Checklist:

Required function		Implement
Functionality 1 F1: (50%)	Design deep neural network with convolutional layer	completed successfully
	Design deep neural network without convolutional layer	completed successfully
Functionality 2 F2: (50%)	Cross-validation	completed successfully
	Confusion matrix	completed successfully
	ROC curve	completed part
	Discussion on the discovery	completed successfully

1. Detailing how to run your program, including the software dependencies

Software Dependencies:

- Python 3.8
- Numpy 1.19.5
- Scikit-learn 0.24.0
- Keras 2.4.3
- Tensorflow 2.4.0
- Matplotlib 3.3.3

How to run program:

Clone to your development environment and run *main.py* with python 3.7 interpreter: python main.py

User interface of python program:

```
1. F1 Load two model and see model construction
2. F2 Model evaluation
3. End
Enter number from 1 to 3:
```

The user selects numbers between 1 and 3 to run different implement, other numbers are not accepted.

2. Explaining how the functionalities and additional requirements are implemented

F1: Use `keras` framework as framework to design and build two different deep neural networks, one with convolutional layer and the other without convolutional layer. And save the model for loading.

F2: Model evaluation

(a) The cross-validation of 5 subsamples

Manually divide five subsets in the given data set for cross validation. Calculate and output the accuracy (and loss rate) of each algorithm

	train set	test set	DNN	CNN	GaussianNB	my_nb
1	$S_2\ S_3\ S_4\ S_5$	S_1	0.92	0.97	0.79	0.91
2	$S_1\ S_3\ S_4\ S_5$	S_2	0.91	0.94	0.79	0.84
3	$S_1\ S_2\ S_4\ S_5$	S_3	0.92	0.97	0.79	0.84
4	$S_1\ S_2\ S_3\ S_5$	S_4	0.97	0.97	0.87	0.89
5	$S_1\ S_2\ S_3\ S_4$	S_5	0.90	0.94	0.81	0.91
Accuracy			0.924	0.958	0.81	0.878

(b) Confusion matrix

First declare a 10*10 null array, iterate and loop through to determine if the predicted value and label are the same, if so, increment the position by 1, resulting in a 10*10 two-dimensional matrix confusion matrix. Then use `matplotlib.pyplot` to output the graph. These graphs are shown below.

(c) ROC curve

Each predicted value and test dataset label are binarized and classified. Then use `roc_curve` and `auc` to calculate the ROC curve of the algorithm. Formula is

$$true\ postivie\ rate(recall) = \frac{TP}{actual\ pos} = \frac{TP}{TP + FN}$$

$$false\ postivie\ rate = \frac{FP}{actual\ neg} = \frac{FP}{TP + FP}$$

In the end, use `matplotlib.pyplot` to output the graph.

(d) Discussion on the discovery

By comparing the results for Assessment 1 and Assessment 2. In the same dataset. Traditional machine learning takes the least time and does not require high performance but need to require more human resources to achieve ideal accuracy. Deep neural network only needs a simple setting to obtain a very ideal accuracy rate. However, the number of parameters in the neural network is very large, and many optimizers, such as RMSProp, SGD, ADAM, etc. are needed to find the local optimal solution. This algorithm is the most time-consuming and most dependent on computer performance.

The appropriate algorithm can be selected according to the requirements of the scene. Traditional machine learning is suitable for scenes with high real-time requirements and low data volume. Neural network is suitable for the scene with very large amount of data and high accuracy

3. Providing the details of your implementation

1. The meaning of parameters and variables

`all_x:` all the data from datasets

`all_y:` all the target of each data from dataset

`tra_x,tes_x,tra_y,tes_y:` Divide the datasets into train and test dataset

`pre_y:` The predicted value(target) obtained by algorithm is used for comparison with `tes_y`

`score:` The accuracy value (and loss value) for the model in test mode

`model:` Model of the algorithm

2. The description of your model evaluation

Two neural networks :

`def dnn():`

Create five layers by `keras.sequential`(show in Figure 1), two of which are `Dropout` to prevent overfitting.

