```
2.204523   0.317883   2.080388   0.5615
1     1.938731   0.660933   1.770403   0.7467
2     1.609755   0.760000   1.423116   0.7956
3     1.283058   0.789517   1.116987   0.8168
4     1.021336   0.810200   0.891695   0.8334
5     0.837351   0.825583   0.740864   0.8465
6     0.714310   0.838500   0.639969   0.8572
7     0.630683   0.849867   0.570973   0.8668
8     0.571758   0.858650   0.521652   0.8742
9     0.528109   0.865400   0.484234   0.8806
10    0.494684   0.871100   0.455319   0.8856
11    0.468315   0.875900   0.432311   0.8901
12    0.446845   0.879833   0.413490   0.8930
13    0.429000   0.883817   0.397645   0.8955
14    0.413854   0.886983   0.384456   0.8983
metrics.plot()

metrics[['loss','val_loss']].plot()

metrics[['accuracy','val_accuracy']].plot()
```

(ii).Evaluate the Model

```
model.evaluate(x_test,y_cat_test,verbose=0)
     #loss      |       #accuracy
[0.38445618748664856, 0.8982999920845032]
```

```
from sklearn.metrics import classification_report,confusion_matrix
predict_x=model.predict(x_test)
classes_x=np.argmax(predict_x,axis=1)
313/313 [==============================] - 2s 7ms/step
print(classification_report(y_test,classes_x))
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       980
           1       0.95      0.98      0.96      1135
           2       0.90      0.87      0.88      1032
           3       0.88      0.89      0.89      1010
           4       0.90      0.91      0.90       982
           5       0.90      0.81      0.85       892
           6       0.91      0.93      0.92       958
           7       0.92      0.87      0.89      1028
           8       0.84      0.86      0.85       974
           9       0.85      0.88      0.87      1009

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000

print(confusion_matrix(y_test,classes_x))
[[ 954    0    4    4    0    2    6    1    7    2]
 [   0 1109    3    4    1    1    4    0   13    0]
 [  13    4  894   15   18    0   23   20   44    1]
 [   3    1   22  903    0   33    4   16   23    5]
```

```
 [   1     3     3     0  892     0    21     2     6    54]
 [  12     5     5    53   17   720    20 1    41    18]
 [  14     6    12     2   10    19   887 1     7     0]
 [   1    19    33     3   12 0     1   895     9    55]
 [  10     9    12    29     7    23     8    17   841    18]
 [  15    10     6    14    36     4     0    25    11   888]]
import seaborn as sns
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,classes_x))
```

(iii).Make Prediction

```
my_num = x_test[1]
classes_x
array([7, 2, 1, ..., 4, 8, 6])
plt.imshow(my_num.reshape(28,28))
```

(iv).Save the Model

```
from tensorflow.keras.models import load_model

model.save('CNN.h5')
print('Model Saved!')

savedModel=load_model('CNN.h5')
```

```
savedModel.summary()

Model Saved!

Model: "sequential"

_________________________________________________________________

 Layer (type)              Output Shape              Param #

=================================================================

 conv2d (Conv2D)           (None, 25, 25, 32)        544


 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)     0

 )


 flatten (Flatten)         (None, 4608)              0


 dense (Dense)             (None, 128)               589952


 dense_1 (Dense)           (None, 10)                1290


=================================================================

Total params: 591,786

Trainable params: 591,786

Non-trainable params: 0

_________________________________________________________________
```