TEAM:PNT2022-TMID10395

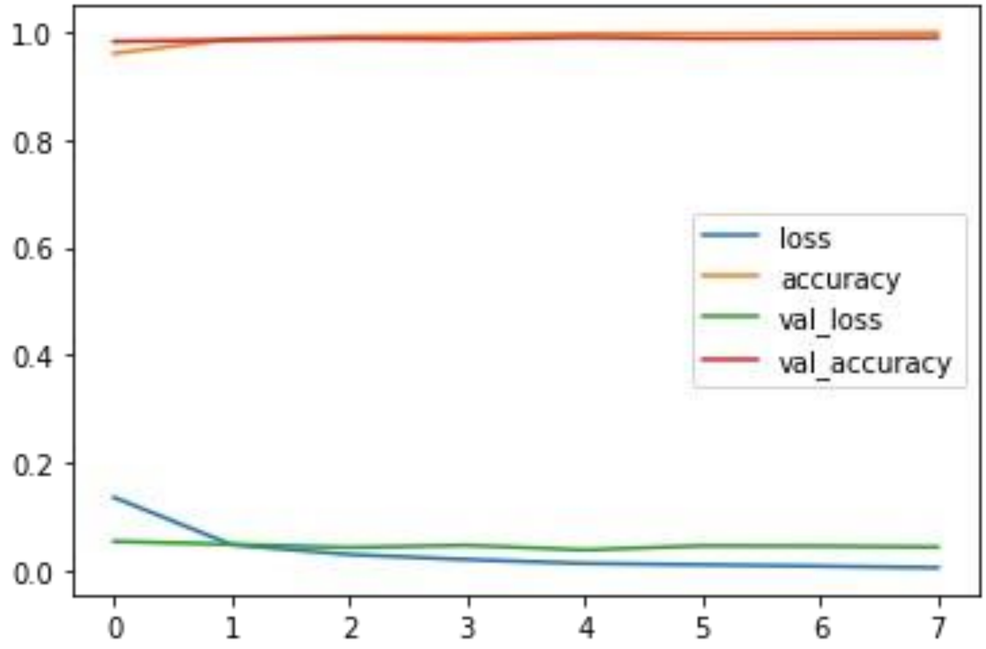
#Performance Analysis (sprint-3) (i).Performance Analysis

```
metrics = pd.DataFrame(model.history.history)
```

```
metrics
```

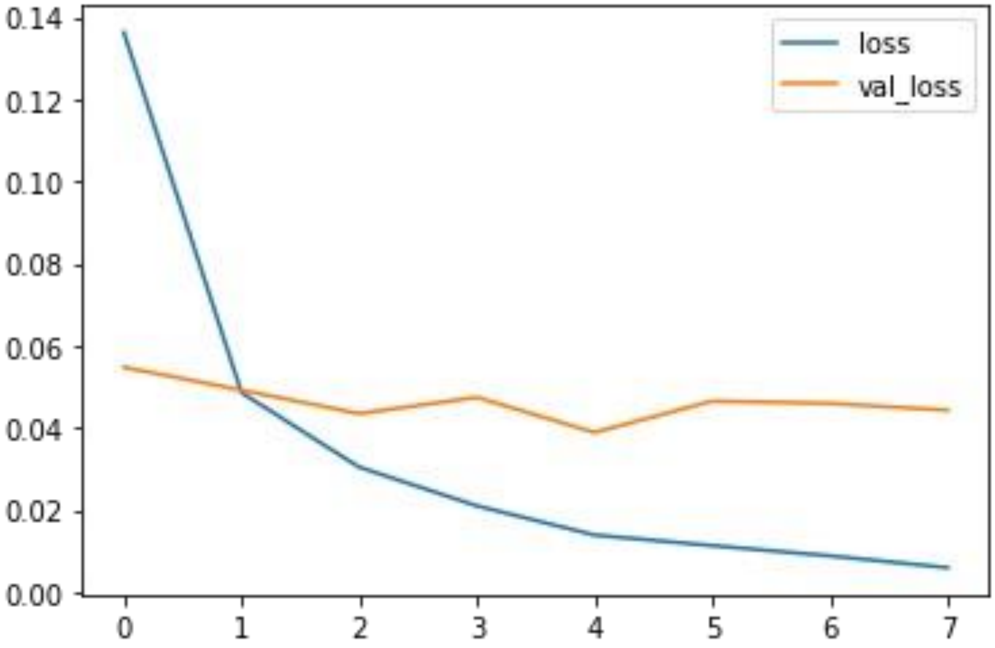
```
       loss  accuracy  val_loss  val_accuracy
0  0.136240  0.959183  0.054753        0.9811
1  0.048557  0.985233  0.049157        0.9839
2  0.030406  0.990800  0.043443        0.9861
3  0.020990  0.993350  0.047409        0.9850
4  0.013883  0.995450  0.038858        0.9890
5  0.011308  0.996183  0.046504        0.9865
6  0.008813  0.996933  0.045933        0.9875 7   0.005928

   0.997917  0.044267        0.9886 metrics.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9be00620d0>
```



