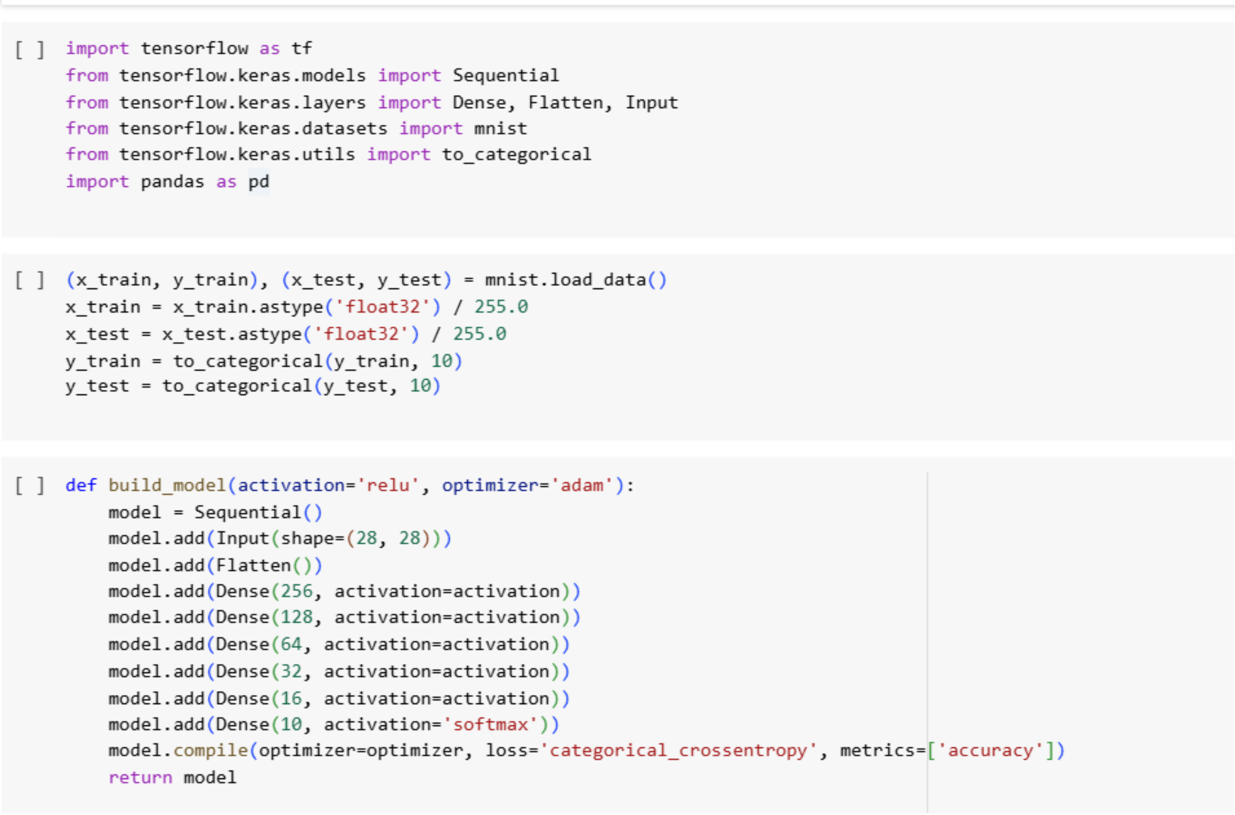
Name : GANESH REDDY LANGATI

ICP6_BIGDATA ANALYTICS

GIT HUB LINK : https://github.com/GANESHREDDYLANGATI/ICP6_BIGDATA

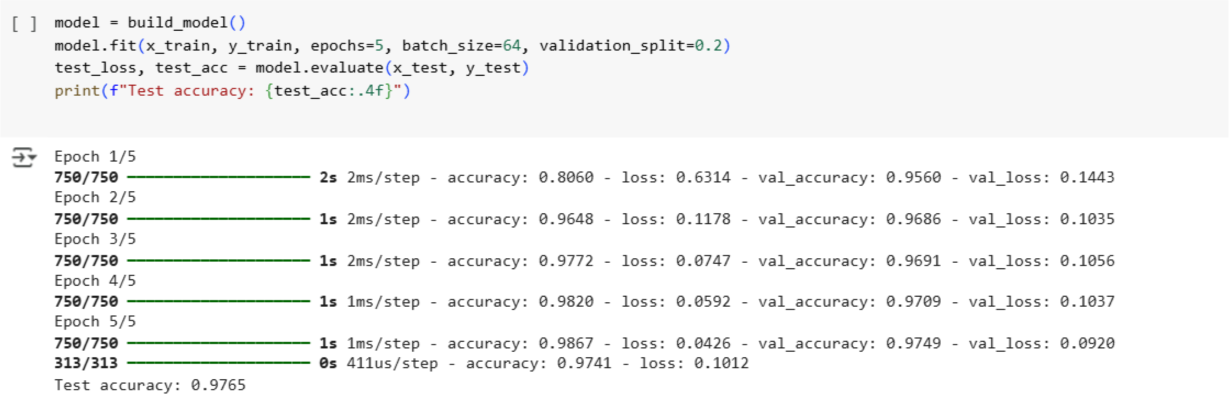
Screen Shots :



```
[ ] import tensorflow as tf
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense, Flatten, Input
    from tensorflow.keras.datasets import mnist
    from tensorflow.keras.utils import to_categorical
    import pandas as pd
```

