程式碼解釋:

E94084032 林耕澤

Import 的 module 或者是 liberary:

```
import random
import pylab
import math
import numpy as np
import sklearn.linear_model #https://scikit-learn.org/stable/
                            #https://en.wikipedia.org/wiki/Logistic_regression
```

常用函式--第一部分:

變異數以及標準差。

```
def variance(X):
    """Assumes that X is a list of numbers.
       Returns the standard deviation of X"""
    mean = sum(X)/len(X)
    tot = 0.0
    for x in X:
        tot += (x - mean)**2
    return tot/len(X)
def stdDev(X):
    """Assumes that X is a list of numbers.
       Returns the standard deviation of X"""
    return variance(X)**0.5
```

常用函式--第二部分:

buildpassagerExamples:建立 example 的函式:先呼叫 getData2，在建立物件，把特徵及標籤及合成物件，把所有物件都變成 list 回傳。

```
def buildpassengerExamples2(fileName):

    data  = getData2(fileName)
    examples = []
    for i in range(len(data['survived'])):
        a = Passenger2(data['c1'][i],data['c2'][i], data['c3'][i],  data['age'][i],data['gender'][i],data['survived'][i])
        examples.append(a)
    return examples

def makeHist(data, bins, title, xLabel, yLabel):
    mean = sum(data)/len(data)
    std = stdDev(data)
    pylab.hist(data, bins, edgecolor='black',label ='Maximum Accuracies'+"\n"+"mean="+str(round(mean,2))+" SD="+str(round(std,2)))
    pylab.title(title)
    pylab.xlabel(xLabel)
    pylab.ylabel(yLabel)
    pylab.legend()
    #pylab.annotate('Mean = ' + str(round(mean, 2)) +              '\nSD = ' + str(round(std, 2)), fontsize = 20,
    #          xy = (0.50, 0.75), xycoords = 'axes fraction')

def makeHist2(data, bins, title, xLabel, yLabel):
    mean = sum(data)/len(data)
    std = stdDev(data)
    pylab.hist(data, bins, edgecolor='black',label ='k Values for Maximum Accuracies'+"\n"+"mean="+str(round(mean,2))+" SD="+str(round(std,2)
    pylab.title(title)
    pylab.xlabel(xLabel)
    pylab.ylabel(yLabel)
    pylab.legend()
```

而 makeHist 及 makeHist2 是用來畫後面的圖，分別為 maximum accuracy 及 k Values for Maximum Accuracies 圖。

Passengers 的 calss:

我把 c1 c2 c3 分別當作一個 features，方便後面題目使用。

```python
class Passenger2(object):
    def __init__ (self,c1,c2,c3, age, gender, survived):
        self.featureVec = (c1,c2,c3, age, gender)
        self.label = survived

    def featureDist(self, other):
        dist = 0.0
        for i in range(len(self.featureVec)):
            dist += abs(self.featureVec[i] - other.featureVec[i])**2
        return dist**0.5

    def cosine_similarity(self,other):
    #compute cosine similarity of v1 to v2: (v1 dot v2)/{||v1||*||v2||)
        sumxx, sumxy, sumyy = 0, 0, 0
        for i in range(len(self.featureVec)):
            x = self.featureVec[i]; y = other.featureVec[i]
            sumxx += x*x
            sumyy += y*y
            sumxy += x*y
        return sumxy/math.sqrt(sumxx*sumyy)

    def getc1(self):
        return self.featureVec[0]

    def getc2(self):
        return self.featureVec[1]

    def getc3(self):
        return self.featureVec[2]

    def getage(self):
        return self.featureVec[3]

    def getgender(self):
        return self.featureVec[4]


    def getlabel(self):
        return self.label

    def getFeatures(self):
        return self.featureVec
```

再來是，
常用函式—第三部分: getData2
讀取 TXT.檔的資料，回傳分類好的 list。

```
def getData2(filename):

    data = {}
    f = open(filename)
    next(f)
    line = f.readline()
    data['c1'], data['c2'], data['c3'] = [], [], []
    data['age'], data['gender'] = [], []
    data['last name'],data['name'] = [], []
    data['survived']  = []

    while line != '':
        split = line.split(',')
        if int(split[0]) == 1:
            data['c1'].append(1)
            data['c2'].append(0)
            data['c3'].append(0)
        elif int(split[0]) == 2:
            data['c1'].append(0)
            data['c2'].append(1)
            data['c3'].append(0)
        elif int(split[0]) == 3:
            data['c1'].append(0)
            data['c2'].append(0)
            data['c3'].append(1)
        data['age'].append(float(split[1]))

        if split[2] == 'M':
            data['gender'].append(int(1))

        elif split[2] == 'F':
            data['gender'].append(int(0))

        data['survived'].append(int(split[3]))
        data['last name'].append(str(split[4]))
        data['name'].append(str(split[5][:-1]))
       #remove \n
        line = f.readline()
    f.close()
    return data
```

題目第一部分:

此函式大致與前面的 buildpassagerdata 差不多，只在於本函式分會把男女資料分開。

```
def sepratepassengerExamples(fileName):
    data  = getData2(fileName)
    examples_F = []
    examples_M = []
    for i in range(len(data['survived'])):
        a = Passenger2(data['c1'][i],data['c2'][i], data['c3'][i],  data['age'][i],data['gender'][i],data['survived'][i])
        if a.getgender() == 1:
            examples_M.append(a)
        else:
            examples_F.append(a)
    return examples_M,examples_F
```

接下來就是把男女資料分開，然後分別去算各性別各 cabins 生存人數。

```
m,f=sepratepassengerExamples('TitanicPassengers.txt')

Mcabin1 = 0
Mcabin1_sur = 0
Mcabin2 = 0
Mcabin2_sur = 0
Mcabin3 = 0
Mcabin3_sur = 0

for i in m:
    if i.getc1() == 1:
        Mcabin1+=1
        if i.getlabel() ==1:
            Mcabin1_sur+=1

    elif i.getc2() == 1:
        Mcabin2 +=1
        if i.getlabel() ==1:
             Mcabin2_sur+=1
    else:
        Mcabin3 +=1
        if i.getlabel() ==1:
            Mcabin3_sur+=1
```

```python
Fcabin1 = 0
Fcabin1_sur = 0
Fcabin2 = 0
Fcabin2_sur = 0
Fcabin3 = 0
Fcabin3_sur = 0

for i in f:
    if i.getc1() == 1:
        Fcabin1+=1
        if i.getlabel() ==1:
            Fcabin1_sur+=1

    elif i.getc2() == 1:
        Fcabin2 +=1
        if i.getlabel() ==1:
             Fcabin2_sur+=1
    else:
        Fcabin3 +=1
        if i.getlabel() ==1:
            Fcabin3_sur+=1
```

最後把結果印出來:

```python
print("_________part1_________")
print("---------male---------")
print("male in cabin 1:",Mcabin1)
print("male in cabin 2:",Mcabin2)
print("male in cabin 3:",Mcabin3)
print("male in cabin 1 survived:",Mcabin1_sur)
print("male in cabin 2 survived:",Mcabin2_sur)
print("male in cabin 3 survived:",Mcabin3_sur)
print("---------female---------")
print("female in cabin 1:",Fcabin1)
print("female in cabin 2:",Fcabin2)
print("female in cabin 3:",Fcabin3)
print("female in cabin 1 survived:",Fcabin1_sur)
print("female in cabin 2 survived:",Fcabin2_sur)
print("female in cabin 3 survived:",Fcabin3_sur)
print("")
```

輸出結果:

```
_________part1_________
---------male----------
male in cabin 1: 151
male in cabin 2: 158
male in cabin 3: 349
male in cabin 1 survived: 53
male in cabin 2 survived: 23
male in cabin 3 survived: 59
---------female----------
female in cabin 1: 133
female in cabin 2: 103
female in cabin 3: 152
female in cabin 1 survived: 128
female in cabin 2 survived: 92
female in cabin 3 survived: 72
```

題目第二部分:

Seprate_M_F:統計男女分別年齡分布。

choose_servived:統計男女生存下來分別年齡分布。

```python
def seprate_M_F(examples):
    Male_age = []
    Female_age = []
    for e in examples:
        if e.getgender() == 1:
            Male_age.append(e.getage())
        elif e.getgender() == 0:
            Female_age.append(e.getage())
    return Male_age , Female_age

survived_male_age = []

survived_female_age = []

def choose_servived(examples):
    survived_age = []
    for e in examples:
        if e.getlabel()== 1 and e.getgender() == 1:
            survived_male_age.append(e.getage())
        elif e.getlabel()== 1 and e.getgender() == 0:
            survived_female_age.append(e.getage())

    return survived_male_age,survived_female_age
```

choose_cabin:分別把在不同 cabin 的男女，以及生存下來的男女，分別記錄。

```
def choose_cabin(examples):
    Male_cabin =[]
    Male_cabin_sur =[]
    #Male_carbin3 =[]
    Female_cabin =[]
    Female_cabin_sur =[]
    #Female_carbin3 =[]
    for e in examples:
        if e.getgender() == 1:
            if e.getc1() == 1:
                Male_cabin.append(int(1))
            elif e.getc2() == 1:
                Male_cabin.append(int(2))
            elif e.getc3() == 1:
                Male_cabin.append(int(3))
            if e.getlabel() ==1:
                if e.getc1() == 1:
                    Male_cabin_sur.append(int(1))
                elif e.getc2() == 1:
                    Male_cabin_sur.append(int(2))
                elif e.getc3() == 1:
                    Male_cabin_sur.append(int(3))
        else:
            if e.getc1() == 1:
                Female_cabin.append(int(1))
            elif e.getc2() == 1:
                Female_cabin.append(int(2))
            elif e.getc3() == 1:
                Female_cabin.append(int(3))
            if e.getlabel() == 1:
                if e.getc1() == 1:
                    Female_cabin_sur.append(int(1))
                elif e.getc2() == 1:
                    Female_cabin_sur.append(int(2))
                elif e.getc3() == 1:
                    Female_cabin_sur.append(int(3))

    return Male_cabin  ,Male_cabin_sur ,Female_cabin ,Female_cabin_sur
```

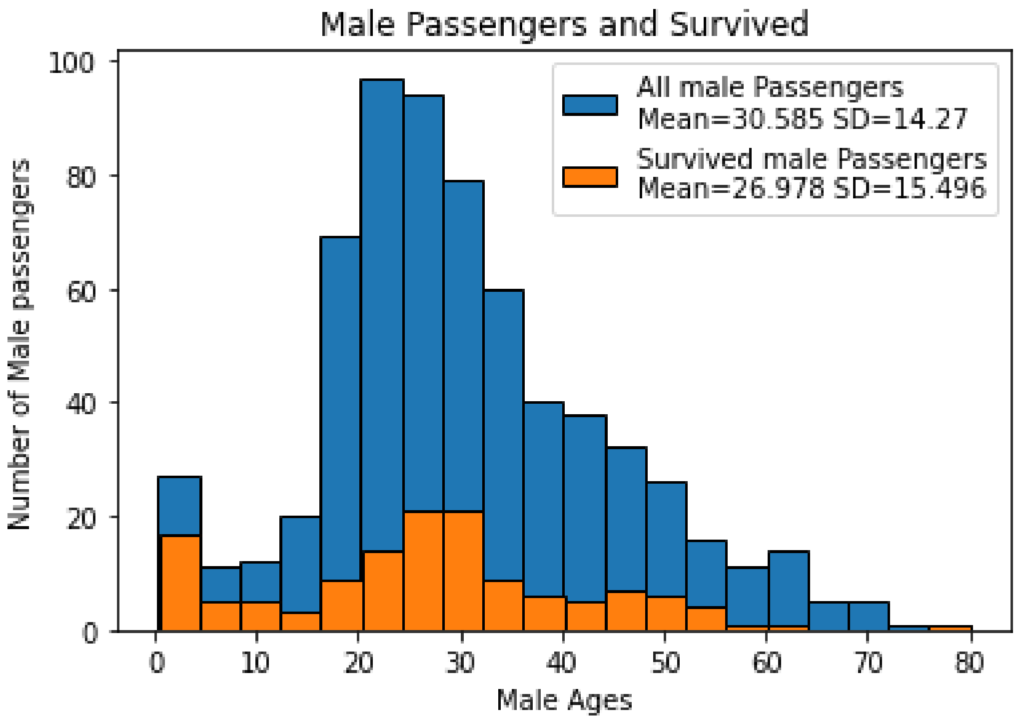
x 為物件 list，存放每一個物件的屬性以及標籤。
將資料男生女生的年齡分別統計，再統計分別生存下來的。

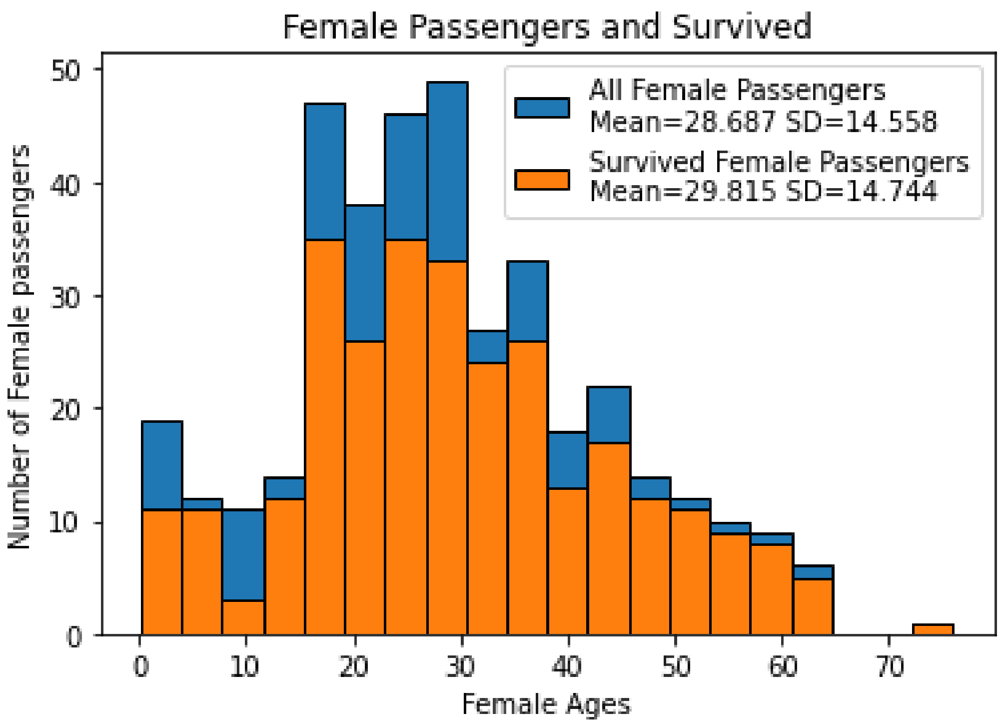
```
x=buildpassengerExamples2('TitanicPassengers.txt')
Male_age , Female_age =seprate_M_F(x)
servived_male_age,servived_female_age = choose_servived(x)
```

