Exercise 5: Solve classification problem by constructing a feedforward neural network using Backpropagation algorithm. (Wheat Seed Data)

```
# importing numpy, pandas libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# loading wheat seeds data into a dataframe
from google.colab import drive
drive.mount("/content/gdrive")
seeds_data = pd.read_csv('gdrive/My Drive/ml/seeds.csv')

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

```
Mounted at /content/gdrive
```

	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kerne
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	

Next steps: Generate code with seeds_data View recommended plots

```
# Extracting Independent Variables

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

	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Ker
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 × 7 columns

Next steps: Generate code with X View recommended plots

```
# 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
```

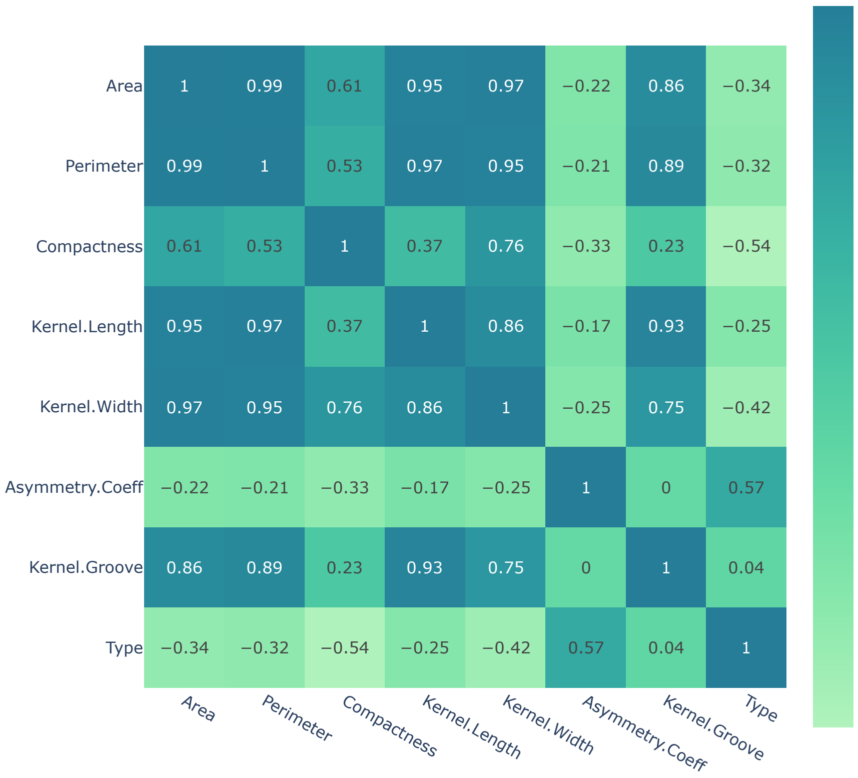
	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry
Area	1.00	0.99	0.61	0.95	0.97	
Perimeter	0.99	1.00	0.53	0.97	0.95	
Compactness	0.61	0.53	1.00	0.37	0.76	
Kernel.Length	0.95	0.97	0.37	1.00	0.86	
Kernel.Width	0.97	0.95	0.76	0.86	1.00	
Asymmetry.Coeff	-0.22	-0.21	-0.33	-0.17	-0.25	
Kernel.Groove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

Next steps: Generate code with `corr` View recommended plots

```
# displaying confusion matrix as a heatmap

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

fig.show()
```



	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kernel.Groove	Type
Area	1	0.99	0.61	0.95	0.97	−0.22	0.86	−0.34
Perimeter	0.99	1	0.53	0.97	0.95	−0.21	0.89	−0.32
Compactness	0.61	0.53	1	0.37	0.76	−0.33	0.23	−0.54
Kernel.Length	0.95	0.97	0.37	1	0.86	−0.17	0.93	−0.25
Kernel.Width	0.97	0.95	0.76	0.86	1	−0.25	0.75	−0.42
Asymmetry.Coeff	−0.22	−0.21	−0.33	−0.17	−0.25	1	0	0.57
Kernel.Groove	0.86	0.89	0.23	0.93	0.75	0	1	0.04
Type	−0.34	−0.32	−0.54	−0.25	−0.42	0.57	0.04	1

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

Next steps: Generate code with X View recommended plots

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)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: scikit-learn>=0.17 in /usr/local/lib/python3.10/dist-packages (from scikit-neuralnetwork) (1.2.2)
Collecting Theano>=0.8 (from scikit-neuralnetwork)
  Downloading Theano-1.0.5.tar.gz (2.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 33.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting Lasagne>=0.1 (from scikit-neuralnetwork)
  Downloading Lasagne-0.1.tar.gz (125 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 125.1/125.1 kB 15.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from Lasagne>=0.1->scikit-neuralnetwork) (1.25.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.17->scikit-neuralnetwor
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.17->scikit-neuralnetwo
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.17->scikit-neur
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from Theano>=0.8->scikit-neuralnetwork) (1.16
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=41668 sha256=de457df34f14a17d175a1
  Stored in directory: /root/.cache/pip/wheels/a7/a0/1c/dad06c626e295c7eb973fa594f481ab8719a0ba053f14c5433
  Building wheel for Lasagne (setup.py) ... done
  Created wheel for Lasagne: filename=Lasagne-0.1-py3-none-any.whl size=79269 sha256=e47972ff390167abcc8dd31b36276b944278e248adf6b6
  Stored in directory: /root/.cache/pip/wheels/26/22/9f/1512e23c2556397304b730376ae4a7b34c5a6e8d68c1273917
  Building wheel for Theano (setup.py) ... done
  Created wheel for Theano: filename=Theano-1.0.5-py3-none-any.whl size=2668109 sha256=6dbac29519002e7b47b43bdd6a335d0789093e9f5f21
  Stored in directory: /root/.cache/pip/wheels/d9/e6/7d/2267d21a99e4ab8276f976f293b4ff23f50c9d809f4a216ebb
Successfully built scikit-neuralnetwork Lasagne Theano
Installing collected packages: Lasagne, Theano, 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_accuracy*100))
```

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