```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
df = pd.read_csv('hcvdat0.csv')
df.head()
```

	Id	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	1	0	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	2	0	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74.0	15.6	76.5
2	3	0	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86.0	33.2	79.3
3	4	0	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7
4	5	0	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7

```
df.shape
```

```
(564, 14)
```

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

```
df.head()
```

	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	0	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	0	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74.0	15.6	76.5
2	0	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86.0	33.2	79.3
3	0	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7
4	0	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7

```
df['Sex'].value_counts()
```

```
m    344
f    220
Name: Sex, dtype: int64
```

```
le = LabelEncoder()
df['Sex'] = le.fit_transform(df['Sex'])
```

```
df['Sex'].value_counts()
```

```
1    344
0    220
```

```
Name: Sex, dtype: int64
```

```
df.isnull().any()
```

```
Category    False
Age         False
Sex         False
ALB         False
ALP          True
ALT          True
AST         False
BIL         False
CHE         False
CHOL         True
CREA        False
GGT         False
PROT        False
dtype: bool
```

```
df.dropna(inplace=True)
```

```
df.isnull().any()
```

```
Category    False
Age         False
Sex         False
ALB         False
ALP         False
ALT         False
AST         False
BIL         False
CHE         False
CHOL        False
CREA        False
GGT         False
PROT        False
dtype: bool
```

```
X = df.drop('Category', axis=1) #X=df.iloc[:, 1:].values
y = df.Category #y=df.iloc[:,0].values
```

```
print(X)
```

```
      Age  Sex   ALB   ALP   ALT    AST   BIL    CHE  CHOL   CREA    GGT  PROT
0      32    1  38.5  52.5   7.7   22.1   7.5   6.93  3.23  106.0   12.1  69.0
1      32    1  38.5  70.3  18.0   24.7   3.9  11.17  4.80   74.0   15.6  76.5
2      32    1  46.9  74.7  36.2   52.6   6.1   8.84  5.20   86.0   33.2  79.3
3      32    1  43.2  52.0  30.6   22.6  18.9   7.33  4.74   80.0   33.8  75.7
4      32    1  39.2  74.1  32.6   24.8   9.6   9.15  4.32   76.0   29.9  68.7
..    ...  ...   ...   ...   ...    ...   ...    ...   ...    ...    ...   ...
559    58    1  43.0  99.1  12.2   63.2  13.0   5.95  6.15  147.3  491.0  65.6
560    33    0  43.0  29.6   3.8   16.7   6.0   6.88  5.72   58.8   11.5  78.2
561    41    0  37.0  31.2   8.2   38.3   7.0   7.08  5.30   60.8   24.7  82.4
562    50    0  40.0  32.7   9.0   46.0  10.0   7.51  4.67   56.6   22.3  70.1
563    61    0  50.0  34.4  27.4  114.4  22.0   9.48  4.62   61.9  169.8  86.0
```

```
     [553 rows x 12 columns]
```

```
sc = StandardScaler()
X = sc.fit_transform(X)
```

```
print(X)
```

```
     [[-1.53376900e+00  7.94524031e-01 -6.43325979e-01 ...  1.72915294e+00
       -5.15811167e-01 -5.72383248e-01]
      [-1.53376900e+00  7.94524031e-01 -6.43325979e-01 ... -3.00247071e-01
       -4.30720437e-01  8.83408450e-01]
      [-1.53376900e+00  7.94524031e-01  8.94538285e-01 ...  4.60777933e-01
       -2.83562494e-03  1.42690402e+00]
      ...
      [-6.17814078e-01 -1.25861517e+00 -9.17944598e-01 ... -1.13737458e+00
       -2.09484540e-01  2.02863125e+00]
      [ 2.98140842e-01 -1.25861517e+00 -3.68707360e-01 ... -1.40373333e+00
       -2.67832469e-01 -3.58867132e-01]
      [ 1.41764130e+00 -1.25861517e+00  1.46208343e+00 ... -1.06761395e+00
        3.31813400e+00  2.72741127e+00]]
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
     (442, 12)
     (111, 12)
     (442,)
     (111,)
```

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

import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt
```

```
model = Sequential()

model.add(Dense(12, activation='relu', input_dim=12)) # Hidden Layer 1
model.add(Dense(15, activation='relu')) # Hidden Layer 2
model.add(Dense(1, activation='sigmoid')) # Output Layer
```

```
model.summary()
```

```
     Model: "sequential_3"
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense_9 (Dense)             (None, 12)                156

 dense_10 (Dense)            (None, 15)                195

 dense_11 (Dense)            (None, 1)                 16

=================================================================
Total params: 367
Trainable params: 367
Non-trainable params: 0
_________________________________________________________________
```

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

```
history = model.fit(X_train, y_train, validation_split=0.2, epochs=30)
np.save('history.npy', history.history)6y
```

