

## LAB EXERCISE-2

Build a MLP and train it using Back Propagation Algorithm

a) Data set used: **HOUSE SALES PREDICTION**

b) Learning rate, no of epochs, accuracy

```

In [19]: hist = model.fit(X_train, Y_train,
                        batch_size=32, epochs=100,
                        validation_data=(X_val, Y_val))

Epoch 1/100
32/32 [=====] - 1s 17ms/step - loss: 0.7013 - accuracy: 0.5059 - val_loss: 0.6991 - val_accuracy: 0.4612
Epoch 2/100
32/32 [=====] - 0s 3ms/step - loss: 0.6939 - accuracy: 0.5098 - val_loss: 0.6933 - val_accuracy: 0.4658
Epoch 3/100
32/32 [=====] - 0s 3ms/step - loss: 0.6873 - accuracy: 0.5137 - val_loss: 0.6881 - val_accuracy: 0.4658
Epoch 4/100
32/32 [=====] - 0s 3ms/step - loss: 0.6814 - accuracy: 0.5147 - val_loss: 0.6833 - val_accuracy: 0.4703
Epoch 5/100
32/32 [=====] - 0s 3ms/step - loss: 0.6763 - accuracy: 0.5215 - val_loss: 0.6793 - val_accuracy: 0.4795
Epoch 6/100
32/32 [=====] - 0s 3ms/step - loss: 0.6719 - accuracy: 0.5587 - val_loss: 0.6756 - val_accuracy: 0.5160
Epoch 7/100
32/32 [=====] - 0s 3ms/step - loss: 0.6675 - accuracy: 0.5587 - val_loss: 0.6712 - val_accuracy: 0.5160

In [20]: model.evaluate(X_test, Y_test)[1]
7/7 [=====] - 0s 2ms/step - loss: 0.2800 - accuracy: 0.8904

Out[20]: 0.8904109597206116
  
```

```

In [26]: hist_2 = model_2.fit(X_train, Y_train,
                             batch_size=32, epochs=100,
                             validation_data=(X_val, Y_val))

Epoch 95/100
32/32 [=====] - 2s 62ms/step - loss: 0.1181 - accuracy: 0.9511 - val_loss: 0.4520 - val_accuracy: 0.8630
Epoch 96/100
32/32 [=====] - 2s 66ms/step - loss: 0.1271 - accuracy: 0.9481 - val_loss: 0.4377 - val_accuracy: 0.8630
Epoch 97/100
32/32 [=====] - 2s 69ms/step - loss: 0.1280 - accuracy: 0.9521 - val_loss: 0.3561 - val_accuracy: 0.8721
Epoch 98/100
32/32 [=====] - 2s 77ms/step - loss: 0.1203 - accuracy: 0.9540 - val_loss: 0.4342 - val_accuracy: 0.8721
Epoch 99/100
32/32 [=====] - 2s 74ms/step - loss: 0.1058 - accuracy: 0.9609 - val_loss: 0.4075 - val_accuracy: 0.8721
Epoch 100/100
32/32 [=====] - 2s 73ms/step - loss: 0.1513 - accuracy: 0.9481 - val_loss: 0.4635 - val_accuracy: 0.8721

In [27]: plt.plot(hist_2.history['loss'])
plt.plot(hist_2.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Loss')
plt.ylabel('Epoch')
  