印出年齡分布:(程式碼較長，字體較小不好意思)

```python
pylab.hist(Male_age,20,label ='All male Passengers'+'\n'+'Mean='+ str(round(sum(Male_age)/len(Male_age),3)) + " SD="+str(round(stdDev(Male_age),3)), edgecolor='black')
pylab.hist(servived_male_age,20,label ='Survived male Passengers'+'\n'+'Mean='+ str(round(sum( servived_male_age)/len( servived_male_age),3)) + " SD="+str(round(stdDev( servived_male_age),3)), edgecolor='black')
pylab.title("Male Passengers and Survived")
pylab.xlabel("Male Ages")
pylab.ylabel("Number of Male passengers")
pylab.legend()
pylab.show()

#makeHist(Female_age,20,"","","")
#makeHist(servived_female_age ,20,"","","")
pylab.show()
pylab.hist(Female_age,20,label ='All Female Passengers'+'\n'+'Mean='+ str(round(sum(Female_age)/len(Female_age),3)) + " SD="+str(round(stdDev(Female_age),3)), edgecolor='black')
pylab.hist(servived_female_age,20,label ='Survived Female Passengers'+'\n'+'Mean='+ str(round(sum(servived_female_age)/len(servived_female_age),3)) + " SD="+str(round(stdDev( servived_female_age),3)), edgecolor='black')
pylab.title("Female Passengers and Survived")
pylab.xlabel("Female Ages")
pylab.ylabel("Number of Female passengers")
pylab.legend()
pylab.show()
```

結果:

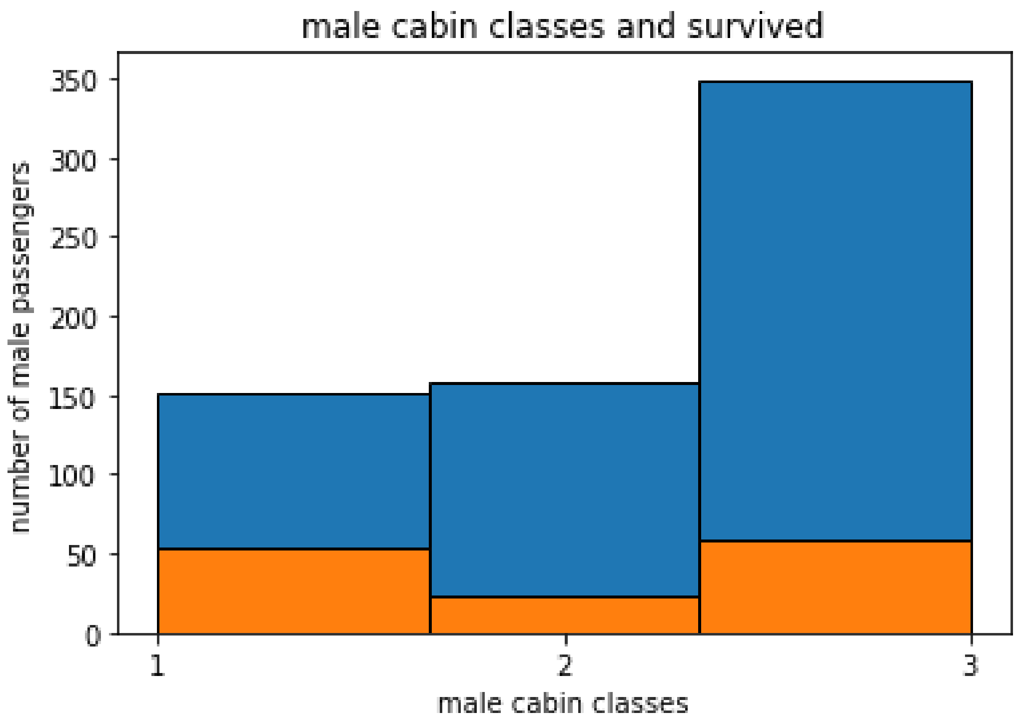


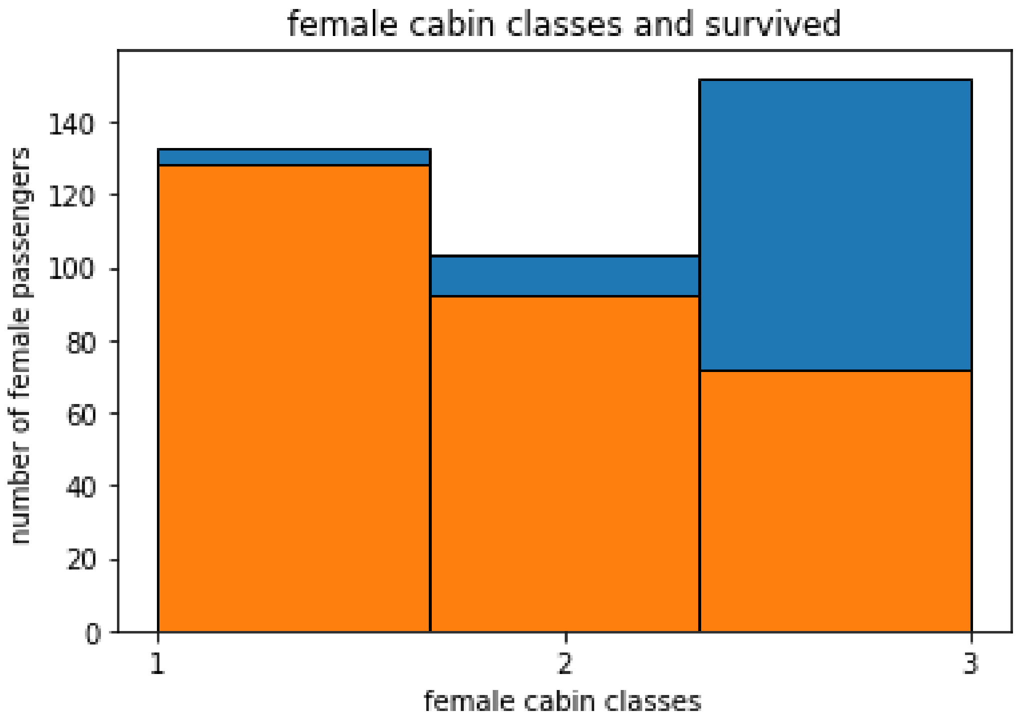


印出不同 cabin 男女人數及生存人數:

```
Male_carbin  ,Male_carbin_ser ,Female_carbin ,Female_carbin_ser=choose_cabin(x)


pylab.hist(Male_carbin,3,range=[1,3], edgecolor='black')
pylab.hist(Male_carbin_ser,3,range=[1,3], edgecolor='black')
pylab.title('male cabin classes and survived')
pylab.xlabel('male cabin classes')
pylab.ylabel('number of male passengers')
pylab.xticks([1,2,3])
#pylab.hist(Male_carbin_ser, edgecolor='black')
pylab.show()


pylab.hist(Female_carbin,3,range=[1,3], edgecolor='black')
pylab.hist(Female_carbin_ser,3,range=[1,3], edgecolor='black')
pylab.title('female cabin classes and survived')
pylab.xlabel('female cabin classes')
pylab.ylabel('number of female passengers')
pylab.xticks([1,2,3])
#pylab.hist(Male_carbin_ser, edgecolor='black')
pylab.show()
```

結果:




female cabin classes and survived
number of female passengers
0
20
40
60
80
100
120
140
1
2
3
female cabin classes

題目第三部分:

常用函式:

applyModel:根據帶入 model 測試集……，去計算真陽性、假陽性、真陰性、假陰性。

getstats:ep 根據真陽性、假陽性、真陰性、假陰性、計算 accuracy、sensitivity、specificity、pos.pred.val.。

devide80_20_1000:分割資料 80 趴 20 趴，分為訓練集集測試集，(後面多取名 1000 原因為後面要切割 1000 次)。

```python
def applyModel(model, testSet, label, prob = 0.5):
    #Create vector containing feature vectors for all test examples
    testFeatureVecs = [e.getFeatures() for e in testSet]
    probs = model.predict_proba(testFeatureVecs)
    truePos, falsePos, trueNeg, falseNeg = 0, 0, 0, 0
    for i in range(len(probs)):
        if probs[i][1] > prob:
            if testSet[i].getlabel() == label:
                truePos += 1
            else:
                falsePos += 1
        else:
            if testSet[i].getlabel() != label:
                trueNeg += 1
            else:
                falseNeg += 1
    return truePos, falsePos, trueNeg, falseNeg


def getStats(truePos, falsePos, trueNeg, falseNeg, toPrint = False):

    accur = accuracy(truePos, falsePos, trueNeg, falseNeg)
    sens = sensitivity(truePos, falseNeg)
    spec = specificity(trueNeg, falsePos)
    ppv = posPredVal(truePos, falsePos)


    if toPrint:
        print(' Accuracy =', round(accur, 3))
        print(' Sensitivity =', round(sens, 3))
        print(' Specificity =', round(spec, 3))
        print(' Pos. Pred. Val. =', round(ppv, 3))
    return accur,sens,spec,ppv

def divide80_20_1000(examples):
    sampleIndices = random.sample(range(len(examples)), len(examples)//5)
    trainingSet, testSet = [], []
    for i in range(len(examples)):
        if i in sampleIndices:
            testSet.append(examples[i])
        else: trainingSet.append(examples[i])
    return trainingSet, testSet
```

accuracy、sensitivity、specificity、pos.pred.val.算法以及 ROC 算法。

```python
def accuracy(truePos, falsePos, trueNeg, falseNeg):
    numerator = truePos + trueNeg
    denominator = truePos + trueNeg + falsePos + falseNeg
    return numerator/denominator
def sensitivity(truePos, falseNeg):
    try:
        return truePos/(truePos + falseNeg)
    except ZeroDivisionError:
        return float('nan')
def specificity(trueNeg, falsePos):
    try:
        return trueNeg/(trueNeg + falsePos)
    except ZeroDivisionError:
        return float('nan')
def posPredVal(truePos, falsePos):
    try:
        return truePos/(truePos + falsePos)
    except ZeroDivisionError:
        return float('nan')
def negPredVal(trueNeg, falseNeg):
    try:
        return trueNeg/(trueNeg + falseNeg)
    except ZeroDivisionError:
        return float('nan')


def buildROC(model, testSet, label, title, plot = True):
    xVals, yVals = [], []
    p = 0.0
    while p <= 1.0:
        truePos, falsePos, trueNeg, falseNeg =\
                               applyModel(model, testSet, label, p)
        xVals.append(1.0 - specificity(trueNeg, falsePos))
        yVals.append(sensitivity(truePos, falseNeg))
        p += 0.01
    auroc = sklearn.metrics.auc(xVals, yVals)
    if plot:
        pylab.plot(xVals, yVals)
        pylab.plot([0,1], [0,1,], '--')
        pylab.title(title +  ' (AUROC = '\
                    + str(round(auroc, 3)) + ')')
        pylab.xlabel('1 - Specificity')
        pylab.ylabel('Sensitivity')
    return auroc
```

第一行的 Accr……及 roc 為用來儲存各數值，每一次分割後的值。
第二行的式用來存每一次的 weight。
MaxK、MaxA 用來儲存最大 K 出現的 accuracy 及最大 accuracy 出現的 K。
mean_accur 為後來畫圖要用。
接下來就是執行 1000 次。

```python
print("__________________part 3__________________")
Accr,Sens,Spec,Ppv = [],[],[],[]
c1_w,c2_w,c3_w,age_w,male_w = [],[],[],[],[]
#age = []
roc = []
MaxK = []
MaxA = []
mean_accur = [None]*201
for i in range (1000):
    trainset , testset = divide80_20_1000(x)
    featureVecs, labels = [], []

    for e in trainset:
        featureVecs.append([e.getc1(),e.getc2(),e.getc3(), e.getage() ,e.getgender()])
        labels.append(e.getlabel())
    model = sklearn.linear_model.LogisticRegression(solver='lbfgs', max_iter=1000).fit(featureVecs, labels)
    c1_w.append(model.coef_[0][0])
    c2_w.append(model.coef_[0][1])
    c3_w.append(model.coef_[0][2])
    age_w.append(model.coef_[0][3])
    male_w.append(model.coef_[0][4])
    ROC=buildROC(model, testset,1, '', plot =False)
    roc.append(ROC)



    truePos, falsePos, trueNeg, falseNeg = applyModel(model, testset, 1, 0.5)
    accur,sens,spec,ppv=getStats(truePos, falsePos, trueNeg, falseNeg)
    Accr.append(accur)
    Sens.append(sens)
    Spec.append(spec)
    Ppv.append(ppv)
    allAccuracy, alltruePos, allfalsePos, alltrueNeg, allfalseNeg= [],[],[],[],[]
    count=0
    maxAccuracy=0
    maxcount=0
    maxk=0
```

此為 0.4~0.601 的 K 值找最大 mean accuracy(也就是為什麼前面
  mean_accuracy
要是大小 201 的 list)

```python
    for k in range(400, 601, 1):
        truePos, falsePos, trueNeg, falseNeg = applyModel(model, testset,1, k/1000)

        alltruePos.append(truePos)
        allfalsePos.append(falsePos)
        alltrueNeg.append(trueNeg)
        allfalseNeg.append(falseNeg)
        accur=accuracy(truePos, falsePos, trueNeg, falseNeg)
        if mean_accur[k-400] == None:
            mean_accur[k-400]=accur/1000
        else:
            mean_accur[k-400]+=accur/1000


        allAccuracy.append(accur)
        if maxAccuracy < accur:
            maxAccuracy = accur
            maxcount=count
            maxk=k/1000
        count+=1
    MAX=max(mean_accur)
    indexk =(mean_accur.index(MAX)+400)/1000
    MaxK.append(maxk)
    MaxA.append(maxAccuracy)
```

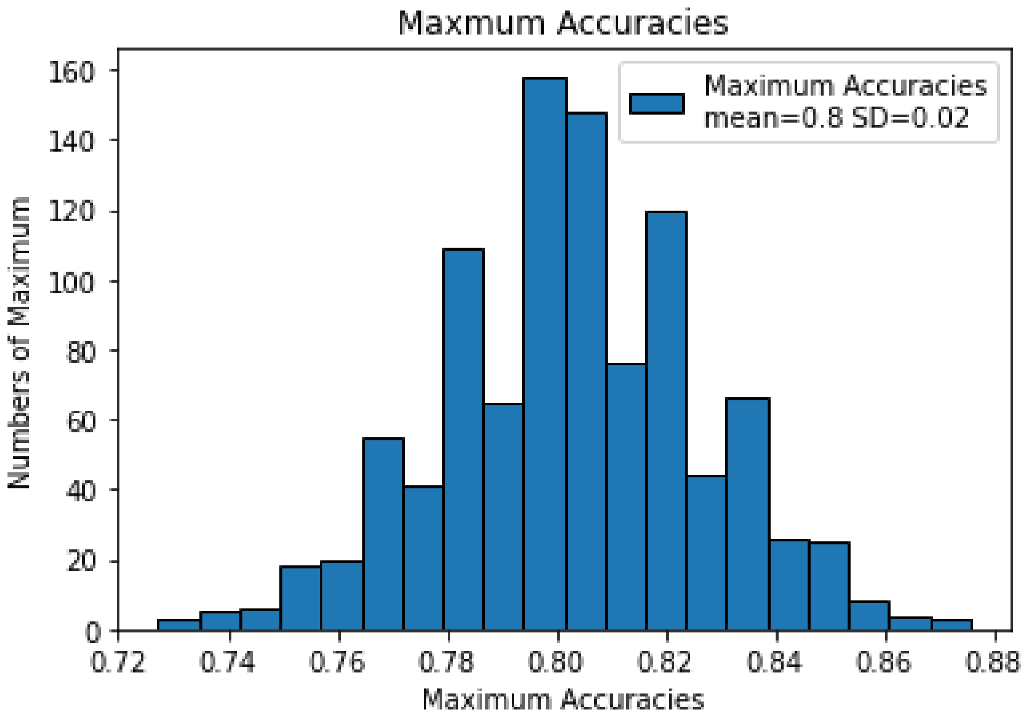
印出結果與圖片:

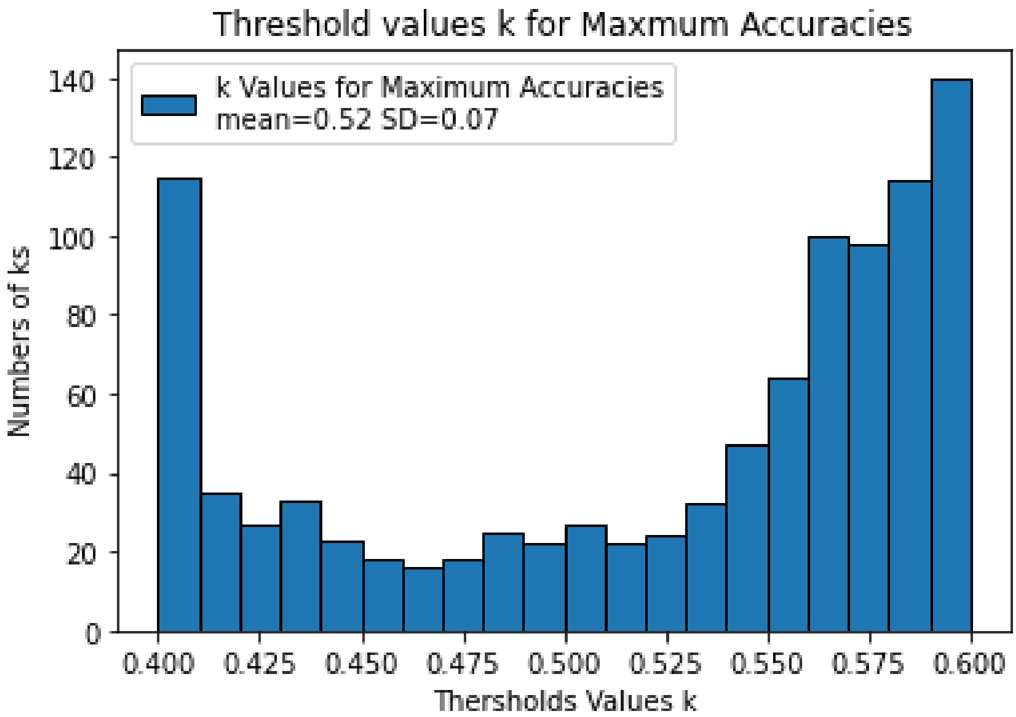
(除以 1000 因為要取 1000 次的平均值)

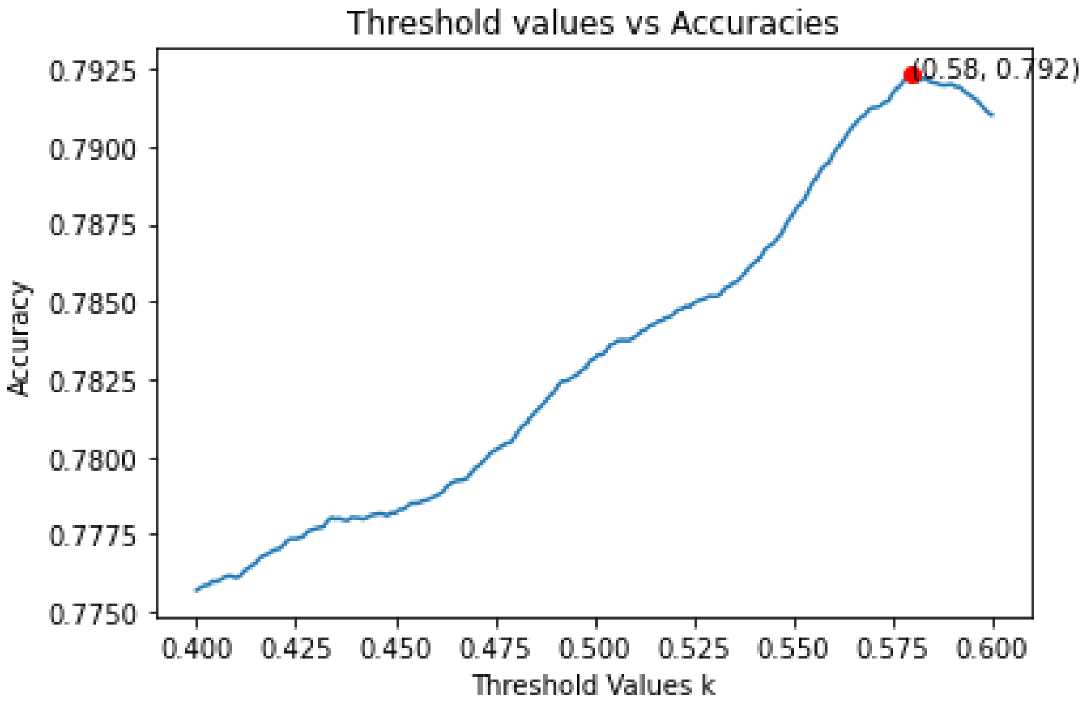
```python
#除以一千的原因為總共做一千次
print("Logistic Regression:")
print("Averages for all examples 1000 trials with k=0.5")
print("Mean weight of C1 =",round((sum(c1_w)/1000),3),",95% confidence interval =",round(1.96*stdDev(c1_w),3))
print("Mean weight of C2 =",round((sum(c2_w)/1000),3),",95% confidence interval =",round(1.96*stdDev(c2_w),3))
print("Mean weight of C3 =",round((sum(c3_w)/1000),3),",95% confidence interval =",round(1.96*stdDev(c3_w),3))
print("Mean weight of Age =",round((sum(age_w)/1000),3),",95% confidence interval =",round(1.96*stdDev(age_w),3))
print("Mean weight of male gender =",round(sum(male_w)/1000,3),",95% confidence interval =",round(1.96*stdDev(male_w),3))
print('Accuracy =', round(sum(Accr)/1000, 3),",95% confidence interval =",round(1.96*stdDev(Accr),3))
print('Sensitivity =', round(sum(Sens)/1000, 3),",95% confidence interval =",round(1.96*stdDev(Sens),3))
print('pecificity =', round(sum(Spec)/1000, 3),",95% confidence interval =",round(1.96*stdDev(Spec),3))
print('Pos. Pred. Val. =', round(sum(Ppv)/1000, 3),",95% confidence interval =",round(1.96*stdDev(Ppv),3))
print('Mean AUROC =',round(sum(roc)/1000,3),",95% confidence interval =",round(1.96*stdDev(roc),3))
print("")

makeHist(MaxA,20,"Maxmum Accuracies","Maximum Accuracies","Numbers of Maximum")
pylab.show()
makeHist2(MaxK, 20, "Threshold values k for Maxmum Accuracies", "Thersholds Values k", " Numbers of ks")
pylab.show()
kValues=[k/1000 for k in range(400, 601, 1)]
pylab.plot(kValues,mean_accur)
pylab.plot(indexk, max(mean_accur),'ro')
pylab.annotate((indexk, round(max(mean_accur),3)), xy=(indexk,MAX))
pylab.title('Threshold values vs Accuracies')
pylab.xlabel('Threshold Values k')
pylab.ylabel('Accuracy')
pylab.show()
```

結果:

```
___________________part 3__________________
Logistic Regression:
Averages for all examples 1000 trials with k=0.5
Mean weight of C1 = 1.14 ,95% confidence interval = 0.122
Mean weight of C2 = -0.081 ,95% confidence interval = 0.098
Mean weight of C3 = -1.059 ,95% confidence interval = 0.12
Mean weight of Age = -0.033 ,95% confidence interval = 0.006
Mean weight of male gender = -2.412 ,95% confidence interval = 0.152
Accuracy = 0.783 ,95% confidence interval = 0.051
Sensitivity = 0.703 ,95% confidence interval = 0.096
pecificity = 0.783 ,95% confidence interval = 0.051
Pos. Pred. Val. = 0.703 ,95% confidence interval = 0.096
Mean AUROC = 0.838 ,95% confidence interval = 0.055
```







第四部份:

函式:

Zscale 及 Iscale 的函式。

buildpassengerExamplesSC:age 經 scaling 的 buildpassengerExamples2

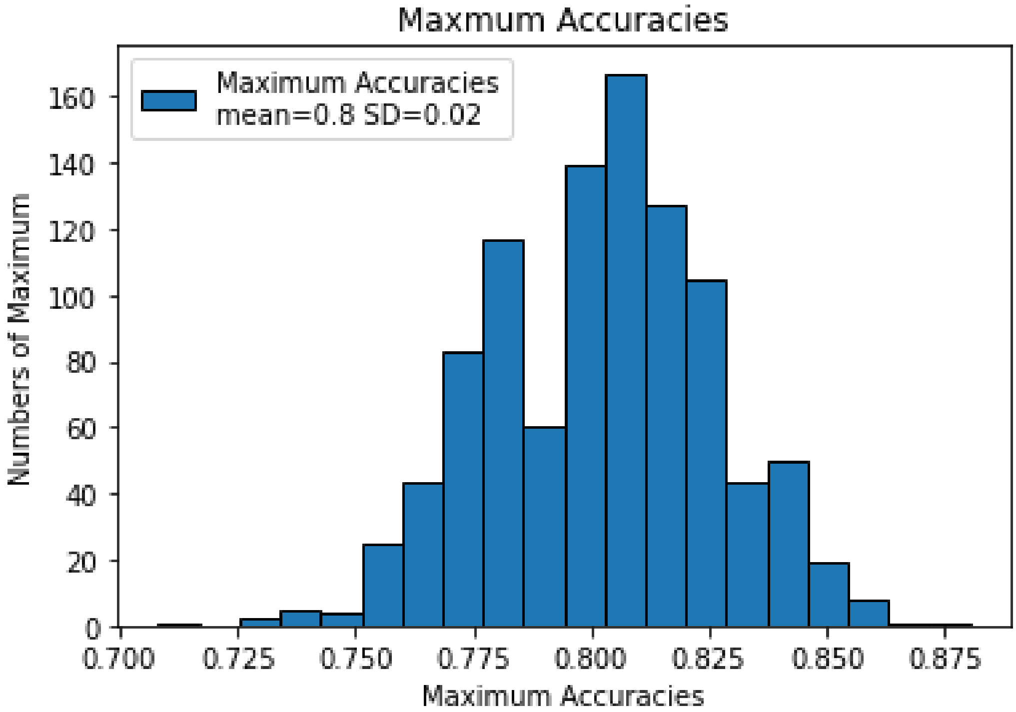
```python
#________________part 4___________________
def zScaleFeatures(vals):
    """Assumes vals is a sequence of floats"""
    result = pylab.array(vals)
    mean = sum(result)/len(result)
    result = result - mean
    return result/stdDev(result)
def iScaleFeatures(vals):
    """Assumes vals is a sequence of floats"""
    minVal, maxVal = min(vals), max(vals)
    fit = pylab.polyfit([minVal, maxVal], [0, 1], 1)
    return pylab.polyval(fit, vals)


def buildpassengerExamplesSC(fileName,scale):

    data  = getData2(fileName)
    #data['c1']=scale(data['c1'])
    #data['c2']=scale(data['c2'])
    #data['c3']=scale(data['c3'])
    data['age']=scale(data['age'])
    #data['gender']=scale(data['gender'])


    examples = []
    for i in range(len(data['survived'])):
        a = Passenger2(data['c1'][i],data['c2'][i], data['c3'][i],  data['age'][i],data['gender'][i],data['survived'][i])
        examples.append(a)
    return examples
```

因為後面程式碼都模稜兩可，就不再截圖多做解釋。

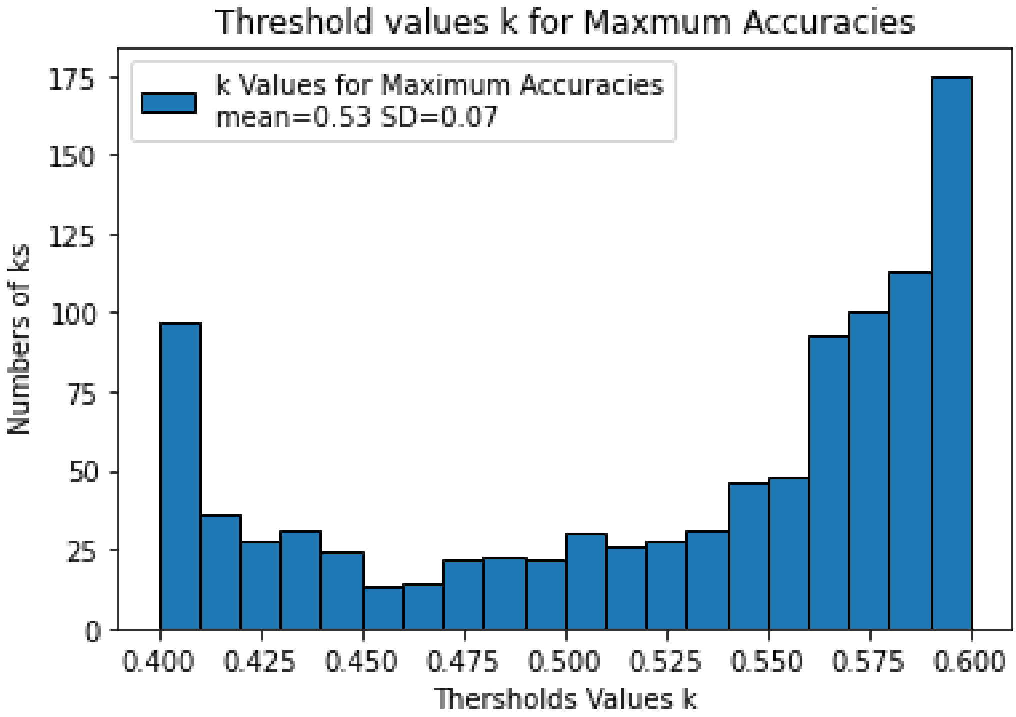
結果:

```
Logistic Regressionwith zScaling:
Averages for all examples 1000 trials with k=0.5
Mean weight of C1 = 1.14 ,95% confidence interval = 0.121
Mean weight of C2 = -0.082 ,95% confidence interval = 0.1
Mean weight of C3 = -1.058 ,95% confidence interval = 0.117
Mean weight of Age = -0.473 ,95% confidence interval = 0.089
Mean weight of male gender = -2.412 ,95% confidence interval = 0.151
Accuracy = 0.782 ,95% confidence interval = 0.05
Sensitivity = 0.703 ,95% confidence interval = 0.095
pecificity = 0.782 ,95% confidence interval = 0.05
Pos. Pred. Val. = 0.703 ,95% confidence interval = 0.095
Mean AUROC = 0.838 ,95% confidence interval = 0.053

Logistic Regression with iScaling:
Averages for all examples 1000 trials with k=0.5
Mean weight of C1 = 1.065 ,95% confidence interval = 0.108
Mean weight of C2 = -0.069 ,95% confidence interval = 0.097
Mean weight of C3 = -0.997 ,95% confidence interval = 0.111
Mean weight of Age = -2.035 ,95% confidence interval = 0.368
Mean weight of male gender = -2.404 ,95% confidence interval = 0.146
Accuracy = 0.781 ,95% confidence interval = 0.051
Sensitivity = 0.698 ,95% confidence interval = 0.093
pecificity = 0.781 ,95% confidence interval = 0.051
Pos. Pred. Val. = 0.698 ,95% confidence interval = 0.093
Mean AUROC = 0.838 ,95% confidence interval = 0.052
```

