Exercise 4

4.1 Problem Statement:

Solve classification problem by constructing a feedforward neural network using Backpropagation algorithm. (Wheat Seed Data)

4.2 Description of Machine Learning Algorithm:

Backpropagation is one of the important concepts of a neural network. Our task is to classify our data best. For this, we have to update the weights of parameter and bias, but how can we do that in a deep neural network? In the linear regression model, we use gradient descent to optimize the parameter. Similarly here we also use gradient descent algorithm using Backpropagation.

For a single training example, **Backpropagation** algorithm calculates the gradient of the **error function**. Backpropagation can be written as a function of the neural network. Backpropagation algorithms are a set of methods used to efficiently train artificial neural networks following a gradient descent approach which exploits the chain rule.

The main features of Backpropagation are the iterative, recursive and efficient method through which it calculates the updated weight to improve the network until it is not able to perform the task for which it is being trained. Derivatives of the activation function to be known at network design time is required to Backpropagation.

4.3 Description of Data Set:

Wheat seed data was acquired from the 'UCI Center for Machine Learning' repository. It contains seven variables for three distinct types of wheat kernels: (*Kama*, *Rosa*, *Canadian*) designated as numerical variables 1, 2 & 3 respectively.

Title of the data set: Wheat Seed Dataset

Number of Attributes or (Features): 7 Attribute Information:

1. Area
2. Perimeter
3. Compactness
4. Kernel Length
5. Kernel Width
6. Asymmetry Coefficient
7. Kernel Groove Length

4.4 Data Preprocessing and Exploratory Data Analysis:

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

Major Tasks in Data Preprocessing:

1. Data cleaning
2. Data integration
3. Data reduction
4. Data transformation

Exploratory data analysis (EDA) is a technique that data professionals can use to understand a dataset before they start to model it. Some people refer to EDA as data exploration. The goal of conducting EDA is to determine the characteristics of the dataset. Conducting EDA can help data analysts make predictions and assumptions about data. Often, EDA involves data visualization, including creating graphs like histograms, scatter plots and box plots.

Major Tasks in EDA:

1. Observe your dataset
2. Find any missing values
3. Categorize your values
4. Find the shape of your dataset
5. Identify relationships in your dataset
6. Locate any outliers in your dataset

4.5 Machine Learning Package Used for Model building:

For classification of feedforward model we use scikit-neural network-

Neural networks are a machine learning method inspired by how the human brain works. They are particularly good at doing pattern recognition and classification tasks, often using images as inputs. They're a well established machine learning technique that has been around since the 1950s but have gone through several iterations since that have overcome fundamental limitations of the previous one. The current state of the art neural networks are often referred to as deep learning

Perceptrons are the building blocks of neural networks, they're an artificial version of a single neuron on the brain. They typically have one or more inputs and a single output. Each input will be multiplied by a weight and the value of all the weighted inputs are then summed together. Finally the summed value is put through an activation function which decides if the neuron "fires" a signal. In some cases this activation function is simply a threshold step function which outputs zero below a certain input and one above it. Other designs of neurons use other activation functions, but typically they have an output between zero and one and are still step like in their nature.

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation). Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

4.6 Implementation:

```
 from google.colab import files

u=files.upload()
```

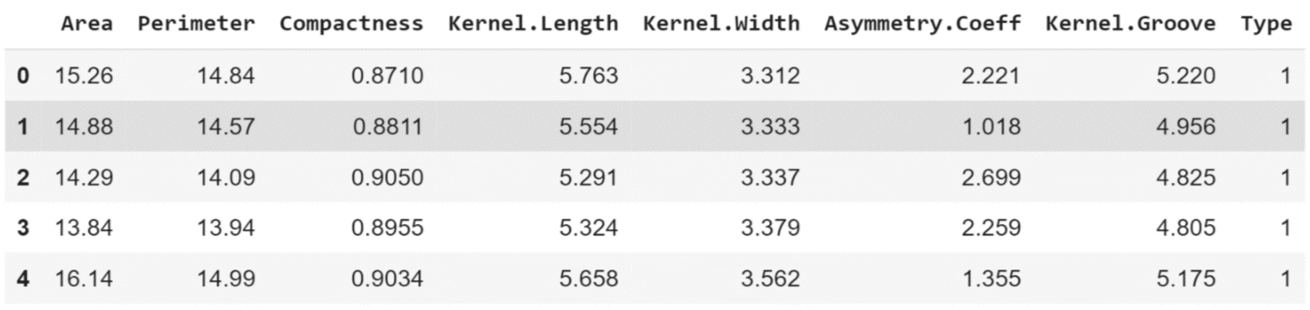
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving seeds.csv to seeds.csv
```