```

## c) Different activation functions:

1.relu

2.sigmoid

```
Home Page - Select or create a ... | MLP using BackPropagation - Ju... | Untitled | Untitled | +
localhost:8891/notebooks/MLP%20using%20BackPropagation.ipynb#
jupyter MLP using BackPropagation Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [24]: model_2 = Sequential([
          Dense(1000, activation='relu', input_shape=(10,)),
          Dense(1000, activation='relu'),
          Dense(1000, activation='relu'),
          Dense(1000, activation='relu'),
          Dense(1, activation='sigmoid'),
        ])

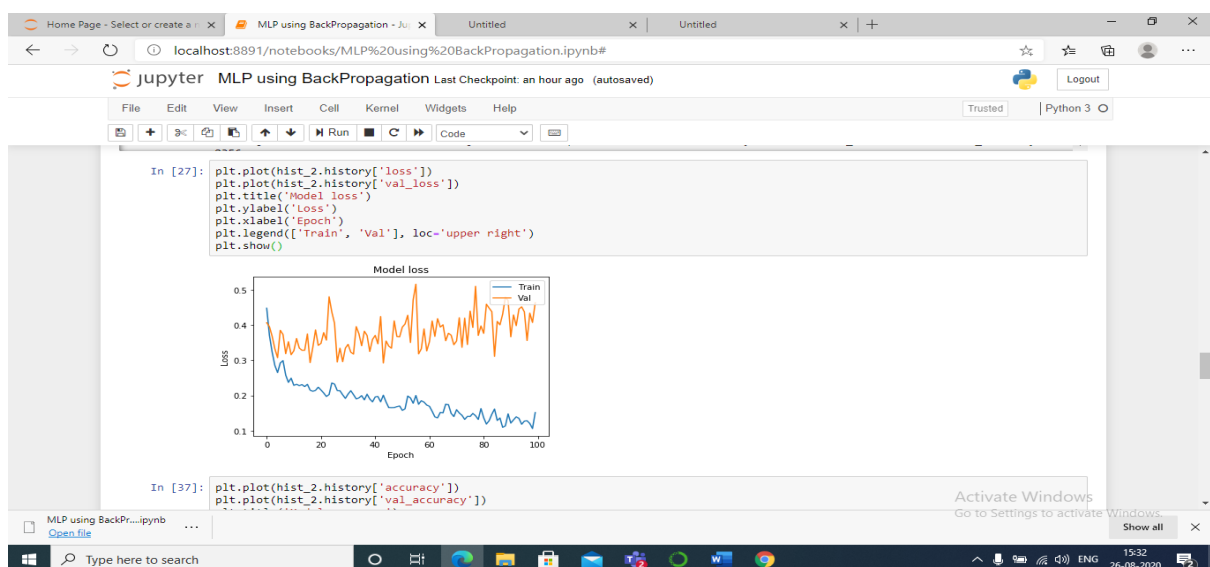
In [25]: model_2.compile(optimizer='adam',
                        loss='binary_crossentropy',
                        metrics=['accuracy'])

In [26]: hist_2 = model_2.fit(X_train, Y_train,
                             batch_size=32, epochs=100,
                             validation_data=(X_val, Y_val))

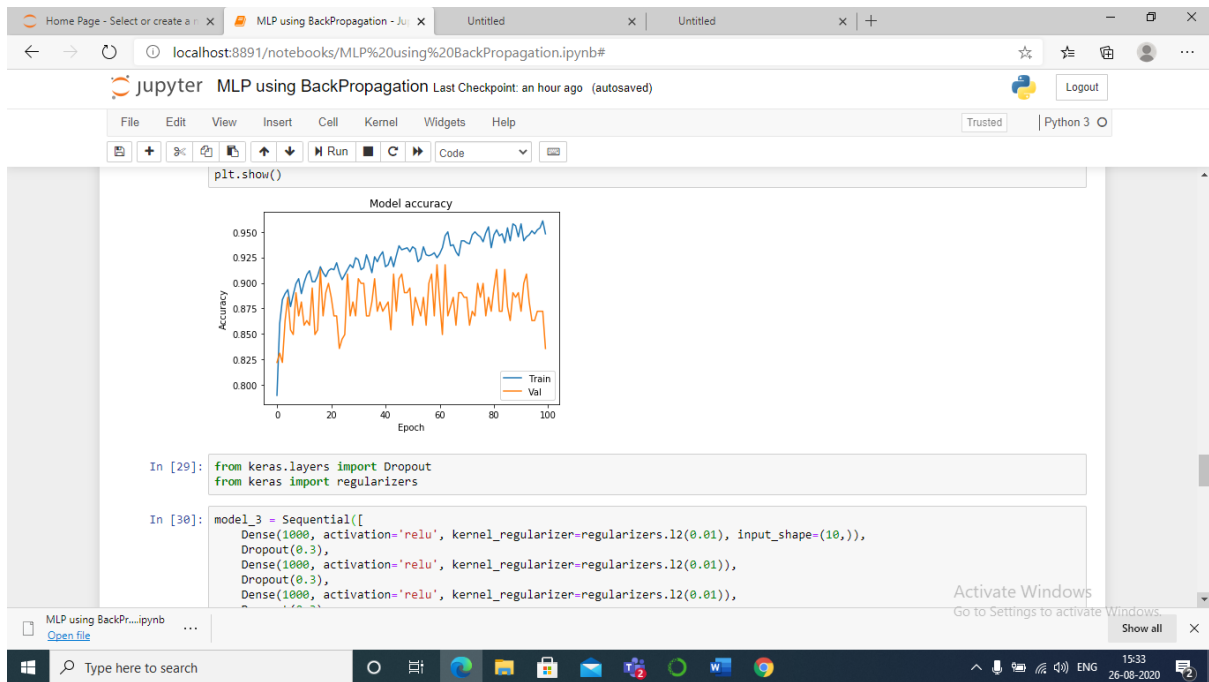
32/32 [=====] - 2s 58ms/step - loss: 0.1348 - accuracy: 0.9472 - val_loss: 0.4457 - val_accuracy: 0.
8813
Epoch 95/100
32/32 [=====] - 2s 62ms/step - loss: 0.1181 - accuracy: 0.9511 - val_loss: 0.4520 - val_accuracy: 0.
8630
Epoch 96/100
32/32 [=====] - 2s 66ms/step - loss: 0.1271 - accuracy: 0.9481 - val_loss: 0.4377 - val_accuracy: 0.
8630
Epoch 97/100
32/32 [=====] - 2s 69ms/step - loss: 0.1280 - accuracy: 0.9521 - val_loss: 0.3561 - val_accuracy: 0.
8721
MLP using BackPr...ipynb ...
Open file
Type here to search
Activate Windows
Go to Settings to activate Windows.
Show all
15:30
26-08-2020
```

