In [1]:
```
import pandas as pd
import tensorflow as tf
from sklearn.preprocessing import Imputer
import numpy as np
from sklearn.cross_validation import train_test_split
from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(2)
```

```
C:\Users\USER\Anaconda3\envs\keras_env\lib\site-packages\h5py\__init__.py:36: F
utureWarning: Conversion of the second argument of issubdtype from `float` to `
np.floating` is deprecated. In future, it will be treated as `np.float64 == np.
dtype(float).type`.
  from ._conv import register_converters as _register_converters
C:\Users\USER\Anaconda3\envs\keras_env\lib\site-packages\sklearn\cross_validati
on.py:41: DeprecationWarning: This module was deprecated in version 0.18 in fav
or of the model_selection module into which all the refactored classes and func
tions are moved. Also note that the interface of the new CV iterators are diffe
rent from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Reading the dataset

In [2]:
```
hcc = pd.read_csv("hcc-data.csv")
```

In [3]:
```
hcc.head(n=2)
```

Out[3]:

	Gender	Symptoms	Alcohol	Hepatitis B Surface Antigen	Hepatitis B e Antigen	Hepatitis B Core Antibody	Hepatitis C Virus Antibody	Cirrhosis	Endemic Countries	Smo
0	1	0.0	1	0.0	0.0	0.0	0.0	1	0.0	
1	0	NaN	0	0.0	0.0	0.0	1.0	1	NaN	

2 rows × 50 columns

In [4]:
```
hcc.columns
```

Out[4]:
```
Index(['Gender', 'Symptoms', 'Alcohol', 'Hepatitis B Surface Antigen',
       'Hepatitis B e Antigen', 'Hepatitis B Core Antibody',
       'Hepatitis C Virus Antibody', 'Cirrhosis', 'Endemic Countries',
       'Smoking', 'Diabetes', 'Obesity', 'Hemochromatosis',
       'Arterial Hypertension', 'Chronic Renal Insufficiency',
       'Human Immunodeficiency Virus', 'Nonalcoholic Steatohepatitis',
       'Esophageal Varices', 'Splenomegaly', 'Portal Hypertension',
       'Portal Vein Thrombosis', 'Liver Metastasis', 'Radiological Hallmark',
       'Age_at_diagnosis', 'Grams of Alcohol per day',
       'Packs of cigarets per year', 'Performance Status',
       'Encefalopathy degree', 'Ascites degree',
       'International Normalised Ratio', 'Alpha-Fetoprotein', 'Haemoglobin',
       'Mean Corpuscular Volume', 'Leukocytes', 'Platelets', 'Albumin',
       'Total Bilirubin', 'Alanine transaminase', 'Aspartate transaminase',
       'Gamma glutamyl transferase', 'Alkaline phosphatase', 'Total Proteins',
       'Creatinine', 'Number of Nodules', 'Major dimension of nodule',
       'Direct Bilirubin', 'Iron', 'Oxygen Saturation', 'Ferritin', 'Class'],
      dtype='object')
```

In [5]:
```
for x in hcc.columns :
    if hcc['Class'].corr(hcc[x]) > 0.1:
        print(x," = ", hcc['Class'].corr(hcc[x]))
```

```
Endemic Countries  =  0.10940519192990868
Haemoglobin  =  0.2923566550222707
Albumin  =  0.2877985579698942
Iron  =  0.2914055776870694
Class  =  1.0
```

In [6]:
```
for x in hcc.columns :
    if hcc['Class'].corr(hcc[x]) < -0.2:
        print(x," = ", hcc['Class'].corr(hcc[x]))
```