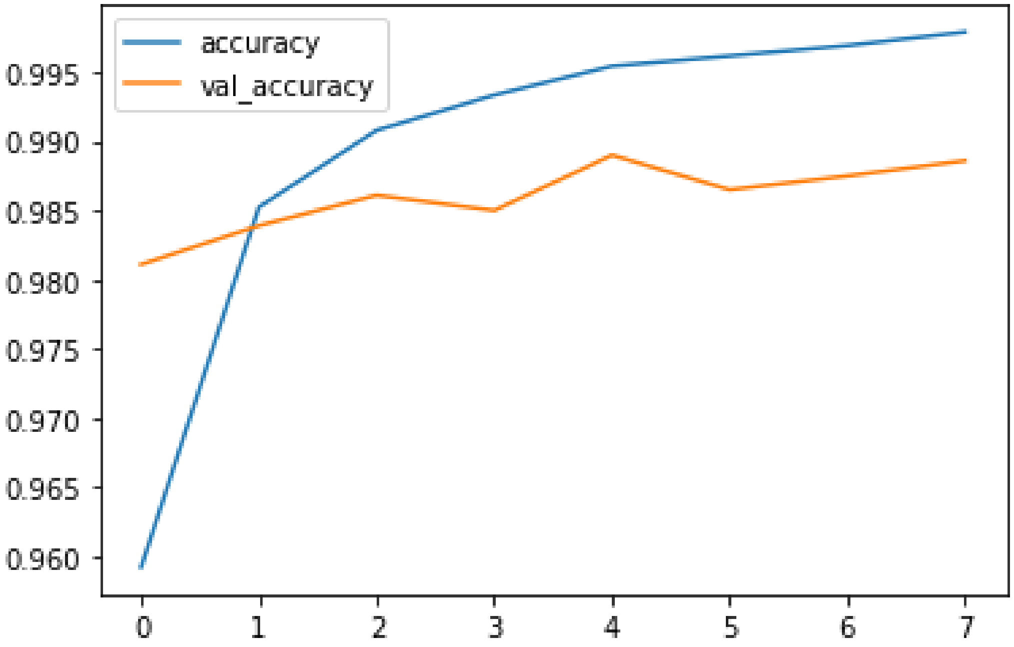
```
metrics[['loss','val_loss']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9b8a38eb90>
```



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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9b8a2a36d0>
```



(ii).Evaluate the Model

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

```
[0.04426722601056099, 0.9886000156402588]
from sklearn.metrics import classification_report,confusion_matrix
```