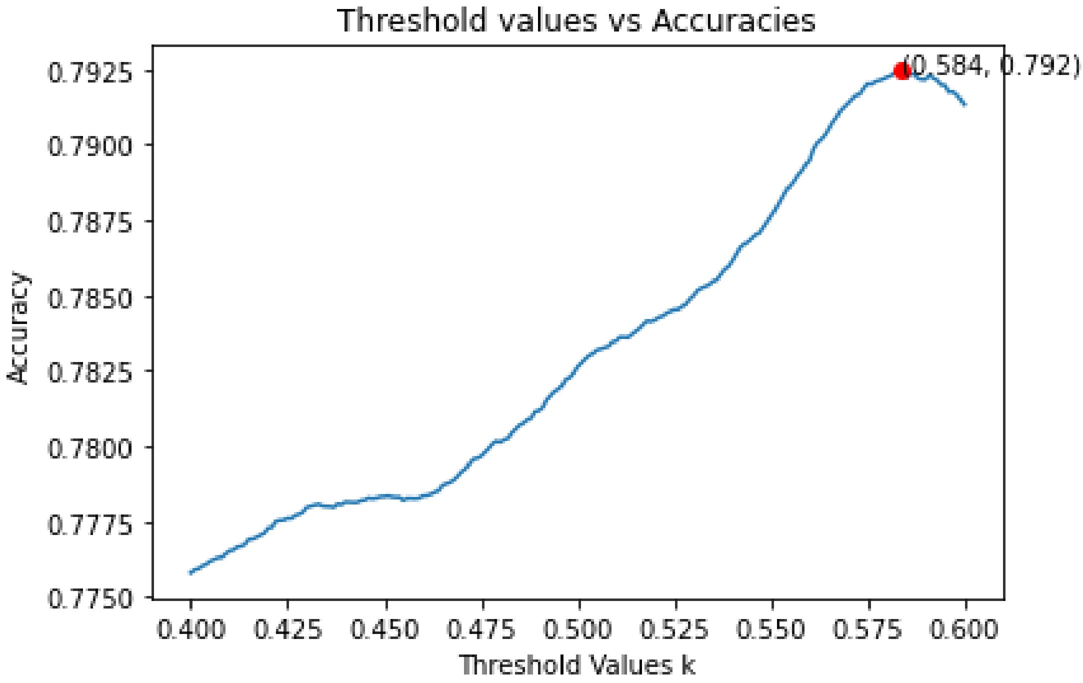

Zscale:



Maxmum Accuracies
Maximum Accuracies
mean=0.8 SD=0.02
Numbers of Maximum
Maximum Accuracies
0
20
40
60
80
100
120
140
160
0.700
0.725
0.750
0.775
0.800
0.825
0.850
0.875

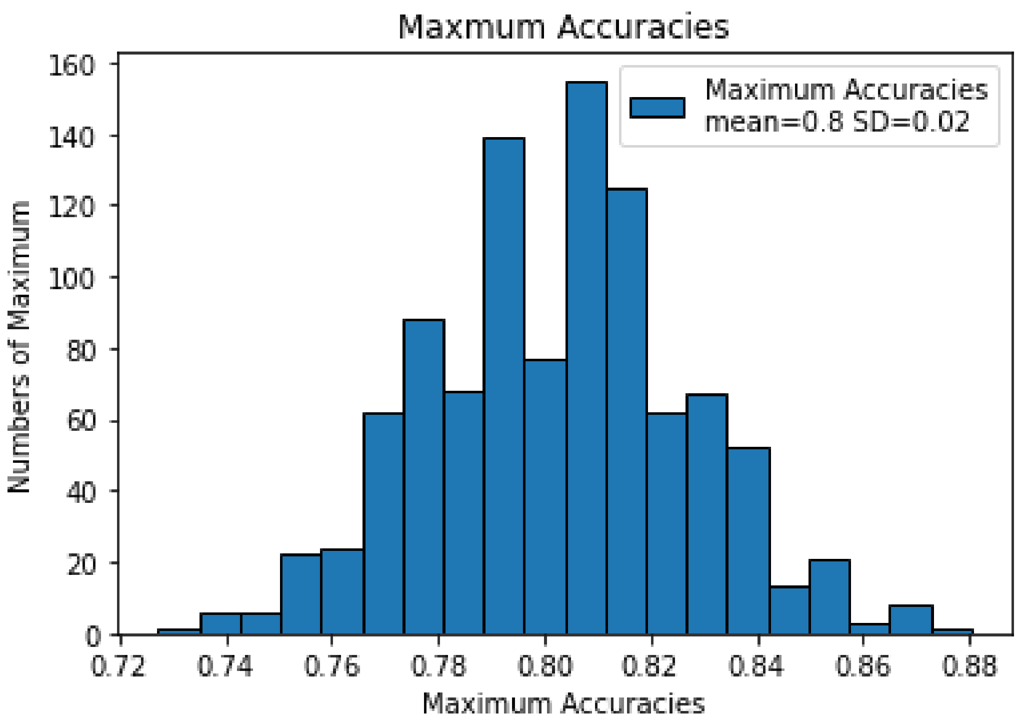


Threshold values k for Maxmum Accuracies
k Values for Maximum Accuracies
mean=0.53 SD=0.07
Numbers of ks
Thersholds Values k
0
25
50
75
100
125
150
175
0.400
0.425
0.450
0.475
0.500
0.525
0.550
0.575
0.600

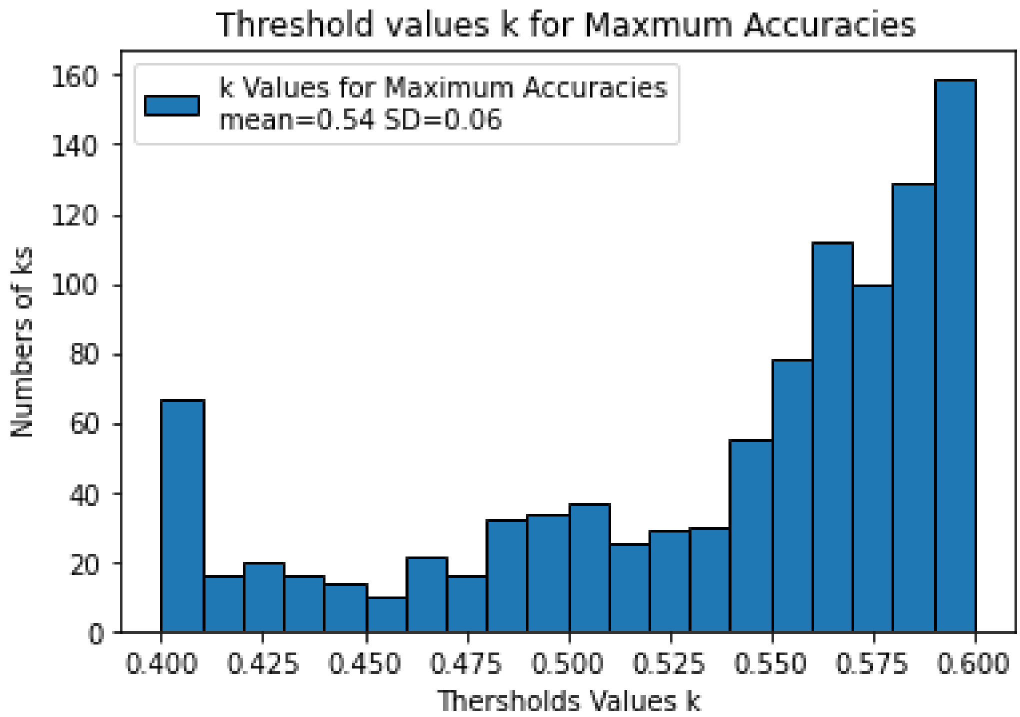


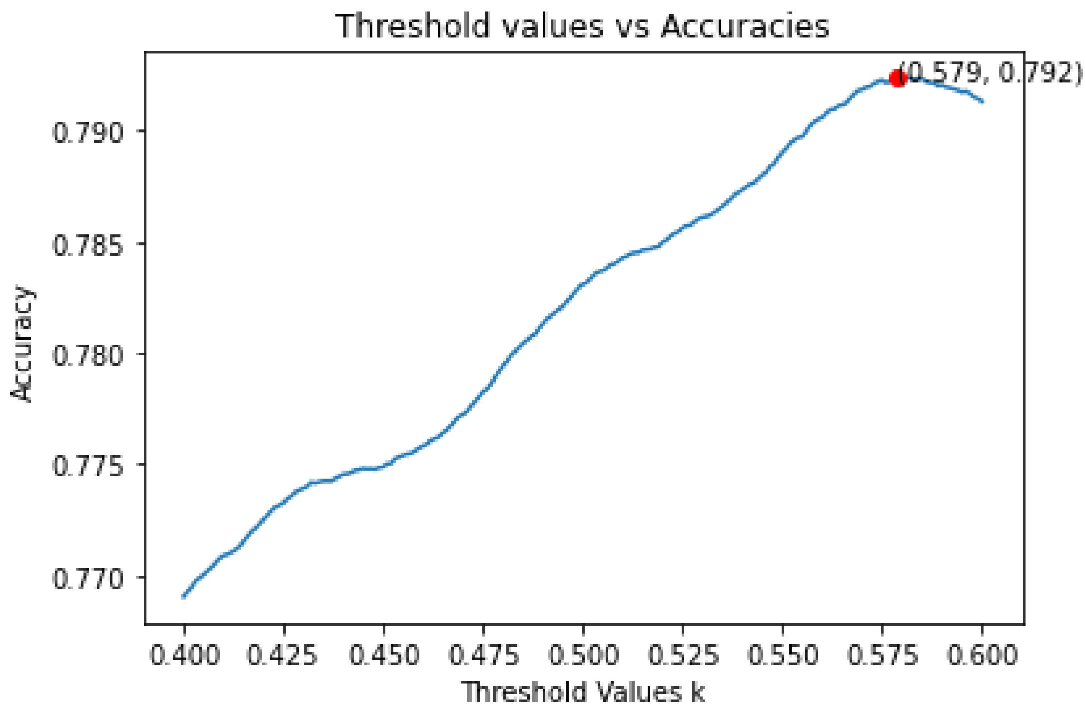
Threshold values vs Accuracies
(0.584, 0.792)
Accuracy
0.7925
0.7900
0.7875
0.7850
0.7825
0.7800
0.7775
0.7750
0.400 0.425 0.450 0.475 0.500 0.525 0.550 0.575 0.600
Threshold Values k

Iscale:



Maxmum Accuracies
Maximum Accuracies
mean=0.8 SD=0.02
Numbers of Maximum
160
140
120
100
80
60
40
20
0
0.72 0.74 0.76 0.78 0.80 0.82 0.84 0.86 0.88
Maximum Accuracies





第五部分:

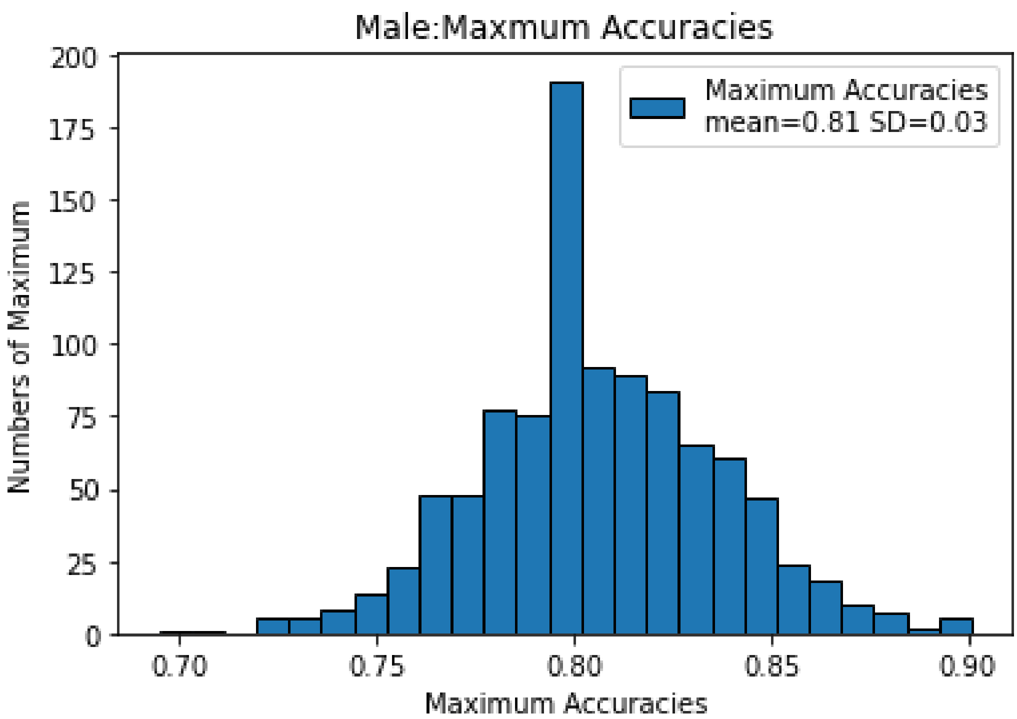
第五部分相較前面稍不一樣的地方為 m, f 分開測試:

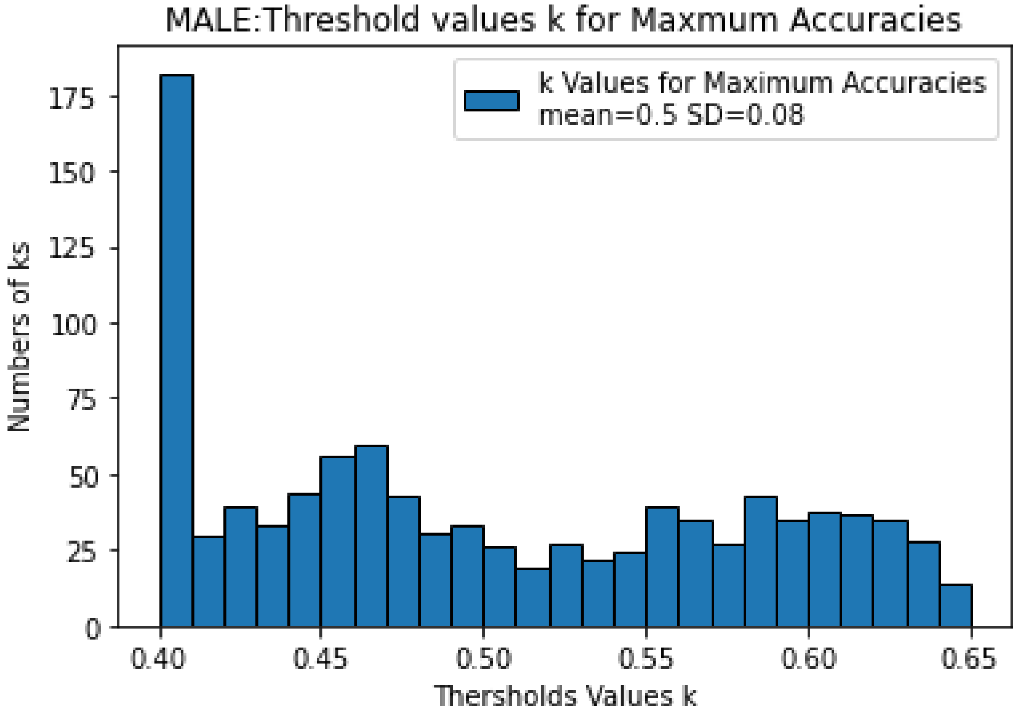
```
trainset , testset = divide80_20_1000(m)
```

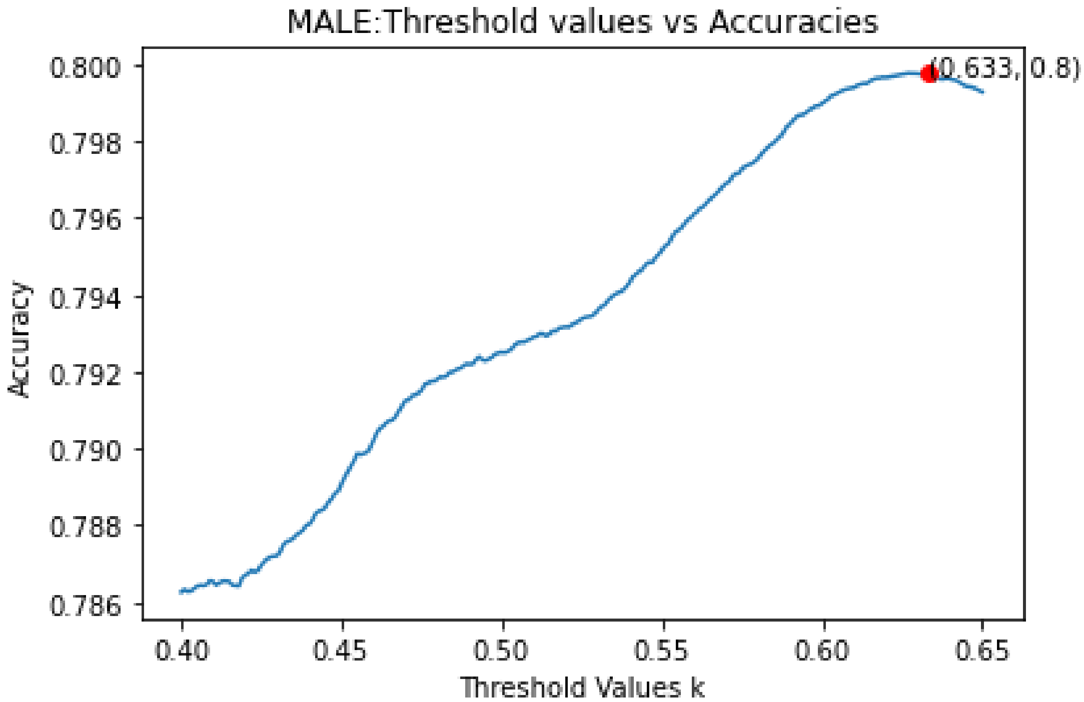
```
trainset , testset = divide80_20_1000(f)
```

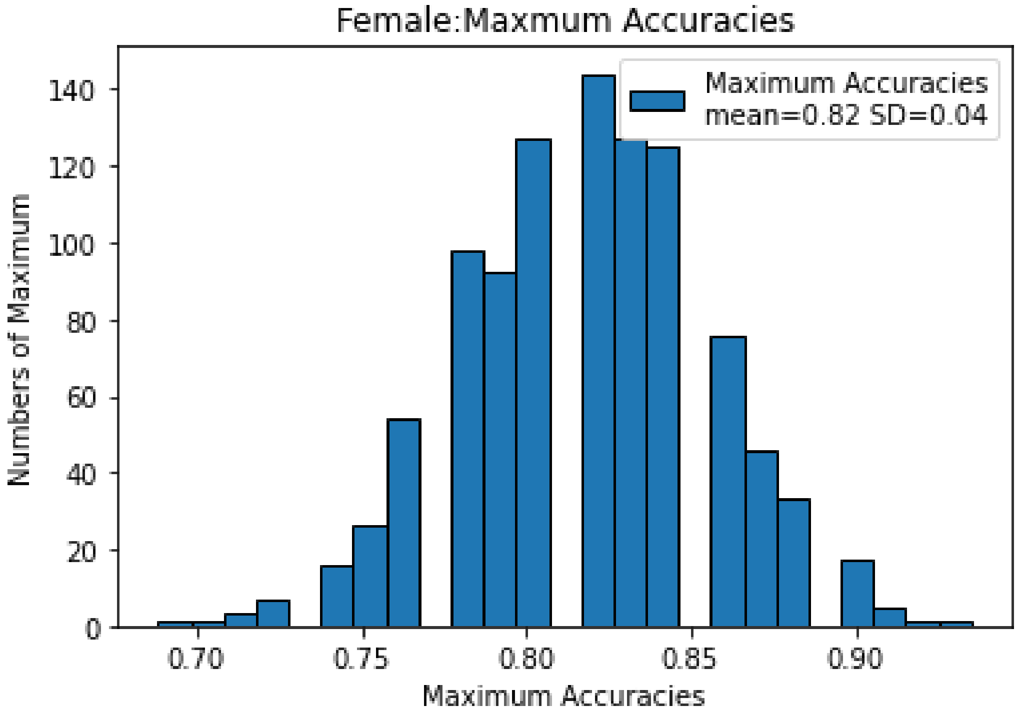
結果:

```
___________part 5___________
Logistic Regression with Male and Female Separated:
Averages for Male Examples 1000 trials with k=0.5
Mean weight of C1 = 1.103 ,95% confidence interval = 0.157
Mean weight of C2 = -0.532 ,95% confidence interval = 0.154
Mean weight of C3 = -0.558 ,95% confidence interval = 0.145
Mean weight of Age = -0.047 ,95% confidence interval = 0.009
Mean weight of male gender = 0.012 ,95% confidence interval = 0.059
Accuracy = 0.793 ,95% confidence interval = 0.063
Sensitivity = 0.079 ,95% confidence interval = 0.094
pecificity = 0.793 ,95% confidence interval = 0.063
Pos. Pred. Val. = 0.079 ,95% confidence interval = 0.094
Mean AUROC = 0.687 ,95% confidence interval = 0.102

Averages for Female Examples 1000 trials with k=0.5
Mean weight of C1 = 1.409 ,95% confidence interval = 0.249
Mean weight of C2 = 0.41 ,95% confidence interval = 0.214
Mean weight of C3 = -1.82 ,95% confidence interval = 0.201
Mean weight of Age = -0.015 ,95% confidence interval = 0.012
Mean weight of male gender = 0.0 ,95% confidence interval = 0.0
Accuracy = 0.766 ,95% confidence interval = 0.088
Sensitivity = 0.853 ,95% confidence interval = 0.15
pecificity = 0.766 ,95% confidence interval = 0.088
Pos. Pred. Val. = 0.853 ,95% confidence interval = 0.15
Mean AUROC = 0.827 ,95% confidence interval = 0.1
```

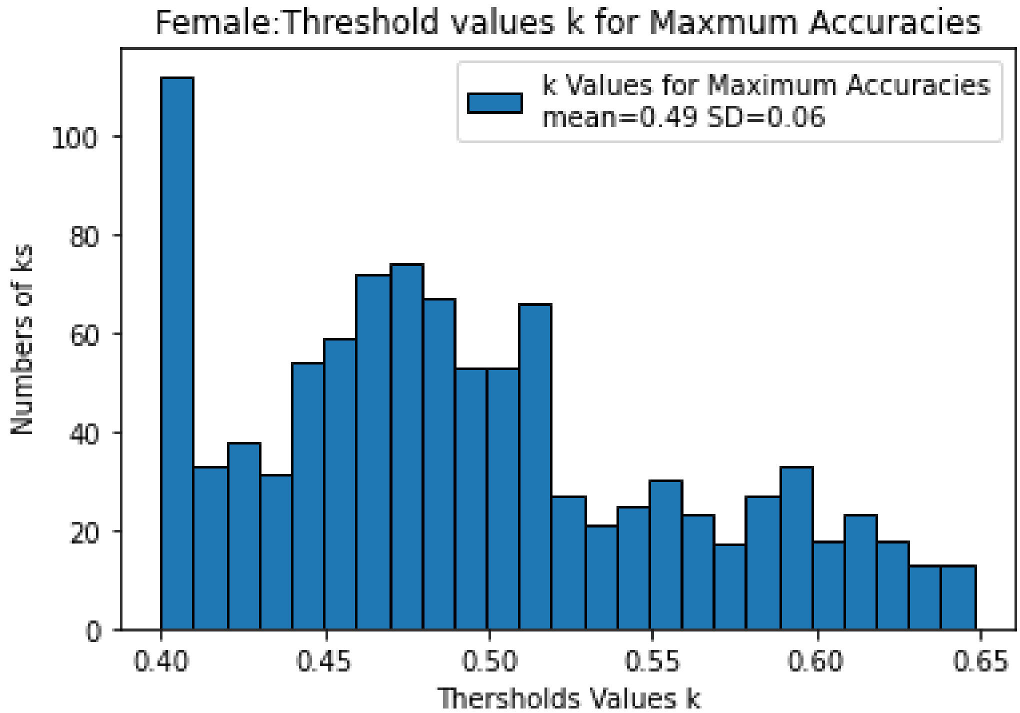




MALE:Threshold values k for Maxmum Accuracies
k Values for Maximum Accuracies
mean=0.5 SD=0.08
Numbers of ks
Thersholds Values k
0
25
50
75
100
125
150
175
0.40
0.45
0.50
0.55
0.60
0.65

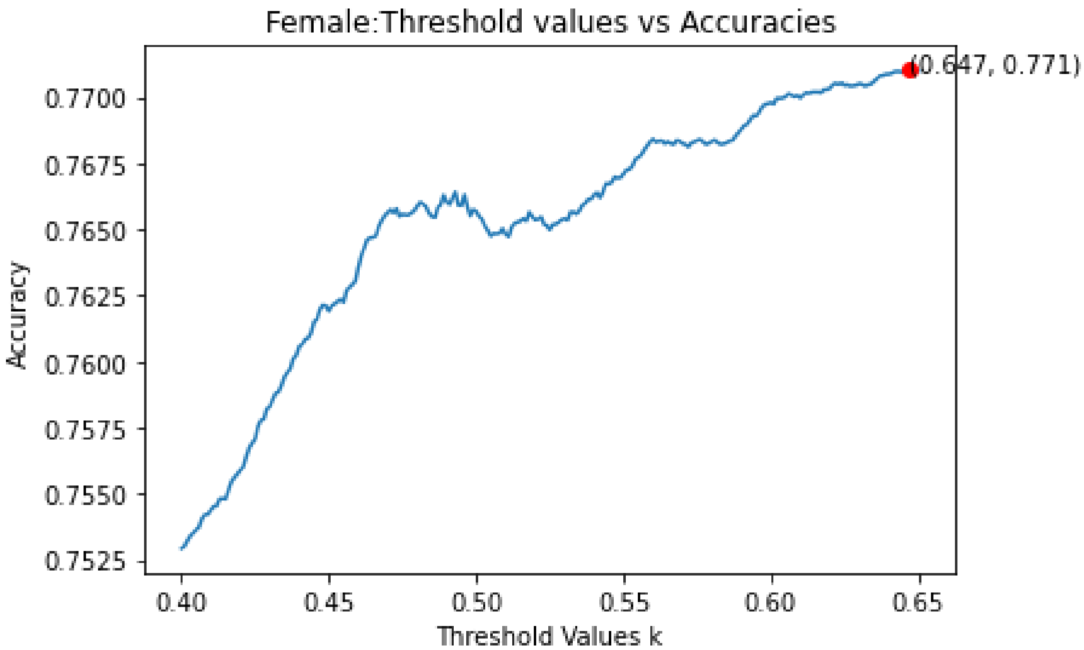

MALE:Threshold values vs Accuracies
(0.633, 0.8)
Accuracy
Threshold Values k
0.786
0.788
0.790
0.792
0.794
0.796
0.798
0.800
0.40
0.45
0.50
0.55
0.60
0.65



Female:Maxmum Accuracies
Maximum Accuracies
mean=0.82 SD=0.04
Numbers of Maximum
Maximum Accuracies
0
20
40
60
80
100
120
140
0.70
0.75
0.80
0.85
0.90

