CSEE5590/490: Python and Deep Learning Programming (2018 Fall)

Lab 3

Team ID:- 13

Partner 1: Kamal Tej Veerapaneni Id - 31

Partner 2: Vinay Maturi Id - 17

YouTube Link: https://youtu.be/km6JzkaLBYU

Introduction:

In this Lab Assignment we have worked on the following tasks.

1)Pick a dataset and plot loss and accuracy values

2)Implement Linear and Logistic regressions

3)Showing the graphs in tensor board

4)Changing various features and calculating loss and accuracy

Objectives:

- Get familiar with Keras Library
- In Linear and Logistic we are gonna use tensorboard to plot loss by using it in training
- We also have to change certain features and observe how loss varies in each case:
  Linear Regression:
  Learning Rate
  Batch size
  Optimizer
  Activation function

Question 1:

1. Implement the **Linear Regression** with any data set of your choice except the datasets being discussed in the class

   a. Show the graph in TensorBoard

   b. Plot the loss and then change the below parameter and report your view how the result changes in each case

   a. learning rate
   b. batch size
   c. optimizer
   d. activation function

Workflow:

The result is shown using Tensor board .here we can see the code and Loss value for each case required.

Important to note: for linear regression cases, you would need to use mse or mae as the loss,and you could't use softmax as activation (since the output of the model isn't supposed to be probabilities).

Observations:

Here we make some observations on loss and accuracy values.Since this question focuses on loss we can see various loss values.As we can observe below when I decrease the batchsize from 128 to 64 with same number of epochs(10) we can see a decrease in loss value form 0.06…to 0.05…

And when we change the optimizer from rms to sgd we can see an increase in loss for epochs=10,batchsize=128

As such many observations can be made which were given below with each feature

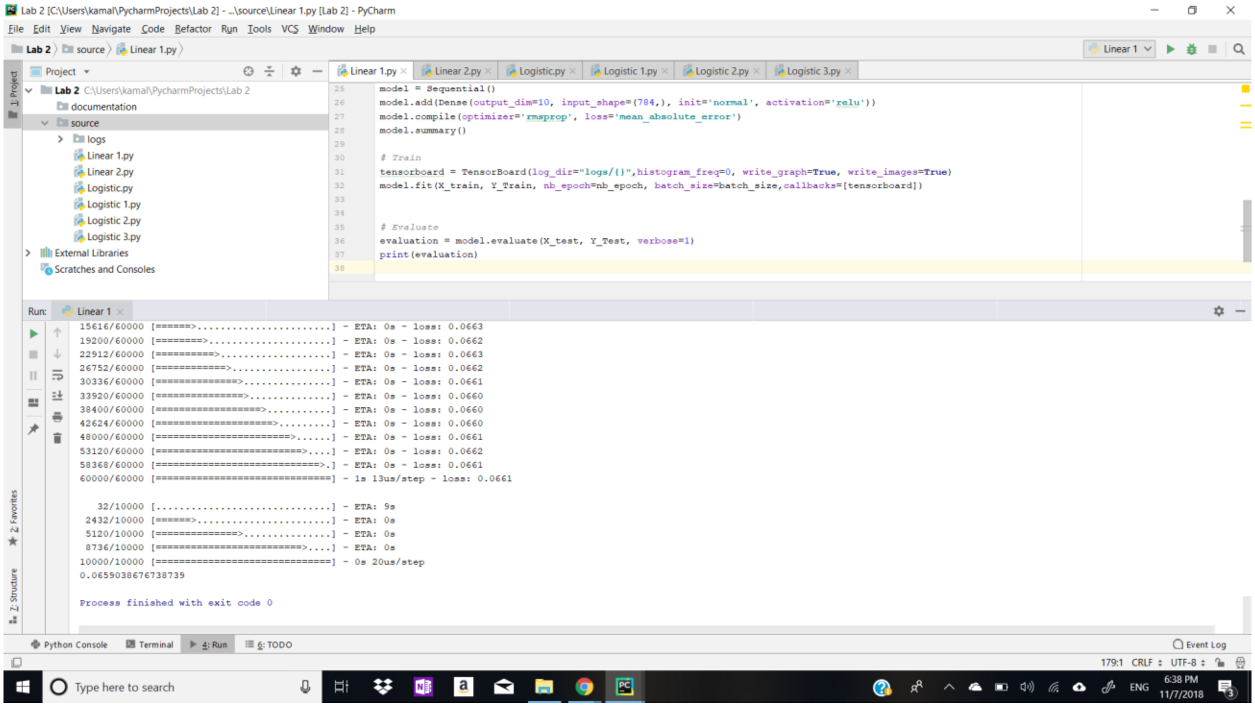
Code Snippet + Loss Values:

