```
#Import required libraries
import keras #library for neural network
import pandas as pd #loading data in table form
import seaborn as sns #visualisation
import matplotlib.pyplot as plt #visualisation
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.preprocessing import normalize #machine learning algorithm library


#Reading data
data=pd.read_csv("Iris.csv")
print("Describing the data: ",data.describe())
print("Info of the data:",data.info())
```

```
Describing the data:                 Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  Pet
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Info of the data: None
```

```
print("10 first samples of the dataset:",data.head(10))
print("10 last samples of the dataset:",data.tail(10))
```

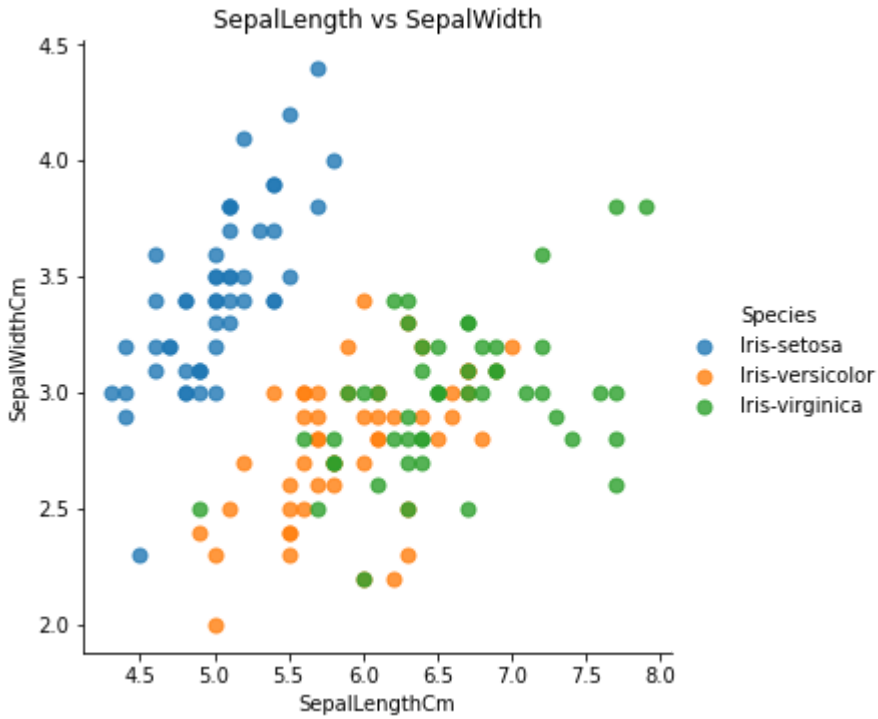
```
10 first samples of the dataset:    Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  Peta
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
5   6            5.4           3.9            1.7           0.4  Iris-setosa
6   7            4.6           3.4            1.4           0.3  Iris-setosa
7   8            5.0           3.4            1.5           0.2  Iris-setosa
```