Optimizer selects `RMSProp`, which is Gradient Descent.

Loss selects `categorical_crossentropy`, the expression is

$$CE(x) = -\sum_{i=1}^{C} y_i \log f_i(x)$$

```
DNN model construction
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 512)               33280
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 301,066
Trainable params: 301,066
Non-trainable params: 0
```

Figure 1

`def cnn():`

Create five layers by `keras.sequential`(show in Figure 2), the `Flatten` layer is used for the transition from the convolution layer to the full join layer, without affecting the batch size.

Optimizer select `Adam`, the parameters are lr=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-08

Loss selects `categorical_crossentropy`.

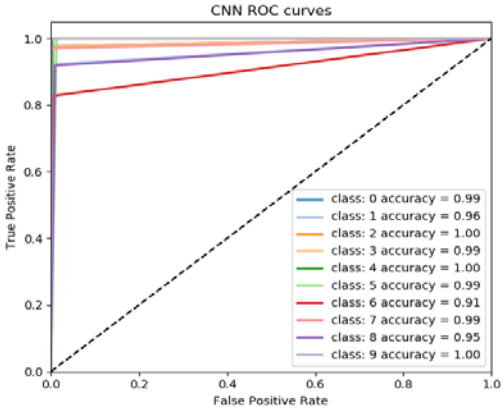
```
CNN model construction
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 6, 6, 32)          320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 4, 4, 64)          18496
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               131200
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 151,306
Trainable params: 151,306
Non-trainable params: 0
```

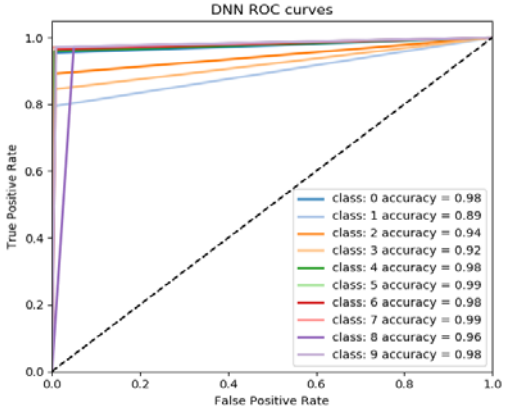
Figure 2

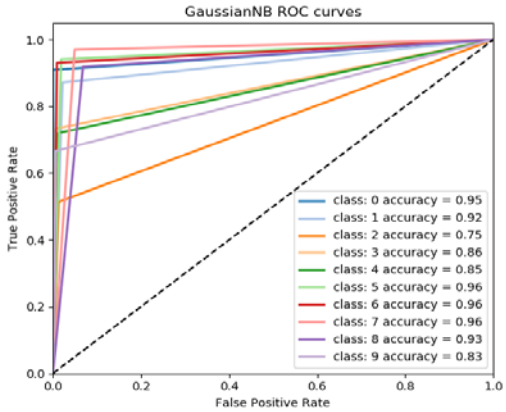
In the end, these models will be saved, respectively are *CNN_model.h5* and *DNN_model.h5* (`model.save`)

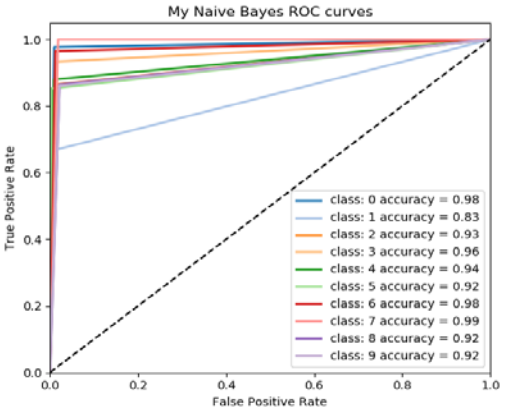
4. Results of graph

ROC cruve









Confusion matrix