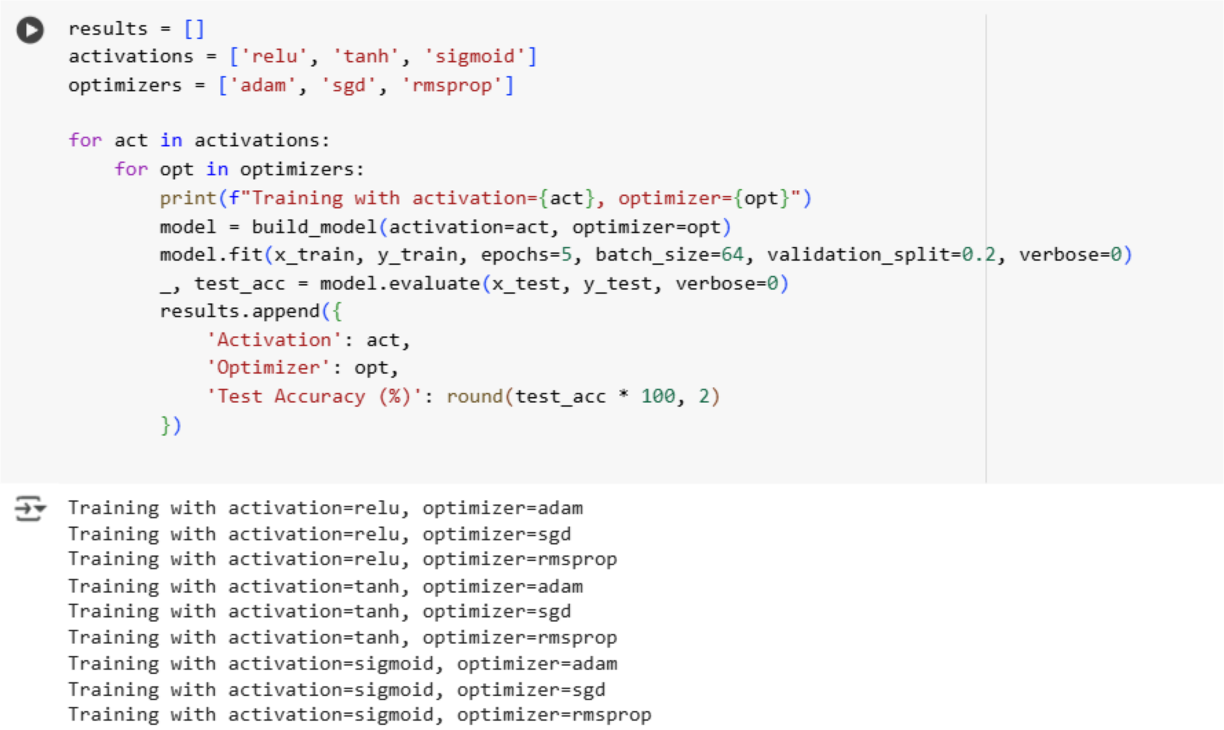
```
[ ] (x_train, y_train), (x_test, y_test) = mnist.load_data()
    x_train = x_train.astype('float32') / 255.0
    x_test = x_test.astype('float32') / 255.0
    y_train = to_categorical(y_train, 10)
    y_test = to_categorical(y_test, 10)
```

```
[ ] def build_model(activation='relu', optimizer='adam'):
        model = Sequential()
        model.add(Input(shape=(28, 28)))
        model.add(Flatten())
        model.add(Dense(256, activation=activation))
        model.add(Dense(128, activation=activation))
        model.add(Dense(64, activation=activation))
        model.add(Dense(32, activation=activation))
        model.add(Dense(16, activation=activation))
        model.add(Dense(10, activation='softmax'))
        model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
        return model
```



```
[ ] model = build_model()
    model.fit(x_train, y_train, epochs=5, batch_size=64, validation_split=0.2)
    test_loss, test_acc = model.evaluate(x_test, y_test)
    print(f"Test accuracy: {test_acc:.4f}")
```

