In [1]:
```
import pickle
ofname = open('/Users/xiyongzhang/documents/MQ/RA_ACST890_notes/dataset_small.pkl','rb')
(x,y)=pickle.load(ofname,encoding='bytes')
```

In [2]:
```
# The first model is KNN

from sklearn import neighbors
# Create an instance of K-nearest neighbor classifier
# We choose k=11
knn = neighbors.KNeighborsClassifier(n_neighbors = 11)

# Estimator interface
knn.fit(x,y)

# Predictor interface
yhat=knn.predict(x)
yhat
```

Out[2]:
```
array([-1., -1., -1., ..., -1., -1., -1.])
```

In [3]:
```
# Score method gives accuracy of model
knn.score(x,y)
```

Out[3]:
```
0.83164251207729467
```

In [4]:
```
# Constructing a confusion matrix

# TP: correct prediction on rejected loan
# FP: false prediction on rejected loan
# FN: false prediction on accepted loan
# TN: correct prediction on accepted loan

# Golden standard:
# TP FP
# FN TN
import numpy as np
TP=sum(np.logical_and(y==-1, yhat==-1))
FP=sum(np.logical_and(y==1, yhat==-1))
FN=sum(np.logical_and(y==-1, yhat==1))
TN=sum(np.logical_and(y==1, yhat==1))
print('TP: ', TP, ', FP: ', FP)
print('FN: ', FN, ', TN: ', TN)
```

```
TP:  3370 , FP:  690
FN:  7 , TN:  73
```

In [5]:
```
# Short cut is sklearn.metrics
from sklearn import metrics
```

```
        # Accuracy score
        print(metrics.accuracy_score(yhat, y))

        # Confusion matrix
        metrics.confusion_matrix(yhat, y, labels=[-1,1])
        # Lables assigns variable orders
        # in this case, "-1", rejection of loan, is treated as positive
```

```
0.831642512077
```

```
Out[5]: array([[3370,  690],
               [   7,   73]])
```

```
In [ ]:
```

```
In [6]: # sklean.cross_validation will be removed in next update
        # Here, we will use sklearn.model_selection
```

```
In [7]: # Train/Test split and Cross sampling
        from sklearn.model_selection import train_test_split
```

```
In [8]: # train_test_split(x,y) gives output as list
        # assign testing size to be 30%
        PRC=0.3
        X_train , X_test , y_train , y_test = train_test_split(x, y, test_size = PRC)
```

```
In [9]: knn = neighbors.KNeighborsClassifier(n_neighbors = 11)
        knn.fit(X_train,y_train)
        yhat_test=knn.predict(X_test)
        print(metrics.accuracy_score(yhat_test, y_test))
        metrics.confusion_matrix(yhat_test, y_test, labels=[-1,1])
```

```
0.834943639291
```

```
Out[9]: array([[1023,  199],
               [   6,   14]])
```

```
In [ ]:
```

```
In [10]: # K-fold Cross validation
         from sklearn import model_selection

         # Firstly, create a CV class
         # n_splits: K
         # shuffle to indicate random selection
         cv=model_selection.KFold(
             n_splits = 2,shuffle=True)
```

```
        # Split method will give a set of INDEX to split data
        cv=cv.split([1,2,3,4,5,6,7,8,9,10])

        # Output is in form:
        # [ split 1: [ Test ], [ Train ] ]
        # [ split 2: [ Test ], [ Train ] ]
        # [ split 3: [ Test ], [ Train ] ]
        # ...
        for i in cv:
            print(i[0],i[1])

        #### Remember: this is index, not data ####
```

```
[1 3 7 8 9] [0 2 4 5 6]
[0 2 4 5 6] [1 3 7 8 9]
```

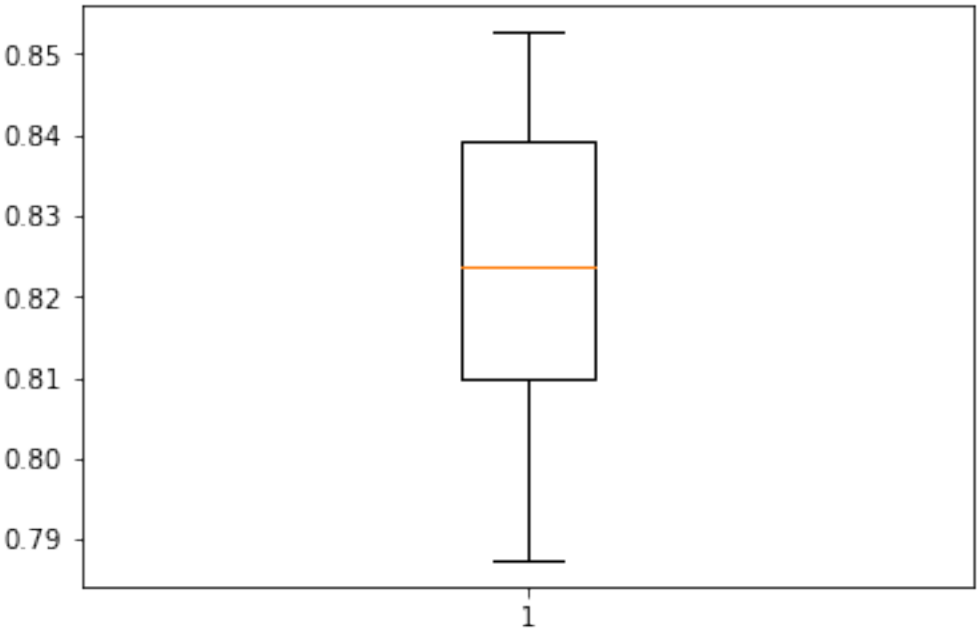
```
In [11]: # Loan example: 10-fold CV
         k_fold=10
         cv=model_selection.KFold(
             n_splits = k_fold,shuffle=True)
         cv=cv.split(x)

         # Make a place-holder for model score
         score_11nn=np.asarray([])

         # Loop to train/test KNN model, k_knn=11
         k_knn=11
         for train, test in cv:
         # Take every element of CV and split into train and test
             x_train=x[train]
             y_train=y[train]
             x_test=x[test]
             y_test=y[test]
             # Train and score model
             knn = neighbors.KNeighborsClassifier(n_neighbors = k_knn)
             knn.fit(x_train,y_train)
             yhat_test=knn.predict(x_test)
             score=metrics.accuracy_score(yhat_test, y_test)
             score_11nn=np.append(score_11nn,score)
         print(score_11nn)
```

```
[ 0.83091787  0.852657    0.79951691  0.78743961  0.80917874  0.8115942
  0.84057971  0.84541063  0.81642512  0.83574879]
```

```
In [12]: import matplotlib.pyplot as plt
         boxp=plt.boxplot(score_11nn)
         plt.show()
```


0.85
0.84
0.83
0.82
0.81
0.80
0.79
1