In [1]:
```python
import pandas
import numpy as np
```

In [2]:
```python
dataset = pandas.read_csv("iris.csv")
dataset = dataset.sample(frac=1)
```

In [3]:
```python
#Getting the input Data.
X = dataset[["sepal_length","sepal_width","petal_length","petal_width"]]
print(X.head())
X = np.array(X)
print(f"\nshape of X is {X.shape}")
```

```
     sepal_length  sepal_width  petal_length  petal_width
142           5.8          2.7           5.1          1.9
52            6.9          3.1           4.9          1.5
36            5.5          3.5           1.3          0.2
1             4.9          3.0           1.4          0.2
18            5.7          3.8           1.7          0.3

shape of X is (150, 4)
```

In [4]:
```python
#Getting the Classes.
Y = dataset["species"]
print(Y.head())
Y = np.array(Y).reshape((150,1))
print(f"\nShape of Y is {Y.shape}")
```

```
142     Iris-virginica
52     Iris-versicolor
36         Iris-setosa
1          Iris-setosa
18         Iris-setosa
Name: species, dtype: object

Shape of Y is (150, 1)
```

In [5]:
```python
#Pre-Processing the output Classes.
Y[Y=="Iris-setosa"] = 0
Y[Y=="Iris-versicolor"] = 1
Y[Y=="Iris-virginica"] = 2
```

In [6]:
```python
print(f"Shape of Y after Pre-Processing is {Y.shape}\n")
print(np.squeeze(Y))
```

```
Shape of Y after Pre-Processing is (150, 1)

[2 1 0 0 0 1 1 2 0 2 0 0 2 0 0 0 0 2 1 0 2 0 1 2 1 1 1 1 0 0 2 2 2 2 1 0 2
 1 0 0 0 2 2 1 0 1 1 2 2 1 0 2 0 1 0 1 0 0 0 2 2 1 2 1 1 2 2 0 1 1 1 0 1 0
 1 0 0 2 1 2 1 0 1 1 0 2 2 0 0 1 2 1 2 2 1 1 0 0 1 1 1 1 2 2 0 0 0 0 1 2 0
 1 2 2 2 1 2 1 1 1 2 0 2 0 1 2 0 2 0 1 2 1 0 2 2 2 0 0 0 2 1 1 1 2 0 1 2 2
 2 2]
```

In [7]:
```python
from sklearn import preprocessing
```

In [8]:
```python
#One Hot Encoding Y.
one_hot = preprocessing.OneHotEncoder()
O_e =  one_hot.fit(Y)
Y_e = O_e.transform(Y)
Y = Y_e.toarray()
print(f"Shape of Y is {Y.shape}")
```

```
Shape of Y is (150, 3)

c:\users\shambu\appdata\local\programs\python\python37\lib\site-packages\sklearn\preprocessi
ng\_encoders.py:415: FutureWarning: The handling of integer data will change in version 0.22. C
urrently, the categories are determined based on the range [0, max(values)], while in the future t
hey will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to intege
rs, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

In [9]:
```python
import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout
```

In [10]:
```python
model = Sequential()
model.add(Dense(10,input_shape=(4,),activation='relu'))
model.add(Dense(10,activation='relu'))
model.add(Dense(10,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(3))
```

In [11]:
```python
predictions = model(X).numpy()
```

```
WARNING:tensorflow:Layer dense is casting an input tensor from dtype float64 to the layer's dty
pe of float32, which is new behavior in TensorFlow 2.  The layer has dtype float32 because it's dt
ype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this warning. If in doubt, this war
ning is likely only an issue if you are porting a TensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backend.set_floatx('float64')`.
To change just this layer, pass dtype='float64' to the layer constructor. If you are the author of thi
s layer, you can disable autocasting by passing autocast=False to the base Layer constructor.
```

In [12]:
```
#Initial Predictions
Y_beta = tf.nn.softmax(predictions).numpy()
print(Y_beta[:5])
print(np.sum(Y_beta[:5]))
Y_beta = np.argmax(Y_beta,axis = 1)
print(Y_beta[:5])
```

