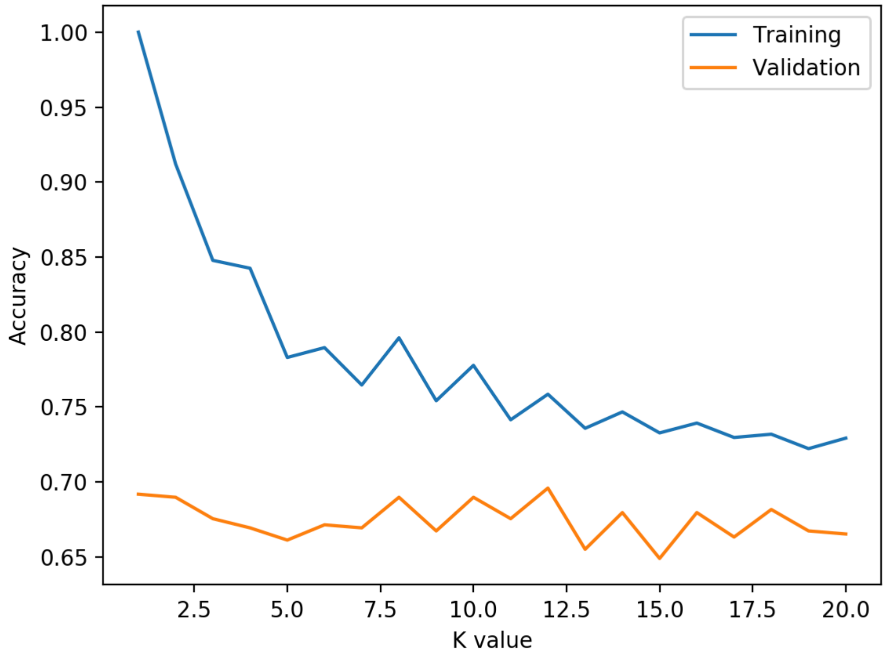
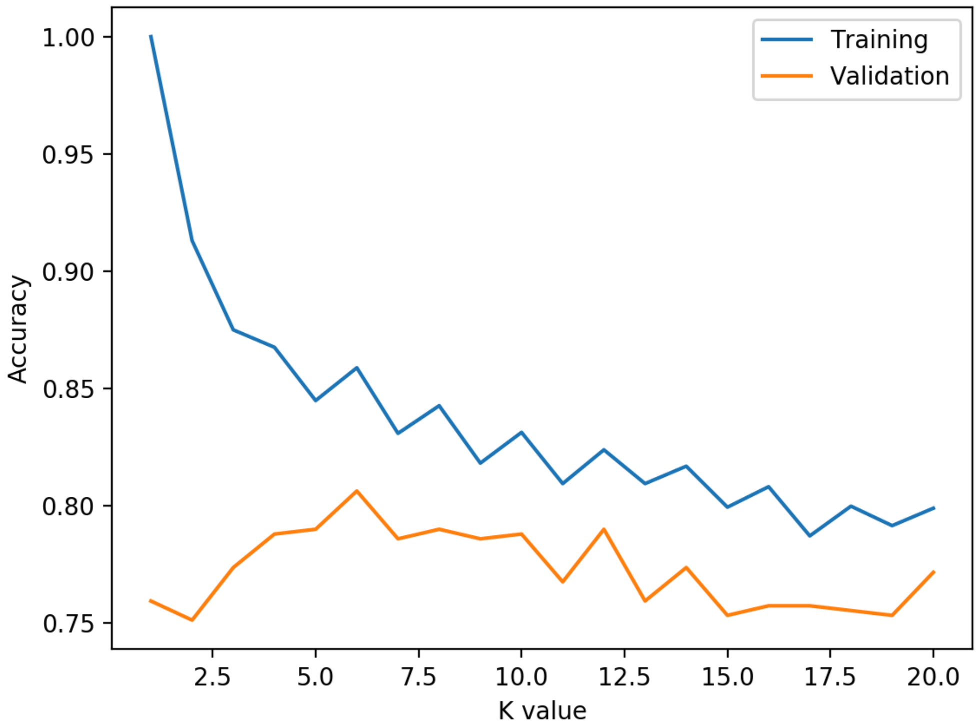
Question 1.

(b).




```
The k value of the best model is: 12. Its accuracy on the test set is 0.6306122448979592.
```

(c).



Claim: the cosine similarity performs better than Euclidean metric in text classification problems.

Consider the example dataset ['cat', 'bulldozer', 'cat cat cat'], which is represented by the following matrix after vectorization.

