Name-Yash Gandhi TE IT Batch A 2017140014

Title : Error backPropagation using MLPClassifier

Theory-

Multilayer perceptrons train on a set of input-output pairs and learn to model the correlation (or dependencies) between those inputs and outputs. Training involves adjusting the parameters, or the weights and biases, of the model in order to minimize error. Backpropagation is used to make those weigh and bias adjustments relative to the error, and the error itself can be measured in a variety of ways, including by root mean squared error.

Multi-layer Perceptron classifier:-

hidden_layer_sizes tuple, length = n_layers - 2, default=(100,)--The ith element represents the number of neurons in the ith hidden layer.
Activation {'identity', 'logistic', 'tanh', 'relu'}, default='relu'--Activation function for the hidden layer.

max_iterint default=200---Maximum number of iterations.

Implementation -

Steps:

1. Import the data and perform preprocessing
2. Removed the null values and split the data frame into 2 dataframe one containing all features other containing output coloumn
3. Perform the train test split.
4. Apply MLPclassifier and initialize a model using relevant parameters.
5. Trained the model using (100) epochs depending on your dataset
6. Evaluated the model using test data.
7. Performed predictions on test data.

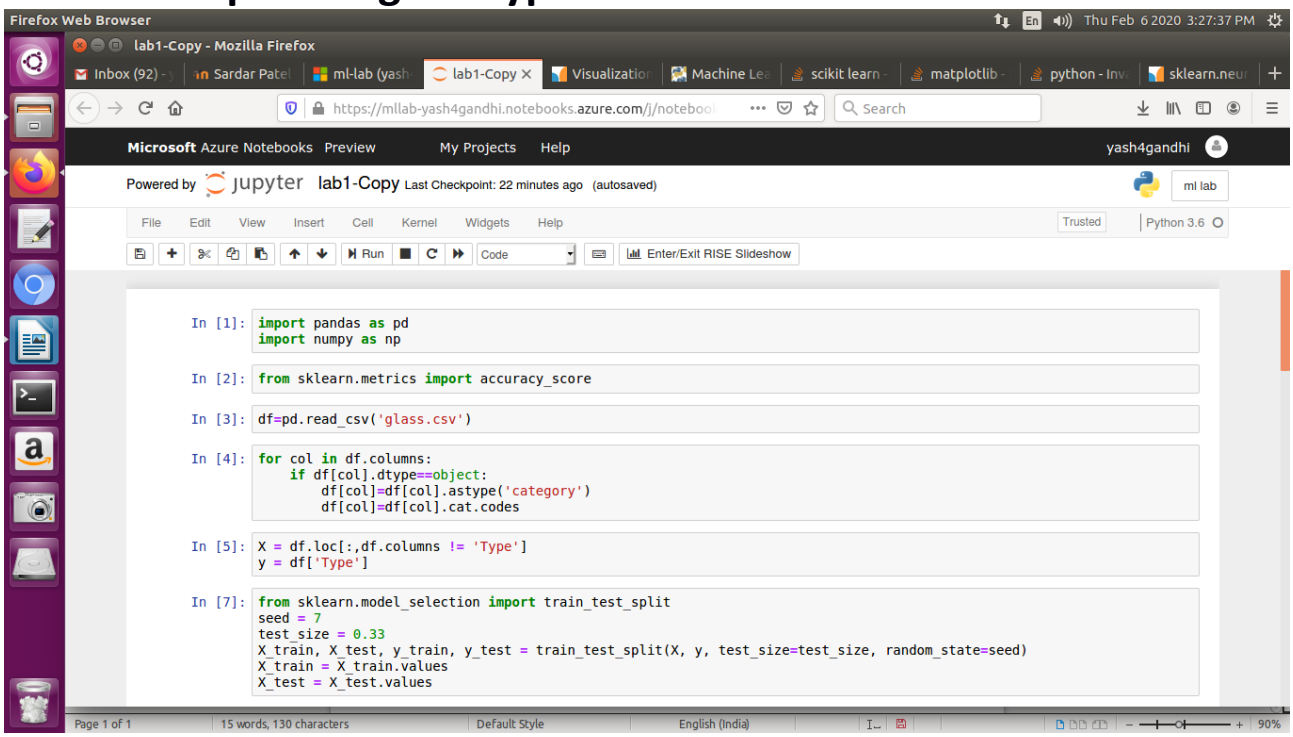
Code-

```
import pandas as pd
import numpy as np

from sklearn.metrics import accuracy_score

df=pd.read_csv('glass.csv')

for col in df.columns:
    if df[col].dtype==object:
        df[col]=df[col].astype('category')
        df[col]=df[col].cat.codes
```

