學號：R05921069 系級： 電機碩一 姓名：黃武昱

1.請說明你實作的 **generative model**， 其訓練方式和準確率為何？

答：

```
def classification(self):

    class_1_numbers = np.where(self.train_data_set[:,-1] == 0)
    class_2_numbers = np.where(self.train_data_set[:,-1] == 1)

    N1 = class_1_numbers[0].size#(19807)
    N2 = class_2_numbers[0].size#(6241)

    class_1_data = np.array([]).reshape(0,int(self.train_data_set[:,:-1].shape[1]))
    class_2_data = np.array([]).reshape(0,int(self.train_data_set[:,:-1].shape[1]))

    for idx in class_1_numbers[0]:
        class_1_data = np.vstack((class_1_data,self.train_data_set[idx,:-1]))#(19807,106)

    for idx2 in class_2_numbers[0]:
        class_2_data = np.vstack((class_2_data,self.train_data_set[idx2,:-1]))#(6241,106)

    mean_u1 = np.array([]).reshape(0,class_1_data.shape[1])
    mean_u2 = np.array([]).reshape(0,class_2_data.shape[1])

    for idx in range(class_1_data.shape[1]):
        m = np.mean(class_1_data[:,idx])
        mean_u1 = np.append(mean_u1,m)

    for idx2 in range(class_2_data.shape[1]):
        m = np.mean(class_2_data[:,idx2])
        mean_u2 = np.append(mean_u2,m)

    mean_u1 = mean_u1.reshape(class_1_data.shape[1],1)#(106,1)
    mean_u2 = mean_u2.reshape(class_2_data.shape[1],1)#(106,1)

    cov1 = np.cov(class_1_data, rowvar = False, bias = True)#(106,106)
    cov2 = np.cov(class_2_data, rowvar = False, bias = True)#(106,106)

    cov_total = ((N1*1.0)/(N1+N2))*cov1+((N2*1.0)/(N1+N2))*cov2#(106,106)

    inverse_cov_total = np.linalg.inv(cov_total)

    w = np.dot((mean_u1-mean_u2).T, inverse_cov_total).T#(106,1)

    b1 = (-1.0/2)*np.dot(np.dot((mean_u1.T), inverse_cov_total), mean_u1)
    b2 = (1.0/2)*np.dot(np.dot((mean_u2.T), inverse_cov_total), mean_u2)
    b = b1+b2+np.log((N1*1.0)/N2)

    return w,b
```

使用全部的特徵，並且分成兩類，分別為 class1 跟 class2，之後再分別算出個別 class 的 mean 跟 covariance matrix，並且假設 covariance matrix 為 class1 的 cov 跟 class2 的 cov 平均分配，最後再運用矩陣乘積算出 weight 跟 bias。

其準確率如右：

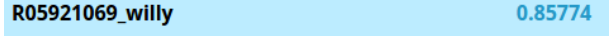
2.請說明你實作的 **discriminative model**， 其訓練方式和準確率為何？

答：

```
w_diff = 0
w2_diff = 0
b_diff = 0

X = self.training_data_set[:,:-1].reshape(self.training_data_set.shape[0],self.training_data_set.shape[1]-1)#(26048,106)

z = X.dot(self.w)+(X**2).dot(self.w2)+self.b#(26048,1)
y = self.sigmoid(z)

Y = self.training_data_set[:,-1].reshape(self.training_data_set.shape[0],1)#(26048,1)

w_diff = -1*np.dot(X.T, (Y-y))+(lamda/y.size)*(self.w.T.dot(self.w)+self.w2.T.dot(self.w2))#(106,1)
w2_diff = -1*np.dot((X**2).T, (Y-y))+(lamda/y.size)*(self.w.T.dot(self.w)+self.w2.T.dot(self.w2))#(106,1)
b_diff = np.sum(-1*(Y-y))

#use Adagrad to improve learning rate
w_diff_total += w_diff**2
w2_diff_total += w2_diff**2
b_diff_total += b_diff**2

ada_w = np.sqrt(w_diff_total)
ada_w2 = np.sqrt(w2_diff_total)
ada_b = np.sqrt(b_diff_total)

ada_w[ada_w == 0] = 1
ada_w2[ada_w2 == 0] = 1
if ada_b == 0: ada_b = 1

self.w -= (self.lr_rate*w_diff)/ada_w
self.w2 -= (self.lr_rate*w2_diff)/ada_w2
self.b -= (self.lr_rate*b_diff)/ada_b
```

先定義我們的 loss function，使用 sigmoid 去限定區間在 0~1 之間，並且運用 gradient descent，根據我們的 interaction 次數去疊加我們的 weight 跟 bias。

其準確率如右：



3.請實作輸入特徵標準化(feature normalization)，並討論其對於你的模型準確率的影響。

答：

```
def feature_normalize(x_train_data, x_test_data):
    # feature normalization with all X
    X_train = x_train_data#(32561,106)
    X_test = x_test_data#(16281,106)

    X_all = np.concatenate((X_train, X_test))#(48842,106)
    mean_value = np.mean(X_all, axis = 0)#(106,)
    SD_value = np.std(X_all, axis = 0)#(106,)

    # only apply normalization on continuos attribute
    feature = [0, 1, 3, 4, 5]
    m = np.zeros(shape = X_all.shape[1])
    s = np.ones(shape = X_all.shape[1])

    m[feature] = mean_value[feature]
    s[feature] = SD_value[feature]

    X_all_normalize = (X_all-m)/s#(48842,106)

    X_train_normalize = X_all_normalize[:X_train.shape[0], :]#(32561,106)
    X_test_normalize = X_all_normalize[X_train.shape[0]:, :]#(16281,106)

    return X_train_normalize, X_test_normalize
```

我只對超過 1 的特徵做 feature normalization

沒做 feature normalization，其準確度：

```
The valid error value is 0.794411
The train error value is 0.787162
```

做 feature normalization，其準確度：

```
The valid error value is 0.856441
The train error value is 0.850852
```

很明顯的可以看出，有 feature normalization 的準確率較高。

4. 請實作 logistic regression 的正規化(regularization)，並討論其對於你的模型準確率的影響。

答：

```
w_diff = -1*np.dot(X.T, (Y-y))+(lamda/y.size)*(self.w.T.dot(self.w)+self.w2.T.dot(self.w2))#(106,1)
w2_diff = -1*np.dot((X**2).T, (Y-y))+(lamda/y.size)*(self.w.T.dot(self.w)+self.w2.T.dot(self.w2))#(106,1)
```

當 lamda=10 時，其準確度為：

```
The valid error value is 0.856441
The train error value is 0.850852
```

當 lamda=100 時，其準確度為：

```
The valid error value is 0.846461
The train error value is 0.848587
```

當 lamda=1000 時，其準確度為：

```
The valid error value is 0.847382
The train error value is 0.845593
```

Regularization 能使 model 防止 overfit，但卻會使準確率下降。

5.請討論你認為哪個 attribute 對結果影響最大？

答：

所有特徵中的 Marital 對於準確率的影響算是蠻大的，雖然婚姻的部分會跟經濟有關，但影響也沒有很大，故如果把這項消除，對於預測經濟是否大於 50 萬，就有很大的差別了。

```
The iterations times: 1000
The valid error value is 0.848918
The train error value is 0.853847
```

此結果為有加 Marital

```
The iterations times: 1000
The valid error value is 0.850760
The train error value is 0.853847
```

此結果為沒加 Marital

很明顯得到，沒有 Marital 可以使精準度上升。