```
# importing numpy, pandas libraries
import pandas as pd
import numpy as np

# loading wheat seeds data into a dataframe
seeds_data = pd.read_csv('seeds.csv')
```

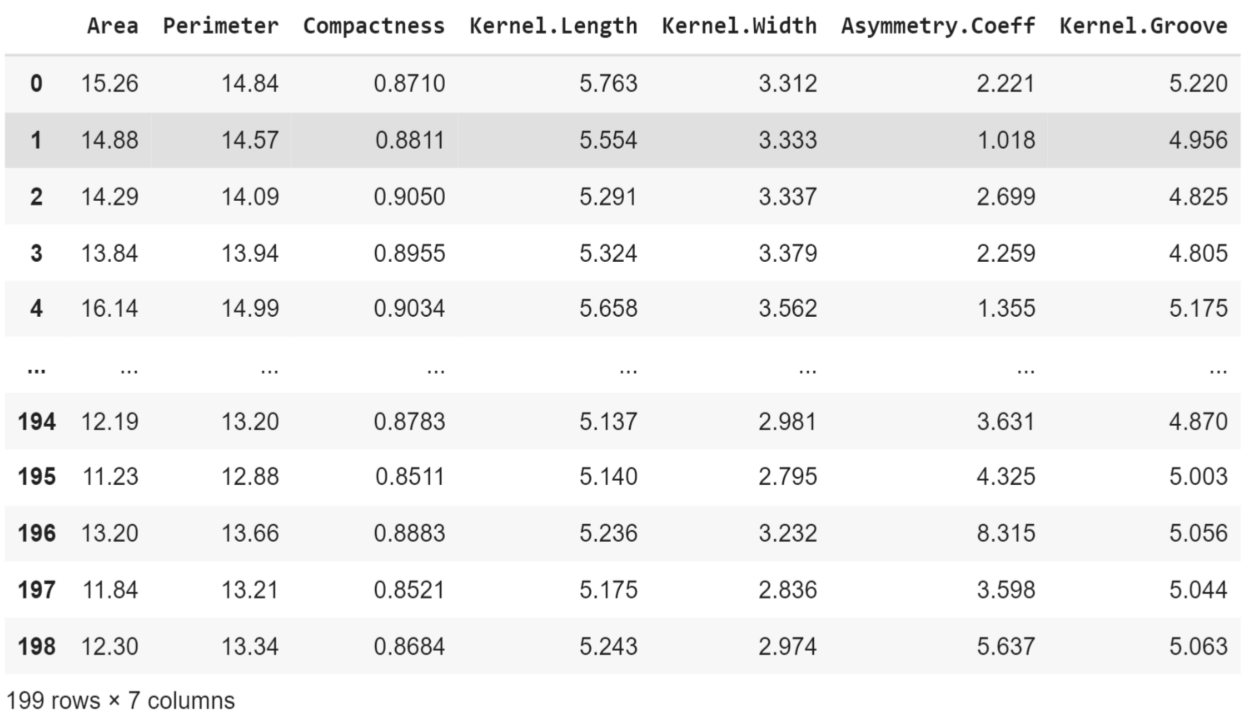
```
# displaying the first 5 rows of wheet seeds data
seeds_data.head()
```



	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kernel.Groove	Type
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1

