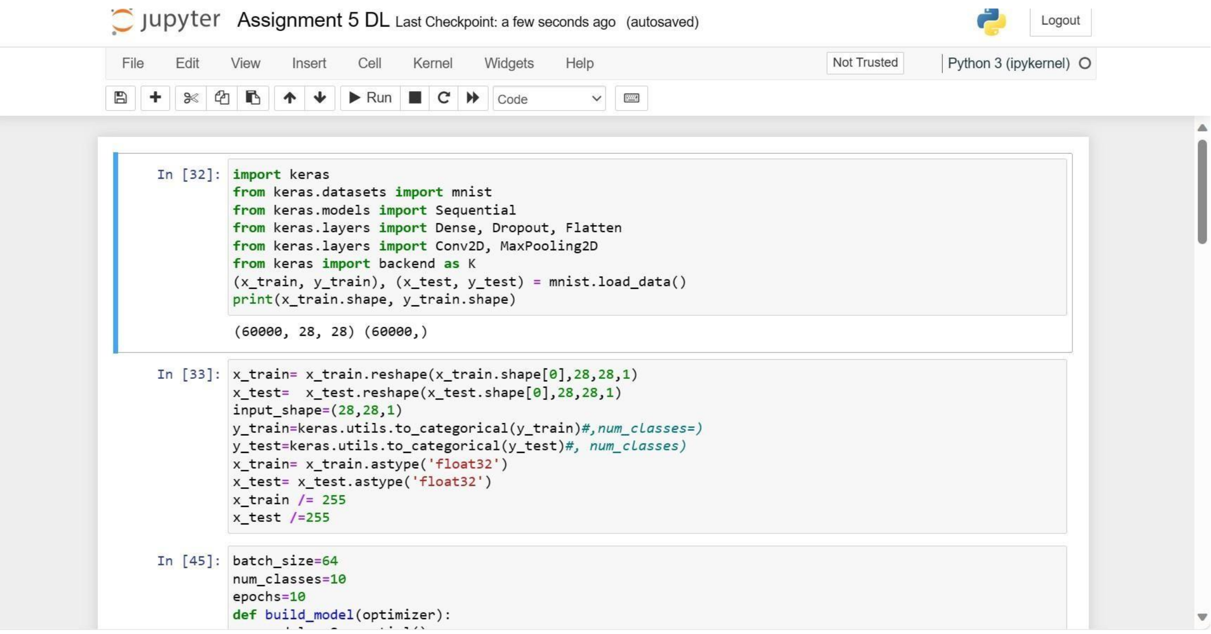
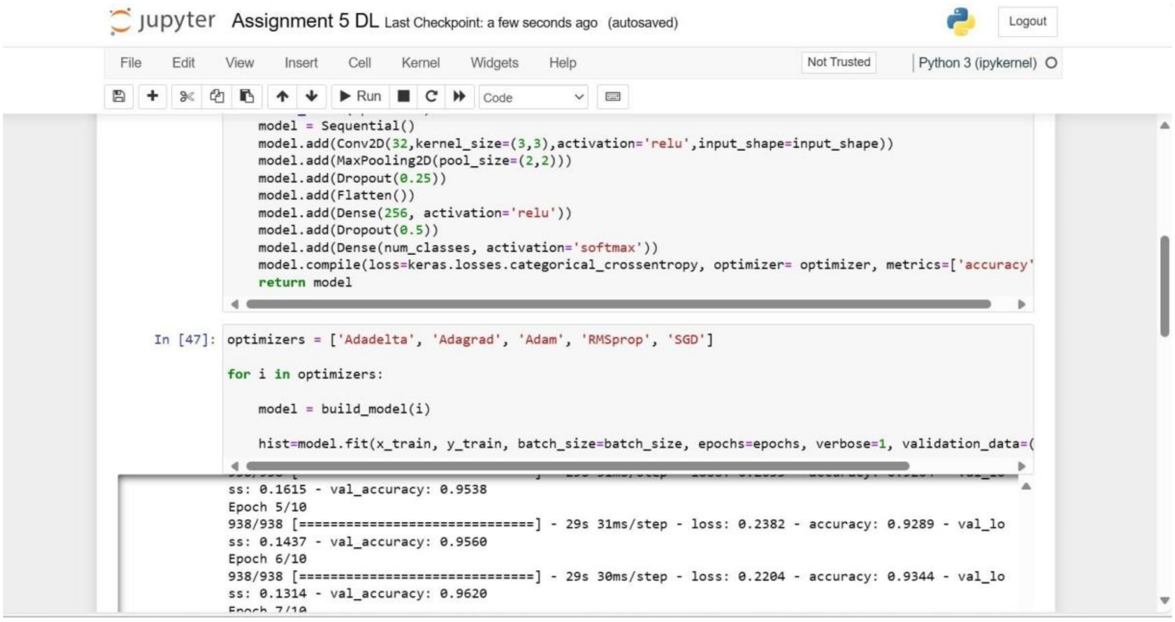
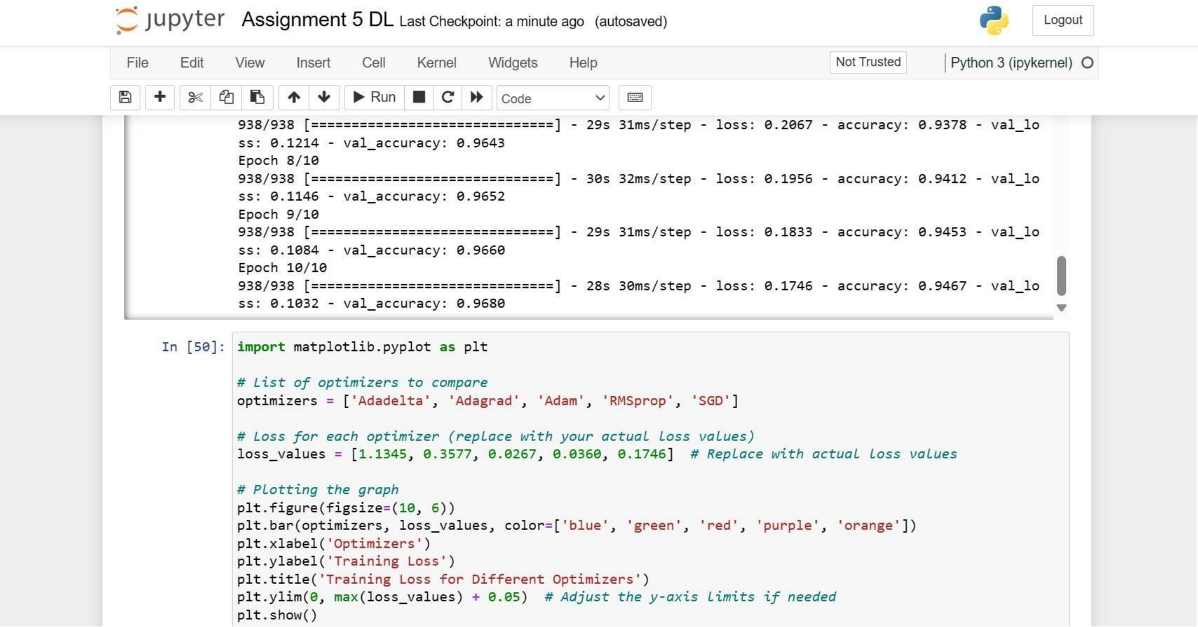


```
In [32]: import keras
         from keras.datasets import mnist
         from keras.models import Sequential
         from keras.layers import Dense, Dropout, Flatten
         from keras.layers import Conv2D, MaxPooling2D
         from keras import backend as K
         (x_train, y_train), (x_test, y_test) = mnist.load_data()
         print(x_train.shape, y_train.shape)

         (60000, 28, 28) (60000,)

In [33]: x_train= x_train.reshape(x_train.shape[0],28,28,1)
         x_test=  x_test.reshape(x_test.shape[0],28,28,1)
         input_shape=(28,28,1)
         y_train=keras.utils.to_categorical(y_train)#,num_classes=)
         y_test=keras.utils.to_categorical(y_test)#, num_classes)
         x_train= x_train.astype('float32')
         x_test= x_test.astype('float32')
         x_train /= 255
         x_test /=255

In [45]: batch_size=64
         num_classes=10
         epochs=10
         def build_model(optimizer):
```

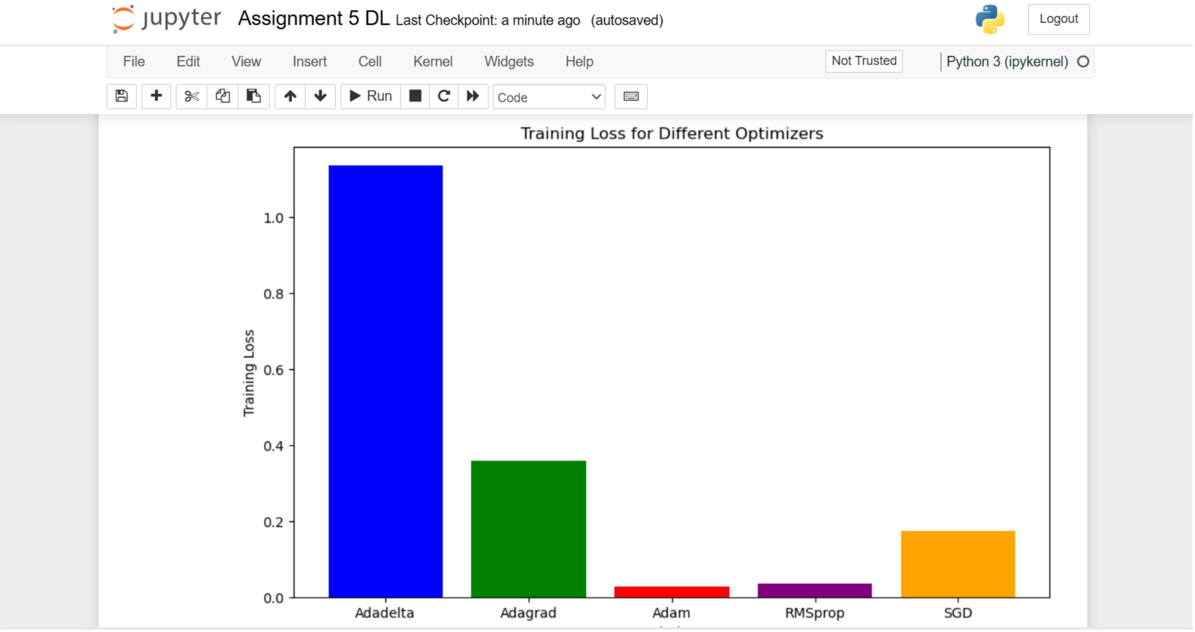


```
             model = Sequential()
             model.add(Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=input_shape))
             model.add(MaxPooling2D(pool_size=(2,2)))
