In [1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
X, y = load_iris(return_X_y=True)
```

In [3]:
```
import pandas as pd
# 儲存模型
metrics = pd.DataFrame({'mean_accuracy':[],'mean_precision':[],'mean_recall':[]}).T
```

In [4]:
```
# 建立LogisticRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RepeatedStratifiedKFold
model_LR = LogisticRegression(multi_class='multinomial', max_iter=300)
                              # max_iter 與後面用 tf.keras建立的 model_tf 的參數 epoch都設定一樣(300)
```

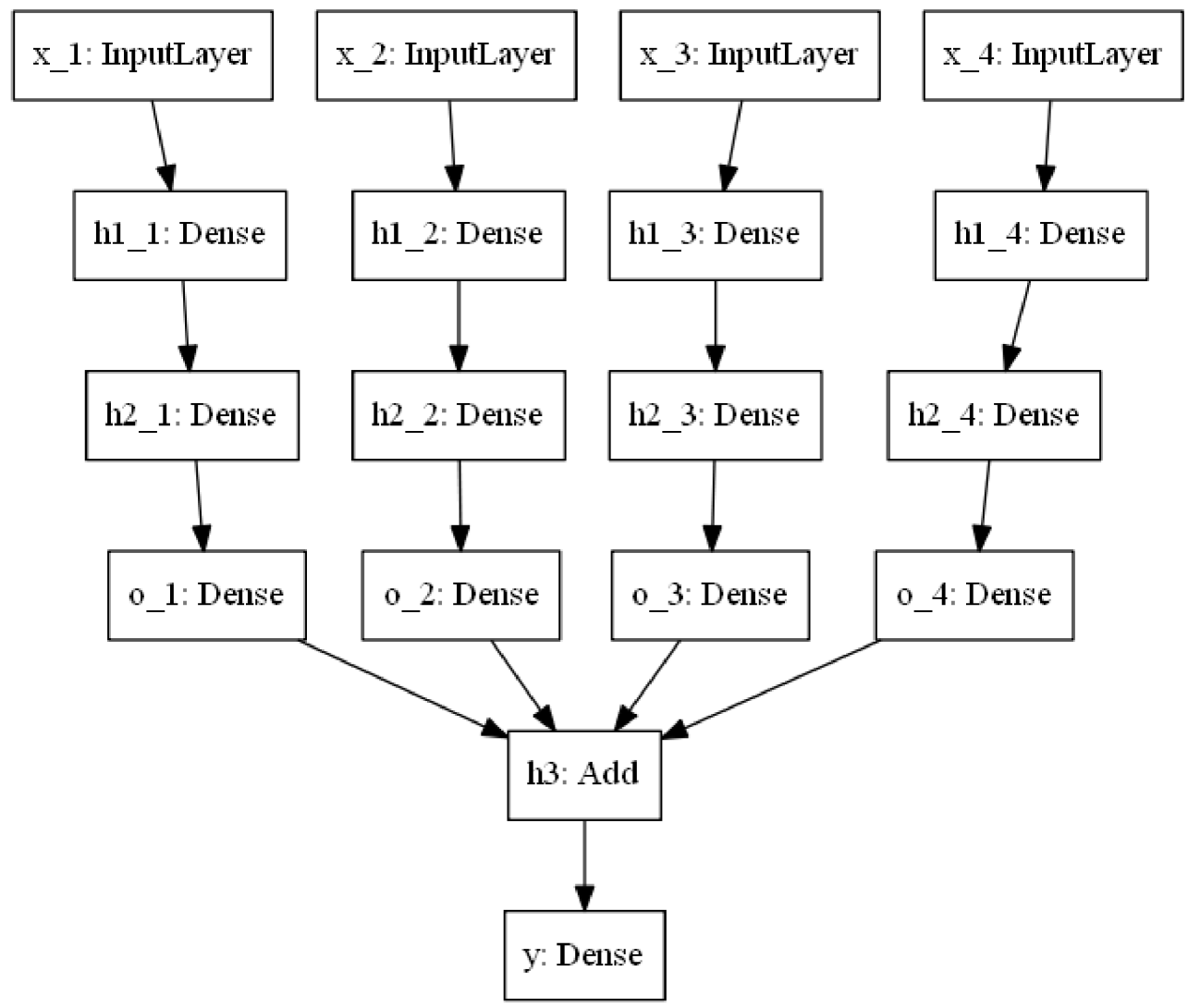
In [5]:
```
from tensorflow import keras
from tensorflow.keras import layers
n, p = X.shape
nc = np.size(np.unique(y))
# 變數 1
x_1 = keras.Input(shape=(1,), name='x_1')
h1_1 = layers.Dense(100,
                    activation='relu',
                    name='h1_1')(x_1)
h2_1 = layers.Dense(20,
                    activation='relu',
                    name='h2_1')(h1_1)
o_1 = layers.Dense(nc, name = 'o_1')(h2_1)
# 變數 2
x_2 = keras.Input(shape=(1,), name='x_2')
h1_2 = layers.Dense(100,
                    activation='relu',
                    name='h1_2')(x_2)
h2_2 = layers.Dense(20,
                    activation='relu',
                    name='h2_2')(h1_2)
o_2 = layers.Dense(nc, name = 'o_2')(h2_2)
# 變數 3
x_3 = keras.Input(shape=(1,), name='x_3')
h1_3 = layers.Dense(100,
                    activation='relu',
                    name='h1_3')(x_3)
h2_3 = layers.Dense(20,
                    activation='relu',
                    name='h2_3')(h1_3)
o_3 = layers.Dense(nc, name = 'o_3')(h2_3)
# 變數 4
x_4 = keras.Input(shape=(1,), name='x_4')
h1_4 = layers.Dense(100,
                    activation='relu',
                    name='h1_4')(x_4)
h2_4 = layers.Dense(20,
                    activation='relu',
                    name='h2_4')(h1_4)
o_4 = layers.Dense(nc, name = 'o_4')(h2_4)
h3 = layers.Add(name='h3')([o_1, o_2, o_3, o_4])
outputs = layers.Dense(nc,
                       activation='softmax',
                       name = 'y')(h3)

model_tf = keras.Model(inputs=[x_1, x_2, x_3, x_4], outputs=outputs)
model_tf.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
x_1 (InputLayer)                [(None, 1)]          0
__________________________________________________________________________________________________
x_2 (InputLayer)                [(None, 1)]          0
__________________________________________________________________________________________________
x_3 (InputLayer)                [(None, 1)]          0
__________________________________________________________________________________________________
x_4 (InputLayer)                [(None, 1)]          0
__________________________________________________________________________________________________
h1_1 (Dense)                    (None, 100)          200         x_1[0][0]
__________________________________________________________________________________________________
h1_2 (Dense)                    (None, 100)          200         x_2[0][0]
__________________________________________________________________________________________________
h1_3 (Dense)                    (None, 100)          200         x_3[0][0]
__________________________________________________________________________________________________
h1_4 (Dense)                    (None, 100)          200         x_4[0][0]
__________________________________________________________________________________________________
h2_1 (Dense)                    (None, 20)           2020        h1_1[0][0]
__________________________________________________________________________________________________
h2_2 (Dense)                    (None, 20)           2020        h1_2[0][0]
__________________________________________________________________________________________________
h2_3 (Dense)                    (None, 20)           2020        h1_3[0][0]
__________________________________________________________________________________________________
h2_4 (Dense)                    (None, 20)           2020        h1_4[0][0]
__________________________________________________________________________________________________
o_1 (Dense)                     (None, 3)            63          h2_1[0][0]
__________________________________________________________________________________________________
o_2 (Dense)                     (None, 3)            63          h2_2[0][0]
__________________________________________________________________________________________________
o_3 (Dense)                     (None, 3)            63          h2_3[0][0]
__________________________________________________________________________________________________
o_4 (Dense)                     (None, 3)            63          h2_4[0][0]
__________________________________________________________________________________________________
h3 (Add)                        (None, 3)            0           o_1[0][0]
                                                                 o_2[0][0]
                                                                 o_3[0][0]
                                                                 o_4[0][0]
```

```
y (Dense)                       (None, 3)            12          h3[0][0]
==================================================================================================
Total params: 9,144
Trainable params: 9,144
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [6]:
```
# 因為畫流程圖套件複雜，此圖僅供理解用
# from tensorflow.keras.utils import plot_model
# plot_model(model_tf)
```

Out[6]:



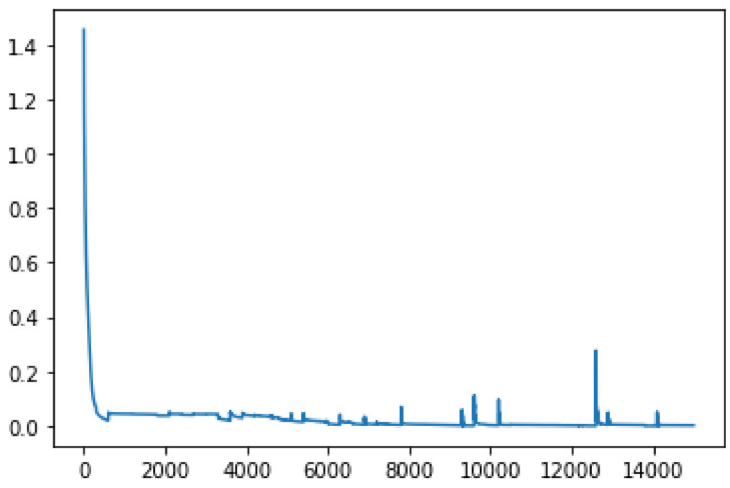
In [7]:
```
model_tf.compile(optimizer='adam',
                 loss=keras.losses.SparseCategoricalCrossentropy())
```

In [8]:
```
# 利用 RepeatedStratifiedKFold 驗證
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import recall_score, accuracy_score, precision_score
```

In [9]:
```
import pandas as pd
# 儲存模型的 metrics
metric_list = ["accuracy", "precision", "recall"]
methods = ["_tf","_LR"]
for metric in metric_list:
    for method in methods:
        locals()[metric+method]=[]
loss = []
# 存結果的位置
results = pd.DataFrame({'mean_accuracy':[],'mean_precision':[],'mean_recall':[]}).T
```

In [10]:
```
# repeat 10-Fold 5 次

rskf = RepeatedStratifiedKFold(n_splits=10, n_repeats = 5)
for train_index, test_index in rskf.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    ### Tensorflow

    ## Fit model
    n = X_train.shape[0]
    history = model_tf.fit({'x_1': X_train[:,0],
                            'x_2': X_train[:,1],
                            'x_3': X_train[:,2],
                            'x_4': X_train[:,3]},{'y': y_train},batch_size=n, epochs=300, verbose=0)
    ##把loss存起來
    loss.extend(history.history['loss'])

    # predict(因為屬於多類別問題，請使用 argmax)
    ypred_tf = np.argmax(model_tf.predict({'x_1': X_test[:,0],
                                           'x_2': X_test[:,1],
                                           'x_3': X_test[:,2],
                                           'x_4': X_test[:,3]}),axis=1)

    accuracy_tf.append(accuracy_score(ypred_tf,y_test))
    precision_tf.append(precision_score(ypred_tf,y_test, average='macro'))
    recall_tf.append(recall_score(ypred_tf,y_test, average='macro'))

    ### LogisticRegression
    ## Fit model
    model_LR.fit(X_train, y_train)

    ## predict
    ypred_LR = model_LR.predict(X_test)
    accuracy_LR.append(accuracy_score(ypred_LR,y_test))
    precision_LR.append(precision_score(ypred_LR,y_test, average='macro'))
    recall_LR.append(recall_score(ypred_LR,y_test, average='macro'))

print('model_tf 的 loss:')
plt.plot(loss)
plt.show()
# 存下來
results['tf.keras']=[np.mean(accuracy_tf),np.mean(precision_tf),np.mean(recall_tf)]
results['LogisticRegression']=[np.mean(accuracy_LR),np.mean(precision_LR),np.mean(recall_LR)]
```

```
model_tf 的 loss:
```



結果

In [11]:
```
results
```

Out[11]:

	tf.keras	LogisticRegression
mean_accuracy	0.980000	0.964000
mean_precision	0.980000	0.964000
mean_recall	0.983206	0.969238

利用 兩種方式fit出來的模型都很不錯，以這三項指標來說，都是以tf.keras建立的模型最好