## Before Regularisation:

LOSS:

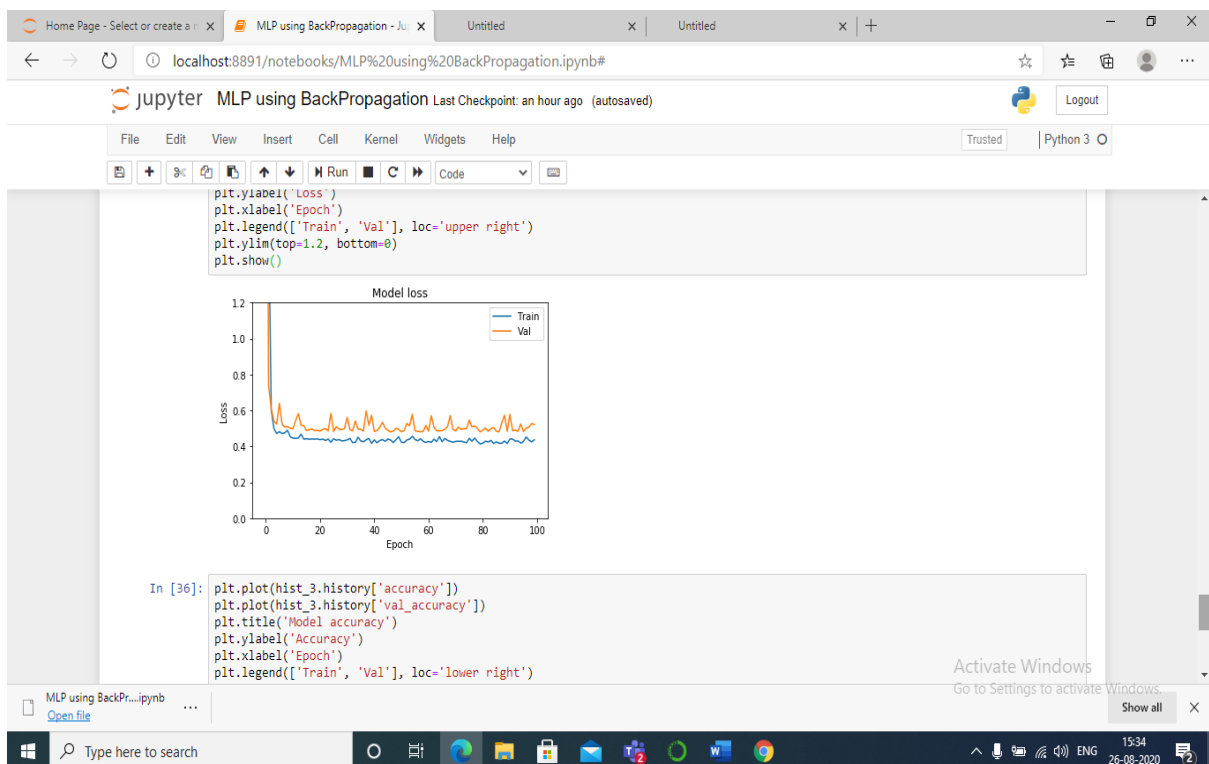


## ACCURACY:



## After Regularization:

## LOSS:



## ACCURACY:

