```
import pandas as pd
```

```
df = pd.read_csv('/content/Iris.csv')
```

```
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

```
df = df.sample(frac = 1)
```

```
df.head()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
118	7.7	2.6	6.9	2.3	2
149	5.9	3.0	5.1	1.8	2
106	4.9	2.5	4.5	1.7	2
109	7.2	3.6	6.1	2.5	2
127	6.1	3.0	4.9	1.8	2

```
df.columns
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

```
df.drop('Id',axis=1,inplace=True)
```

```
df
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
118	7.7	2.6	6.9	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica
106	4.9	2.5	4.5	1.7	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
127	6.1	3.0	4.9	1.8	Iris-virginica
...	...	...	...	...	...
37	4.9	3.1	1.5	0.1	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
df['Species'].unique()
```

```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa'], dtype=object)
```

128 6.4 2.8 5.6 2.1 Iris-virginica

```
Species_dict = {
    'Iris-setosa' : 0, 'Iris-versicolor' : 1, 'Iris-virginica':2
}
```

```
df['Species'] = df['Species'].map(Species_dict)
```

```
df.head()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
118	7.7	2.6	6.9	2.3	2
149	5.9	3.0	5.1	1.	2
106	4.9	2.5	4.5	1.7	2
109	7.2	3.6	6.1	2.5	2
127	6.1	3.0	4.9	1.8	2

```
import keras
```

```
from tensorflow.keras.utils import to_categorical
```

```
out_col = to_categorical(df['Species'])
```

```
# out_col
```

```
input_col = df.drop("Species",axis=1)
```

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


model = Sequential()


l1 = Dense(16)
l2 = Dense(32)
l3 = Dense(64)
l4 = Dense(128)
l5 = Dense(64)
l6 = Dense(32)
out_layer = Dense(3,activation='softmax')


model.add(l1)
model.add(l2)
model.add(l3)
model.add(l4)
model.add(l5)
model.add(l6)
model.add(out_layer)
```

```
# model.summary()
```

```
import tensorflow as tf
```

```
model.compile(loss ='categorical_crossentropy',optimizer='Adam',metrics='accuracy')
```

```
input_col.head()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
118	7.7	2.6	6.9	2.3
149	5.9	3.0	5.1	1.8
106	4.9	2.5	4.5	1.7
109	7.2	3.6	6.1	2.5
127	6.1	3.0	4.9	1.8

```
out_col[:5]
```

```
array([[0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.]], dtype=float32)
```

```
model.fit(x=input_col,y=out_col,epochs=300)
```

```
5/5 [==============================] - 0s 7ms/step - loss: 0.0537 - accuracy: 0.9
Epoch 273/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0528 - accuracy: 0.9
Epoch 274/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0558 - accuracy: 0.9
Epoch 275/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0616 - accuracy: 0.9
Epoch 276/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0716 - accuracy: 0.9
Epoch 277/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0566 - accuracy: 0.9
Epoch 278/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0628 - accuracy: 0.9
Epoch 279/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0627 - accuracy: 0.9
Epoch 280/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0567 - accuracy: 0.9
Epoch 281/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0674 - accuracy: 0.9
Epoch 282/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0555 - accuracy: 0.9
Epoch 283/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0666 - accuracy: 0.9
Epoch 284/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0535 - accuracy: 0.9
Epoch 285/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0551 - accuracy: 0.9
Epoch 286/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0578 - accuracy: 0.9
Epoch 287/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0753 - accuracy: 0.9
Epoch 288/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0694 - accuracy: 0.9
Epoch 289/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0494 - accuracy: 0.9
Epoch 290/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0759 - accuracy: 0.9
Epoch 291/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0525 - accuracy: 0.9
Epoch 292/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0571 - accuracy: 0.9
Epoch 293/300
5/5 [==============================] - 0s 6ms/step - loss: 0.0724 - accuracy: 0.9
Epoch 294/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0607 - accuracy: 0.9
Epoch 295/300
5/5 [==============================] - 0s 5ms/step - loss: 0.0566 - accuracy: 0.9
Epoch 296/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0735 - accuracy: 0.9
Epoch 297/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0484 - accuracy: 0.9
Epoch 298/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0691 - accuracy: 0.9
```

```
Epoch 299/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0630 - accuracy: 0.9
Epoch 300/300
5/5 [==============================] - 0s 4ms/step - loss: 0.0559 - accuracy: 0.9
<tensorflow.python.keras.callbacks.History at 0x7f72a654d750>
```

```
model.predict([[1.2,2.4,1.3,4.2]])
```

```
array([[5.6293100e-12, 1.6773407e-12, 1.0000000e+00]], dtype=float32)
```

```
df
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
118	7.7	2.6	6.9	2.3	2
149	5.9	3.0	5.1	1.8	2
106	4.9	2.5	4.5	1.7	2
109	7.2	3.6	6.1	2.5	2
127	6.1	3.0	4.9	1.8	2
...	...	...	...	...	...
37	4.9	3.1	1.5	0.1	0
4	5.0	3.6	1.4	0.2	0
90	5.5	2.6	4.4	1.2	1
1	4.9	3.0	1.4	0.2	0
128	6.4	2.8	5.6	2.1	2

150 rows × 5 columns

```
import numpy as np
np.argmax(model.predict([[4.9,  3.1,  1.5 ,0.1  ]]))
```

```
0
```

```
import pickle
```

```
# pickle.dump(model,open('/content/Irismodel','wb'))
```

```
---------------------------------------------------------------------------
```

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
```

SEARCH STACK OVERFLOW