Batch Size:128

Epoch:10

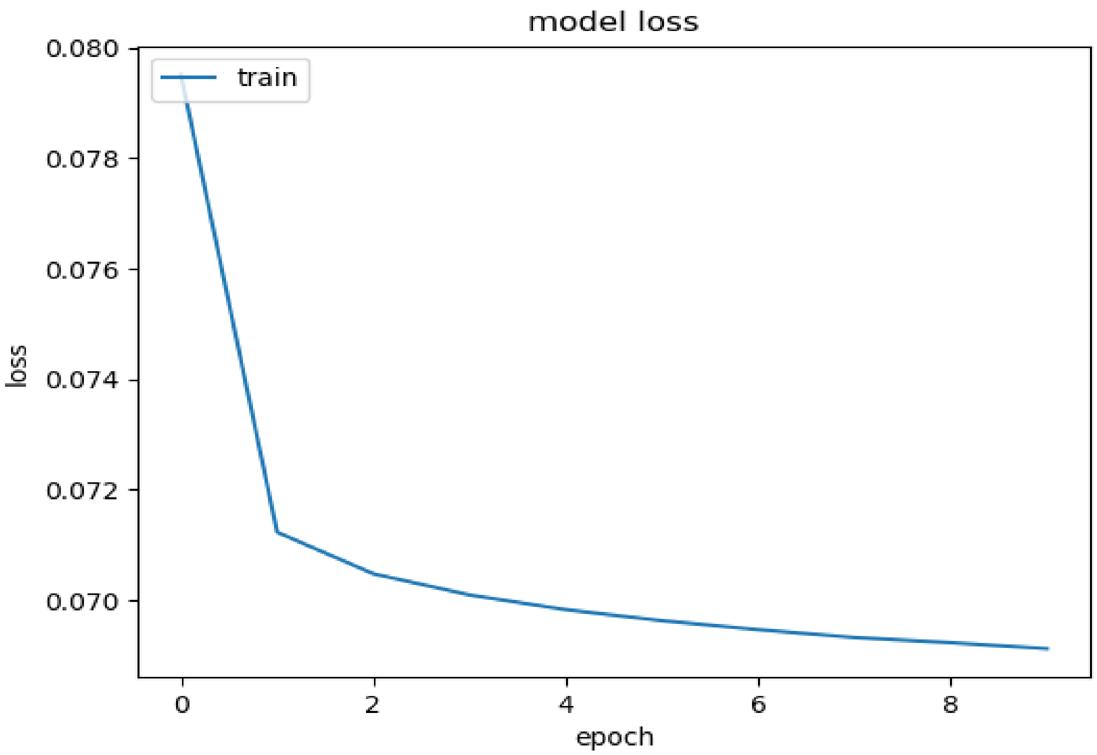
Optimizer: rmsprop

Activation Function: relu Loss:0.0659

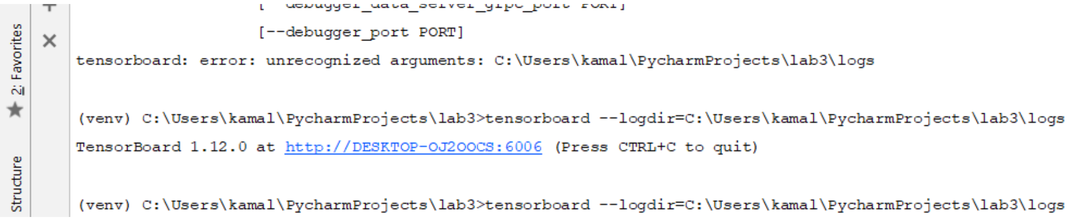


```
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='relu'))
model.compile(optimizer='rmsprop', loss='mean_absolute_error')
model.summary()

# Train
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
model.fit(X_train, Y_Train, nb_epoch=nb_epoch, batch_size=batch_size,callbacks=[tensorboard])


# Evaluate
evaluation = model.evaluate(X_test, Y_Test, verbose=1)
print(evaluation)
```

```
15616/60000 [======>.......................] - ETA: 0s - loss: 0.0663
19200/60000 [========>.....................] - ETA: 0s - loss: 0.0662
22912/60000 [==========>...................] - ETA: 0s - loss: 0.0663
26752/60000 [============>.................] - ETA: 0s - loss: 0.0662
30336/60000 [==============>...............] - ETA: 0s - loss: 0.0661
33920/60000 [===============>..............] - ETA: 0s - loss: 0.0660
38400/60000 [==================>...........] - ETA: 0s - loss: 0.0660
42624/60000 [====================>.........] - ETA: 0s - loss: 0.0660
48000/60000 [=======================>......] - ETA: 0s - loss: 0.0661
53120/60000 [=========================>....] - ETA: 0s - loss: 0.0662
58368/60000 [============================>.] - ETA: 0s - loss: 0.0661
60000/60000 [==============================] - 1s 13us/step - loss: 0.0661

   32/10000 [..............................] - ETA: 9s
 2432/10000 [======>.......................] - ETA: 0s
 5120/10000 [==============>...............] - ETA: 0s
 8736/10000 [=========================>....] - ETA: 0s
10000/10000 [==============================] - 0s 20us/step
0.0659038676738739

Process finished with exit code 0
```



Graph Link for tensor board:



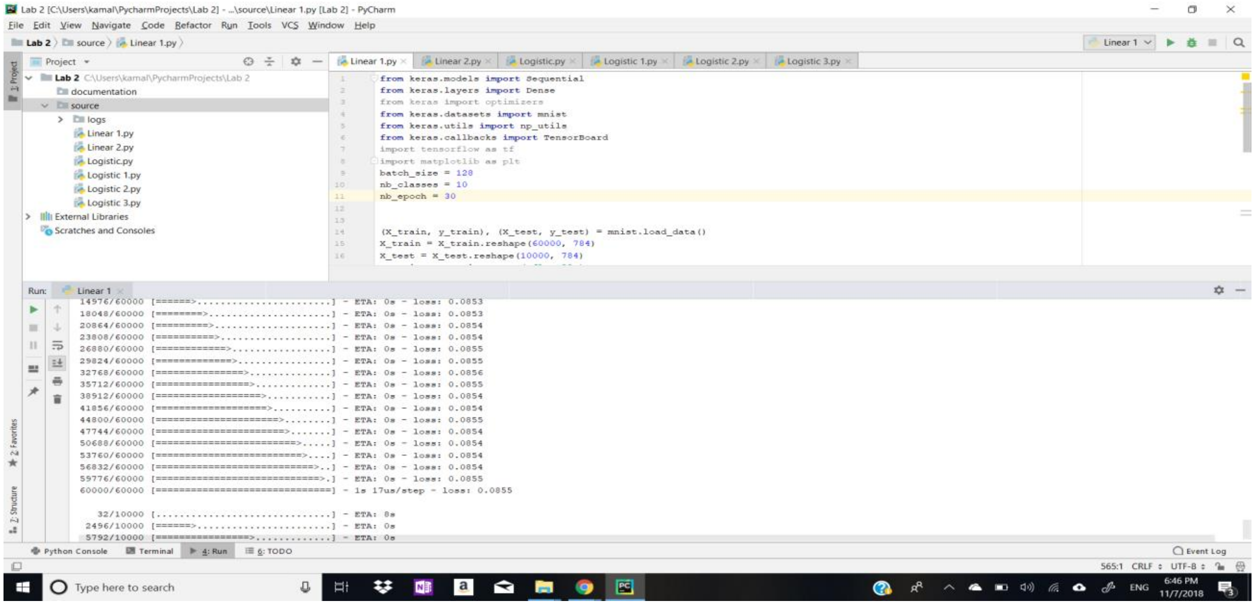
```
                        [--debugger_data_server_grpc_port PORT]
                        [--debugger_port PORT]
tensorboard: error: unrecognized arguments: C:\Users\kamal\PycharmProjects\lab3\logs

(venv) C:\Users\kamal\PycharmProjects\lab3>tensorboard --logdir=C:\Users\kamal\PycharmProjects\lab3\logs
TensorBoard 1.12.0 at http://DESKTOP-OJ2OOCS:6006 (Press CTRL+C to quit)

(venv) C:\Users\kamal\PycharmProjects\lab3>tensorboard --logdir=C:\Users\kamal\PycharmProjects\lab3\logs
```

Batch Size:128

Epoch:30

Optimizer: rmsprop

Activation Function: relu



```python
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from keras.datasets import mnist
from keras.utils import np_utils
from keras.callbacks import TensorBoard
import tensorflow as tf
import matplotlib as plt
batch_size = 128
nb_classes = 10
nb_epoch = 30


(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
```

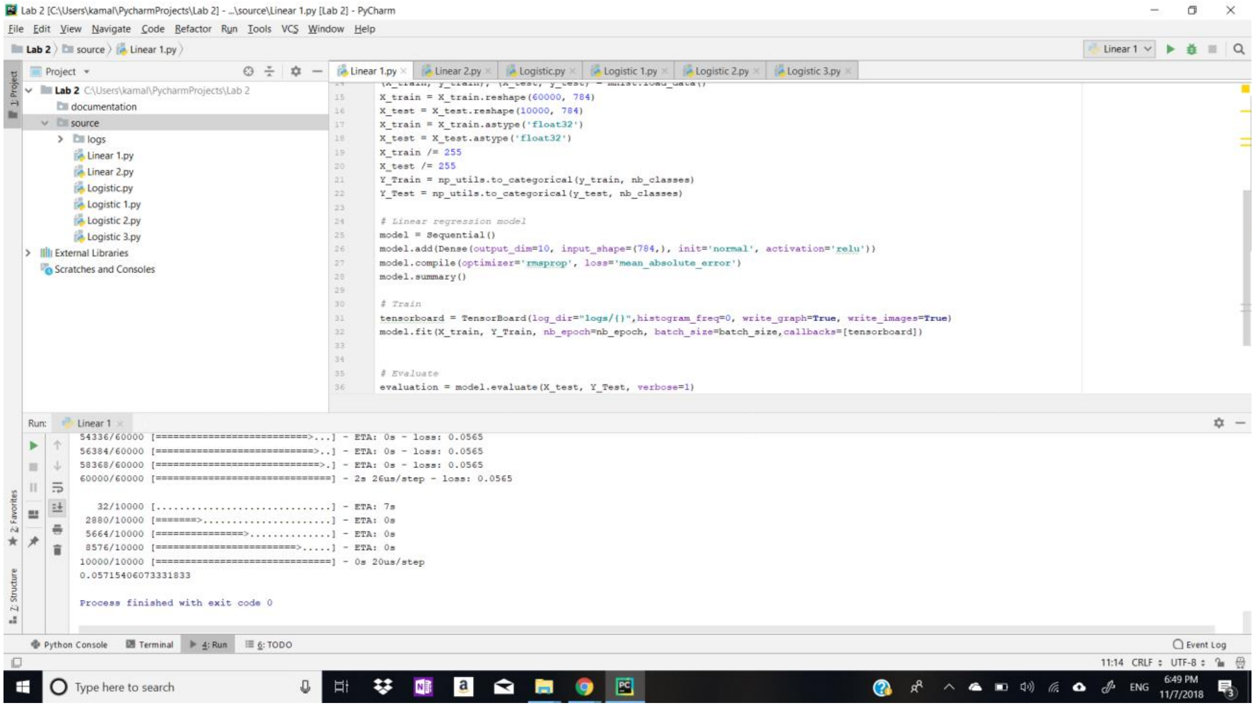
```
14976/60000 [======>.......................] - ETA: 0s - loss: 0.0853
18048/60000 [========>.....................] - ETA: 0s - loss: 0.0853
20864/60000 [=========>....................] - ETA: 0s - loss: 0.0854
23808/60000 [==========>...................] - ETA: 0s - loss: 0.0854
26880/60000 [============>.................] - ETA: 0s - loss: 0.0855
29824/60000 [=============>................] - ETA: 0s - loss: 0.0855
32768/60000 [===============>..............] - ETA: 0s - loss: 0.0856
35712/60000 [================>.............] - ETA: 0s - loss: 0.0855
38912/60000 [==================>...........] - ETA: 0s - loss: 0.0854
41856/60000 [===================>..........] - ETA: 0s - loss: 0.0854
44800/60000 [=====================>........] - ETA: 0s - loss: 0.0855
47744/60000 [======================>.......] - ETA: 0s - loss: 0.0854
50688/60000 [========================>.....] - ETA: 0s - loss: 0.0854
53760/60000 [=========================>....] - ETA: 0s - loss: 0.0854
56832/60000 [===========================>..] - ETA: 0s - loss: 0.0854
59776/60000 [============================>.] - ETA: 0s - loss: 0.0855
60000/60000 [==============================] - 1s 17us/step - loss: 0.0855

   32/10000 [..............................] - ETA: 8s
 2496/10000 [======>.......................] - ETA: 0s
 5792/10000 [================>.............] - ETA: 0s
```