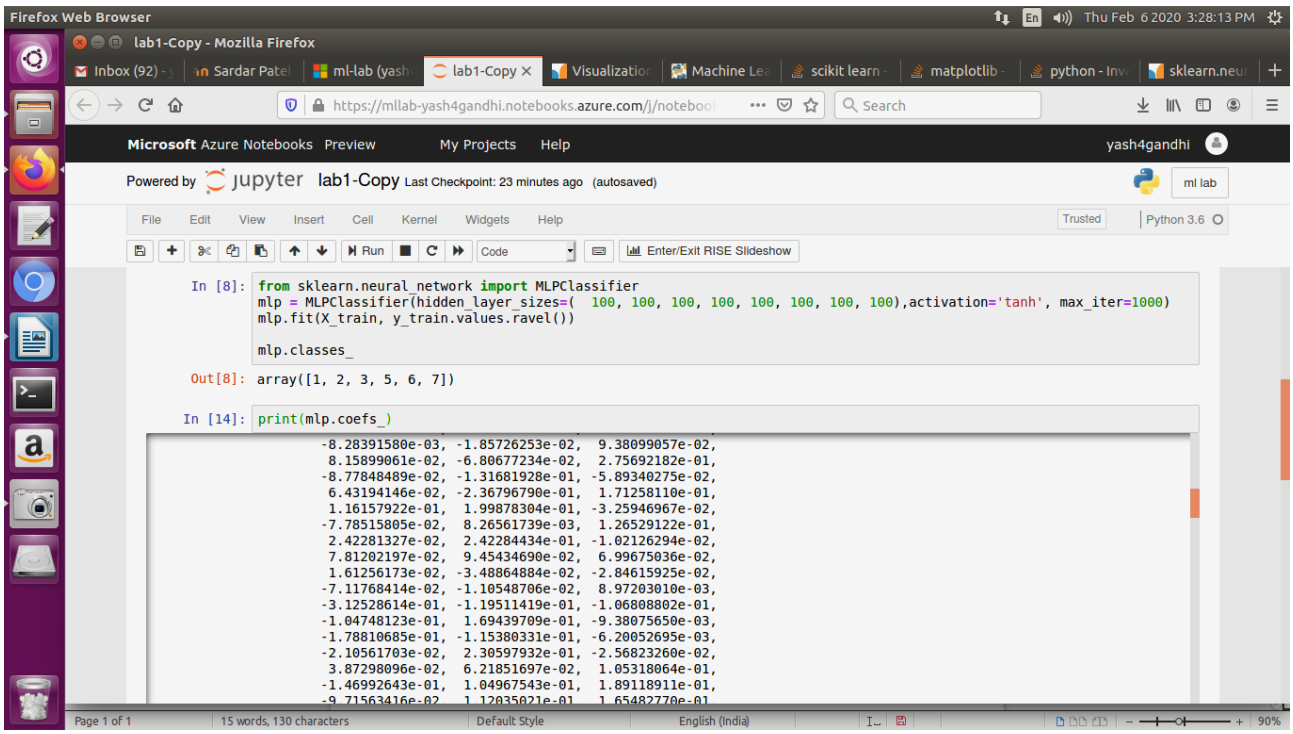
```
X = df.loc[:,df.columns != 'Type']
y = df['Type']


from sklearn.model_selection import train_test_split
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)
X_train = X_train.values
X_test = X_test.values


from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(  100, 100, 100, 100, 100, 100, 100, 100),activation='tanh', max_iter=1000)
mlp.fit(X_train, y_train.values.ravel())

mlp.classes_

print(mlp.coefs_)

predictions = mlp.predict(X_test)
from sklearn.metrics import accuracy_score
X_test
predictions

import matplotlib.pyplot as plt

print("The accuracy is %s" %(accuracy_score(y_test,predictions)))

plt.ylabel('LOSS_FUNCTION')
plt.xlabel('ITERATIONS')

plt.plot(mlp.loss_curve_)
plt.show()
```