```
# Extracting Independent Variables

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

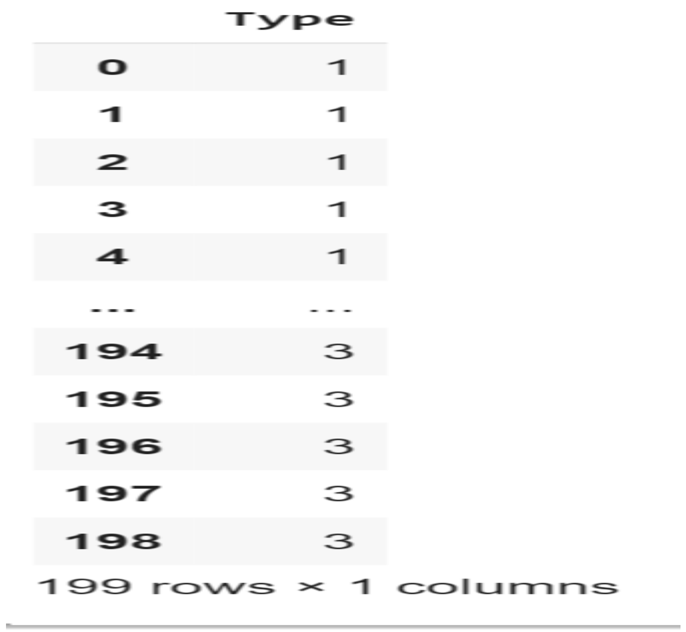


	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kernel.Groove
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175
...	...	...	...	...	...	...	...
194	12.19	13.20	0.8783	5.137	2.981	3.631	4.870
195	11.23	12.88	0.8511	5.140	2.795	4.325	5.003
196	13.20	13.66	0.8883	5.236	3.232	8.315	5.056
197	11.84	13.21	0.8521	5.175	2.836	3.598	5.044
198	12.30	13.34	0.8684	5.243	2.974	5.637	5.063

199 rows × 7 columns

```
# Extracting Target Variable

Y = seeds_data.loc[:, seeds_data.columns == 'Type']
Y
```



	Type
0	1
1	1
2	1
3	1
4	1
...	...
194	3
195	3
196	3
197	3
198	3

199 rows × 1 columns

Split Data for training and testing

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X , Y ,
                                                    test_size = 0.2,
                                                    random_state = 523)
```

Training the Perceptron Classifier

```
# importing Perceptron Class
from sklearn.linear_model import Perceptron


# Creating an insance of Perceptron Class
perc = Perceptron( random_state = 15)


# Training the perceptron classifier
perc.fit(X_train, np.ravel(y_train))

# importing metrics for evaluating perceptron classifier
from sklearn.metrics import accuracy_score

# Using perceptron classifier to make predictions on test data
pred_test = perc.predict(X_test)

# calculating and displaying accuracy score of Perceptron classifier
accuracy = accuracy_score(y_test, pred_test)
print('% of Accuracy using Linear Perceptron: ', accuracy * 100)
```

```
% of Accuracy using Linear Perceptron:  67.5
```

Correlation between two variables can be either a positive correlation, a negative correlation, or no correlation.

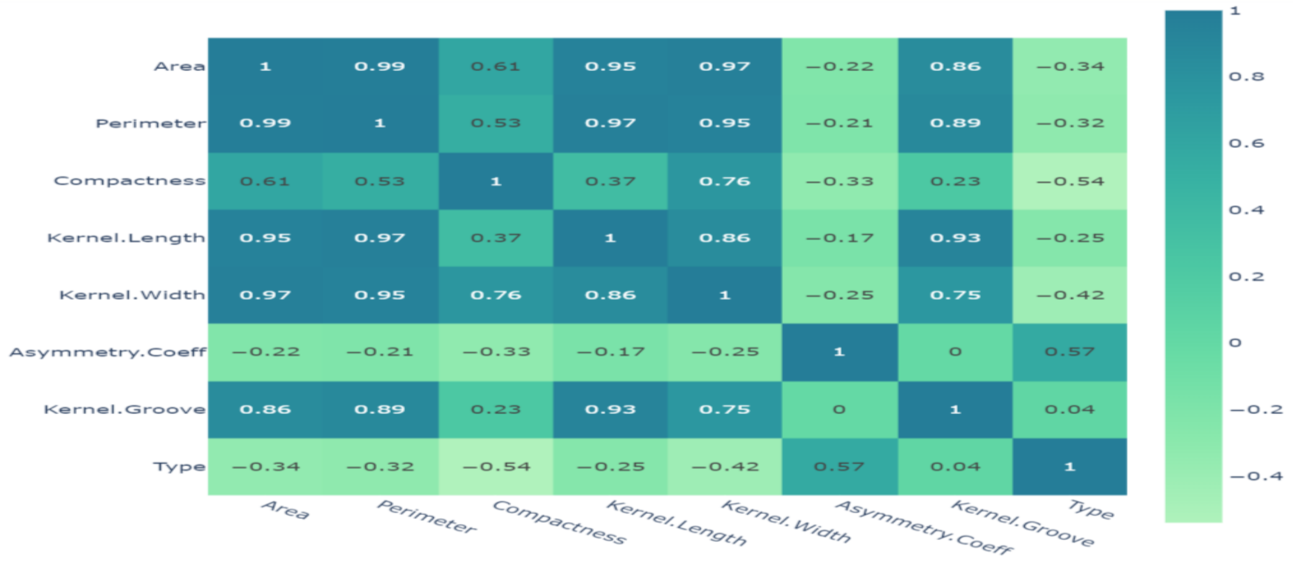
```
# Importing plotly.express
import plotly.express as px
# Finding the correlation of Independent variables on Target Variable
corr = seeds_data.corr()

corr = corr.round(2)
corr
```

	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kernel.Groove	Type
Area	1.00	0.99	0.61	0.95	0.97	-0.22	0.86	-0.34
Perimeter	0.99	1.00	0.53	0.97	0.95	-0.21	0.89	-0.32
Compactness	0.61	0.53	1.00	0.37	0.76	-0.33	0.23	-0.54
Kernel.Length	0.95	0.97	0.37	1.00	0.86	-0.17	0.93	-0.25
Kernel.Width	0.97	0.95	0.76	0.86	1.00	-0.25	0.75	-0.42
Asymmetry.Coeff	-0.22	-0.21	-0.33	-0.17	-0.25	1.00	-0.00	0.57
Kernel.Groove	0.86	0.89	0.23	0.93	0.75	-0.00	1.00	0.04
Type	-0.34	-0.32	-0.54	-0.25	-0.42	0.57	0.04	1.00

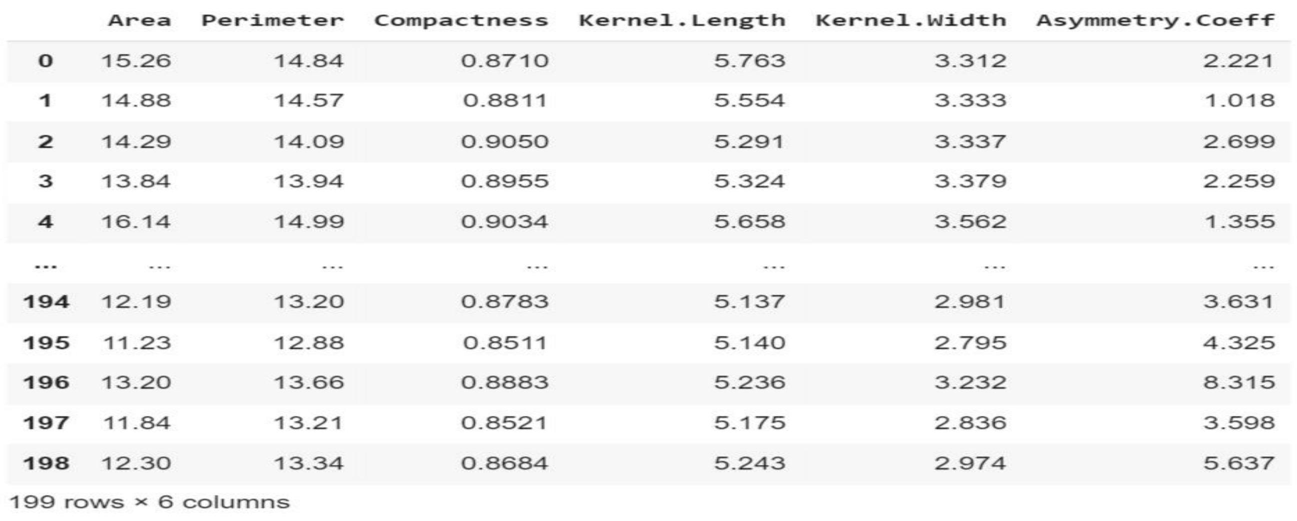
```
# displaying confusion matrix as a heatmap

fig = px.imshow(corr ,
                width = 700,
                height = 700 ,
                text_auto = True,
                color_continuous_scale = 'tealgrn',
                )
fig.show()
```



It can be observed that the attribute "Kernel.Groove" has very least correlation on the target variable

```
# remove Kernel.Groove attribute from X
X = X.loc[:, X.columns != 'Kernel.Groove']
X
```



	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff
0	15.26	14.84	0.8710	5.763	3.312	2.221
1	14.88	14.57	0.8811	5.554	3.333	1.018
2	14.29	14.09	0.9050	5.291	3.337	2.699
3	13.84	13.94	0.8955	5.324	3.379	2.259
4	16.14	14.99	0.9034	5.658	3.562	1.355
...	...	...	...	...	...	...
194	12.19	13.20	0.8783	5.137	2.981	3.631
195	11.23	12.88	0.8511	5.140	2.795	4.325
196	13.20	13.66	0.8883	5.236	3.232	8.315
197	11.84	13.21	0.8521	5.175	2.836	3.598
198	12.30	13.34	0.8684	5.243	2.974	5.637

199 rows × 6 columns

Resplitting Data for training and testing

```
X_train, X_test, y_train, y_test = train_test_split(X , Y ,
                                                    test_size = 0.2,
                                                    random_state = 523)
```

Retraining the Perceptron Classifier