Batch Size: 64

Epoch: 10

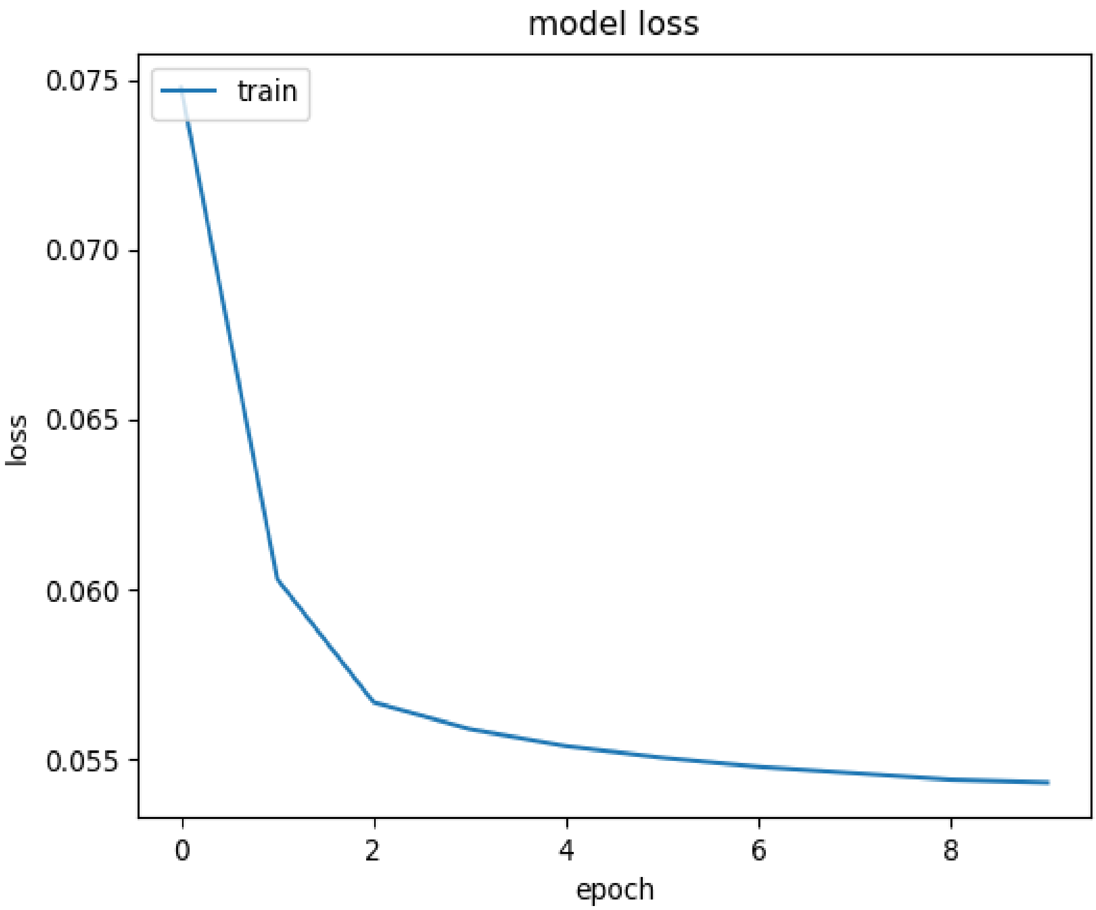
Optimizer: rms prop

Activation Function: relu



```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Linear regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='relu'))
model.compile(optimizer='rmsprop', loss='mean_absolute_error')
model.summary()

# Train
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
model.fit(X_train, Y_Train, nb_epoch=nb_epoch, batch_size=batch_size,callbacks=[tensorboard])


# Evaluate
evaluation = model.evaluate(X_test, Y_Test, verbose=1)
```

```
54336/60000 [==========================>...] - ETA: 0s - loss: 0.0565
56384/60000 [===========================>..] - ETA: 0s - loss: 0.0565
58368/60000 [============================>.] - ETA: 0s - loss: 0.0565
60000/60000 [==============================] - 2s 26us/step - loss: 0.0565

   32/10000 [..............................] - ETA: 7s
 2880/10000 [=======>......................] - ETA: 0s
 5664/10000 [===============>..............] - ETA: 0s
 8576/10000 [========================>.....] - ETA: 0s
10000/10000 [==============================] - 0s 20us/step
0.05715406073331833

Process finished with exit code 0
```

