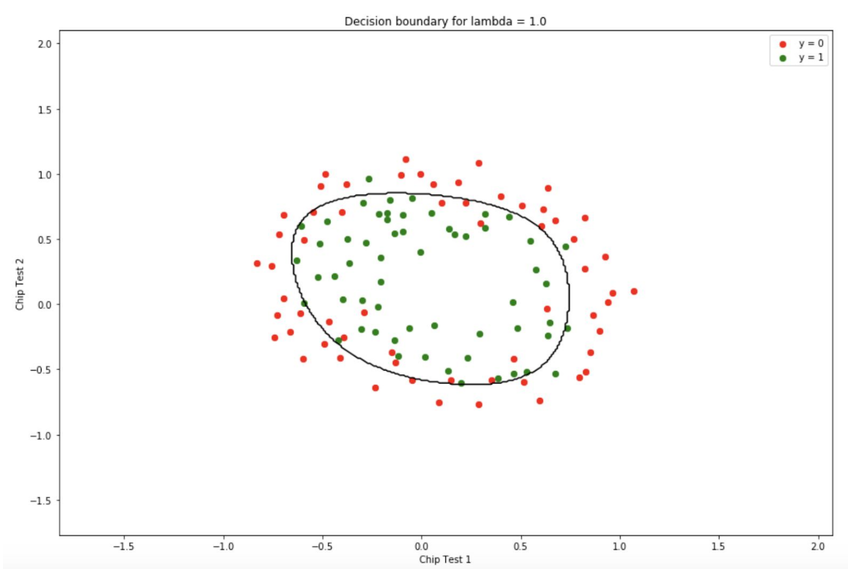


Problem 4A3: Prediction using a logistic regression model (5 points)

Accuracy on the training set = 0.8900

Problem 4, Part B: Regularized logistic regression (20 points)

Problem 4B1: Cost function and gradient for regularized logistic regression (10 points)



Problem 4B2: Prediction using the model (2 points)
Accuracy on the training set = 0.8305