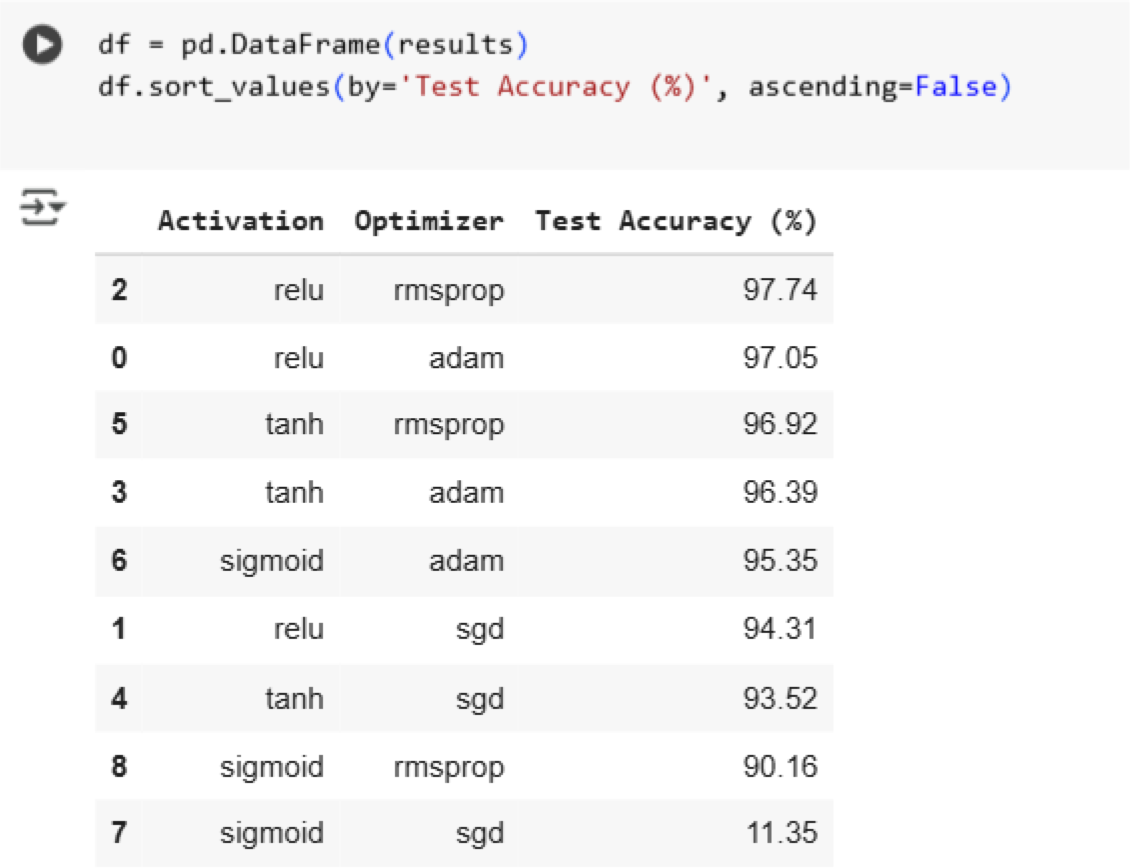
```
Epoch 1/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8060 - loss: 0.6314 - val_accuracy: 0.9560 - val_loss: 0.1443
Epoch 2/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9648 - loss: 0.1178 - val_accuracy: 0.9686 - val_loss: 0.1035
Epoch 3/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9772 - loss: 0.0747 - val_accuracy: 0.9691 - val_loss: 0.1056
Epoch 4/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9820 - loss: 0.0592 - val_accuracy: 0.9709 - val_loss: 0.1037
Epoch 5/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9867 - loss: 0.0426 - val_accuracy: 0.9749 - val_loss: 0.0920
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 411us/step - accuracy: 0.9741 - loss: 0.1012
Test accuracy: 0.9765
```



```
results = []
activations = ['relu', 'tanh', 'sigmoid']
optimizers = ['adam', 'sgd', 'rmsprop']

for act in activations:
    for opt in optimizers:
        print(f"Training with activation={act}, optimizer={opt}")
        model = build_model(activation=act, optimizer=opt)
        model.fit(x_train, y_train, epochs=5, batch_size=64, validation_split=0.2, verbose=0)
        _, test_acc = model.evaluate(x_test, y_test, verbose=0)
        results.append({
            'Activation': act,
            'Optimizer': opt,
            'Test Accuracy (%)': round(test_acc * 100, 2)
        })
```

```
Training with activation=relu, optimizer=adam
Training with activation=relu, optimizer=sgd
Training with activation=relu, optimizer=rmsprop
Training with activation=tanh, optimizer=adam
Training with activation=tanh, optimizer=sgd
Training with activation=tanh, optimizer=rmsprop
Training with activation=sigmoid, optimizer=adam
Training with activation=sigmoid, optimizer=sgd
Training with activation=sigmoid, optimizer=rmsprop
```



```python
df = pd.DataFrame(results)
df.sort_values(by='Test Accuracy (%)', ascending=False)
```

	Activation	Optimizer	Test Accuracy (%)
2	relu	rmsprop	97.74
0	relu	adam	97.05
5	tanh	rmsprop	96.92
3	tanh	adam	96.39
6	sigmoid	adam	95.35
1	relu	sgd	94.31
4	tanh	sgd	93.52
8	sigmoid	rmsprop	90.16
7	sigmoid	sgd	11.35