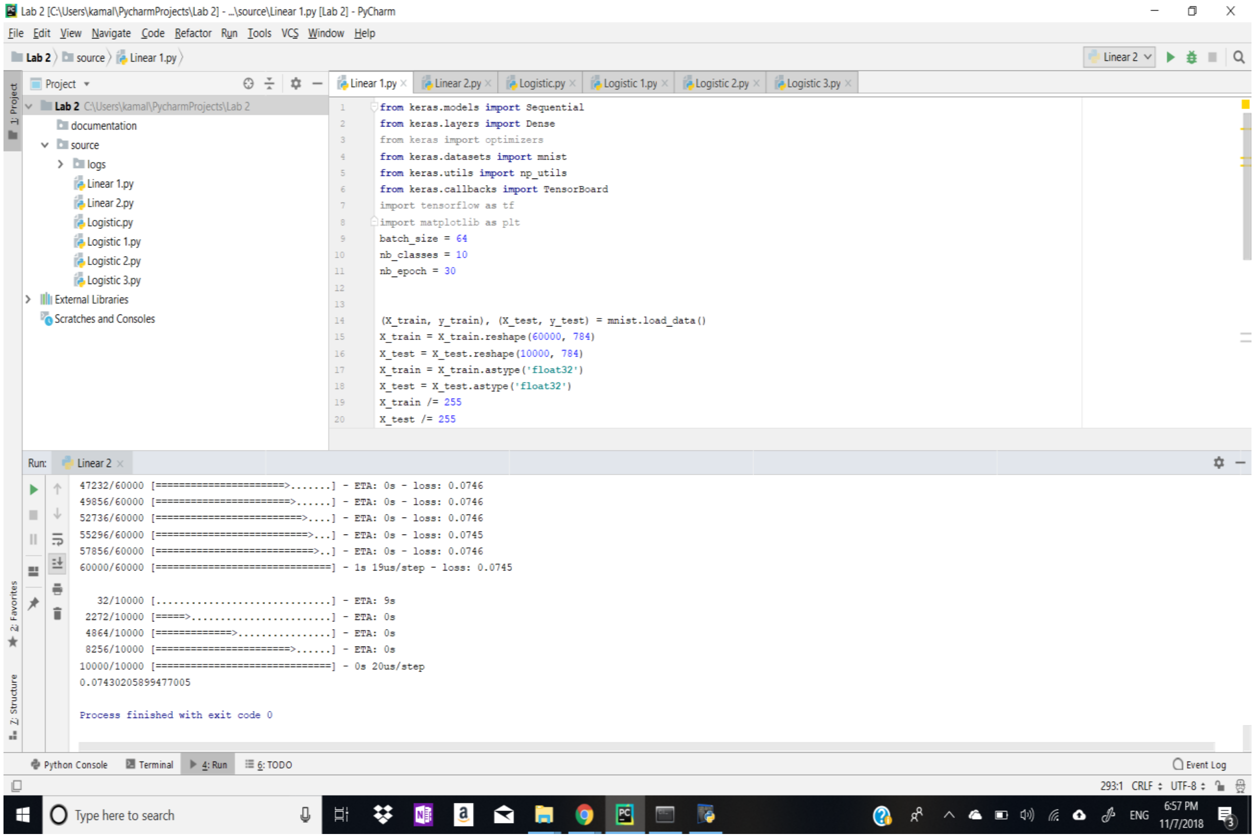


Batch Size:64

Epoch:30

Optimizer:rms prop

Activation Function:relu



```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from keras.datasets import mnist
from keras.utils import np_utils
from keras.callbacks import TensorBoard
import tensorflow as tf
import matplotlib as plt
batch_size = 64
nb_classes = 10
nb_epoch = 30


(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

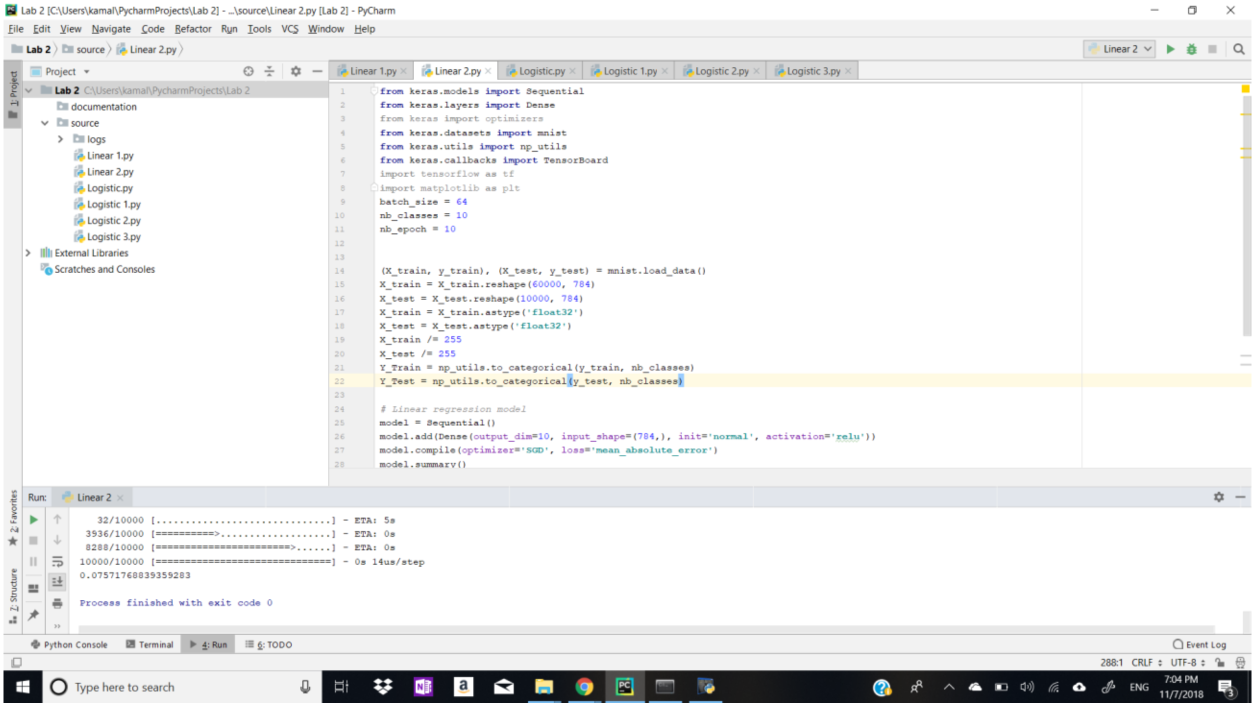
```
47232/60000 [======================>.......] - ETA: 0s - loss: 0.0746
49856/60000 [=======================>......] - ETA: 0s - loss: 0.0746
52736/60000 [=========================>....] - ETA: 0s - loss: 0.0746
55296/60000 [==========================>...] - ETA: 0s - loss: 0.0745
57856/60000 [===========================>..] - ETA: 0s - loss: 0.0746
60000/60000 [==============================] - 1s 19us/step - loss: 0.0745

   32/10000 [..............................] - ETA: 9s
 2272/10000 [=====>........................] - ETA: 0s
 4864/10000 [=============>................] - ETA: 0s
 8256/10000 [=======================>......] - ETA: 0s
10000/10000 [==============================] - 0s 20us/step
0.07430205899477005

Process finished with exit code 0
```

Batch Size:64

Epoch:10

Optimizer:sgd

Activation Function:relu



```python
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from keras.datasets import mnist
from keras.utils import np_utils
from keras.callbacks import TensorBoard
import tensorflow as tf
import matplotlib as plt
batch_size = 64
nb_classes = 10
nb_epoch = 10