Problem 4B3: Varying λ

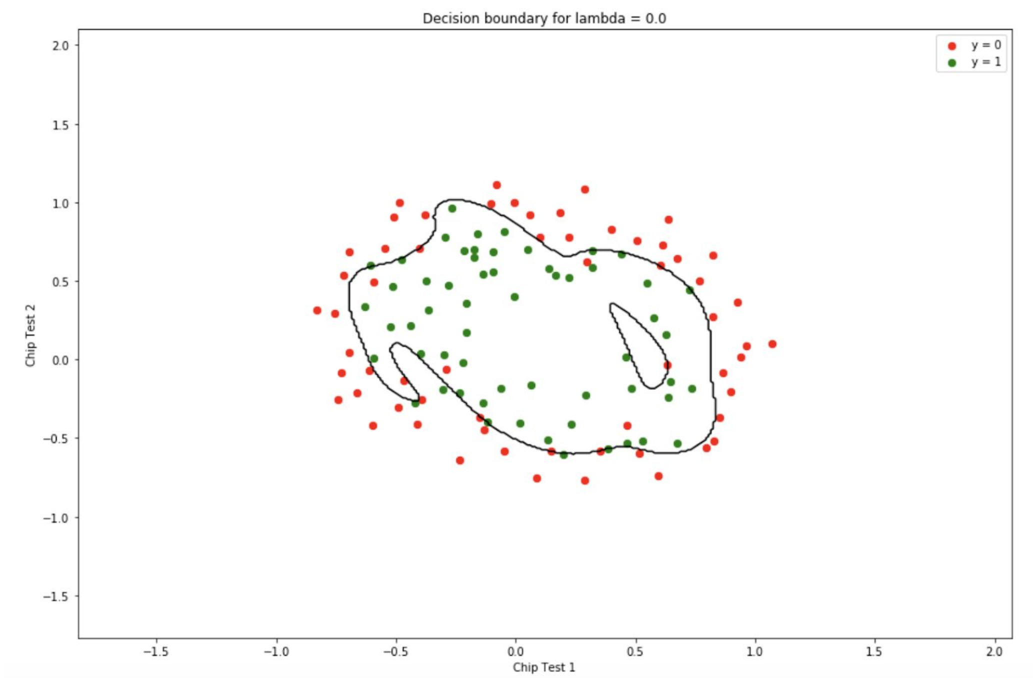


Figure 2. Decision boundary for lambda = 0

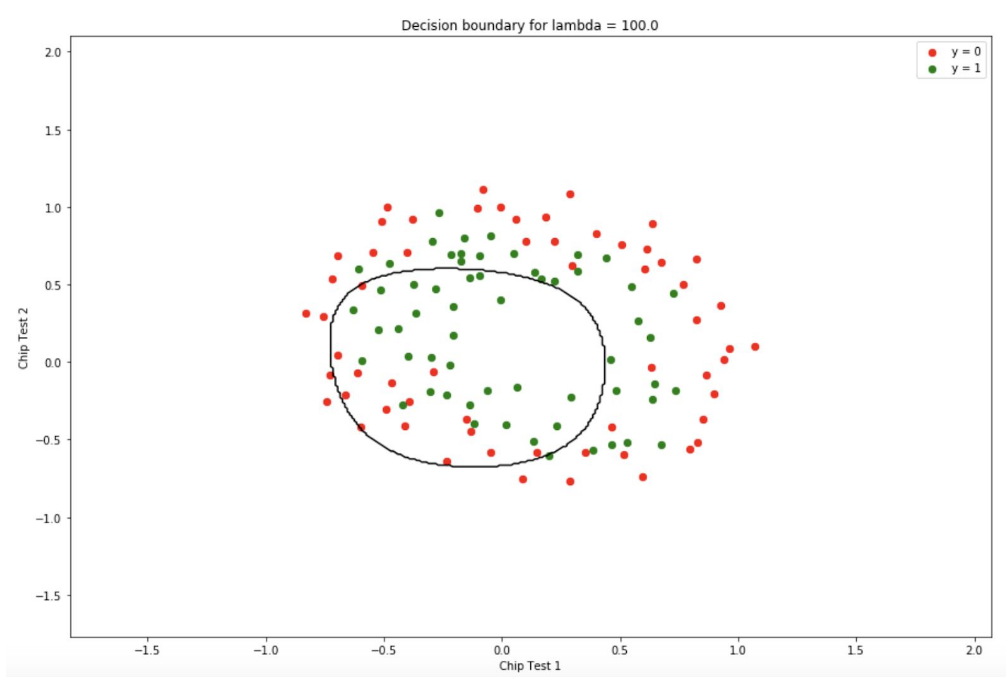
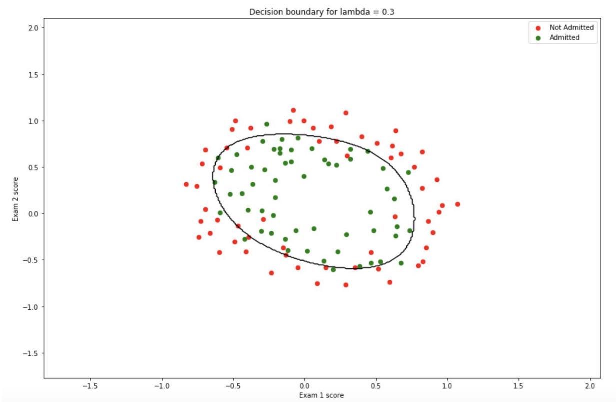
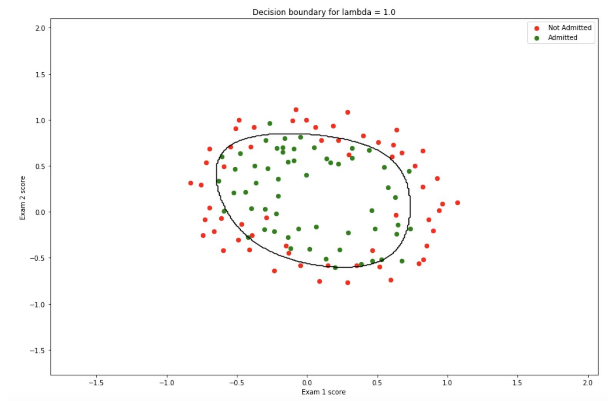