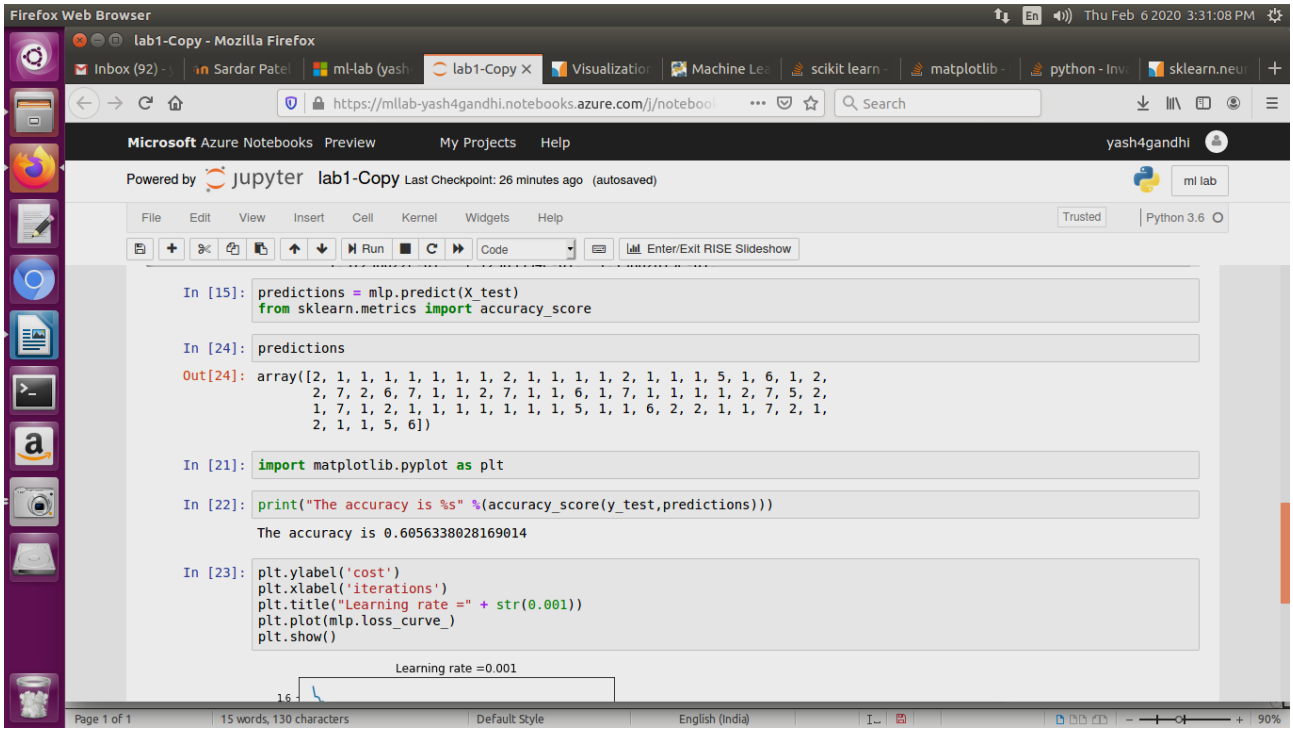
Output-

Dataset-to predict glass type

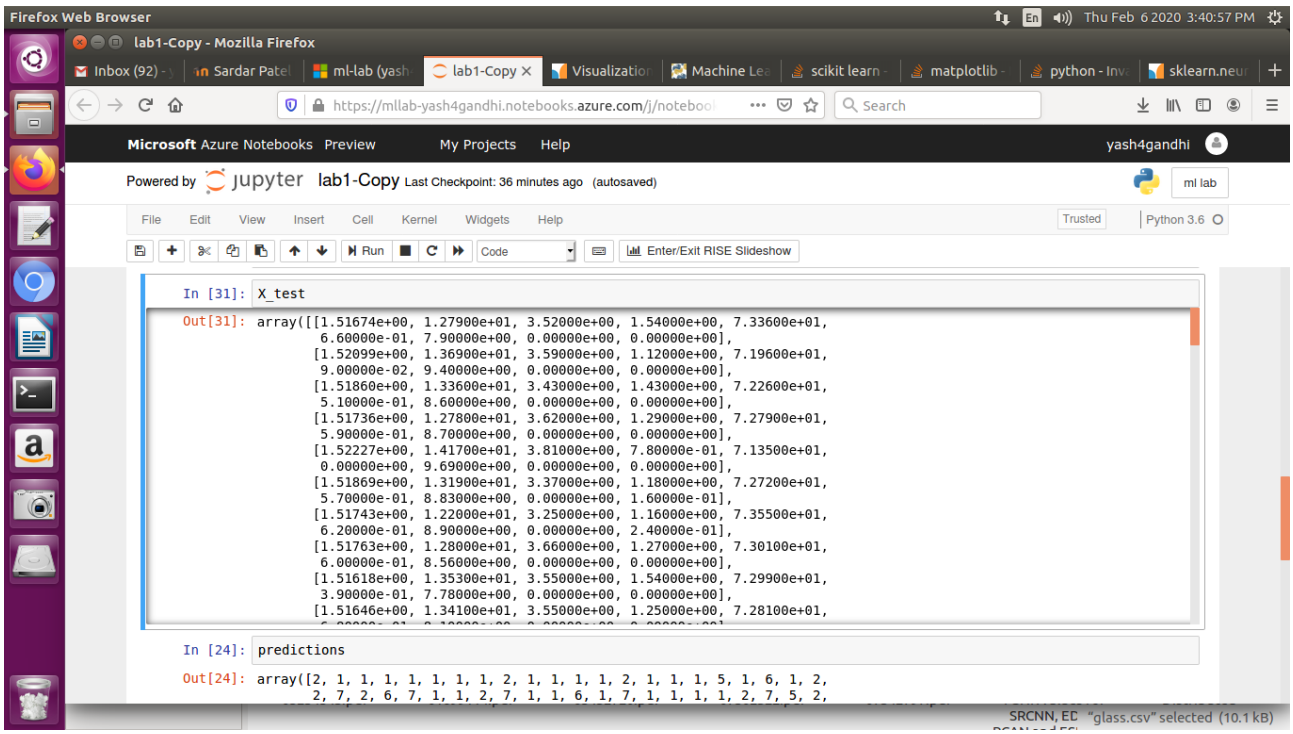


Training model-

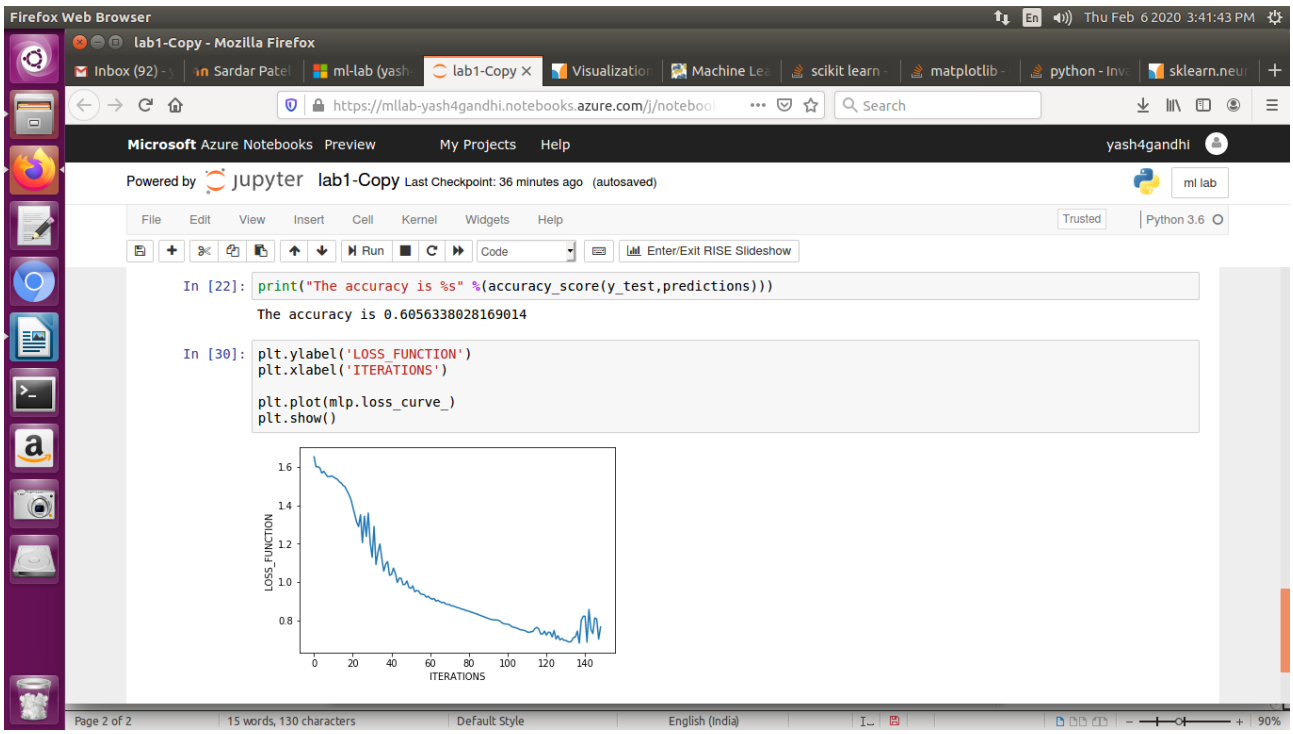




Predictions-



Loss function curve



Conclusion-

Using Backpropagation technique in MLPClassifier hidden layers were set appropriately to in crease the accuracy.

8 hiddenm layers with each layer having 100 nodes were used .

100 iterations were used for the process .

1. On any other number of hiddenlayers the accuracy was less than 60%
2. Activation function tanh was used in this model.