             model.add(Dropout(0.25))
             model.add(Flatten())
             model.add(Dense(256, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(num_classes, activation='softmax'))
             model.compile(loss=keras.losses.categorical_crossentropy, optimizer= optimizer, metrics=['accuracy'
             return model

In [47]: optimizers = ['Adadelta', 'Adagrad', 'Adam', 'RMSprop', 'SGD']

         for i in optimizers:

             model = build_model(i)

             hist=model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(

         ss: 0.1615 - val_accuracy: 0.9538
         Epoch 5/10
         938/938 [==============================] - 29s 31ms/step - loss: 0.2382 - accuracy: 0.9289 - val_lo
         ss: 0.1437 - val_accuracy: 0.9560
         Epoch 6/10
         938/938 [==============================] - 29s 30ms/step - loss: 0.2204 - accuracy: 0.9344 - val_lo
         ss: 0.1314 - val_accuracy: 0.9620
         Epoch 7/10
```



```
938/938 [==============================] - 29s 31ms/step - loss: 0.2067 - accuracy: 0.9378 - val_lo
ss: 0.1214 - val_accuracy: 0.9643
Epoch 8/10
938/938 [==============================] - 30s 32ms/step - loss: 0.1956 - accuracy: 0.9412 - val_lo
ss: 0.1146 - val_accuracy: 0.9652