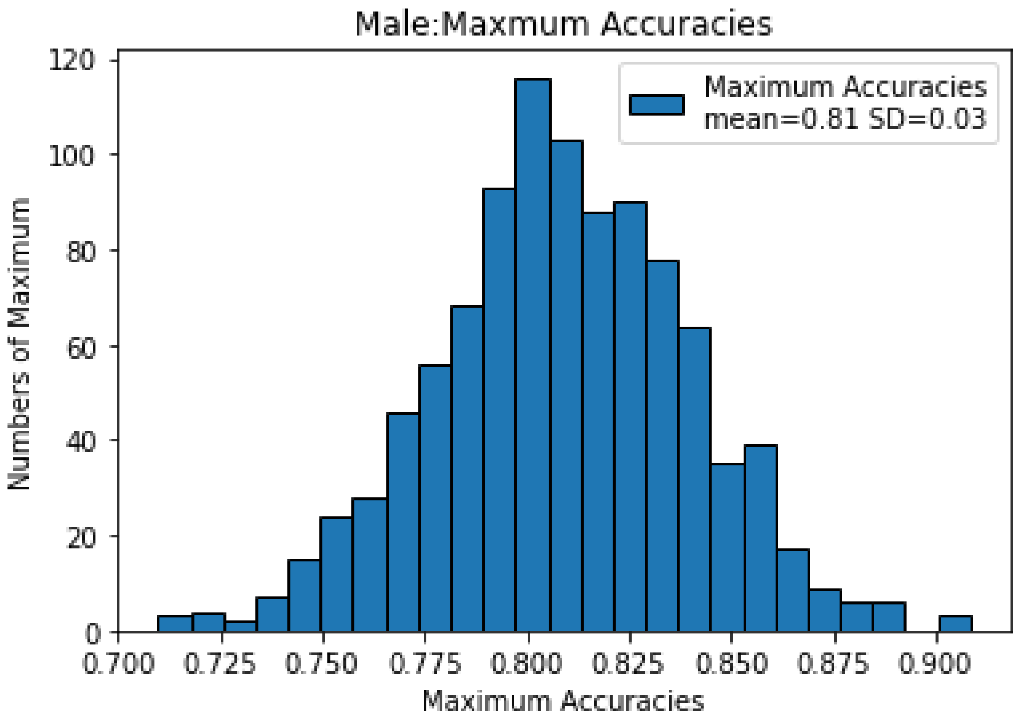


Female:Threshold values k for Maxmum Accuracies
k Values for Maximum Accuracies
mean=0.49 SD=0.06
Numbers of ks
Thersholds Values k
0
20
40
60
80
100
0.40
0.45
0.50
0.55
0.60
0.65

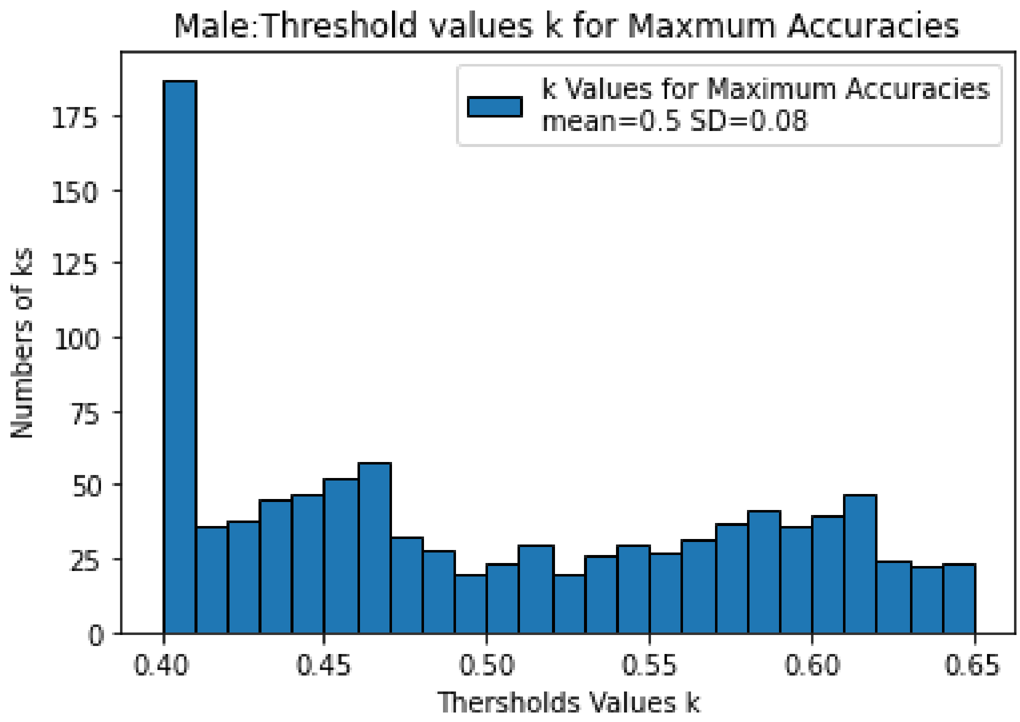


Zscale:

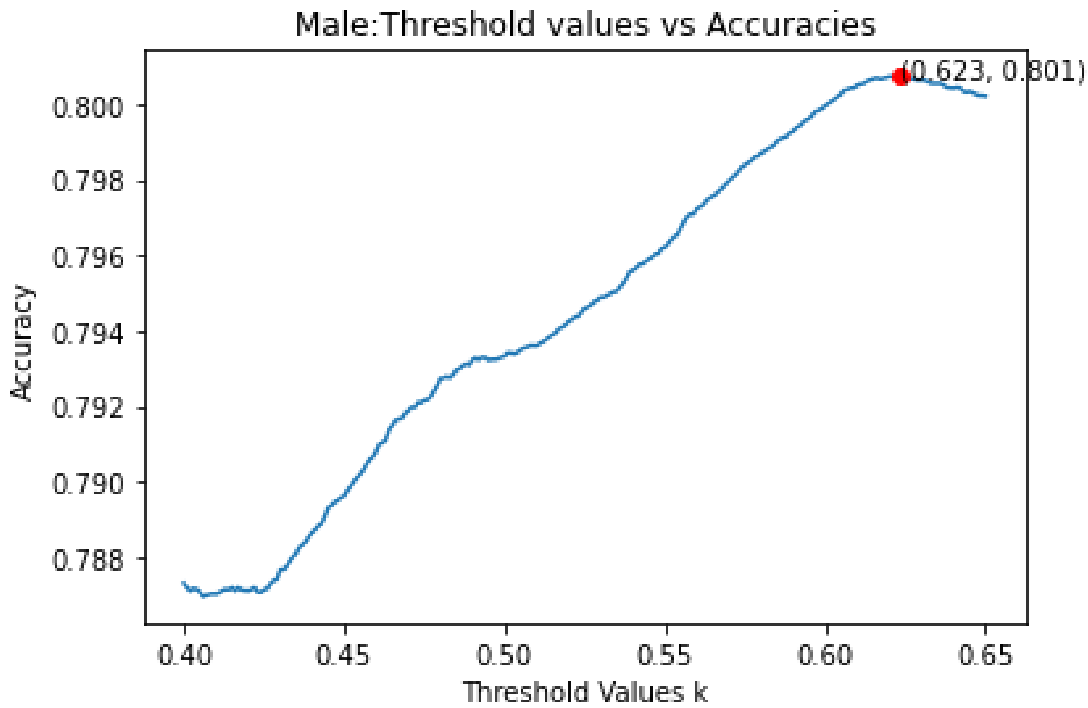
```
Logistic Regression with Male and Female Separated with z-scaling:
Averages for Male Examples 1000 trials with k=0.5
Mean weight of C1 = 1.092 ,95% confidence interval = 0.159
Mean weight of C2 = -0.536 ,95% confidence interval = 0.147
Mean weight of C3 = -0.556 ,95% confidence interval = 0.138
Mean weight of Age = -0.666 ,95% confidence interval = 0.124
Mean weight of male gender = 0.0 ,95% confidence interval = 0.0
Accuracy = 0.793 ,95% confidence interval = 0.061
Sensitivity = 0.078 ,95% confidence interval = 0.096
pecificity = 0.793 ,95% confidence interval = 0.061
Pos. Pred. Val. = 0.078 ,95% confidence interval = 0.096
Mean AUROC = 0.687 ,95% confidence interval = 0.106

Averages for Female Examples 1000 trials with k=0.5
Mean weight of C1 = 1.087 ,95% confidence interval = 0.151
Mean weight of C2 = -0.533 ,95% confidence interval = 0.148
Mean weight of C3 = -0.555 ,95% confidence interval = 0.142
Mean weight of Age = -0.664 ,95% confidence interval = 0.122
Mean weight of male gender = 0.0 ,95% confidence interval = 0.0
Accuracy = 0.794 ,95% confidence interval = 0.06
Sensitivity = 0.078 ,95% confidence interval = 0.094
pecificity = 0.794 ,95% confidence interval = 0.06
Pos. Pred. Val. = 0.078 ,95% confidence interval = 0.094
Mean AUROC = 0.688 ,95% confidence interval = 0.105
```

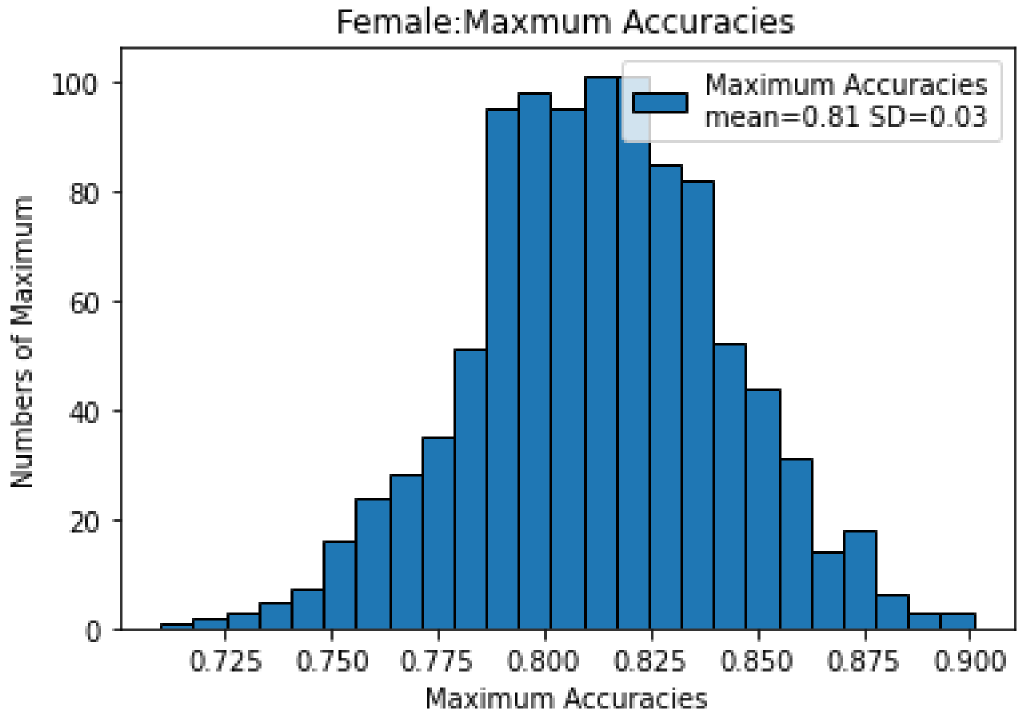


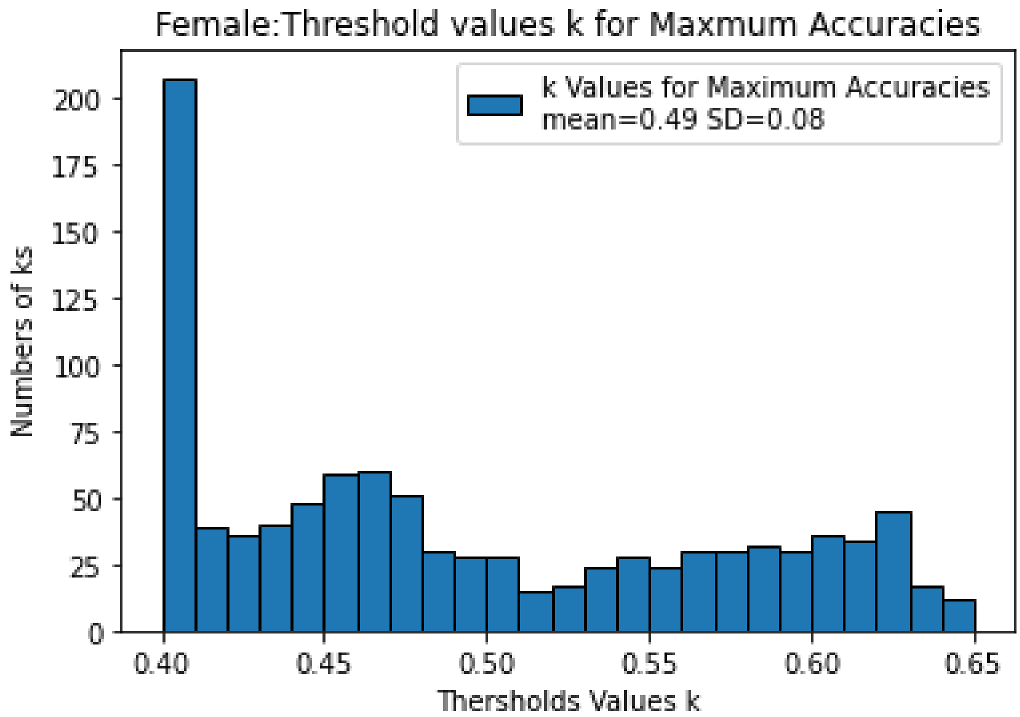
Male:Maxmum Accuracies
Maximum Accuracies
mean=0.81 SD=0.03
Numbers of Maximum
Maximum Accuracies
0.700
0.725
0.750
0.775
0.800
0.825
0.850
0.875
0.900
0
20
40
60
80
100
120

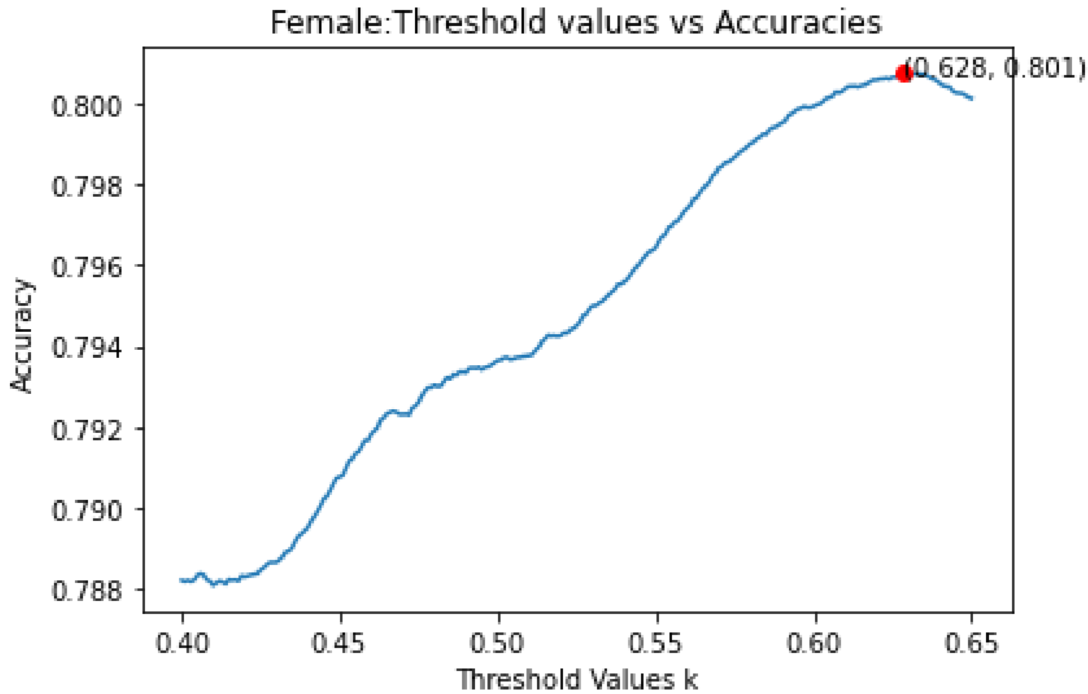


Male:Threshold values k for Maxmum Accuracies
k Values for Maximum Accuracies
mean=0.5 SD=0.08
Numbers of ks
Thersholds Values k
0.40
0.45
0.50
0.55
0.60
0.65
0
25
50
75
100
125
150
175


