Importing Packages

```
import pandas as pd
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn import datasets
from sklearn.model_selection import train_test_split
```

Loading Dataset

```
data = datasets.load_iris()
print(data)
print("\nFeature Names in the data")
print(data.feature_names)
```

```
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
```

```
        [6.3, 2.7, 4.9, 1.8],
        [6.7, 3.3, 5.7, 2.1],
        [7.2, 3.2, 6. , 1.8],
        [6.2, 2.8, 4.8, 1.8],
        [6.1, 3. , 4.9, 1.8],
        [6.4, 2.8, 5.6, 2.1],
        [7.2, 3. , 5.8, 1.6],
        [7.4, 2.8, 6.1, 1.9],
        [7.9, 3.8, 6.4, 2. ],
        [6.4, 2.8, 5.6, 2.2],
        [6.3, 2.8, 5.1, 1.5],
        [6.1, 2.6, 5.6, 1.4],
        [7.7, 3. , 6.1, 2.3],
        [6.3, 3.4, 5.6, 2.4],
        [6.4, 3.1, 5.5, 1.8],
        [6. , 3. , 4.8, 1.8],
        [6.9, 3.1, 5.4, 2.1],
        [6.7, 3.1, 5.6, 2.4],
        [6.9, 3.1, 5.1, 2.3],
        [5.8, 2.7, 5.1, 1.9],
        [6.8, 3.2, 5.9, 2.3],
        [6.7, 3.3, 5.7, 2.5],
        [6.7, 3. , 5.2, 2.3],
        [6.3, 2.5, 5. , 1.9],
        [6.5, 3. , 5.2, 2. ],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target_names': array(['setosa', 'versicol

Feature Names in the data
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
x = data.data
y = data.target
```

```
z = data.target_names
print("First row in the data")
print(x[0])
```

```
First row in the data
[5.1 3.5 1.4 0.2]
```

Split the Dataset as per the requirement in the code

```
x_train, x_test,  y_train, y_test = train_test_split(x, y, test_size=0.3)
```

Building the Model

```
model= Sequential()
model.add(Dense(100,input_shape=(4,), activation="relu"))
model.add(Dense(3, activation='softmax'))
```

Compile the Model

```
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
```

Fit the Model

```
model.fit(x_train,y_train, epochs=50)
```

```
Epoch 1/50
4/4 [==============================] - 0s 3ms/step - loss: 1.5305 - accuracy: 0.1048
Epoch 2/50
4/4 [==============================] - 0s 4ms/step - loss: 1.3598 - accuracy: 0.3143
Epoch 3/50
4/4 [==============================] - 0s 4ms/step - loss: 1.2005 - accuracy: 0.3619
Epoch 4/50
```

```
4/4 [==============================] - 0s 4ms/step - loss: 1.0683 - accuracy: 0.3619
Epoch 5/50
4/4 [==============================] - 0s 4ms/step - loss: 0.9602 - accuracy: 0.3619
Epoch 6/50
4/4 [==============================] - 0s 4ms/step - loss: 0.8726 - accuracy: 0.5619
Epoch 7/50
4/4 [==============================] - 0s 5ms/step - loss: 0.8100 - accuracy: 0.7238
Epoch 8/50
4/4 [==============================] - 0s 4ms/step - loss: 0.7688 - accuracy: 0.7238
Epoch 9/50
4/4 [==============================] - 0s 5ms/step - loss: 0.7349 - accuracy: 0.7238
Epoch 10/50
4/4 [==============================] - 0s 4ms/step - loss: 0.7075 - accuracy: 0.7238
Epoch 11/50
4/4 [==============================] - 0s 6ms/step - loss: 0.6824 - accuracy: 0.7238
Epoch 12/50
4/4 [==============================] - 0s 4ms/step - loss: 0.6565 - accuracy: 0.7333
Epoch 13/50
4/4 [==============================] - 0s 4ms/step - loss: 0.6299 - accuracy: 0.7429
Epoch 14/50
4/4 [==============================] - 0s 5ms/step - loss: 0.6093 - accuracy: 0.7238
Epoch 15/50
4/4 [==============================] - 0s 3ms/step - loss: 0.5898 - accuracy: 0.7238
Epoch 16/50
4/4 [==============================] - 0s 4ms/step - loss: 0.5710 - accuracy: 0.7238
Epoch 17/50
4/4 [==============================] - 0s 4ms/step - loss: 0.5552 - accuracy: 0.7429
Epoch 18/50
4/4 [==============================] - 0s 5ms/step - loss: 0.5409 - accuracy: 0.7619
Epoch 19/50
4/4 [==============================] - 0s 5ms/step - loss: 0.5263 - accuracy: 0.7524
Epoch 20/50
4/4 [==============================] - 0s 3ms/step - loss: 0.5157 - accuracy: 0.7238
Epoch 21/50
4/4 [==============================] - 0s 3ms/step - loss: 0.5038 - accuracy: 0.7333
Epoch 22/50
4/4 [==============================] - 0s 3ms/step - loss: 0.4924 - accuracy: 0.7429
Epoch 23/50
4/4 [==============================] - 0s 5ms/step - loss: 0.4830 - accuracy: 0.7905
Epoch 24/50
4/4 [==============================] - 0s 4ms/step - loss: 0.4726 - accuracy: 0.8095
```

```
Epoch 25/50
4/4 [==============================] - 0s 3ms/step - loss: 0.4633 - accuracy: 0.8286
Epoch 26/50
4/4 [==============================] - 0s 4ms/step - loss: 0.4552 - accuracy: 0.8857
Epoch 27/50
4/4 [==============================] - 0s 4ms/step - loss: 0.4474 - accuracy: 0.8857
Epoch 28/50
4/4 [==============================] - 0s 5ms/step - loss: 0.4389 - accuracy: 0.8857
Epoch 29/50
```

Evaluate the Model and then predict for the first 10 Observations

```
model.evaluate(x_test, y_test)
print("\n")
pred=model.predict(x_test[:10])
print(pred)
```

```
2/2 [==============================] - 0s 7ms/step - loss: 0.3923 - accuracy: 0.9556


[[0.00461665 0.31616974 0.6792136 ]
 [0.00619281 0.4368465  0.55696064]
 [0.05222719 0.63185096 0.31592184]
 [0.00100379 0.2362021  0.7627941 ]
 [0.01477385 0.48641855 0.4988076 ]
 [0.0320243  0.5327655  0.43521014]
 [0.93326294 0.06360903 0.00312798]
 [0.8965067  0.09634187 0.00715148]
 [0.00658231 0.34724262 0.6461751 ]
 [0.04095796 0.5449838  0.41405818]]
```

```
p=np.argmax(pred, axis=1)
print(p)
print(y_test[:10])
```

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

Prediction Result

```
for i in p:
  print("Predicted-Class: {},   Name: {}".format(i,z[i]))
```

```
Predicted-Class: 2,   Name: virginica
Predicted-Class: 2,   Name: virginica
Predicted-Class: 1,   Name: versicolor
Predicted-Class: 2,   Name: virginica
Predicted-Class: 2,   Name: virginica
Predicted-Class: 1,   Name: versicolor
Predicted-Class: 0,   Name: setosa
Predicted-Class: 0,   Name: setosa
Predicted-Class: 2,   Name: virginica
Predicted-Class: 1,   Name: versicolor
```