(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Linear regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='relu'))
model.compile(optimizer='SGD', loss='mean_absolute_error')
model.summary()
```

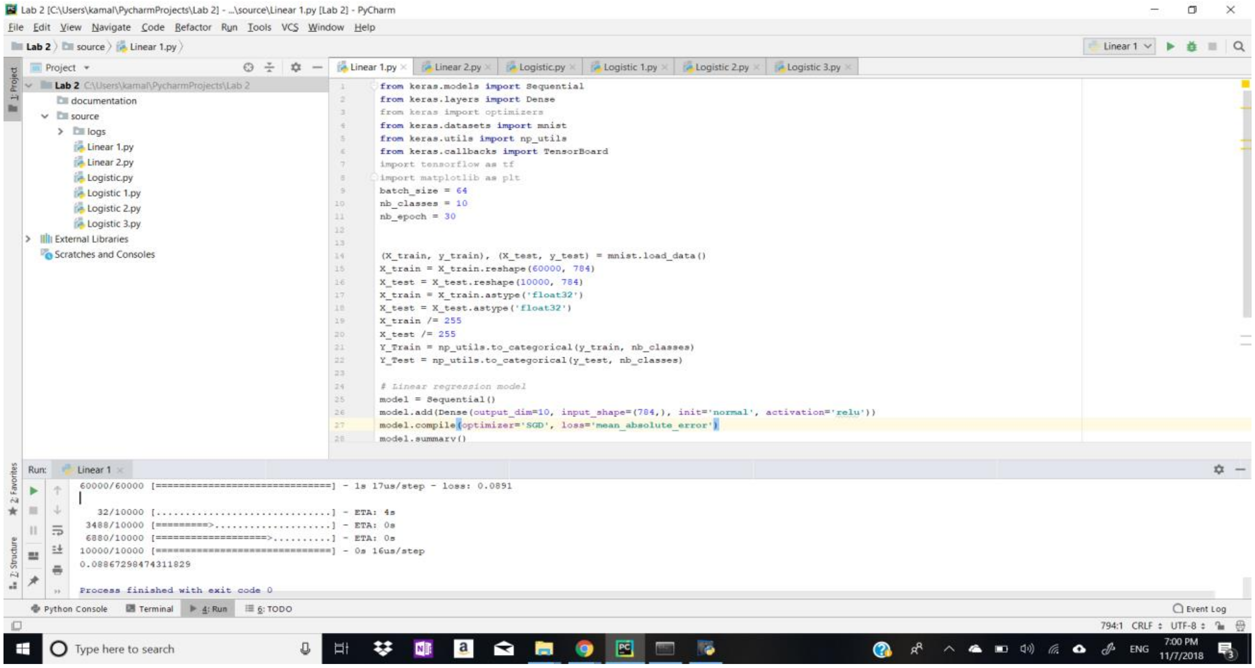
```
   32/10000 [..............................] - ETA: 5s
 3936/10000 [==========>...................] - ETA: 0s
 8288/10000 [=======================>......] - ETA: 0s
10000/10000 [==============================] - 0s 14us/step
0.07571768839359283

Process finished with exit code 0
```

Batch Size:64

Epoch:30

Optimizer:sgd

Activation Function:relu



```python
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from keras.datasets import mnist
from keras.utils import np_utils
from keras.callbacks import TensorBoard
import tensorflow as tf
import matplotlib as plt
batch_size = 64
nb_classes = 10
nb_epoch = 30


(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Linear regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='relu'))
model.compile(optimizer='SGD', loss='mean_absolute_error')
model.summary()
```

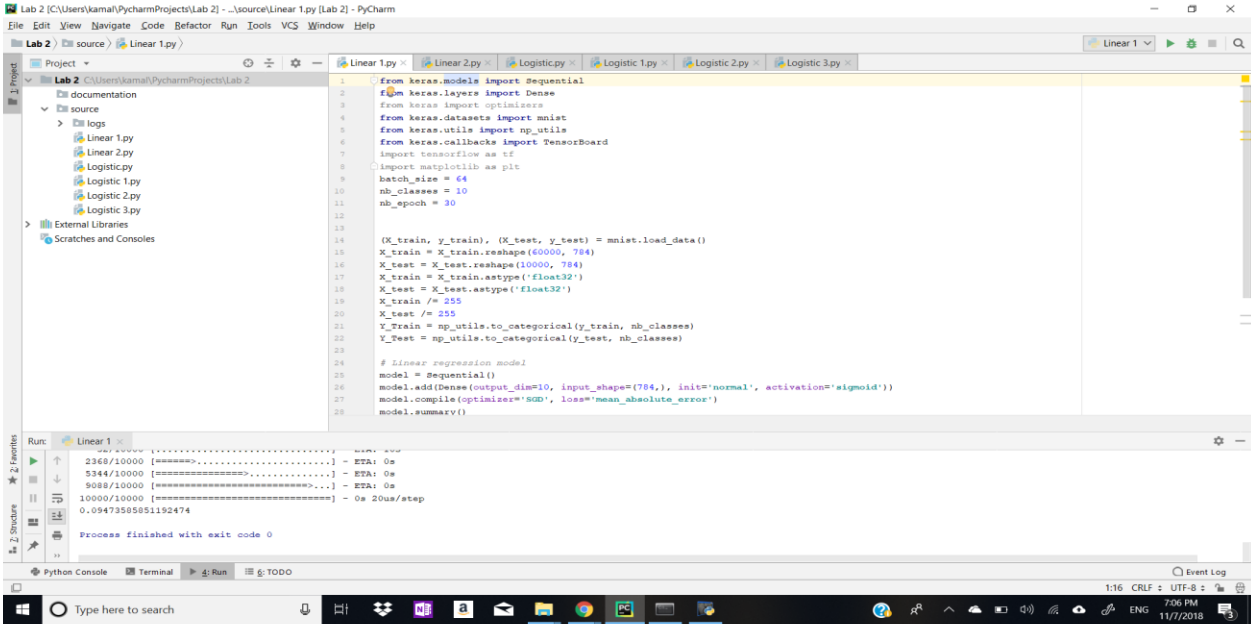
```
60000/60000 [==============================] - 1s 17us/step - loss: 0.0891


   32/10000 [..............................] - ETA: 4s
 3488/10000 [=========>....................] - ETA: 0s
 6880/10000 [===================>..........] - ETA: 0s
10000/10000 [==============================] - 0s 16us/step
0.08867298474311829

Process finished with exit code 0
```

Batch Size:64

Epoch:10

Optimizer:sgd

Activation Function: sigmoid



```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from keras.datasets import mnist
from keras.utils import np_utils
from keras.callbacks import TensorBoard
import tensorflow as tf
import matplotlib as plt
batch_size = 64
nb_classes = 10
nb_epoch = 30


(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Linear regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='sigmoid'))
model.compile(optimizer='SGD', loss='mean_absolute_error')
model.summary()
```

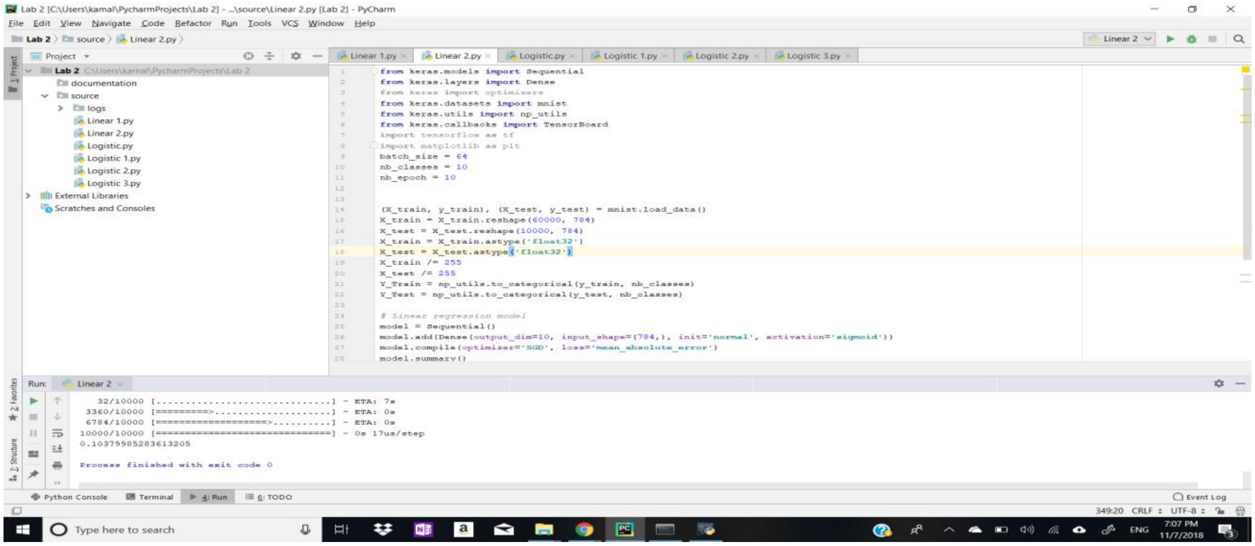
```
 2368/10000 [======>.......................] - ETA: 0s
 5344/10000 [===============>..............] - ETA: 0s
 9088/10000 [==========================>...] - ETA: 0s
10000/10000 [==============================] - 0s 20us/step
0.09473585851192474

Process finished with exit code 0
```

Batch Size:30

Epoch:64

Optimizer: sgd

Activation Function: sigmoid



```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from keras.datasets import mnist
from keras.utils import np_utils
from keras.callbacks import TensorBoard
import tensorflow as tf
import matplotlib as plt
batch_size = 64
nb_classes = 10
nb_epoch = 10


(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Linear regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='sigmoid'))
model.compile(optimizer='SGD', loss='mean_absolute_error')
model.summary()
```

```
   32/10000 [..............................] - ETA: 7s
 3360/10000 [=========>....................] - ETA: 0s
 6784/10000 [===================>..........] - ETA: 0s
10000/10000 [==============================] - 0s 17us/step
0.10379985283613205

Process finished with exit code 0
```

Question 2:

Implement *Logistic Regression with any data set of your choice.

a. Show the graph in TensorBoard
b. Show the Loss in TensorBoard
c. use **score=model.evaluate(x_text,y_test)** and then **print('test accuracy', score[1])** to print the accuracy
d. Change three hyperparameter and report how the accuracy changes

Workflow: Dataset is taken (mnist) graph and loss are shown in tensor board

along with accuracy and the outputs on how accuracy changes by changing hyper parameters is shown

Observations:

Here we make some observations on loss and accuracy values..As we can see below when I decrease the batchsize from 128 to 50 with same number of epochs(10) we can see a decrease in loss value form 0.38…to 0.32… and accuracy increases from 0.90 to 0.91

And when we change the optimizer from sgd to rmsprop we can see an decrease in loss from 0.38 to 0.27 and accuracy increases from 0.90 to 0.93 for epochs=10,batchsize=128

As such all the observations that can be made which were given below with each feature

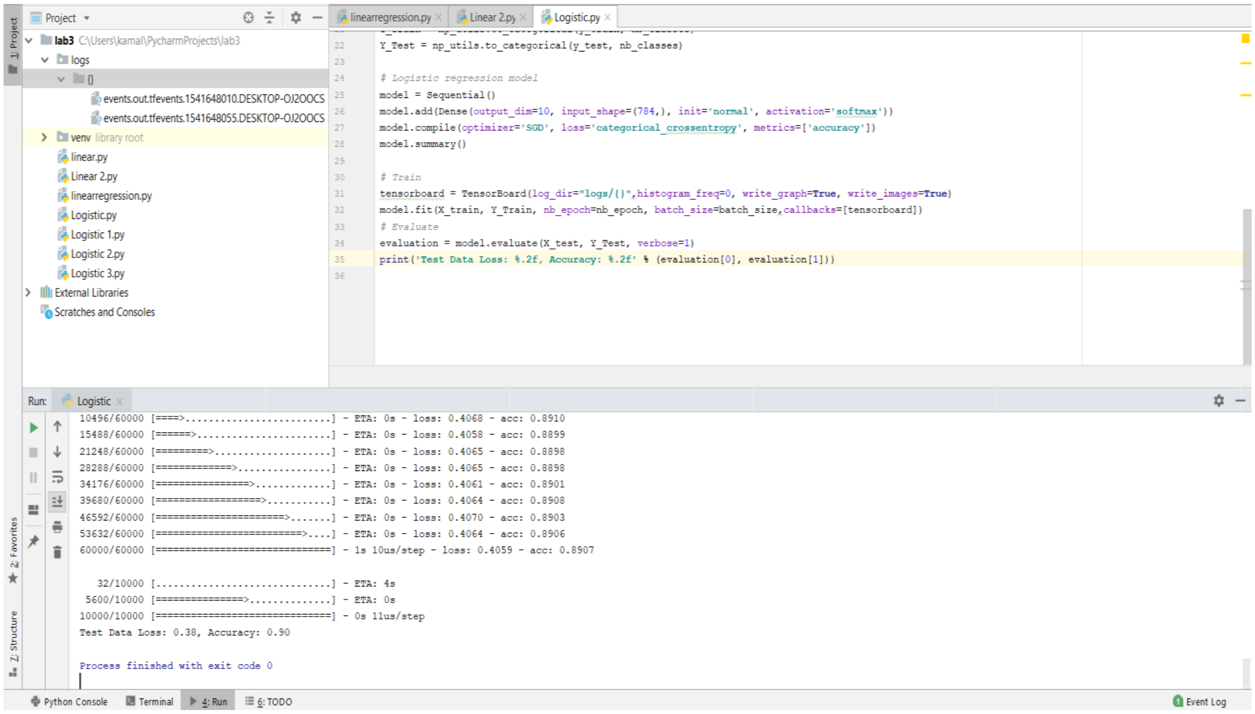
Loss and Accuracy Values:

Batch Size:128

Epoch:10

Optimizer:SGD

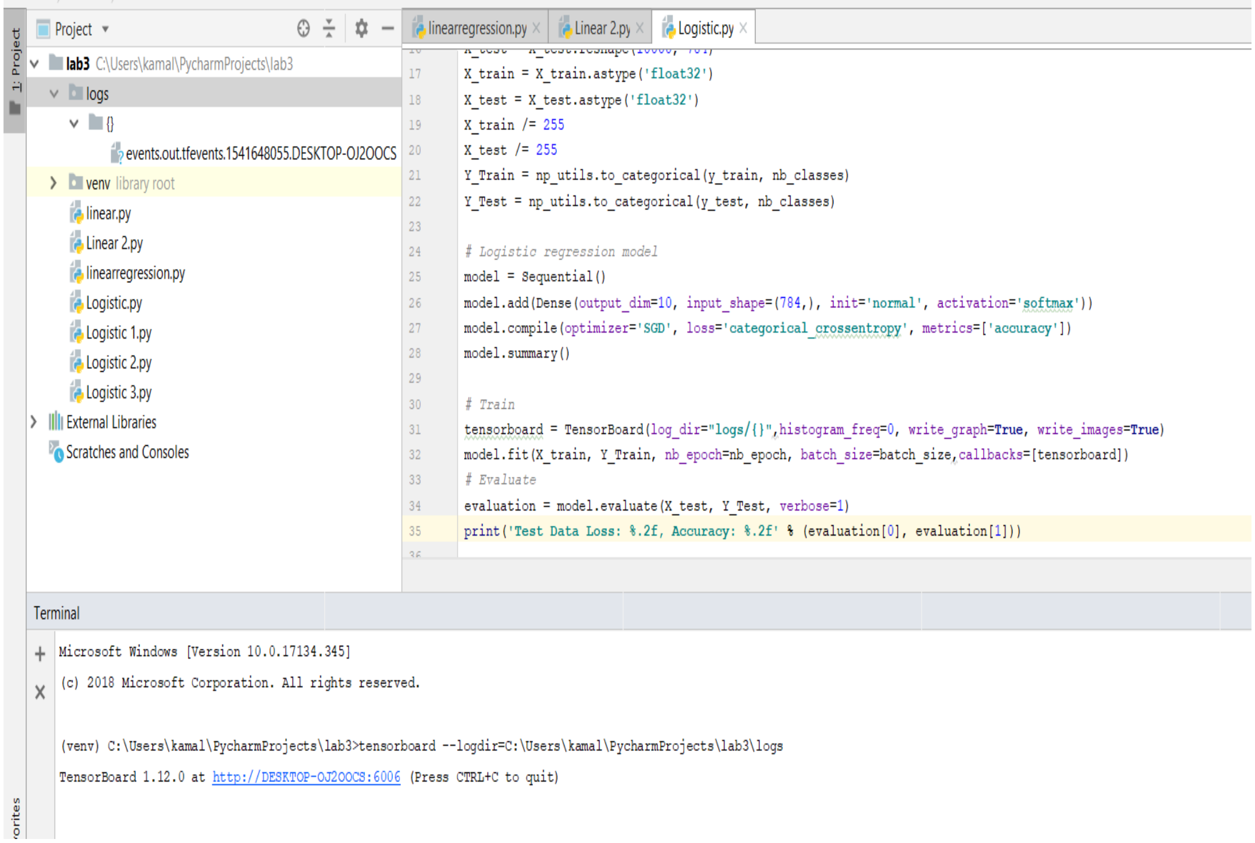
Activation Function:softmax