Epoch 9/10
938/938 [==============================] - 29s 31ms/step - loss: 0.1833 - accuracy: 0.9453 - val_lo
ss: 0.1084 - val_accuracy: 0.9660
Epoch 10/10
938/938 [==============================] - 28s 30ms/step - loss: 0.1746 - accuracy: 0.9467 - val_lo
ss: 0.1032 - val_accuracy: 0.9680
```

```
In [50]: import matplotlib.pyplot as plt

         # List of optimizers to compare
         optimizers = ['Adadelta', 'Adagrad', 'Adam', 'RMSprop', 'SGD']

         # Loss for each optimizer (replace with your actual loss values)
         loss_values = [1.1345, 0.3577, 0.0267, 0.0360, 0.1746]  # Replace with actual loss values

         # Plotting the graph
         plt.figure(figsize=(10, 6))
         plt.bar(optimizers, loss_values, color=['blue', 'green', 'red', 'purple', 'orange'])
         plt.xlabel('Optimizers')
         plt.ylabel('Training Loss')
         plt.title('Training Loss for Different Optimizers')
         plt.ylim(0, max(loss_values) + 0.05)  # Adjust the y-axis limits if needed
         plt.show()
```



## Conclusion:

Through this experiment we have successfully developed a model to test the accuracy, loss and convergence time of various optimizers such as stochastic gradient descent, gradient descent,adam,adagrad,rmsprop and nadam.