Male:Threshold values vs Accuracies
(0.623, 0.801)
0.800
0.798
0.796
0.794
0.792
0.790
0.788
Accuracy
0.40
0.45
0.50
0.55
0.60
0.65
Threshold Values k


Female:Maxmum Accuracies
Maximum Accuracies
mean=0.81 SD=0.03
100
80
60
40
20
0
Numbers of Maximum
0.725
0.750
0.775
0.800
0.825
0.850
0.875
0.900
Maximum Accuracies

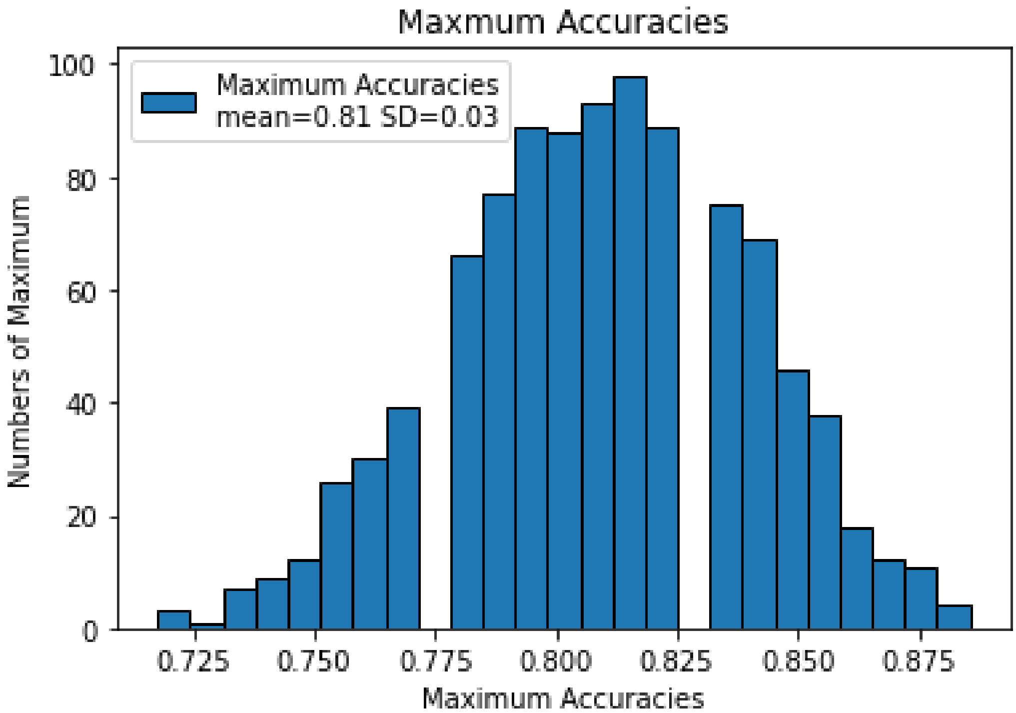

Female:Threshold values k for Maxmum Accuracies
k Values for Maximum Accuracies
mean=0.49 SD=0.08
Numbers of ks
Thersholds Values k
0
25
50
75
100
125
150
175
200
0.40
0.45
0.50
0.55
0.60
0.65

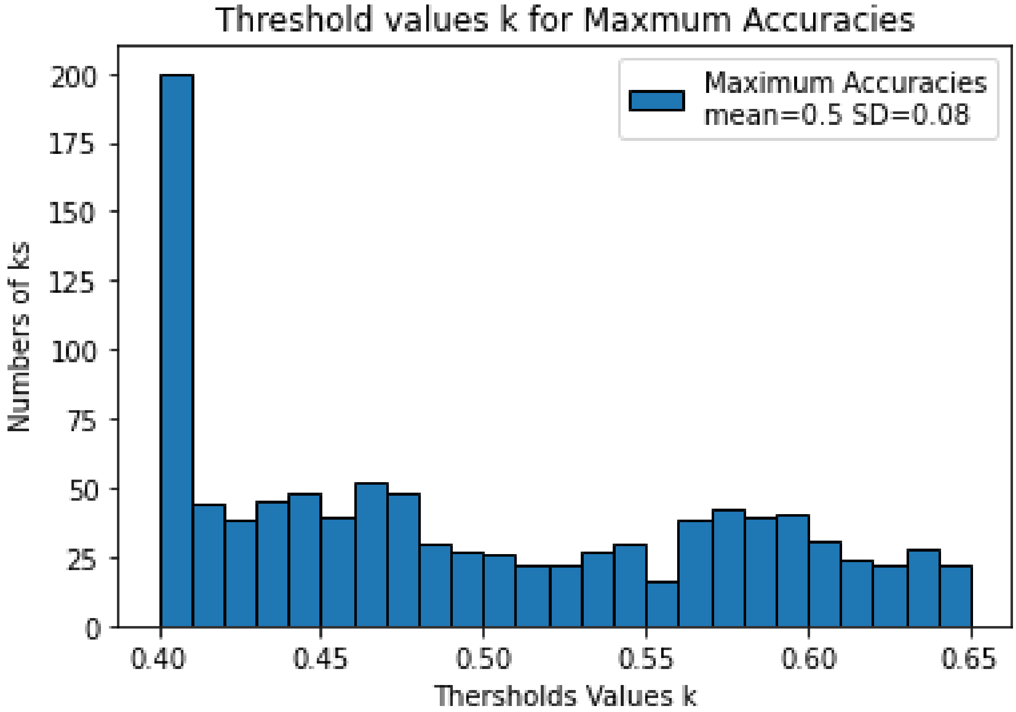

Female:Threshold values vs Accuracies
(0.628, 0.801)
Accuracy
Threshold Values k
0.788
0.790
0.792
0.794
0.796
0.798
0.800
0.40
0.45
0.50
0.55
0.60
0.65

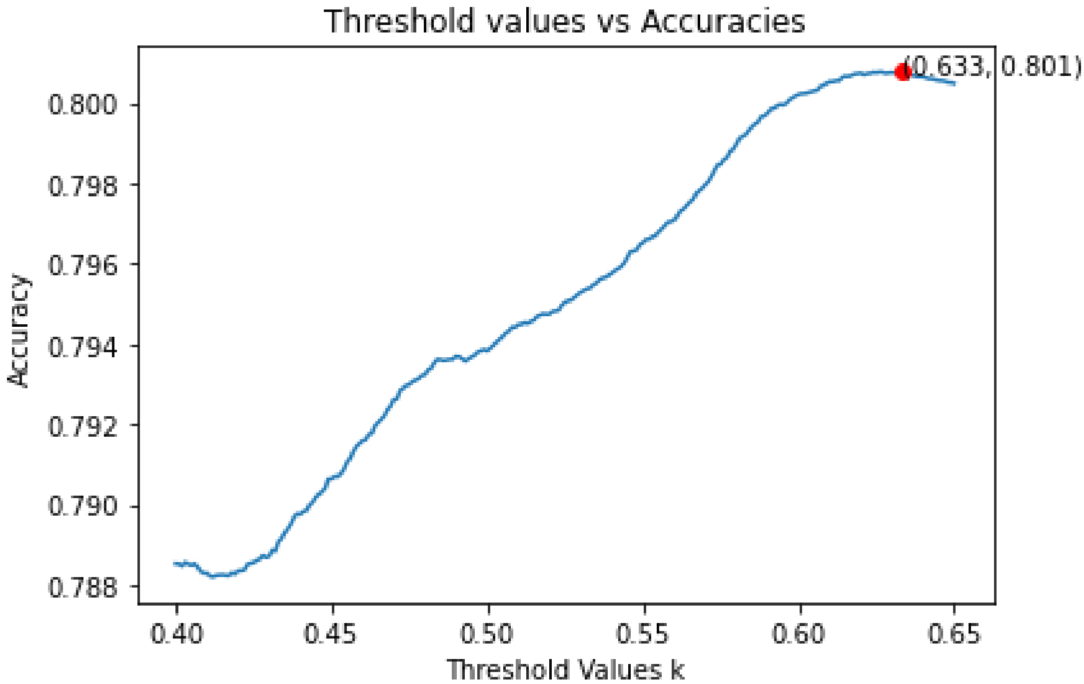
Iscale:

```
Logistic Regression with Male and Female Separated with i-scaling:
Averages for Male Examples 1000 trials with k=0.5
Mean weight of C1 = 1.092 ,95% confidence interval = 0.159
Mean weight of C2 = -0.537 ,95% confidence interval = 0.161
Mean weight of C3 = -0.555 ,95% confidence interval = 0.145
Mean weight of Age = -0.668 ,95% confidence interval = 0.125
Mean weight of male gender = 0.0 ,95% confidence interval = 0.0
Accuracy = 0.794 ,95% confidence interval = 0.062
Sensitivity = 0.082 ,95% confidence interval = 0.1
pecificity = 0.794 ,95% confidence interval = 0.062
Pos. Pred. Val. = 0.082 ,95% confidence interval = 0.1
Mean AUROC = 0.684 ,95% confidence interval = 0.107

Averages for Female Examples 1000 trials with k=0.5
Mean weight of C1 = 1.09 ,95% confidence interval = 0.157
Mean weight of C2 = -0.537 ,95% confidence interval = 0.151
Mean weight of C3 = -0.553 ,95% confidence interval = 0.139
Mean weight of Age = -0.662 ,95% confidence interval = 0.124
Mean weight of male gender = 0.0 ,95% confidence interval = 0.0
Accuracy = 0.794 ,95% confidence interval = 0.06
Sensitivity = 0.079 ,95% confidence interval = 0.094
pecificity = 0.794 ,95% confidence interval = 0.06
Pos. Pred. Val. = 0.079 ,95% confidence interval = 0.094
```

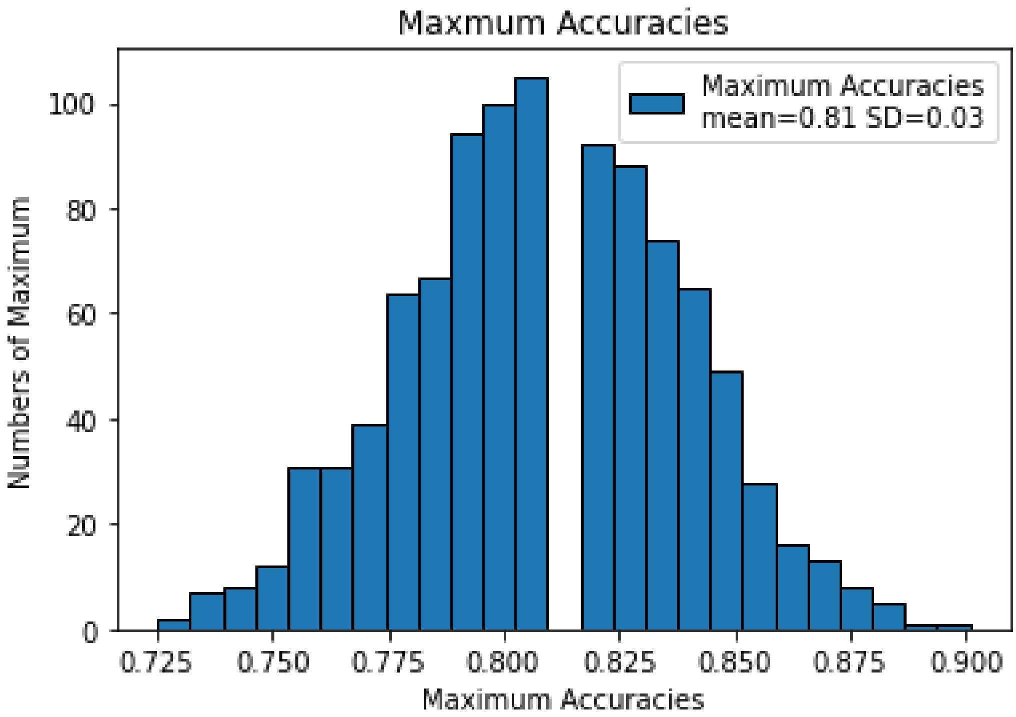
Male:



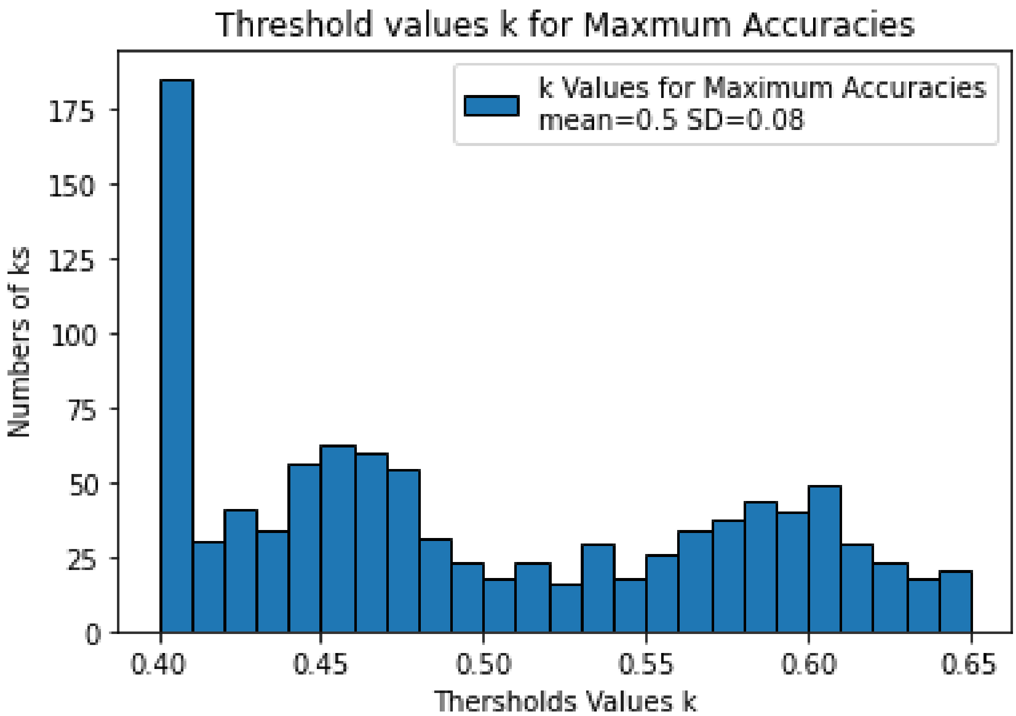

Threshold values k for Maxmum Accuracies
Maximum Accuracies
mean=0.5 SD=0.08
Numbers of ks
0
25
50
75
100
125
150
175
200
0.40
0.45
0.50
0.55
0.60
0.65
Thersholds Values k


Threshold values vs Accuracies
(0.633, 0.801)
Accuracy
0.788
0.790
0.792
0.794
0.796
0.798
0.800
0.40
0.45
0.50
0.55
0.60
0.65
Threshold Values k