```
# retraining the perceptron classifier
perc.fit(X_train, np.ravel(y_train))


# Using perceptron classifier to make predictions on test data
pred_test = perc.predict(X_test)

# calculating and displaying accuracy score of Perceptron classifier
accuracy = accuracy_score(y_test, pred_test)
print('% of Accuracy using Linear Perceptron: ', accuracy * 100)
```

```
% of Accuracy using Linear Perceptron:  75.0
```

Install scikit-neuralnetwork

```
#scikit-neuralnetwork works withscikit-learn 0.18 and above

# installing scikit-neuralnetwork if not already installed
!pip install scikit-neuralnetwork
```

```
Collecting scikit-neuralnetwork
  Downloading scikit-neuralnetwork-0.7.tar.gz (33 kB)
Requirement already satisfied: scikit-learn>=0.17 in
/usr/local/lib/python3.7/dist-packages (from scikit-neuralnetwork) (1.0.2)
Collecting Theano>=0.8
  Downloading Theano-1.0.5.tar.gz (2.8 MB)
     |████████████████████████████████| 2.8 MB 20.9 MB/s
Collecting Lasagne>=0.1
  Downloading Lasagne-0.1.tar.gz (125 kB)
     |████████████████████████████████| 125 kB 65.3 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from Lasagne>=0.1->scikit-neuralnetwork) (1.21.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.17->scikit-
neuralnetwork) (3.1.0)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=0.17->scikit-neuralnetwork) (1.4.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=0.17->scikit-neuralnetwork) (1.1.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-
packages (from Theano>=0.8->scikit-neuralnetwork) (1.15.0)
Building wheels for collected packages: scikit-neuralnetwork, Lasagne, Theano
  Building wheel for scikit-neuralnetwork (setup.py) ... done
  Created wheel for scikit-neuralnetwork: filename=scikit_neuralnetwork-0.7-
py3-none-any.whl size=41698
sha256=eb62b6b622eab53e47b166951c79aafed2c45dae67c48c870f38d828778029ed
  Stored in directory:
/root/.cache/pip/wheels/7d/42/93/b99bd6392fb56ec7831a695cb7a23dd9c73382b258614b
62ed
  Building wheel for Lasagne (setup.py) ... done
  Created wheel for Lasagne: filename=Lasagne-0.1-py3-none-any.whl size=79283
sha256=dd918baee132e2877a3a45b2de0fd64425814286e1b60da5828a3f1f9fea19e2
  Stored in directory:
/root/.cache/pip/wheels/a3/72/b6/89bbeb6140ee3756fa2bdd2fb03003dd60d289851314b3
5fd7
  Building wheel for Theano (setup.py) ... done
  Created wheel for Theano: filename=Theano-1.0.5-py3-none-any.whl size=2668111
sha256=14fd92a00aa4addce42d4edef829812ee371c2dd062e0ea0008c5bdbb41686b8
  Stored in directory:
/root/.cache/pip/wheels/26/68/6f/745330367ce7822fe0cd863712858151f5723a0a5e322c
c144
Successfully built scikit-neuralnetwork Lasagne Theano
Installing collected packages: Theano, Lasagne, scikit-neuralnetwork
Successfully installed Lasagne-0.1 Theano-1.0.5 scikit-neuralnetwork-0.7
```

Training the Multilayer Perceptron Classifier using Backpropagation algorithm

```
# importing required library
import sklearn.neural_network as nn
```

```
# Creating an instance of MLPClassifier class
# Taking maximum number of iterations = 1000
# constructing MLP network with 3 hidden layers with
#            100 neurons in hidden layer 1,
#            75 neurons in hidden layer 2,
#            50 neurons in hidden layer 3

mlp = nn.MLPClassifier(random_state = 560,
                       hidden_layer_sizes = [100, 75, 50],
                       max_iter = 1000)
# Training the MLP classifier
mlp.fit(X_train, np.ravel(y_train))

pred_test = mlp.predict(X_test)
mlp_accuracy = accuracy_score(y_test, pred_test)
print('% of Accuracy using MultiLayer Perceptron: ', "{0:0.2f}".format(mlp_accu
racy*100))
```

```
% of Accuracy using MultiLayer Perceptron:  87.50
```

4.7 Results and Discussion:

The problem to constructing a feedforward neural network using Backpropagation algorithm (Wheat Seed Data) is classified Successfully.

when we classify the dataset using perceptron model we get less Accuracy than classifying dataset by using multilayer perceptron backpropagation algorithm.