Figure 3. Decision boundary for lambda = 100.0

Problem 4B4: Exploring L1 and L2 penalized logistic regression

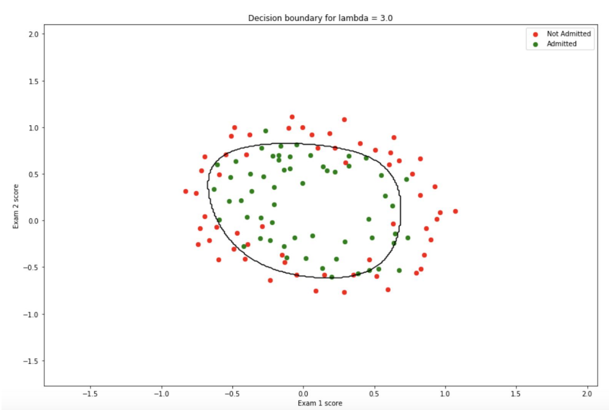
1. L2 regularization

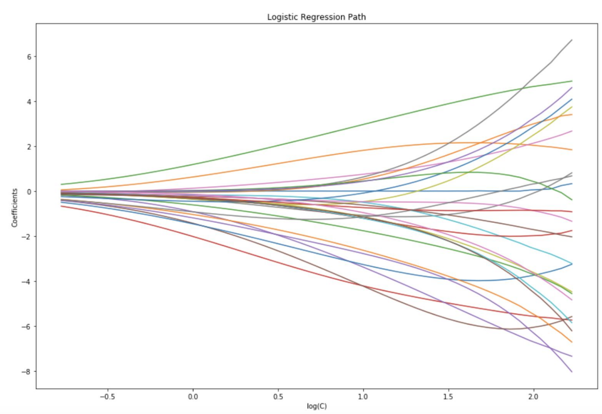




(a) lambda = 0.3: loss = 0.3946
Non-zero coefficients = 28

(b) lambda = 1.0: loss = 0.4684
Non-zero coefficients = 28

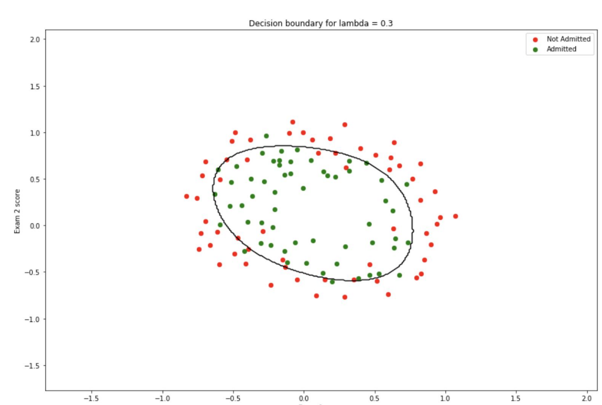


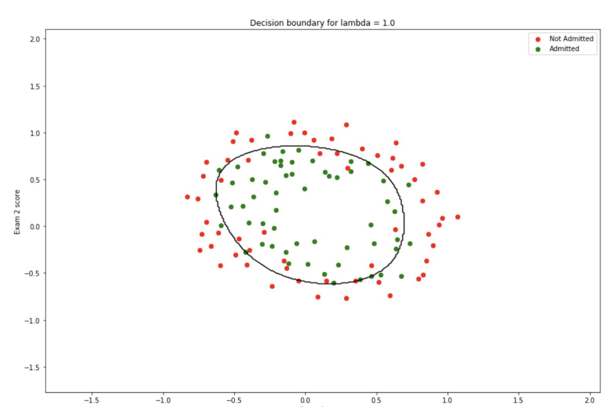


(c) lambda = 3.0: loss = 0.5478
Non-zero coefficients = 28

(d) learning path

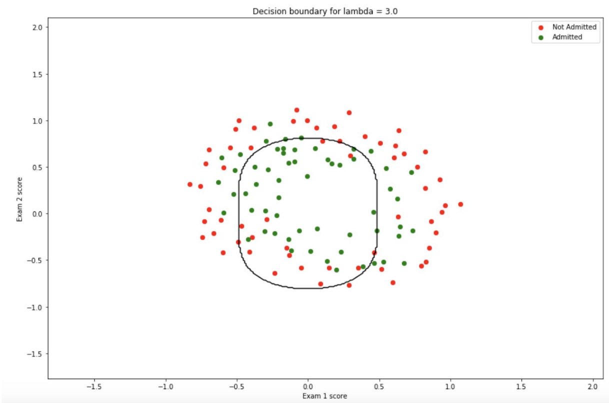
2. L1 regularization



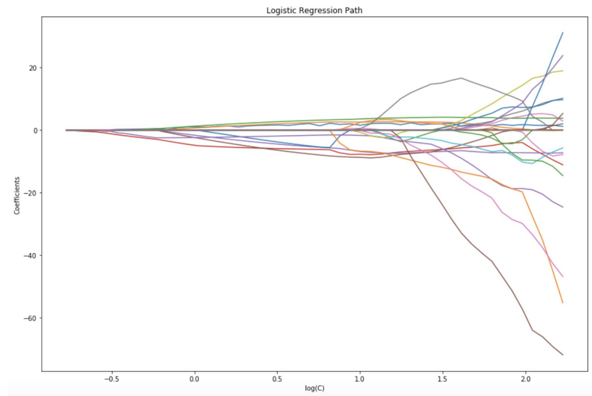


(a) lambda = 0.3: loss = 0.3573
Non-zero coefficients = 8

(b) lambda = 1.0: loss = 0.4381
Non-zero coefficients = 7



(c) lambda = 3.0: loss = 0.6137
Non-zero coefficients = 3



(d) learning path

From the learning path, we see that the coefficients of L1 regularized model shrink faster than that of L2 regularized model as lambda (i.e. 1/C) increases. When lambda is large, the L1 regularization provides larger penalty (loss) than L2 regularization, which results in less non-zero coefficients and a simpler model.

Problem 4 Part C: Logistic regression for spam classification
Fitting regularized logistic regression models (L2 and L1)

L2 penalty

a. Standardize features
   Accuracy = 0.9219
   # Non-zero coefficients = 58
b. Log transform features
   Accuracy = 0.9434
   # Non-zero coefficients = 58
c. Binarize features
   Accuracy = 0.9284
   # Non-zero coefficients = 58

L1 penalty

a. Standardize features
   Accuracy = 0.9225
   # Non-zero coefficients = 52
b. Log transform features

Accuracy = 0.9453
# Non-zero coefficients = 49

c. Binarize features
Accuracy = 0.9284
# Non-zero coefficients = 48

L1 penalty results in more sparse models (model with less non-zero coefficients). I will use model trained by log transform features with L1 penalty because it produces a simpler model with the best accuracy.