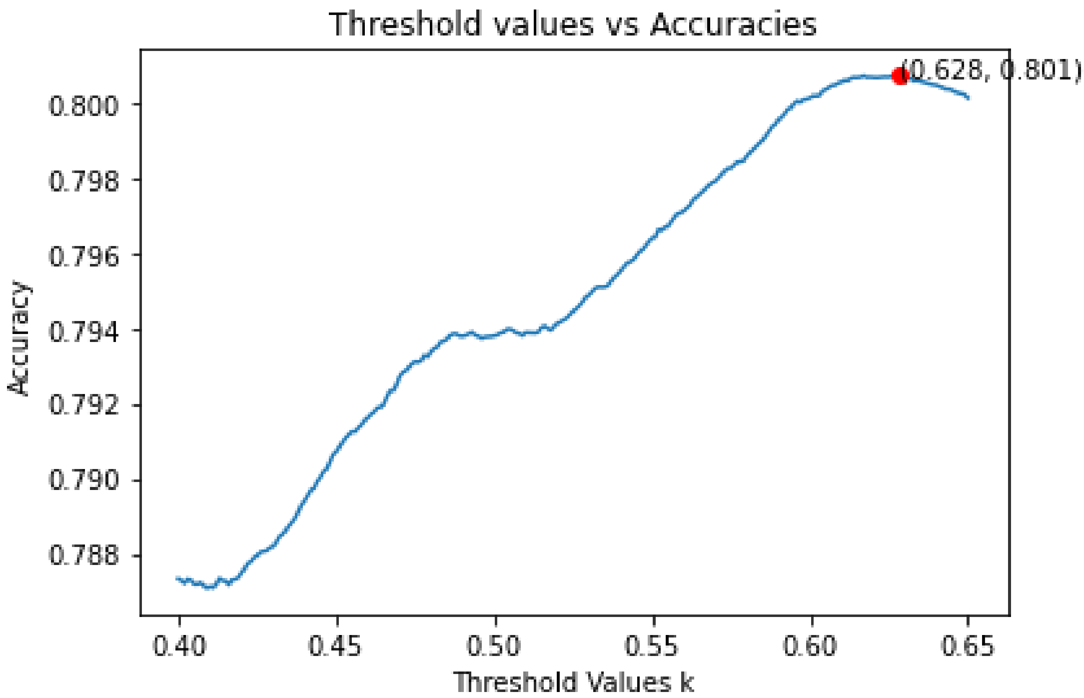
Female:



Maxmum Accuracies
Maximum Accuracies
mean=0.81 SD=0.03
Numbers of Maximum
0
20
40
60
80
100
0.725
0.750
0.775
0.800
0.825
0.850
0.875
0.900
Maximum Accuracies



Threshold values k for Maxmum Accuracies
k Values for Maximum Accuracies
mean=0.5 SD=0.08
Numbers of ks
0
25
50
75
100
125
150
175
0.40
0.45
0.50
0.55
0.60
0.65
Thersholds Values k



第六部分:

函式:

為上課使用的 KNN 函式，分別為找鄰近點的函式，以及計算真陽性、假陽性、真陰性、假陰性的函式。

再來，是用來列印 confusionMatrix 的函式，以及 n-ford 函式(回傳最大 K 值，以及各 K 值的 accuracy)(皆與課程程式碼差不多)。

```
print("____________part 6____________")
def findKNearest(example, exampleSet, k):
    kNearest, distances = [], []
    #Build lists containing first k examples and their distances
    for i in range(k):
        kNearest.append(exampleSet[i])
        distances.append(example.featureDist(exampleSet[i]))
    maxDist = max(distances) #Get maximum distance
    #Look at examples not yet considered
    for e in exampleSet[k:]:
        dist = example.featureDist(e)
        if dist < maxDist:
            #replace farther neighbor by this one
            maxIndex = distances.index(maxDist)
            kNearest[maxIndex] = e
            distances[maxIndex] = dist
            maxDist = max(distances)
    return kNearest, distances
```

```
def KNearestClassify(training, testSet, label, k):
    """Assumes training and testSet lists of examples, k an int
       Uses a k-nearest neighbor classifier to predict
         whether each example in testSet has the given label
       Returns number of true positives, false positives,
          true negatives, and false negatives"""
    truePos, falsePos, trueNeg, falseNeg = 0, 0, 0, 0
    for e in testSet:
#        nearest, distances = findKNearest(e, training, k)
        nearest, similarities = findKNearest(e, training, k)
        #conduct vote
        numMatch = 0
        for i in range(len(nearest)):
            if nearest[i].getlabel() == label:
                numMatch += 1
        if numMatch > k//2: #guess label
            if e.getlabel() == label:
                truePos += 1
            else:
                falsePos += 1
        else: #guess not label
            if e.getlabel() != label:
                trueNeg += 1
            else:
                falseNeg += 1
    return truePos, falsePos, trueNeg, falseNeg
```

再來，是用來列印 confusionMatrix 的函式，以及 n-ford 函式(回傳最大 K 值，以及各 K 值的 accuracy)(皆與課程程式碼差不多)。

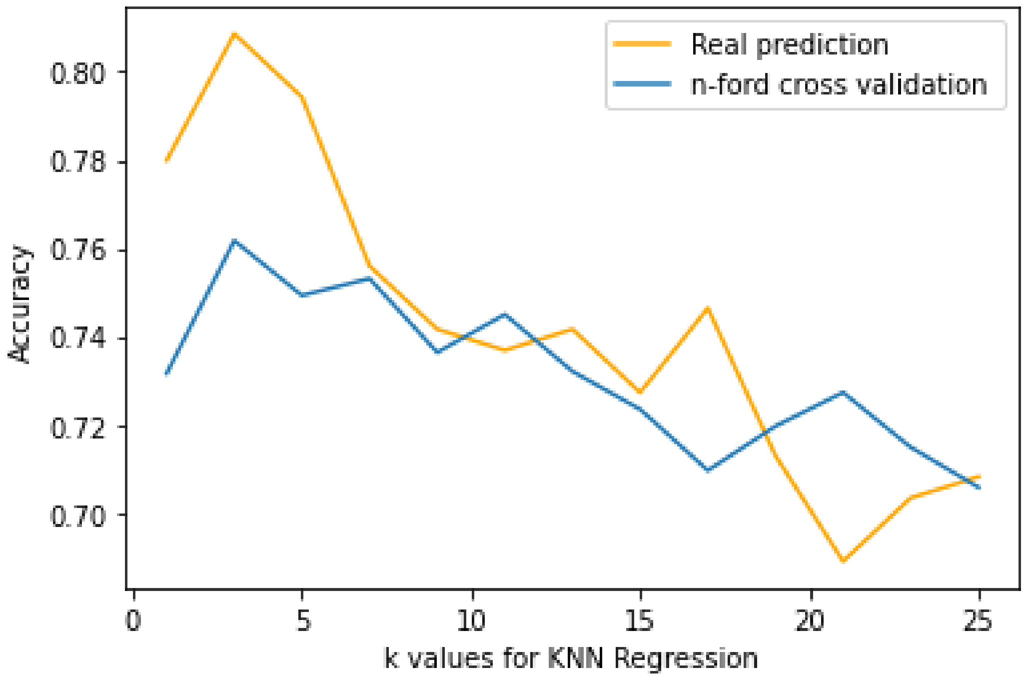
```
def confusionMatrix(truePos, falsePos, trueNeg, falseNeg, k):
    #print('\nk = ', k)
    print('TP,FP,TN,FN = ', truePos, falsePos, trueNeg, falseNeg)
    print('                    ', 'TP', '\t', 'FP')
    print('Confusion Matrix is: ', truePos, '\t', falsePos)
    print('                    ', trueNeg, '\t', falseNeg)
    print('                    ', 'TN', '\t', 'FN' )
    getStats(truePos, falsePos, trueNeg, falseNeg,True)
    return

def findK(training, minK, maxK, numFolds, label):
    #Find average accuracy for range of odd values of k
    accuracies = []
    for k in range(minK, maxK + 1, 2):
        score = 0.0
        for i in range(numFolds): #downsample to reduce computation time
            fold = random.sample(training, min(5000, len(training)))
            examples, testSet = divide80_20_1000(fold)
            truePos, falsePos, trueNeg, falseNeg = KNearestClassify(examples, testSet, label, k)
            score += accuracy(truePos, falsePos, trueNeg, falseNeg)
            #confusionMatrix(truePos, falsePos, trueNeg, falseNeg, k)
        accuracies.append(score/numFolds)
    k_max=2*accuracies.index(max(accuracies))+1

    #for k in range(minK, maxK + 1, 2):


    return accuracies,k_max
```

再來就是資料分割、預測、列印出 confusionMatrix，以及交叉驗證找最大 K 值，最後印出 real predicted 與 n-ford(交叉驗證)的 accuracy 分布比較。

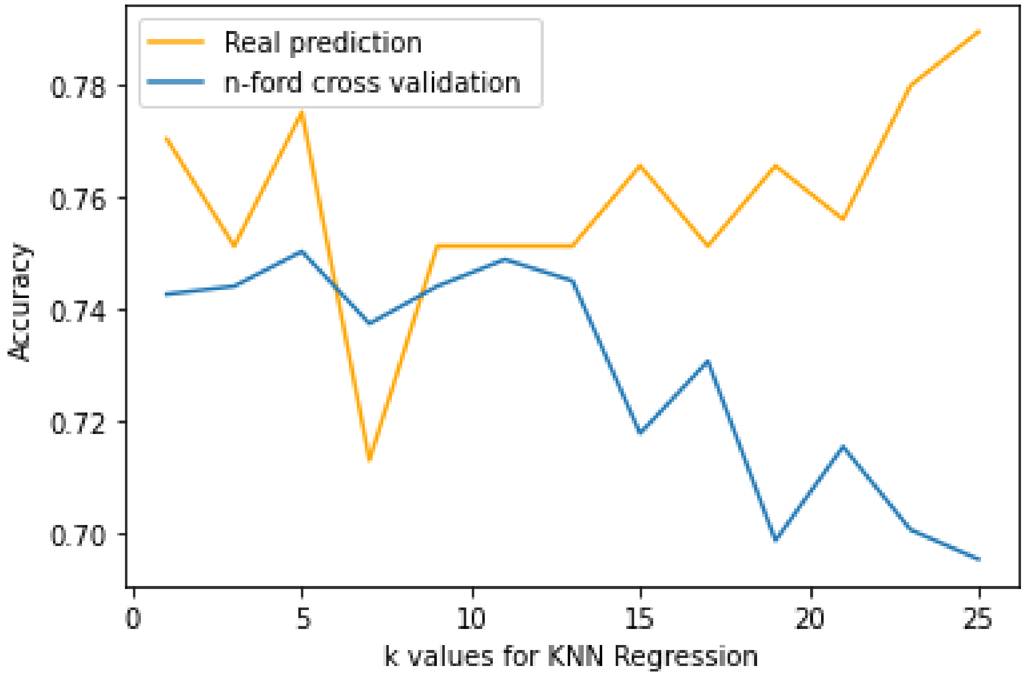
```
trainset , testset = divide80_20_1000(x)
truePos, falsePos, trueNeg, falseNeg = KNearestClassify(trainset, testset, 1, 3)
print('k-NN Prediction for Survive with k=3:')
confusionMatrix(truePos, falsePos, trueNeg, falseNeg, 3)
#accuracy(truePos, falsePos, trueNeg, falseNeg)
Accurs,max_k=findK(x, 1, 25, 10, 1)
print("K for Maximum Accuracy is:", max_k)
confusionMatrix(truePos, falsePos, trueNeg, falseNeg, k)
true_acc = []
for i in range(1,26,2):
    truePos, falsePos, trueNeg, falseNeg = KNearestClassify(trainset, testset, 1, i)
    true_acc.append(accuracy(truePos, falsePos, trueNeg, falseNeg))



x_diff =[1,3,5,7,9,11,13,15,17,19,21,23,25]
pylab.plot(x_diff,true_acc,label ='Real prediction',color ='orange')
pylab.plot(x_diff,Accurs,label = 'n-ford cross validation ')
pylab.xlabel('k values for KNN Regression')
pylab.ylabel("Accuracy")
pylab.legend()
pylab.show()
```

結果:

```
_____________part 6_____________
k-NN Prediction for Survive with k=3:
TP,FP,TN,FN =  66 15 103 25
                      TP      FP
Confusion Matrix is:  66      15
                      103     25
                      TN      FN
 Accuracy = 0.809
 Sensitivity = 0.725
 Specificity = 0.873
 Pos. Pred. Val. = 0.815
K for Maximum Accuracy is: 3
TP,FP,TN,FN =  66 15 103 25
                      TP      FP
Confusion Matrix is:  66      15
                      103     25
                      TN      FN
 Accuracy = 0.809
 Sensitivity = 0.725
 Specificity = 0.873
 Pos. Pred. Val. = 0.815
```



上二圖為本次執行結果，本次 n-ford 最大 K 值為 3(大部分會為 5，如下圖(其他次執行結果))。



每一次執行都會有不同的圖不同的結果(資料分割的不同導致)。

第七部分:

用到前面，男女分開資料的 m、f。

```
print("__________part 7__________")
trainsetm , testsetm = divide80_20_1000(m)
trainsetf , testsetf = divide80_20_1000(f)
trainset , testset = divide80_20_1000(x)
truePos, falsePos, trueNeg, falseNeg = KNearestClassify(trainsetm, testsetm, 1, 3)
print("For Male:")
confusionMatrix(truePos, falsePos, trueNeg, falseNeg, 3)


truePos, falsePos, trueNeg, falseNeg = KNearestClassify(trainsetf, testsetf, 1, 3)
print("For Female:")
confusionMatrix(truePos, falsePos, trueNeg, falseNeg, 3)


truePos, falsePos, trueNeg, falseNeg = KNearestClassify(trainset, testset, 1, 3)
print("Combined Predictions Statistics:")
confusionMatrix(truePos, falsePos, trueNeg, falseNeg, 3)
```

結果:

```
___________part 7___________
For Male:
TP,FP,TN,FN =  7 3 98 23
                      TP     FP
Confusion Matrix is:  7      3
                      98     23
                      TN     FN
 Accuracy = 0.802
 Sensitivity = 0.233
 Specificity = 0.97
 Pos. Pred. Val. = 0.7
For Female:
TP,FP,TN,FN =  48 7 13 9
                      TP     FP
Confusion Matrix is:  48     7
                      13     9
                      TN     FN
 Accuracy = 0.792
 Sensitivity = 0.842
 Specificity = 0.65
 Pos. Pred. Val. = 0.873
```

```
Combined Predictions Statistics:
TP,FP,TN,FN =  55 19 108 27
                      TP     FP
Confusion Matrix is:  55     19
                      108    27
                      TN     FN
 Accuracy = 0.78
 Sensitivity = 0.671
 Specificity = 0.85
 Pos. Pred. Val. = 0.743
```

與前面一樣，會因為資料分割不同，預測結果會有所差異。