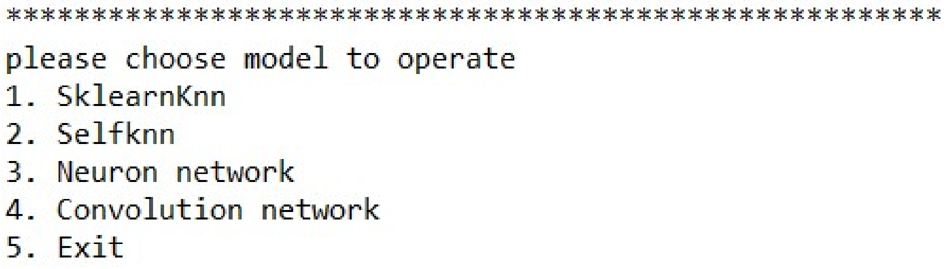
1. check list indicating

F1	Neural networks	Model and save	√ in def **modle_neuron_network**:
	Convolutional neural networks	Model and save	√ in def **modle_convolution_network**:
F2	sklearn	cross validation	√ in def **cross_validation_sklearnknn**: def **getAccuracy**:
		confusion matrix	√ in def **confusion_matrix_sklearnknn**: def **plot_confusion_matrix**:
	selfknn	cross validation	√ in def **cross_validation_selfknn**: def **getAccuracy**:
		confusion matrix	√ in def **confusion_matrix_selfknn**: def **plot_confusion_matrix**:
	Neural networks	cross validation	√ in def **cross_validation_neuron__network:** def **getAccuracy:**
		confusion matrix	√ in def **confusion_matrix_neuron__network:** def **plot_confusion_matrix**:
		ROC curve	√ in def **roc_neuron_network:** def **roc_curve**:
	Convolutional neural networks	cross validation	√ in def **cross_validation_convolution_network:** def **getAccuracy:**
		confusion matrix	√ in def **confusion_matrix_convolution_network:** def **plot_confusion_matrix:**
		ROC curve	√ in def **roc_convolution_network:** def **roc_curve:**
	discussion on the discovery		√ **in this document**

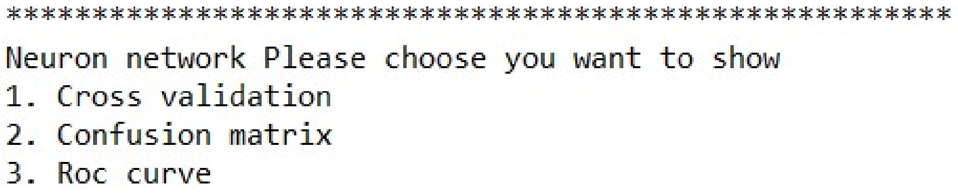
2. How to run

- Just run the code, then choose model to show (enter 1-5)



```
*******************************************************
please choose model to operate
1. SklearnKnn
2. Selfknn
3. Neuron network
4. Convolution network
5. Exit
```

- Choose which component want to execute.

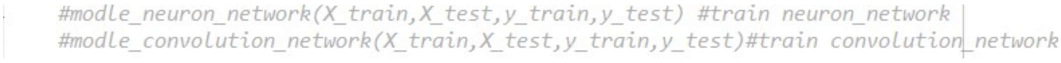


```
*******************************************************
Neuron network Please choose you want to show
1. Cross validation
2. Confusion matrix
3. Roc curve
```

- You can re choose model to show (enter 1-5)(this program is loop, you can choose you model and execute component repeat)

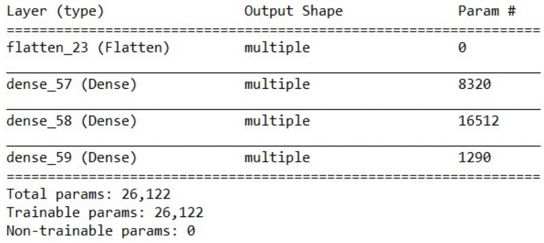
3. **Additional Requirements**

- **Additional requirement 1:** This two lines in def **main()**: are how train and save Neural networks and Convolutional neural networks, I already delete it, and results of functionality **f1** by loading the saved models, without training, when it required.



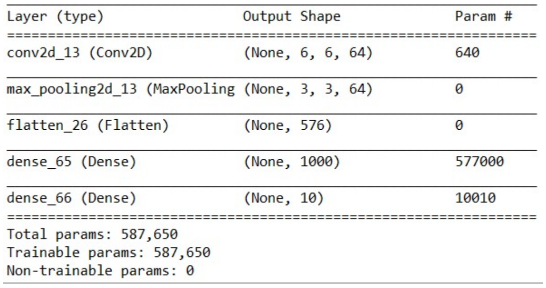
```
#modle_neuron_network(X_train,X_test,y_train,y_test) #train neuron_network
#modle_convolution_network(X_train,X_test,y_train,y_test)#train convolution_network
```