```
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Logistic regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='softmax'))
model.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# Train
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
model.fit(X_train, Y_Train, nb_epoch=nb_epoch, batch_size=batch_size,callbacks=[tensorboard])
# Evaluate
evaluation = model.evaluate(X_test, Y_Test, verbose=1)
print('Test Data Loss: %.2f, Accuracy: %.2f' % (evaluation[0], evaluation[1]))
```

```
10496/60000 [====>.........................] - ETA: 0s - loss: 0.4068 - acc: 0.8910
15488/60000 [======>.......................] - ETA: 0s - loss: 0.4058 - acc: 0.8899
21248/60000 [=========>....................] - ETA: 0s - loss: 0.4065 - acc: 0.8898
28288/60000 [=============>................] - ETA: 0s - loss: 0.4065 - acc: 0.8898
34176/60000 [================>.............] - ETA: 0s - loss: 0.4061 - acc: 0.8901
39680/60000 [==================>...........] - ETA: 0s - loss: 0.4064 - acc: 0.8908
46592/60000 [======================>.......] - ETA: 0s - loss: 0.4070 - acc: 0.8903
53632/60000 [=========================>....] - ETA: 0s - loss: 0.4064 - acc: 0.8906
60000/60000 [==============================] - 1s 10us/step - loss: 0.4059 - acc: 0.8907

   32/10000 [..............................] - ETA: 4s
 5600/10000 [===============>..............] - ETA: 0s
10000/10000 [==============================] - 0s 11us/step
Test Data Loss: 0.38, Accuracy: 0.90

Process finished with exit code 0
```

Graph Link for tensor board:



```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Logistic regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='softmax'))
model.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# Train
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
model.fit(X_train, Y_Train, nb_epoch=nb_epoch, batch_size=batch_size,callbacks=[tensorboard])
# Evaluate
evaluation = model.evaluate(X_test, Y_Test, verbose=1)
print('Test Data Loss: %.2f, Accuracy: %.2f' % (evaluation[0], evaluation[1]))
```

```
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.


(venv) C:\Users\kamal\PycharmProjects\lab3>tensorboard --logdir=C:\Users\kamal\PycharmProjects\lab3\logs
TensorBoard 1.12.0 at http://DESKTOP-OJ2OOCS:6006 (Press CTRL+C to quit)
```

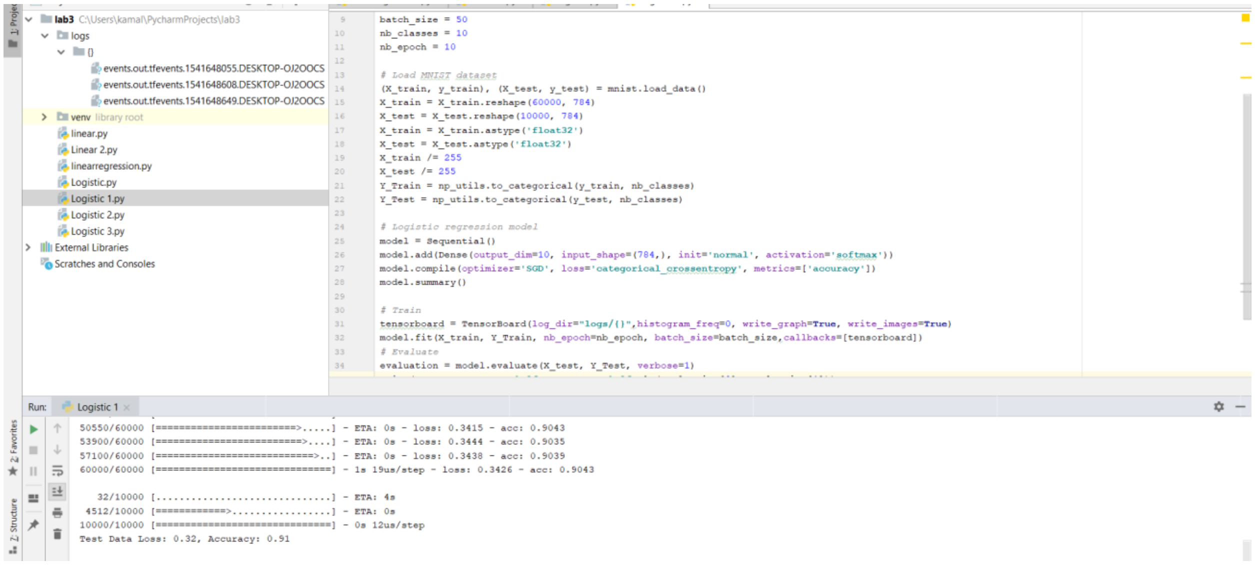
Loss and Accuracy Values:

Batch Size:50

Epoch:10

Optimizer:SGD

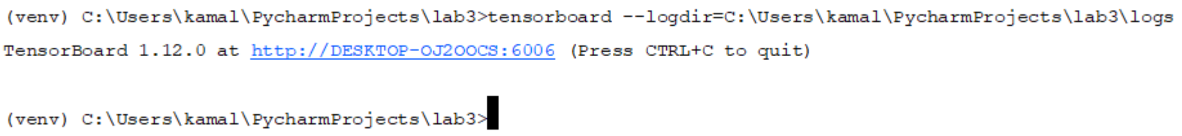
Activation Function:softmax



```
batch_size = 50
nb_classes = 10
nb_epoch = 10

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Logistic regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='softmax'))
model.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# Train
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
model.fit(X_train, Y_Train, nb_epoch=nb_epoch, batch_size=batch_size,callbacks=[tensorboard])
# Evaluate
evaluation = model.evaluate(X_test, Y_Test, verbose=1)
```

```
50550/60000 [========================>.....] - ETA: 0s - loss: 0.3415 - acc: 0.9043
53900/60000 [==========================>....] - ETA: 0s - loss: 0.3444 - acc: 0.9035
57100/60000 [===========================>..] - ETA: 0s - loss: 0.3438 - acc: 0.9039
60000/60000 [==============================] - 1s 19us/step - loss: 0.3426 - acc: 0.9043

   32/10000 [..............................] - ETA: 4s
 4512/10000 [============>.................] - ETA: 0s
10000/10000 [==============================] - 0s 12us/step
Test Data Loss: 0.32, Accuracy: 0.91
```

Graph Link for tensor board:



```
(venv) C:\Users\kamal\PycharmProjects\lab3>tensorboard --logdir=C:\Users\kamal\PycharmProjects\lab3\logs
TensorBoard 1.12.0 at http://DESKTOP-OJ2OOCS:6006 (Press CTRL+C to quit)