```
[[0.30060497 0.60891485 0.09048025]
 [0.3051782  0.61609834 0.07872337]
 [0.28733787 0.5755495  0.13711268]
 [0.30517185 0.5413889  0.15343921]
 [0.2868019  0.5829144  0.13028374]]
5.0
[1 1 1 1 1]
```

In [13]:
```
lossSoft = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
```

In [14]:
```
print(lossSoft(Y[:5],predictions[:5]).numpy())
```

```
1.3139826
```

In [15]:
```
X_train =  X[:-50]
Y_train = Y[:-50]
X_test = X[100:-1]
Y_test = Y[100:-1]
model.compile(optimizer = "adam",loss=lossSoft,metrics=["accuracy"])
```

In [21]:
```
model.fit(X,Y,epochs=10)
```

```
Train on 150 samples
Epoch 1/10
150/150 [==============================] - 0s 164us/sample - loss: 0.2053 - accuracy:
0.9067
Epoch 2/10
150/150 [==============================] - 0s 121us/sample - loss: 0.2443 - accuracy:
0.9000
Epoch 3/10
150/150 [==============================] - 0s 125us/sample - loss: 0.2462 - accuracy:
0.8867
Epoch 4/10
150/150 [==============================] - 0s 130us/sample - loss: 0.2603 - accuracy:
0.8600
Epoch 5/10
150/150 [==============================] - 0s 154us/sample - loss: 0.2465 - accuracy:
0.8800
Epoch 6/10
150/150 [==============================] - 0s 127us/sample - loss: 0.2363 - accuracy:
0.9000
Epoch 7/10
150/150 [==============================] - 0s 116us/sample - loss: 0.1963 - accuracy:
0.9467
Epoch 8/10
150/150 [==============================] - 0s 155us/sample - loss: 0.2425 - accuracy:
0.8933
Epoch 9/10
150/150 [==============================] - 0s 109us/sample - loss: 0.2504 - accuracy:
0.9200
Epoch 10/10
150/150 [==============================] - 0s 143us/sample - loss: 0.2027 - accuracy:
0.9400
```

Out[21]:
```
<tensorflow.python.keras.callbacks.History at 0x1aad68aae48>
```

In [22]:
```
probability_model = tf.keras.Sequential([
  model,
  tf.keras.layers.Softmax()
])
```

In [23]:

```
print(np.argmax(probability_model(X[:100]).numpy(),axis=1))
print(Y[:100])
```

```
[2 1 0 0 0 1 1 2 0 2 0 0 2 0 0 0 0 2 1 0 2 0 1 2 1 1 1 1 0 0 2 2 2 2 1 0 2
 1 0 0 0 2 2 2 0 1 1 2 2 1 0 2 0 1 0 1 0 0 0 2 2 1 2 1 1 2 2 0 1 1 1 0 1 0
 2 0 0 2 2 2 1 0 1 1 0 2 2 0 0 1 2 1 2 2 1 1 0 0 1 1]
[[0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
```

```
[1. 0. 0.]
[0. 1. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[0. 0. 1.]
[0. 0. 1.]
[0. 1. 0.]
[0. 0. 1.]
[0. 1. 0.]
[0. 1. 0.]
[0. 0. 1.]
[0. 0. 1.]
[1. 0. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[1. 0. 0.]
[0. 1. 0.]
[1. 0. 0.]
[0. 1. 0.]
[1. 0. 0.]
[1. 0. 0.]
[0. 0. 1.]
[0. 1. 0.]
[0. 0. 1.]
[0. 1. 0.]
[1. 0. 0.]
[0. 1. 0.]
[0. 1. 0.]
[1. 0. 0.]
[0. 0. 1.]
[0. 0. 1.]
[1. 0. 0.]
[1. 0. 0.]
[0. 1. 0.]
[0. 0. 1.]
[0. 1. 0.]
[0. 0. 1.]
[0. 0. 1.]
[0. 1. 0.]
[0. 1. 0.]
[1. 0. 0.]
[1. 0. 0.]
[0. 1. 0.]
[0. 1. 0.]]
```

In [ ]: