一.1

```
data=pd.read_csv('C:/Users/Admin/Downloads/framingham_train.csv')
data.head()
```

Out[150]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate
0	1	0.184211	0.333333	1	0.428571	0	0	0	0	0.176591	0.191489	0.402116	0.190257	0.323232
1	1	0.736842	0.000000	0	0.000000	0	0	0	0	0.262834	0.170213	0.328042	0.221037	0.161616
2	0	0.157895	0.666667	1	0.500000	0	0	0	0	0.104723	0.125296	0.296296	0.201406	0.242424
3	0	0.868421	0.333333	0	0.000000	0	0	1	0	0.293634	0.309693	0.507937	0.344401	0.464646
4	0	0.947368	0.666667	1	0.285714	0	0	1	0	0.297741	0.352246	0.486772	0.390208	0.363636

In [151]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model  import LogisticRegression
y_data = data['TenYearCHD']
x_data = data.drop('TenYearCHD', axis = 1)


lr=LogisticRegression(solver='lbfgs')


lr=lr.fit(x_data,y_data)
accuracy = lr.score(x_data,y_data)
print(accuracy)
```

```
0.8522975929978118
```

一.2

```
data1=pd.read_csv('C:/Users/Admin/Downloads/test_sample_1.csv')

y1_data = data1['TenYearCHD']
x1_data = data1.drop('TenYearCHD', axis = 1)


accuracy = lr.score(x1_data,y1_data)
print(accuracy)


a100=((1.28)**2)/(2*100)
b100=1.28*((((accuracy/100)-(((accuracy)**2)/100))+(((1.28)**2)/(4*((100)**2))))**0.5)
c100=(1+(((1.28)**2)/100))
print((accuracy+a100-b100)/c100)
print((accuracy+a100+b100)/c100)


u100=((1.65)**2)/(2*100)
v100=1.65*((((accuracy/100)-(((accuracy)**2)/100))+(((1.65)**2)/(4*((100)**2))))**0.5)
w100=(1+(((1.65)**2)/100))
print((accuracy+u100-v100)/w100)
print((accuracy+u100+v100)/w100)

u100=((2.58)**2)/(2*100)
v100=2.58*((((accuracy/100)-(((accuracy)**2)/100))+(((2.58)**2)/(4*((100)**2))))**0.5)
w100=(1+(((2.58)**2)/100))
print((accuracy+u100-v100)/w100)
print((accuracy+u100+v100)/w100)


data4=pd.read_csv('C:/Users/Admin/Downloads/test_sample_4.csv')
y4_data = data4['TenYearCHD']
```

```
accuracy4 = lr.score(x4_data,y4_data)
print(accuracy4)

e1000=((1.28)**2)/(2*1000)
f1000=1.28*((((accuracy4/1000)-(((accuracy4)**2)/1000))+(((1.28)**2)/(4*((1000)**2))))**0.5)
g1000=(1+(((1.28)**2)/1000))
print((accuracy4+e1000-f1000)/g1000)
print((accuracy4+e1000+f1000)/g1000)

n1000=((1.65)**2)/(2*1000)
o1000=1.65*((((accuracy4/1000)-(((accuracy4)**2)/1000))+(((1.65)**2)/(4*((1000)**2))))**0.5)
p1000=(1+(((1.65)**2)/1000))
print((accuracy4+n1000-o1000)/p1000)
print((accuracy4+n1000+o1000)/p1000)


x1000=((2.58)**2)/(2*1000)
y1000=2.58*((((accuracy4/1000)-(((accuracy4)**2)/1000))+(((2.58)**2)/(4*((1000)**2))))**0.5)
z1000=(1+(((2.58)**2)/1000))
print((accuracy4+x1000-y1000)/z1000)
print((accuracy4+x1000+y1000)/z1000)



data5=pd.read_csv('C:/Users/Admin/Downloads/test_sample_5.csv')
y5_data = data5['TenYearCHD']
x5_data = data5.drop('TenYearCHD', axis = 1)
accuracy5 = lr.score(x5_data,y5_data)
print(accuracy5)

h1800=((1.28)**2)/(2*1800)
```

```
print((accuracy5+h1800-i1800)/j1800)
print((accuracy5+h1800+i1800)/j1800)




q1800=((1.65)**2)/(2*1800)
r1800=1.65*((((accuracy5/1800)-(((accuracy5)**2)/1800))+(((1.65)**2)/(4*((1800)**2))))**0.5)
t1800=(1+(((1.65)**2)/1800))
print((accuracy5+q1800-r1800)/t1800)
print((accuracy5+q1800+r1800)/t1800)



zz1800=((2.58)**2)/(2*1800)
yy1800=2.58*((((accuracy5/1800)-(((accuracy5)**2)/1800))+(((2.58)**2)/(4*((1800)**2))))**0.5)
xx1800=(1+(((2.58)**2)/1800))
print((accuracy5+zz1800-yy1800)/xx1800)
print((accuracy5+zz1800+yy1800)/xx1800)

aa={'80%':[((accuracy+a100-b100)/c100),((accuracy4+e1000-f1000)/g1000),((accuracy5+h1800-i1800)/j1800)]
    ,'90%':[((accuracy+k100-l100)/m100),((accuracy4+n1000-o1000)/p1000),((accuracy5+q1800-r1800)/t1800)]
    ,'99%':[((accuracy+u100-v100)/w100),((accuracy4+x1000-y1000)/z1000),((accuracy5+zz1800-yy1800)/xx1800)]}
bb=pd.DataFrame(aa,index=['100 lower','1000 lower','1800 lower'])
print(bb)

cc={'80%':[((accuracy+a100+b100)/c100),((accuracy4+e1000+f1000)/g1000),((accuracy5+h1800+i1800)/j1800)]
    ,'90%':[((accuracy+k100+l100)/m100),((accuracy4+n1000+o1000)/p1000),((accuracy5+q1800+r1800)/t1800)]
    ,'99%':[((accuracy+u100+v100)/w100),((accuracy4+x1000+y1000)/z1000),((accuracy5+zz1800+yy1800)/xx1800)]}
dd=pd.DataFrame(cc,index=['100 upper','1000 upper','1800 upper'])
```

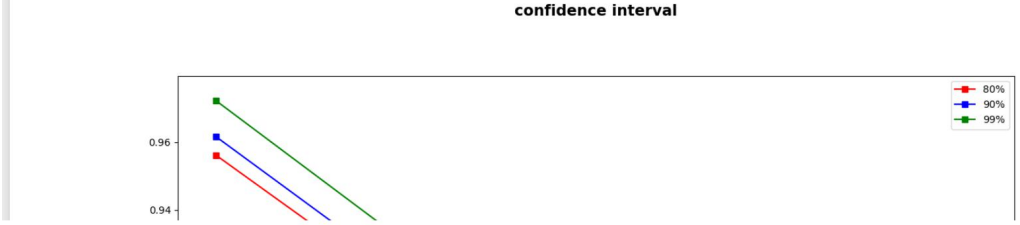
```
0.93
0.8899406428070985
0.9561962503356508
0.8755308600825573
0.9616761782975445
0.8340041590227228
0.9723234500096465
0.857
0.8422456518280976
0.8705864440659895
0.837763830777712
0.8742975820259301
0.8260749430838155
0.8832038137988706
0.8588888888888889
0.848058993832633
0.8690660408025083
0.8448069463103083
0.8718868321275822
0.8363937440183226
0.8787394711298374
                80%       90%       99%
100 lower   0.889941  0.875531  0.834004
1000 lower  0.842246  0.837764  0.826075
1800 lower  0.848059  0.844807  0.836394
                80%       90%       99%
100 upper   0.956196  0.961676  0.972323
1000 upper  0.870586  0.874298  0.883204
1800 upper  0.869066  0.871887  0.878739
```

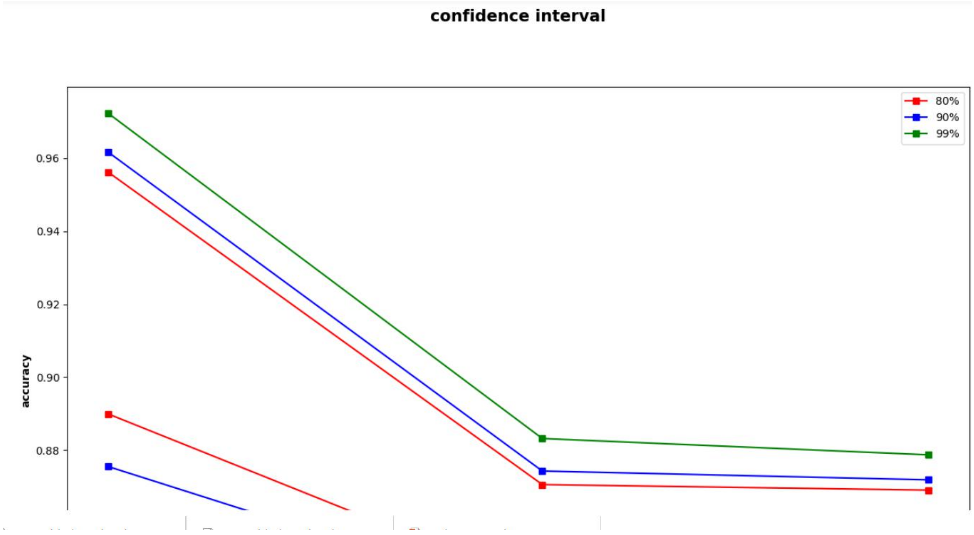
一.3

```
from matplotlib.font_manager import FontProperties
plt.figure(figsize=(15,10),dpi=100,linewidth = 2)
samples=[100,1000,1800]

plt.plot(samples,bb['80%'],'s-',color = 'r',label='80%')
plt.plot(samples,bb['90%'],'s-',color = 'b',label='90%')
plt.plot(samples,bb['99%'],'s-',color = 'g',label='99%')
plt.plot(samples,dd['80%'],'s-',color = 'r')
plt.plot(samples,dd['90%'],'s-',color = 'b')
plt.plot(samples,dd['99%'],'s-',color = 'g')
plt.xlabel("samples", fontweight = "bold")
plt.ylabel("accuracy", fontweight = "bold")
plt.legend( loc = 'best')
plt.title("confidence interval", fontsize = 15, fontweight = "bold", y = 1.1)
```

```
Out[155]: Text(0.5, 1.1, 'confidence interval')
```





一.4

In [156]:
```
#(a)樣本數越多:信賴區間越小，代表越準確，越小的範圍中資料反而比較多
#(b)信心水準越大，信賴區間越大，表示較多的資料在這範圍中
```

二.1

In [158]:
```
from sklearn.model_selection import cross_val_score,cross_val_predict
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from sklearn.naive_bayes import GaussianNB
data = data.fillna(0)
clf = GaussianNB()
X=data.drop('diagnosis', axis = 1)
y=data['diagnosis']
clf = clf.fit(X,y)
```

In [159]:
```
acc=cross_val_score(clf,X,y,cv=10,scoring='accuracy')
scores = cross_val_predict(clf,X,y,cv=10)
print(acc)
print(acc.mean())
print(scores)
```

```
[0.61403509 0.57894737 0.63157895 0.63157895 0.63157895 0.63157895
 0.63157895 0.59649123 0.63157895 0.625     ]
0.6203947368421053
['B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
```

In [160]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y,scores)
```

Out[160]:
```
array([[351,   6],
       [210,   2]], dtype=int64)
```

In [161]:
```
from sklearn import svm
data = data.fillna(0)
clf = svm.SVC(kernel='rbf',C=1,gamma='auto')
X=data.drop('diagnosis', axis = 1)
y=data['diagnosis']
clf = clf.fit(X,y)
```

In [162]:
```
score= cross_val_score(clf,X,y,cv=10,scoring='accuracy')
scores = cross_val_predict(clf,X,y,cv=10)
print(score)
print(score.mean())
print(scores)
```

```
[0.61403509 0.61403509 0.63157895 0.63157895 0.63157895 0.63157895
 0.63157895 0.63157895 0.63157895 0.625     ]
0.6274122807017544
['B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
```

In [163]:
```
confusion_matrix(y,scores)
```

Out[163]:
```
array([[357,   0],
       [212,   0]], dtype=int64)
```

In [164]:
```
from sklearn import tree
data = data.fillna(0)
clf = tree.DecisionTreeClassifier(criterion='gini',max_depth=3)
X=data.drop('diagnosis', axis = 1)
y=data['diagnosis']
clf = clf.fit(X,y)
```

In [165]:
```
score= cross_val_score(clf,X,y,cv=10,scoring='accuracy')
scores = cross_val_predict(clf,X,y,cv=10)
print(score)
print(score.mean())
print(scores)
```

```
[0.9122807  0.84210526 0.9122807  0.89473684 0.98245614 0.89473684
 0.89473684 0.94736842 0.85964912 0.96428571]
0.9104636591478696
['M' 'M' 'M' 'M' 'M' 'M' 'M' 'B' 'M' 'M' 'B' 'M' 'M' 'B' 'M' 'M' 'M' 'M'
 'M' 'B' 'B' 'B' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'M'
 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'M' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'M'
 'M' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'M' 'B' 'M' 'M' 'B' 'B' 'M' 'B' 'M' 'B'
 'M' 'B' 'B' 'M' 'B' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'M' 'B' 'M' 'B' 'B'
 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B'
 'M' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'M' 'B' 'B' 'B'
 'B' 'M' 'B' 'M' 'B' 'M' 'M' 'B' 'M' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'B' 'B'
 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'M']
```



```
'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'M' 'B' 'B' 'B'
'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'M' 'B' 'M'
'B' 'B' 'B' 'B' 'M' 'B' 'M' 'B' 'B' 'B' 'B' 'M' 'B' 'M' 'B' 'M' 'B' 'B'
'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B'
'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'M' 'M' 'M' 'B' 'M' 'M' 'B' 'M' 'M' 'B'
'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'M' 'B' 'B'
'M' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B'
'M' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B'
'M' 'M' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'B' 'M' 'B' 'M' 'M'
'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B'
'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'M' 'B'
'M' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'M' 'M' 'M' 'M' 'B' 'M'
'M' 'M' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'M' 'M' 'B' 'B' 'M' 'M'
'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'M' 'B' 'B' 'B' 'B'
'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'
'B' 'B' 'B' 'B' 'M' 'M' 'M' 'M' 'M' 'M' 'B']
```

In [166]:
```
confusion_matrix(y,scores)
#由上述比較可知，decision預測準確度>SVM>naive
```

Out[166]:
```
array([[336,  21],
       [ 30, 182]], dtype=int64)
```