(venv) C:\Users\kamal\PycharmProjects\lab3>
```

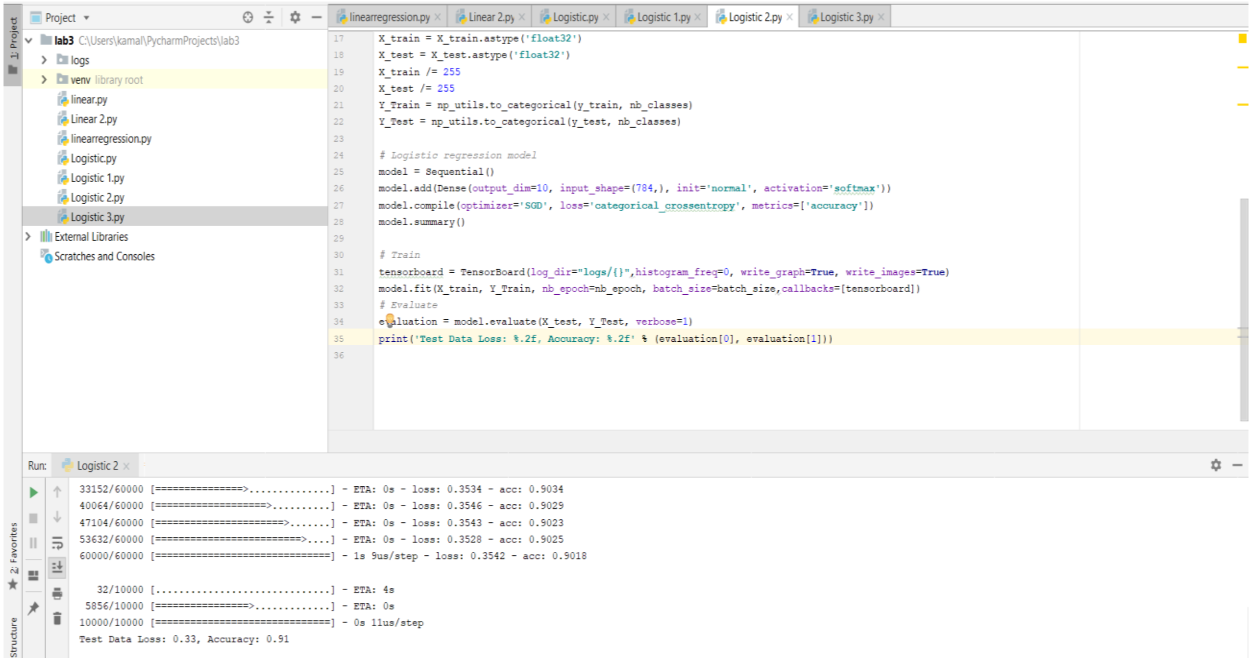
Loss and Accuracy Values:

Batch Size:128

Epoch:20

Optimizer:SGD

Activation Function:softmax



```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Logistic regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='softmax'))
model.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# Train
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
model.fit(X_train, Y_Train, nb_epoch=nb_epoch, batch_size=batch_size,callbacks=[tensorboard])
# Evaluate
evaluation = model.evaluate(X_test, Y_Test, verbose=1)
print('Test Data Loss: %.2f, Accuracy: %.2f' % (evaluation[0], evaluation[1]))
```

```
33152/60000 [===============>..............] - ETA: 0s - loss: 0.3534 - acc: 0.9034
40064/60000 [===================>..........] - ETA: 0s - loss: 0.3546 - acc: 0.9029
47104/60000 [======================>.......] - ETA: 0s - loss: 0.3543 - acc: 0.9023
53632/60000 [=========================>....] - ETA: 0s - loss: 0.3528 - acc: 0.9025
60000/60000 [==============================] - 1s 9us/step - loss: 0.3542 - acc: 0.9018

   32/10000 [..............................] - ETA: 4s
 5856/10000 [================>.............] - ETA: 0s
10000/10000 [==============================] - 0s 11us/step
Test Data Loss: 0.33, Accuracy: 0.91
```

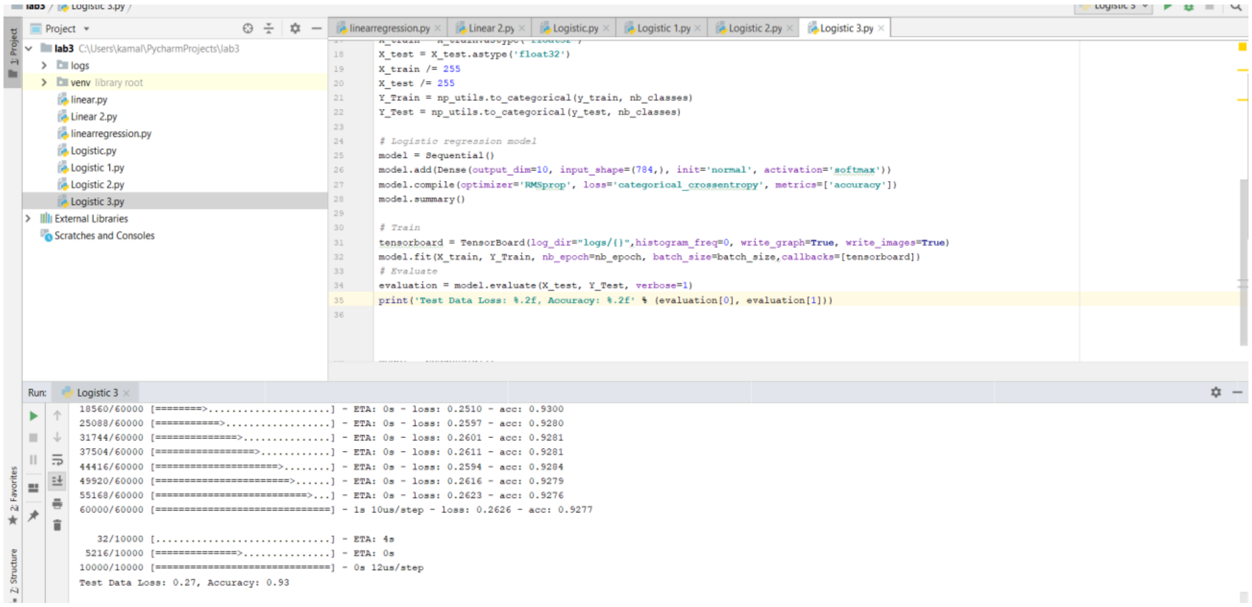
Loss and Accuracy Values:

Batch Size:128

Epoch:10

Optimizer:RMS prop

Activation Function:softmax



```
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
Y_Train = np_utils.to_categorical(y_train, nb_classes)
Y_Test = np_utils.to_categorical(y_test, nb_classes)

# Logistic regression model
model = Sequential()
model.add(Dense(output_dim=10, input_shape=(784,), init='normal', activation='softmax'))
model.compile(optimizer='RMSprop', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# Train
tensorboard = TensorBoard(log_dir="logs/{}",histogram_freq=0, write_graph=True, write_images=True)
model.fit(X_train, Y_Train, nb_epoch=nb_epoch, batch_size=batch_size,callbacks=[tensorboard])
# Evaluate
evaluation = model.evaluate(X_test, Y_Test, verbose=1)
print('Test Data Loss: %.2f, Accuracy: %.2f' % (evaluation[0], evaluation[1]))
```

```
18560/60000 [========>.....................] - ETA: 0s - loss: 0.2510 - acc: 0.9300
25088/60000 [===========>..................] - ETA: 0s - loss: 0.2597 - acc: 0.9280
31744/60000 [==============>...............] - ETA: 0s - loss: 0.2601 - acc: 0.9281
37504/60000 [=================>............] - ETA: 0s - loss: 0.2611 - acc: 0.9281
44416/60000 [=====================>........] - ETA: 0s - loss: 0.2594 - acc: 0.9284
49920/60000 [=======================>......] - ETA: 0s - loss: 0.2616 - acc: 0.9279
55168/60000 [==========================>...] - ETA: 0s - loss: 0.2623 - acc: 0.9276
60000/60000 [==============================] - 1s 10us/step - loss: 0.2626 - acc: 0.9277

   32/10000 [..............................] - ETA: 4s
 5216/10000 [==============>...............] - ETA: 0s
10000/10000 [==============================] - 0s 12us/step
Test Data Loss: 0.27, Accuracy: 0.93
```