```
Epoch 1/30
12/12 [==============================] - 1s 29ms/step - loss: 0.7493 - accuracy:
Epoch 2/30
12/12 [==============================] - 0s 10ms/step - loss: 0.6376 - accuracy:
Epoch 3/30
12/12 [==============================] - 0s 11ms/step - loss: 0.5501 - accuracy:
Epoch 4/30
12/12 [==============================] - 0s 8ms/step - loss: 0.4830 - accuracy: 0
Epoch 5/30
12/12 [==============================] - 0s 8ms/step - loss: 0.4269 - accuracy: 0
Epoch 6/30
12/12 [==============================] - 0s 8ms/step - loss: 0.3808 - accuracy: 0
Epoch 7/30
12/12 [==============================] - 0s 8ms/step - loss: 0.3440 - accuracy: 0
Epoch 8/30
12/12 [==============================] - 0s 7ms/step - loss: 0.3138 - accuracy: 0
Epoch 9/30
12/12 [==============================] - 0s 10ms/step - loss: 0.2892 - accuracy:
Epoch 10/30
12/12 [==============================] - 0s 8ms/step - loss: 0.2679 - accuracy: 0
Epoch 11/30
12/12 [==============================] - 0s 8ms/step - loss: 0.2490 - accuracy: 0
Epoch 12/30
12/12 [==============================] - 0s 8ms/step - loss: 0.2324 - accuracy: 0
Epoch 13/30
12/12 [==============================] - 0s 10ms/step - loss: 0.2170 - accuracy:
Epoch 14/30
12/12 [==============================] - 0s 8ms/step - loss: 0.2033 - accuracy: 0
Epoch 15/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1906 - accuracy: 0
Epoch 16/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1797 - accuracy: 0
Epoch 17/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1689 - accuracy: 0
Epoch 18/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1595 - accuracy: 0
Epoch 19/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1514 - accuracy: 0
```

```
Epoch 20/30
12/12 [==============================] - 0s 6ms/step - loss: 0.1438 - accuracy: 0
Epoch 21/30
12/12 [==============================] - 0s 7ms/step - loss: 0.1371 - accuracy: 0
Epoch 22/30
12/12 [==============================] - 0s 7ms/step - loss: 0.1299 - accuracy: 0
Epoch 23/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1233 - accuracy: 0
Epoch 24/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1171 - accuracy: 0
Epoch 25/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1115 - accuracy: 0
Epoch 26/30
12/12 [==============================] - 0s 7ms/step - loss: 0.1067 - accuracy: 0
Epoch 27/30
12/12 [==============================] - 0s 8ms/step - loss: 0.1016 - accuracy: 0
Epoch 28/30
12/12 [==============================] - 0s 8ms/step - loss: 0.0969 - accuracy: 0
Epoch 29/30
```

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

```
4/4 [==============================] - 0s 3ms/step - loss: 0.0817 - accuracy: 0.9730
[0.08171148598194122, 0.9729729890823364]
```

```
y_predict = model.predict(X_test)
classes = np.argmax(y_predict, axis=1)
print(classes)
```

```
[[1.64979100e-02]
 [2.78153121e-02]
 [1.36108994e-02]
 [6.45458102e-02]
 [2.65879333e-02]
 [8.36057663e-02]
 [5.18247426e-01]
 [1.12000704e-02]
 [7.71045804e-01]
 [8.25917721e-03]
 [6.00565076e-02]
 [2.01325417e-02]
 [1.18838549e-01]
 [1.45123005e-02]
 [2.00536847e-03]
 [1.97711289e-02]
 [1.05224431e-01]
 [8.77333879e-02]
 [3.77695858e-02]
 [2.53579021e-03]
 [5.15030026e-02]
 [2.95187831e-02]
 [7.54981637e-02]
 [6.19610548e-02]
 [1.77247226e-02]
 [5.44127524e-02]
 [2.44077742e-02]
 [5.73926568e-02]
```

```
[5.38637340e-02]
[7.23284483e-03]
[2.15710998e-02]
[3.94894183e-02]
[6.85346231e-06]
[3.43999863e-02]
[1.65128708e-02]
[6.02293015e-03]
[4.20926511e-02]
[3.61141860e-02]
[2.88188457e-04]
[8.85488689e-02]
[2.27972865e-02]
[8.41523111e-02]
[7.08445907e-03]
[1.88476145e-02]
[6.40061796e-02]
[1.76147819e-02]
[1.06801033e-01]
[1.70432925e-02]
[2.36445963e-02]
[2.19001114e-01]
[1.26354396e-02]
[2.43576765e-02]
[6.09049201e-02]
[8.61006975e-03]
[4.70441580e-02]
[1.59436464e-02]
[4.60806489e-03]
```