- **Additional requirement 2:**
  In Neural networks without convolutional:
  add 4 layers, which one have 1-128-128-10 units. Use adam optimizer to model, and use **model.save('network_model.h5')** to save it.



```
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_23 (Flatten)         multiple                  0         
_________________________________________________________________
dense_57 (Dense)             multiple                  8320      
_________________________________________________________________
dense_58 (Dense)             multiple                  16512     
_________________________________________________________________
dense_59 (Dense)             multiple                  1290      
=================================================================
Total params: 26,122
Trainable params: 26,122
Non-trainable params: 0
```

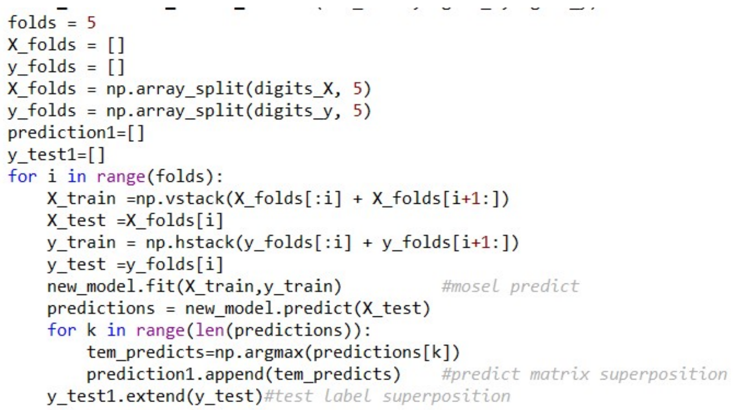
  In convolutional Neural networks:
  add 5 layers, which one have 64-64-576-1000-10 units. Use adam optimizer to model, and epochs=3 to train 3 times. Use **model.save(' convolution_network_model.h5')** to save it.



```
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_13 (Conv2D)           (None, 6, 6, 64)          640       
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 3, 3, 64)          0         
_________________________________________________________________
flatten_26 (Flatten)         (None, 576)               0         
_________________________________________________________________
dense_65 (Dense)             (None, 1000)              577000    
_________________________________________________________________
dense_66 (Dense)             (None, 10)                10010     
=================================================================
Total params: 587,650
Trainable params: 587,650
Non-trainable params: 0
```

4. Cross validation

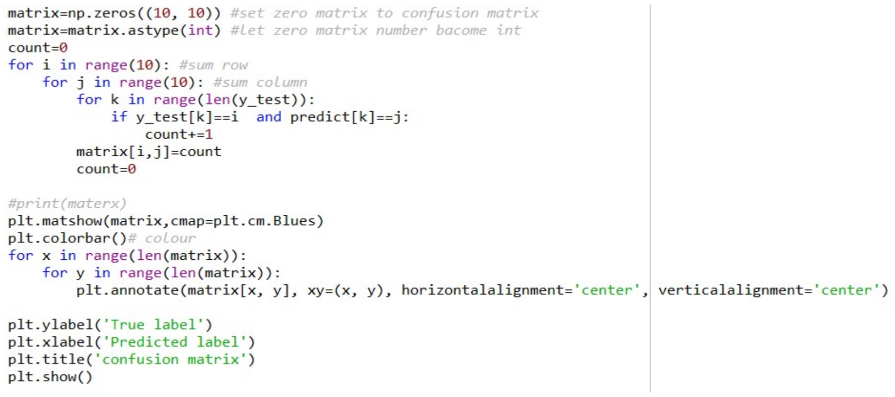
Split dataset into 5 folds, then set the first, second, third, fourth and fifth folds as test dataset, the others as train dataset respectively. Get all predicts and get accuracy average.



```
folds = 5
X_folds = []
y_folds = []
X_folds = np.array_split(digits_X, 5)
y_folds = np.array_split(digits_y, 5)
prediction1=[]
y_test1=[]
for i in range(folds):
    X_train =np.vstack(X_folds[:i] + X_folds[i+1:])
    X_test =X_folds[i]
    y_train = np.hstack(y_folds[:i] + y_folds[i+1:])
    y_test =y_folds[i]
    new_model.fit(X_train,y_train)          #mosel predict
    predictions = new_model.predict(X_test)
    for k in range(len(predictions)):
        tem_predicts=np.argmax(predictions[k])
        prediction1.append(tem_predicts)     #predict matrix superposition
    y_test1.extend(y_test)#test label superposition
```

5. Confusion matrix

Get every model predicts, then if true value=predict value, count it. Then save count as matrix, draw it.



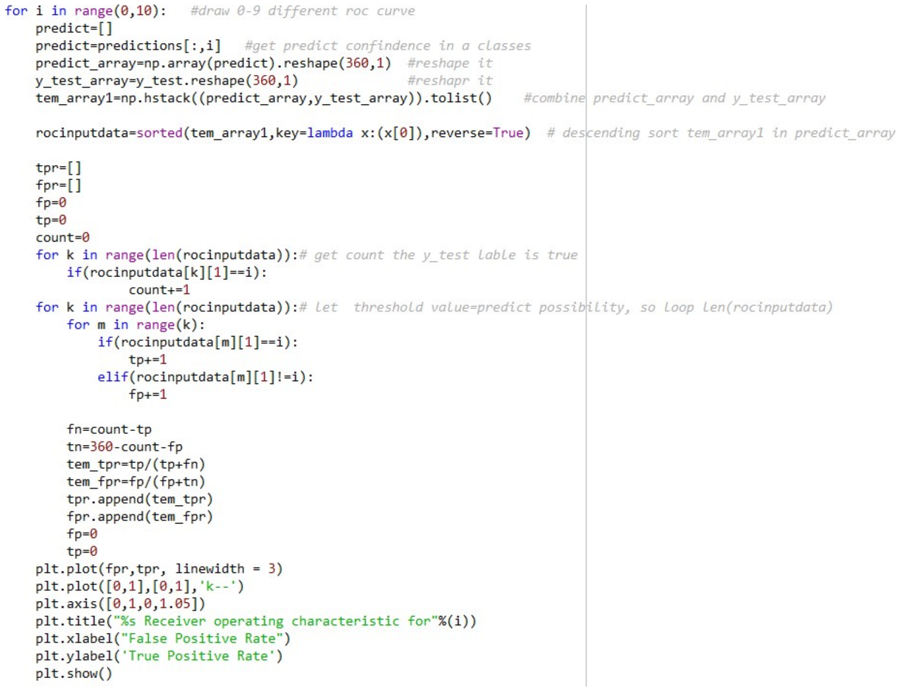
```
matrix=np.zeros((10, 10)) #set zero matrix to confusion matrix
matrix=matrix.astype(int) #let zero matrix number bacome int
count=0
for i in range(10): #sum row
    for j in range(10): #sum column
        for k in range(len(y_test)):
            if y_test[k]==i  and predict[k]==j:
                count+=1
        matrix[i,j]=count
        count=0

#print(materx)
plt.matshow(matrix,cmap=plt.cm.Blues)
plt.colorbar()# colour
for x in range(len(matrix)):
    for y in range(len(matrix)):
        plt.annotate(matrix[x, y], xy=(x, y), horizontalalignment='center', verticalalignment='center')

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.title('confusion matrix')
plt.show()
```

6. ROC curve (because draw roc needs predict confidence to compare with threshold value. But knn algorithm didn't have predict confidence, because knn algorithm just choose Nearest Neighbor, so didn't have predict confidence)

- Because we have 10 class, so I choose 1 class as true, the others are false. loop 10 times to get every roc curve.
- Get every model predict confidence in 1 class, then put confidence and labels in same matrix, descending order it in confidence. Choose every confidence as threshold value respectively to count tpr and fpr(in every loop, I just choose before threshold value's rows in matrix, it can reduce decision whether predict confidence > threshold value, because matrix is descending order).
- Get the tpr and fpr in every threshold value, combine these as matrix, draw it.



```
for i in range(0,10):   #draw 0-9 different roc curve
    predict=[]
    predict=predictions[:,i]   #get predict confindence in a classes
    predict_array=np.array(predict).reshape(360,1)  #reshape it
    y_test_array=y_test.reshape(360,1)              #reshapr it
    tem_array1=np.hstack((predict_array,y_test_array)).tolist()    #combine predict_array and y_test_array

    rocinputdata=sorted(tem_array1,key=lambda x:(x[0]),reverse=True)  # descending sort tem_array1 in predict_array

    tpr=[]
    fpr=[]
    fp=0
    tp=0
    count=0
    for k in range(len(rocinputdata)):# get count the y_test lable is true
        if(rocinputdata[k][1]==i):
                count+=1
    for k in range(len(rocinputdata)):# let  threshold value=predict possibility, so loop len(rocinputdata)
        for m in range(k):
            if(rocinputdata[m][1]==i):
                tp+=1
            elif(rocinputdata[m][1]!=i):
                fp+=1

        fn=count-tp
        tn=360-count-fp
        tem_tpr=tp/(tp+fn)
        tem_fpr=fp/(fp+tn)
        tpr.append(tem_tpr)
        fpr.append(tem_fpr)
        fp=0
        tp=0
    plt.plot(fpr,tpr, linewidth = 3)
    plt.plot([0,1],[0,1],'k--')
    plt.axis([0,1,0,1.05])
    plt.title("%s Receiver operating characteristic for"%(i))
    plt.xlabel("False Positive Rate")
    plt.ylabel('True Positive Rate')
    plt.show()
```

7. discussion on the discovery

Discovery: neuron network has better accuracy than traditional machine learning algorithms

Algorithm	sklearn	selfknn	Neuron network Without Convolutional	Convolutional neural networks
Accuracy	0.96438	0.9287	0.9855	0.99721

Accuracy: Convolutional neural networks> Neuron network without convolutional> sklearn> selfknn

Discussion: neural networks have sparsity of connections, however knn algorithm is full connection. In the recognition of handwritten digits, every features are different, if use full connection, every features may effect, so neural networks is better than traditional machine learning algorithms in recognition of handwritten digits.