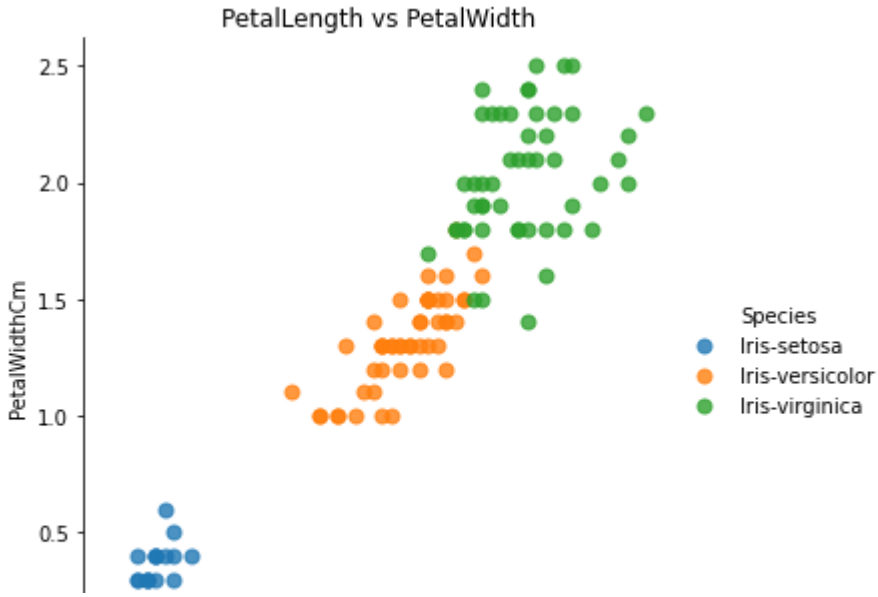
```
8   9            4.4           2.9            1.4           0.2  Iris-setosa
9  10            4.9           3.1            1.5           0.1  Iris-setosa
10 last samples of the dataset:       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  Pe
140  141            6.7           3.1            5.6           2.4
141  142            6.9           3.1            5.1           2.3
142  143            5.8           2.7            5.1           1.9
143  144            6.8           3.2            5.9           2.3
144  145            6.7           3.3            5.7           2.5
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
140  Iris-virginica
141  Iris-virginica
142  Iris-virginica
143  Iris-virginica
144  Iris-virginica
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

```
sns.lmplot('SepalLengthCm', 'SepalWidthCm',
           data=data,
           fit_reg=False,
           hue="Species",
           scatter_kws={"marker": "D",
                        "s": 50})
plt.title('SepalLength vs SepalWidth')

sns.lmplot('PetalLengthCm', 'PetalWidthCm',
           data=data,
           fit_reg=False,
           hue="Species",
           scatter_kws={"marker": "D",
                        "s": 50})
plt.title('PetalLength vs PetalWidth')

sns.lmplot('SepalLengthCm', 'PetalLengthCm',
           data=data,
           fit_reg=False,
           hue="Species",
           scatter_kws={"marker": "D",
                        "s": 50})
plt.title('SepalLength vs PetalLength')

sns.lmplot('SepalWidthCm', 'PetalWidthCm',
           data=data,
```

```
            fit_reg=False,
            hue="Species",
            scatter_kws={"marker": "D",
                         "s": 50})
plt.title('SepalWidth vs PetalWidth')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
```




```
print(data["Species"].unique())
```

```
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

SepalLength vs PetalLength

```
data.loc[data["Species"]=="Iris-setosa","Species"]=0
data.loc[data["Species"]=="Iris-versicolor","Species"]=1
data.loc[data["Species"]=="Iris-virginica","Species"]=2
print(data.head())
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm Species
0   1            5.1           3.5            1.4           0.2       0
1   2            4.9           3.0            1.4           0.2       0
2   3            4.7           3.2            1.3           0.2       0
3   4            4.6           3.1            1.5           0.2       0
4   5            5.0           3.6            1.4           0.2       0
```

```
data=data.iloc[np.random.permutation(len(data))]
print(data.head())
```

```
     Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm Species
82   83            5.8           2.7            3.9           1.2       1
105 106            7.6           3.0            6.6           2.1       2
44   45            5.1           3.8            1.9           0.4       0
88   89            5.6           3.0            4.1           1.3       1
12   13            4.8           3.0            1.4           0.1       0
```

```
X=data.iloc[:,1:5].values
y=data.iloc[:,5].values

print("Shape of X",X.shape)
print("Shape of y",y.shape)
print("Examples of X\n",X[:3])
print("Examples of y\n",y[:3])
```

```
Shape of X (150, 4)
Shape of y (150,)
Examples of X
 [[5.8 2.7 3.9 1.2]
 [7.6 3.  6.6 2.1]
 [5.1 3.8 1.9 0.4]]
Examples of y
 [1 2 0]
```

```
X_normalized=normalize(X,axis=0)
print("Examples of X_normalised\n",X_normalized[:3])
```

```
Examples of X_normalised
 [[0.08024771 0.07147336 0.07673657 0.06901797]
 [0.10515217 0.07941484 0.12986189 0.12078145]
 [0.07056264 0.10059213 0.03738448 0.02300599]]
```

```
#Creating train,test and validation data
'''
80% -- train data
20% -- test data
'''
total_length=len(data)
train_length=int(0.8*total_length)
```

```
test_length=int(0.2*total_length)

X_train=X_normalized[:train_length]
X_test=X_normalized[train_length:]
y_train=y[:train_length]
y_test=y[train_length:]

print("Length of train set x:",X_train.shape[0],"y:",y_train.shape[0])
print("Length of test set x:",X_test.shape[0],"y:",y_test.shape[0])
```

```
Length of train set x: 120 y: 120
Length of test set x: 30 y: 30
```

```
#Neural network module
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from keras.layers import BatchNormalization
from keras.utils import np_utils


#Change the label to one hot vector
'''
[0]--->[1 0 0]
[1]--->[0 1 0]
[2]--->[0 0 1]
'''
y_train=np_utils.to_categorical(y_train,num_classes=3)
y_test=np_utils.to_categorical(y_test,num_classes=3)
print("Shape of y_train",y_train.shape)
print("Shape of y_test",y_test.shape)
```

```
Shape of y_train (120, 3)
Shape of y_test (30, 3)
```

```
model=Sequential()
model.add(Dense(1000,input_dim=4,activation='relu'))
model.add(Dense(500,activation='relu'))
model.add(Dense(300,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(3,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1000)              5000
```

```
 dense_1 (Dense)             (None, 500)               500500

 dense_2 (Dense)             (None, 300)               150300

 dropout (Dropout)           (None, 300)               0

 dense_3 (Dense)             (None, 3)                 903

=================================================================
Total params: 656,703
Trainable params: 656,703
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(X_train,y_train,validation_data=(X_test,y_test),batch_size=20,epochs=10,verbose=1)
```

```
Epoch 1/10
6/6 [==============================] - 1s 62ms/step - loss: 1.0797 - accuracy: 0.5500 -
Epoch 2/10
6/6 [==============================] - 0s 14ms/step - loss: 1.0065 - accuracy: 0.7750 -
Epoch 3/10
6/6 [==============================] - 0s 13ms/step - loss: 0.8619 - accuracy: 0.9250 -
Epoch 4/10
6/6 [==============================] - 0s 14ms/step - loss: 0.6553 - accuracy: 0.9167 -
Epoch 5/10
6/6 [==============================] - 0s 14ms/step - loss: 0.4624 - accuracy: 0.9500 -
Epoch 6/10
6/6 [==============================] - 0s 17ms/step - loss: 0.3240 - accuracy: 0.9833 -
Epoch 7/10
6/6 [==============================] - 0s 17ms/step - loss: 0.2772 - accuracy: 0.8750 -
Epoch 8/10
6/6 [==============================] - 0s 16ms/step - loss: 0.2956 - accuracy: 0.8500 -
Epoch 9/10
6/6 [==============================] - 0s 14ms/step - loss: 0.2079 - accuracy: 0.9417 -
Epoch 10/10
6/6 [==============================] - 0s 18ms/step - loss: 0.1800 - accuracy: 0.9417 -
<keras.callbacks.History at 0x7fa93e098f50>
```

```
prediction=model.predict(X_test)
length=len(prediction)
y_label=np.argmax(y_test,axis=1)
predict_label=np.argmax(prediction,axis=1)

accuracy=np.sum(y_label==predict_label)/length * 100
print("Accuracy of the dataset",accuracy )
```

```
Accuracy of the dataset 96.66666666666667
```