```
predict_x=model.predict(x_test)
classes_x=np.argmax(predict_x,axis=1)
```

```
313/313 [==============================] - 1s 2ms/step
```

```
print(classification_report(y_test,classes_x))
```

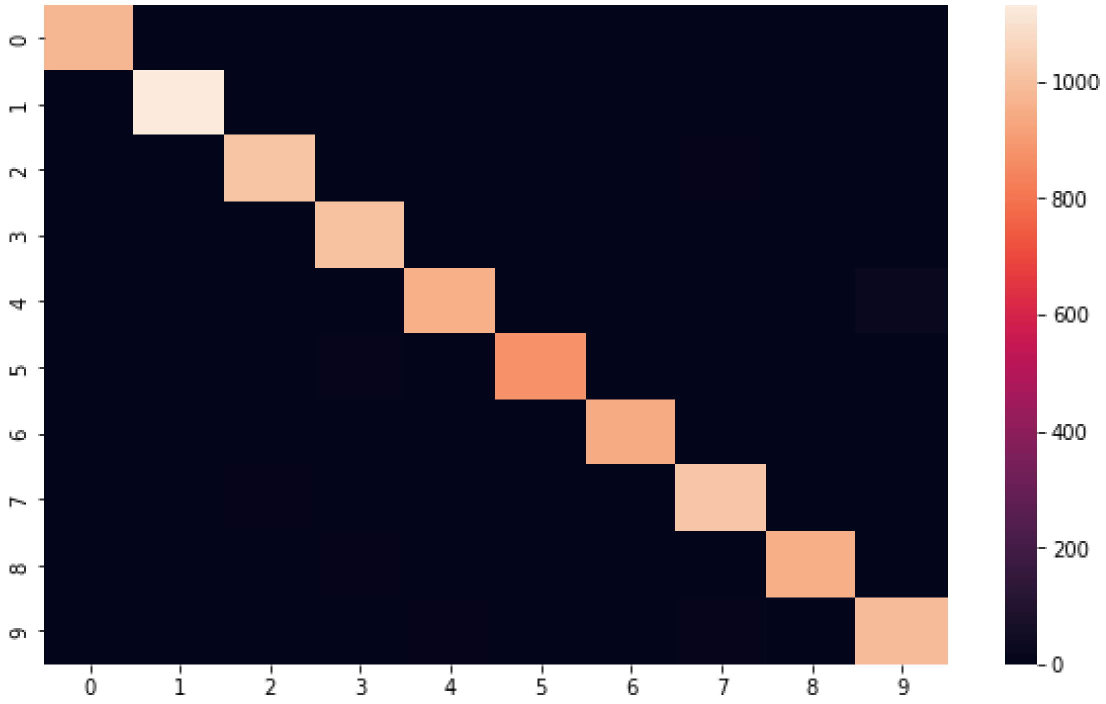
```
precision    recall  f1-score   support

0       0.99      1.00      0.99       980
1       0.99      1.00      1.00      1135
2       0.99      0.99      0.99      1032
3       0.98      1.00      0.99      1010
4       0.99      0.98      0.99       982
5       1.00      0.99      0.99       892
6       0.99      0.98      0.99       958
7       0.98      0.99      0.99      1028
8       0.99      0.98      0.99       974             9       0.98
        0.98      0.98      1009

    accuracy                           0.99     10000
macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

```
print(confusion_matrix(y_test,classes_x))
```

```
[[ 977    0    0    1    0    0    1    0    1    0]
 [    0 1132    1    2    0    0    0    0    0    0]
 [    1    1 1017    2    0    0    2    7    2    0]
 [    0    0    1 1005    0    1    0    1    1    1]
 [    0    0    0    0  963    0    1    0    0   18]
 [    0    0    0   12    0  879    1    0    0    0]
 [    4    2    1    0    4    2  943    0    2    0]
 [    0    2    5    0    0    0    0 1019    1    1]
 [    4    0    1    6    0    0    0    1  959    3]
[   0    1    0    1    5    1    0    9    0  992]]
```

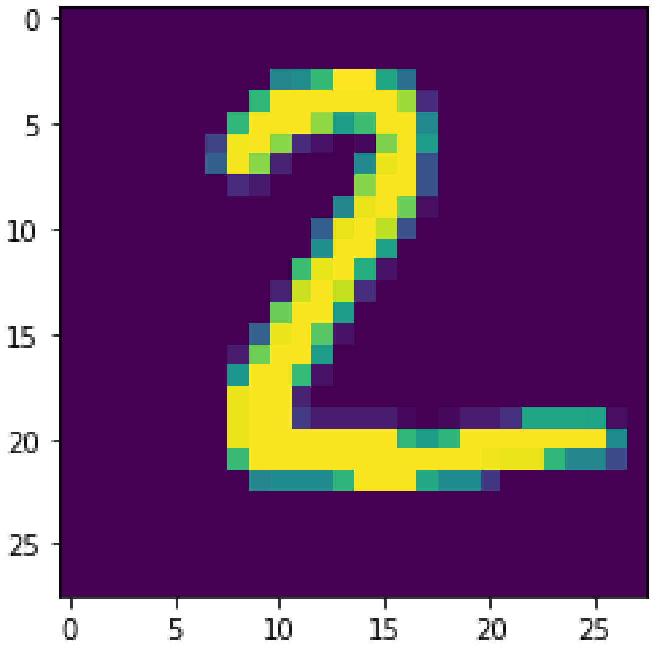
```
import seaborn as sns
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_test,classes_x))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9b73f53750>
```



(iii).Make Prediction

```
my_num = x_test[1] classes_x

array([7, 2, 1, ..., 4, 5, 6])

plt.imshow(my_num.reshape(28,28))

<matplotlib.image.AxesImage at 0x7f9b73a95b10>
```



(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

_________________________________________________________________
```