In [60]:

```python
import numpy as np
import pandas as pd

df = pd.read_csv('Data_for_UCI_named.csv')
df.head()
```

Out[60]:

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g
0	2.959060	3.079885	8.381025	9.780754	3.763085	-0.782604	-1.257395	-1.723086	0.65045
1	9.304097	4.902524	3.047541	1.369357	5.067812	-1.940058	-1.872742	-1.255012	0.41344
2	8.971707	8.848428	3.046479	1.214518	3.405158	-1.207456	-1.277210	-0.920492	0.16304
3	0.716415	7.669600	4.486641	2.340563	3.963791	-1.027473	-1.938944	-0.997374	0.44620
4	3.134112	7.608772	4.943759	9.857573	3.525811	-1.125531	-1.845975	-0.554305	0.79711

In [61]:

```python
df = df.drop('stab', axis = 1)
```

In [62]:

```python
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

stabf_dummies = pd.get_dummies(df.stabf)
df_new = pd.concat([df, stabf_dummies], axis = 1)
df_new.head()
```

Out[62]:

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g
0	2.959060	3.079885	8.381025	9.780754	3.763085	-0.782604	-1.257395	-1.723086	0.65045
1	9.304097	4.902524	3.047541	1.369357	5.067812	-1.940058	-1.872742	-1.255012	0.41344
2	8.971707	8.848428	3.046479	1.214518	3.405158	-1.207456	-1.277210	-0.920492	0.16304
3	0.716415	7.669600	4.486641	2.340563	3.963791	-1.027473	-1.938944	-0.997374	0.44620
4	3.134112	7.608772	4.943759	9.857573	3.525811	-1.125531	-1.845975	-0.554305	0.79711

In [63]:

```python
df_new.drop(['stabf', 'unstable'], axis = 1, inplace = True)
df_new.head()
```

Out[63]:

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g
0	2.959060	3.079885	8.381025	9.780754	3.763085	-0.782604	-1.257395	-1.723086	0.65045
1	9.304097	4.902524	3.047541	1.369357	5.067812	-1.940058	-1.872742	-1.255012	0.41344
2	8.971707	8.848428	3.046479	1.214518	3.405158	-1.207456	-1.277210	-0.920492	0.16304
3	0.716415	7.669600	4.486641	2.340563	3.963791	-1.027473	-1.938944	-0.997374	0.44620
4	3.134112	7.608772	4.943759	9.857573	3.525811	-1.125531	-1.845975	-0.554305	0.79711

In [64]:

```python
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   tau1    10000 non-null  float64
 1   tau2    10000 non-null  float64
 2   tau3    10000 non-null  float64
 3   tau4    10000 non-null  float64
 4   p1      10000 non-null  float64
 5   p2      10000 non-null  float64
 6   p3      10000 non-null  float64
 7   p4      10000 non-null  float64
 8   g1      10000 non-null  float64
 9   g2      10000 non-null  float64
 10  g3      10000 non-null  float64
 11  g4      10000 non-null  float64
 12  stable  10000 non-null  uint8  
dtypes: float64(12), uint8(1)
memory usage: 947.4 KB
```

In [65]:

```
X = df_new.drop('stable', axis = 1)
Y = df_new[['stable']]
X.head()
```

Out[65]:

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g
0	2.959060	3.079885	8.381025	9.780754	3.763085	-0.782604	-1.257395	-1.723086	0.65045
1	9.304097	4.902524	3.047541	1.369357	5.067812	-1.940058	-1.872742	-1.255012	0.41344
2	8.971707	8.848428	3.046479	1.214518	3.405158	-1.207456	-1.277210	-0.920492	0.16304
3	0.716415	7.669600	4.486641	2.340563	3.963791	-1.027473	-1.938944	-0.997374	0.44620
4	3.134112	7.608772	4.943759	9.857573	3.525811	-1.125531	-1.845975	-0.554305	0.79711

In [66]:

```
Y.head()
```

Out[66]:

	stable
0	0
1	1
2	0
3	0
4	0

In [67]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size = 0.75)
```

In [68]:

```
X_train.info()
print('*'*25)
Y_train.info()
print('*'*25)
X_test.info()
print('*'*25)
Y_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7500 entries, 5414 to 7126
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   tau1    7500 non-null   float64
 1   tau2    7500 non-null   float64
 2   tau3    7500 non-null   float64
 3   tau4    7500 non-null   float64
 4   p1      7500 non-null   float64
 5   p2      7500 non-null   float64
 6   p3      7500 non-null   float64
 7   p4      7500 non-null   float64
 8   g1      7500 non-null   float64
 9   g2      7500 non-null   float64
 10  g3      7500 non-null   float64
 11  g4      7500 non-null   float64
dtypes: float64(12)
memory usage: 761.7 KB
*************************
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7500 entries, 5414 to 7126
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   stable  7500 non-null   uint8
dtypes: uint8(1)
memory usage: 65.9 KB
*************************
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2500 entries, 2038 to 8947
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   tau1    2500 non-null   float64
 1   tau2    2500 non-null   float64
 2   tau3    2500 non-null   float64
 3   tau4    2500 non-null   float64
 4   p1      2500 non-null   float64
 5   p2      2500 non-null   float64
 6   p3      2500 non-null   float64
 7   p4      2500 non-null   float64
 8   g1      2500 non-null   float64
 9   g2      2500 non-null   float64
 10  g3      2500 non-null   float64
 11  g4      2500 non-null   float64
dtypes: float64(12)
memory usage: 253.9 KB
*************************
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2500 entries, 2038 to 8947
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   stable  2500 non-null   uint8
dtypes: uint8(1)
memory usage: 22.0 KB
```

In [69]:

```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import callbacks

model = keras.Sequential([
    layers.Dense(units = 420, activation = 'relu', input_shape = [12]),
    layers.Dense(units = 1, activation = 'sigmoid')
])

model.compile(
    optimizer = 'adam',
    loss = 'mae',
    metrics = ['accuracy']
)

early_stopping = callbacks.EarlyStopping(
    min_delta = 0.001,
    patience = 15,
    restore_best_weights = True
)

history = model.fit(
    X_train, Y_train,
    validation_data = (X_test, Y_test),
    batch_size = 50,
    epochs = 200,
    callbacks = [early_stopping]
)
```

```
Epoch 1/200
150/150 [==============================] - 2s 6ms/step - loss: 0.3157 -
accuracy: 0.7077 - val_loss: 0.2630 - val_accuracy: 0.7568
Epoch 2/200
150/150 [==============================] - 1s 7ms/step - loss: 0.2419 -
accuracy: 0.7907 - val_loss: 0.2203 - val_accuracy: 0.8056
Epoch 3/200
150/150 [==============================] - 1s 8ms/step - loss: 0.2066 -
accuracy: 0.8304 - val_loss: 0.1913 - val_accuracy: 0.8444
Epoch 4/200
150/150 [==============================] - 1s 6ms/step - loss: 0.1822 -
accuracy: 0.8536 - val_loss: 0.1713 - val_accuracy: 0.8584
Epoch 5/200
150/150 [==============================] - 1s 8ms/step - loss: 0.1681 -
accuracy: 0.8657 - val_loss: 0.1642 - val_accuracy: 0.8596
Epoch 6/200
150/150 [==============================] - 1s 8ms/step - loss: 0.1592 -
accuracy: 0.8688 - val_loss: 0.1535 - val_accuracy: 0.8688
Epoch 7/200
150/150 [==============================] - 1s 8ms/step - loss: 0.1484 -
```

In [71]:

```
model.evaluate(X_test, Y_test)
```

```
79/79 [==============================] - 1s 5ms/step - loss: 0.0598 - accu
racy: 0.9496
```

Out[71]:

```
[0.059837933629751205, 0.9495999813079834]
```

In [73]:

```
model1 = keras.Sequential([
    layers.Dense(units = 420, activation = 'relu', input_shape = [12]),
    layers.Dense(units = 1, activation = 'sigmoid')
])

model1.compile(
    optimizer = 'sgd',
    loss = 'mse',
    metrics = ['accuracy']
)

history = model1.fit(
    X_train, Y_train,
    validation_data = (X_test, Y_test),
    batch_size = 50,
    epochs = 200,
    callbacks = [early_stopping]
)
```

```
Epoch 1/200
150/150 [==============================] - 2s 8ms/step - loss: 0.2005 -
accuracy: 0.6635 - val_loss: 0.1835 - val_accuracy: 0.7328
Epoch 2/200
150/150 [==============================] - 1s 7ms/step - loss: 0.1811 -
accuracy: 0.7384 - val_loss: 0.1718 - val_accuracy: 0.7428
Epoch 3/200
150/150 [==============================] - 1s 4ms/step - loss: 0.1725 -
accuracy: 0.7588 - val_loss: 0.1664 - val_accuracy: 0.7584
Epoch 4/200
150/150 [==============================] - 1s 4ms/step - loss: 0.1668 -
accuracy: 0.7703 - val_loss: 0.1615 - val_accuracy: 0.7604
Epoch 5/200
150/150 [==============================] - 1s 4ms/step - loss: 0.1628 -
accuracy: 0.7777 - val_loss: 0.1586 - val_accuracy: 0.7640
Epoch 6/200
150/150 [==============================] - 1s 4ms/step - loss: 0.1595 -
accuracy: 0.7816 - val_loss: 0.1554 - val_accuracy: 0.7744
Epoch 7/200
150/150 [==============================] - 1s 4ms/step - loss: 0.1566 -
```

In [74]:

```
model1.evaluate(X_test,Y_test)
```

```
79/79 [==============================] - 1s 4ms/step - loss: 0.0727 - accu
racy: 0.9048
```

Out[74]:

```
[0.07269883155822754, 0.9047999978065491]
```

In [75]:

```
model2 = keras.Sequential([
    layers.Dense(units = 420, activation = 'relu', input_shape = [12]),
    layers.Dense(units = 1, activation = 'sigmoid')
])

model2.compile(
    optimizer = 'adam',
    loss = 'binary_crossentropy',
    metrics = ['accuracy']
)

history = model2.fit(
    X_train, Y_train,
    validation_data = (X_test, Y_test),
    batch_size = 50,
    epochs = 200,
    callbacks = [early_stopping]
)
```

```
Epoch 1/200
150/150 [==============================] - 2s 7ms/step - loss: 0.5030 -
accuracy: 0.7544 - val_loss: 0.4096 - val_accuracy: 0.8248
Epoch 2/200
150/150 [==============================] - 1s 6ms/step - loss: 0.3893 -
accuracy: 0.8271 - val_loss: 0.3678 - val_accuracy: 0.8456
Epoch 3/200
150/150 [==============================] - 1s 4ms/step - loss: 0.3455 -
accuracy: 0.8456 - val_loss: 0.3168 - val_accuracy: 0.8604
Epoch 4/200
150/150 [==============================] - 1s 7ms/step - loss: 0.3088 -
accuracy: 0.8669 - val_loss: 0.3073 - val_accuracy: 0.8636
Epoch 5/200
150/150 [==============================] - 1s 8ms/step - loss: 0.2852 -
accuracy: 0.8775 - val_loss: 0.2735 - val_accuracy: 0.8712
Epoch 6/200
150/150 [==============================] - 1s 7ms/step - loss: 0.2620 -
accuracy: 0.8900 - val_loss: 0.2507 - val_accuracy: 0.8884
Epoch 7/200
150/150 [==============================] - 1s 5ms/step - loss: 0.2477
```

In [76]:

```
model2.evaluate(X_test,Y_test)
```

```
79/79 [==============================] - 0s 2ms/step - loss: 0.0882 - accu
racy: 0.9640
```

Out[76]:

```
[0.08819855004549026, 0.9639999866485596]
```

In [77]:

```
model3 = keras.Sequential([
    layers.Dense(units = 420, activation = 'relu', input_shape = [12]),
    layers.Dense(units = 1, activation = 'sigmoid')
])

model3.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['binary_accuracy'],
)

history = model3.fit(
    X_train, Y_train,
    validation_data = (X_test, Y_test),
    batch_size = 50,
    epochs = 200,
    callbacks = [early_stopping]
)
```

```
Epoch 1/200
150/150 [==============================] - 2s 6ms/step - loss: 0.5186 -
binary_accuracy: 0.7491 - val_loss: 0.4159 - val_binary_accuracy: 0.814
8
Epoch 2/200
150/150 [==============================] - 1s 7ms/step - loss: 0.3918 -
binary_accuracy: 0.8275 - val_loss: 0.3578 - val_binary_accuracy: 0.847
2
Epoch 3/200
150/150 [==============================] - 1s 5ms/step - loss: 0.3455 -
binary_accuracy: 0.8499 - val_loss: 0.3391 - val_binary_accuracy: 0.832
8
Epoch 4/200
150/150 [==============================] - 1s 8ms/step - loss: 0.3134 -
binary_accuracy: 0.8639 - val_loss: 0.3094 - val_binary_accuracy: 0.850
0
Epoch 5/200
150/150 [==============================] - 1s 6ms/step - loss: 0.2853 -
binary_accuracy: 0.8759 - val_loss: 0.2884 - val_binary_accuracy: 0.877
```

In [78]:

```
model3.evaluate(X_test,Y_test)
```

```
79/79 [==============================] - 0s 2ms/step - loss: 0.0998 - bina
ry_accuracy: 0.9608
```

Out[78]:

```
[0.0997513085603714, 0.9607999920845032]
```

In [79]:

```
model4 = keras.Sequential([
    layers.Dense(units = 420, activation = 'relu', input_shape = [12]),
    layers.Dense(units = 420, activation = 'relu'),
    layers.Dense(units = 420, activation = 'relu'),
    layers.Dropout(0.5),
    layers.Dense(units = 1, activation = 'sigmoid')
])

model4.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['binary_accuracy'],
)

history = model4.fit(
    X_train, Y_train,
    validation_data = (X_test, Y_test),
    batch_size = 50,
    epochs = 200,
    callbacks = [early_stopping]
)
```

```
Epoch 1/200
150/150 [==============================] - 4s 13ms/step - loss: 0.4511
- binary_accuracy: 0.7847 - val_loss: 0.3609 - val_binary_accuracy: 0.8
324
Epoch 2/200
150/150 [==============================] - 2s 11ms/step - loss: 0.3326
- binary_accuracy: 0.8484 - val_loss: 0.2998 - val_binary_accuracy: 0.8
680
Epoch 3/200
150/150 [==============================] - 2s 12ms/step - loss: 0.2731
- binary_accuracy: 0.8775 - val_loss: 0.2669 - val_binary_accuracy: 0.8
860
Epoch 4/200
150/150 [==============================] - 2s 11ms/step - loss: 0.2248
- binary_accuracy: 0.8999 - val_loss: 0.2391 - val_binary_accuracy: 0.8
972
Epoch 5/200
150/150 [==============================] - 2s 11ms/step - loss: 0.2006
- binary_accuracy: 0.9139 - val_loss: 0.1604 - val_binary_accuracy: 0.9
202
```

In [80]:

```
model4.evaluate(X_test,Y_test)
```

```
79/79 [==============================] - 1s 4ms/step - loss: 0.0809 - bina
ry_accuracy: 0.9696
```

Out[80]:

```
[0.08094283193349838, 0.9696000218391418]
```

In [4]:

```
!pip install nbconvert[webpdf]
```

```
Collecting nbconvert[webpdf]

  WARNING: Failed to write executable - trying to use .deleteme logic
ERROR: Could not install packages due to an OSError: [WinError 2] The syst
em cannot find the file specified: 'C:\\Python311\\Scripts\\pygmentize.ex
e' -> 'C:\\Python311\\Scripts\\pygmentize.exe.deleteme'


[notice] A new release of pip available: 22.3.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
  Using cached nbconvert-7.4.0-py3-none-any.whl (285 kB)
Collecting beautifulsoup4
  Using cached beautifulsoup4-4.12.2-py3-none-any.whl (142 kB)
Collecting bleach
  Using cached bleach-6.0.0-py3-none-any.whl (162 kB)
Collecting defusedxml
  Using cached defusedxml-0.7.1-py2.py3-none-any.whl (25 kB)
Collecting jinja2>=3.0
  Using cached Jinja2-3.1.2-py3-none-any.whl (133 kB)
Collecting jupyter-core>=4.7
  Using cached jupyter_core-5.3.0-py3-none-any.whl (93 kB)
Collecting jupyterlab-pygments
  Using cached jupyterlab_pygments-0.2.2-py2.py3-none-any.whl (21 kB)
Collecting markupsafe>=2.0
  Using cached MarkupSafe-2.1.2-cp311-cp311-win_amd64.whl (16 kB)
Collecting mistune<3,>=2.0.3
  Using cached mistune-2.0.5-py2.py3-none-any.whl (24 kB)
Collecting nbclient>=0.5.0
  Using cached nbclient-0.8.0-py3-none-any.whl (73 kB)
Collecting nbformat>=5.1
  Using cached nbformat-5.8.0-py3-none-any.whl (77 kB)
Collecting packaging
  Using cached packaging-23.1-py3-none-any.whl (48 kB)
Collecting pandocfilters>=1.4.1
  Using cached pandocfilters-1.5.0-py2.py3-none-any.whl (8.7 kB)
Collecting pygments>=2.4.1
  Using cached Pygments-2.15.1-py3-none-any.whl (1.1 MB)
Collecting tinycss2
  Using cached tinycss2-1.2.1-py3-none-any.whl (21 kB)
Collecting traitlets>=5.0
  Using cached traitlets-5.9.0-py3-none-any.whl (117 kB)
Collecting pyppeteer<1.1,>=1
  Using cached pyppeteer-1.0.2-py3-none-any.whl (83 kB)
Collecting platformdirs>=2.5
  Using cached platformdirs-3.5.1-py3-none-any.whl (15 kB)
Requirement already satisfied: pywin32>=300 in c:\python311\lib\site-packa
ges (from jupyter-core>=4.7->nbconvert[webpdf]) (306)
Collecting jupyter-client>=6.1.12
  Using cached jupyter_client-8.2.0-py3-none-any.whl (103 kB)
Collecting fastjsonschema
  Using cached fastjsonschema-2.17.1-py3-none-any.whl (23 kB)
Collecting jsonschema>=2.6
  Using cached jsonschema-4.17.3-py3-none-any.whl (90 kB)
Collecting appdirs<2.0.0,>=1.4.3
  Using cached appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Collecting certifi>=2021
  Using cached certifi-2023.5.7-py3-none-any.whl (156 kB)
Collecting importlib-metadata>=1.4
```

```
In [ ]:
```