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 3 & 0 \end{pmatrix}$$

Euclidean metric:

Distance between 'cat', 'bulldozer' $= \sqrt{(1-0)^2 + (0-1)^2} = \sqrt{2}$

Distance between 'cat cat cat', 'cat' $= \sqrt{(3-1)^2 + 0^2} = 2$

Hence 'cat' and 'bulldozer' are closer, which is obviously wrong.

Cosine Similarity:

cos(angle between 'cat'. 'bulldozer') = 0

cos(angle between 'cat'. 'cat cat cat') = 1

The result is correct.

Since the distance between words is large because of the # of it appears, in this case, the cosine similarity can still give a smaller angle between them.

Question 2.
(a).

$$w_j \leftarrow (1 - \alpha\beta_j)w_j - \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})x_j^{(i)}$$

$$b \leftarrow \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})$$

Because we subtract $\alpha \vec{\beta}^T \vec{w}$ every time when we update $\vec{w}$.

Details:

$$\frac{\partial J_{reg}^{\beta}}{\partial w_j} = \frac{\partial}{\partial w_j}\left[\frac{1}{2N}\sum_{i=1}^{N}\left(\sum_{k=1}^{D} w_k x_k^{(i)} + b - t^{(i)}\right)^2\right] + \frac{\partial}{\partial w_j}\left[\frac{1}{2}\sum_{i=1}^{D}\beta_i w_i^2\right]$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - t^{(i)}\right)\cdot\frac{\partial}{\partial w_j}\left[w_j x_j^{(i)}\right] + \frac{\partial}{\partial w_j}\left[\frac{1}{2}\beta_j w_j^2\right]$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})\cdot x_j^{(i)} + \beta_j w_j$$

so $w_j \leftarrow w_j - \alpha\frac{\partial J_{reg}^{\beta}}{\partial w_j}$

$$\leftarrow (1-\alpha\beta_j)w_j - \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})\cdot x_j^{(i)}$$

$$\frac{\partial J_{reg}^{\beta}}{\partial b} = \frac{1}{N}\sum_{i=1}^{1}(y^{(i)} - t^{(i)})$$

$$b \leftarrow b - \alpha\frac{\partial J_{reg}^{\beta}}{\partial b}$$

$$\leftarrow b - \frac{\alpha}{N}\sum_{i=1}^{1}(y^{(i)} - t^{(i)})$$

(b).

By (a).

$$\frac{\partial J_{reg}^{\beta}}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})\cdot x_j^{(i)} + \beta_j w_j$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j'=1}^{D} x_{j'}^{(i)} w_{j'} - t^{(i)}\right)x_j^{(i)} + \beta_j w_j$$

$$= \left(\sum_{j'=1}^{D}\frac{1}{N}\sum_{i=1}^{N} x_{j'}^{(i)} x_j^{(i)} w_{j'}\right) + \beta_j w_j - \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}\cdot t^{(i)} = 0$$

so

$$A_{jj'} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N}\left(x_j^{(i)} x_{j'}^{(i)}\right) + \beta_j & , \text{ if } j=j' \\ \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)} & , \text{ if } j\neq j' \end{cases}$$

$$c_j = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} \cdot t^{(i)}$$

(c). By (b). Clearly:

$$A = \frac{1}{N} X^T X + \operatorname{diag}(\beta_1, \beta_2, \cdots, \beta_D)$$

$$\vec{c} = \frac{1}{N} X^T \vec{t}$$

Since $A\vec{\omega} - \vec{c} = 0$

$$\vec{\omega} = A^{-1}\vec{c}$$

$$= \left[\frac{1}{N}X^T X + \operatorname{diag}(\beta_1, \cdots, \beta_D)\right]^{-1} \cdot \frac{1}{N} X^T \vec{t}$$

$$\vec{\omega} = \left[X^T X + N\operatorname{diag}(\beta_1, \cdots, \beta_D)\right]^{-1} X^T \vec{t}$$

Question 3.

$$y = X\vec{\omega} + b \cdot \vec{1}$$

$$\frac{\partial J}{\partial y} = \frac{\partial}{\partial y}\left[\frac{1}{N}\sum_{i=1}^{N} 1 - \cos(y^{(i)} - t^{(i)})\right]$$

$$= \frac{1}{N}\sin(\vec{y} - \vec{t})$$

$$\frac{\partial J}{\partial \vec{w}} = \frac{\partial J}{\partial \vec{y}} \frac{\partial \vec{y}}{\partial \vec{w}}, \qquad \text{by } \frac{\partial y^{(i)}}{\partial w_j} = x_j^{(i)},$$

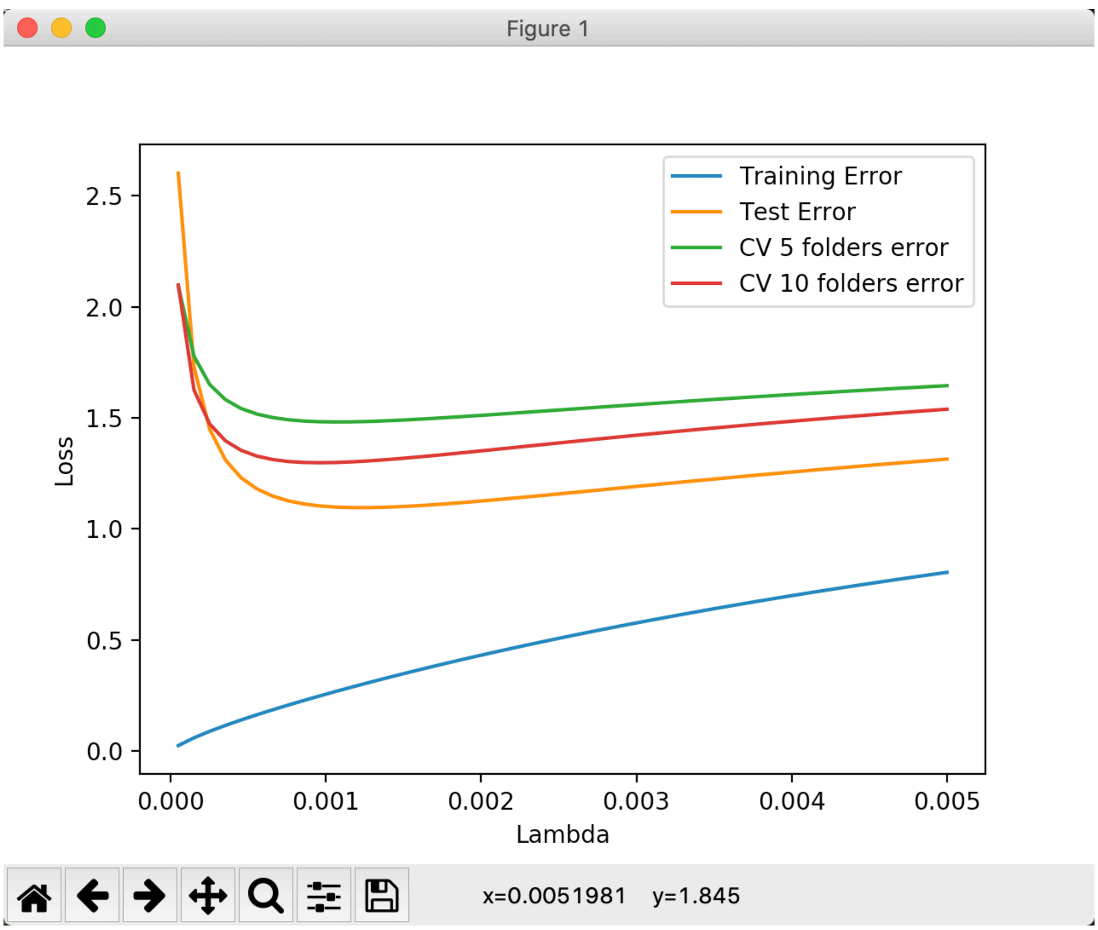
$$= \frac{1}{N} X^T \cdot sm(\vec{y} - \vec{t})$$

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial \vec{y}} \frac{\partial \vec{y}}{\partial b} \qquad \text{by } \frac{\partial y^{(i)}}{\partial b} = 1,$$

$$= \frac{1}{N} \left[sm(\vec{y} - \vec{t})\right]^T \cdot \vec{1}$$

Question 4.
(b).
Order:
Firstly, shuffle_data(data) + split_data(data, num_folders, fold) to preprocess the dataset.
Then train our model by calling train_model(data, lambd).
Next, predict the result of the training set by predict(data, model) and use loss() to compute the loss.

(c) & (d).



```
Best lambda by 5-folder CV: 0.0009591836734693879
Best lambda by 10-folder CV: 0.0010602040816326530
```

The training error increases as the coefficient of L^2 penalty becoming greater.
The cross validation errors reach to their local minimum near lambda = 0.001, which is the proposed parameter.
Notice the initial loss is very big if we set the penalty to 0 (i.e. lambda=0), that's how regularization works.