```
Symptoms  =  -0.2970349024975779
Portal Vein Thrombosis  =  -0.21320071635561033
Liver Metastasis  =  -0.24925942952179742
Performance Status  =  -0.37970806342311314
Ascites degree  =  -0.2611445142577042
International Normalised Ratio  =  -0.20234811096417377
Total Bilirubin  =  -0.22396084388025556
Alkaline phosphatase  =  -0.29387264675701524
Direct Bilirubin  =  -0.26490310100015263
Ferritin  =  -0.32148949023643725
```

Handling the values for continuous attributes

In [7]:
```
continuous_x = hcc[['Symptoms','Ferritin','Portal Vein Thrombosis','Liver Metasta
```

In [8]:
```
continuous_x.head()
```

Out[8]:

	Symptoms	Ferritin	Portal Vein Thrombosis	Liver Metastasis	Performance Status	Ascites degree	International Normalised Ratio	Total Bilirubin	phos
0	0.0	NaN	0.0	0.0	0	1.0	1.53	2.1	
1	NaN	NaN	0.0	0.0	0	1.0	NaN	NaN	
2	0.0	16.0	0.0	1.0	2	2.0	0.96	0.4	
3	1.0	NaN	0.0	1.0	0	1.0	0.95	0.4	
4	1.0	22.0	0.0	0.0	0	1.0	0.94	0.7	

In [9]:
```
y_value = hcc.iloc[: , 49]
```

In [10]:
```
type(y_value)
```

Out[10]:
```
pandas.core.series.Series
```

In [11]:
```
imputer = Imputer(missing_values = "NaN" ,strategy = 'mean', axis = 0)
```

In [12]:
```
imputer = imputer.fit(continuous_x)
```

In [13]:
```
continuous_x = imputer.transform(continuous_x)
```

In [14]:
```
print(continuous_x)
```

```
[[  0.         438.99764706   0.         ...  13.7          3.4
   85.59883721]
 [  0.63945578 438.99764706   0.         ...  12.87901235   3.44553459
   85.59883721]
 [  0.          16.           0.         ...   8.9          3.3
   28.        ]
 ...
 [  0.         438.99764706   0.         ...  13.3          4.3
   85.59883721]
 [  0.         438.99764706   1.         ...  15.6          4.8
   85.59883721]
 [  1.         438.99764706   0.         ...  12.7          2.2
   85.59883721]]
```

In [15]:
```
continuous_x = pd.DataFrame(data = continuous_x)
```

In [16]:
```
continuous_x = continuous_x.apply(lambda x: (x - x.min()) / (x.max() - x.min()))
```

In [17]:
```
continuous_x.head()
```

Out[17]:

	0	1	2	3	4	5	6	7	8	9	10	
0	0.000000	0.196860	0.0	0.0	0.0	0.0	0.173367	0.044776	0.151954	0.013699	0.000000	0.6350
1	0.639456	0.196860	0.0	0.0	0.0	0.0	0.146194	0.069352	0.215518	0.062671	0.079365	0.5751
2	0.000000	0.007175	0.0	1.0	0.5	0.5	0.030151	0.002488	0.110062	0.000000	0.000000	0.2846
3	1.000000	0.196860	0.0	1.0	0.0	0.0	0.027638	0.002488	0.176475	0.003425	0.000000	0.6131
4	1.000000	0.009865	0.0	0.0	0.0	0.0	0.025126	0.009950	0.110062	0.062671	0.000000	0.6788

In [18]:
```
#X_train, X_test, y_train, y_test = train_test_split(continuous_x,y_value,test_si
```

In [19]:
```
#X_train.columns = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','
#continuous_x.columns = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n',
#X_test.columns = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p
```

In [20]:
```
continuous_x.head()
```

Out[20]:

	0	1	2	3	4	5	6	7	8	9	10	
0	0.000000	0.196860	0.0	0.0	0.0	0.0	0.173367	0.044776	0.151954	0.013699	0.000000	0.6350
1	0.639456	0.196860	0.0	0.0	0.0	0.0	0.146194	0.069352	0.215518	0.062671	0.079365	0.5751
2	0.000000	0.007175	0.0	1.0	0.5	0.5	0.030151	0.002488	0.110062	0.000000	0.000000	0.2846
3	1.000000	0.196860	0.0	1.0	0.0	0.0	0.027638	0.002488	0.176475	0.003425	0.000000	0.6131
4	1.000000	0.009865	0.0	0.0	0.0	0.0	0.025126	0.009950	0.110062	0.062671	0.000000	0.6788

Handling the values for nominal values

In [21]:
```
nom_x = hcc.iloc[:,0:23]
```

In [22]:
```
nom_x.head()
```

Out[22]:

	Gender	Symptoms	Alcohol	Hepatitis B Surface Antigen	Hepatitis B e Antigen	Hepatitis B Core Antibody	Hepatitis C Virus Antibody	Cirrhosis	Endemic Countries	Smo
0	1	0.0	1	0.0	0.0	0.0	0.0	1	0.0	
1	0	NaN	0	0.0	0.0	0.0	1.0	1	NaN	
2	1	0.0	1	1.0	0.0	1.0	0.0	1	0.0	
3	1	1.0	1	0.0	0.0	0.0	0.0	1	0.0	
4	1	1.0	1	1.0	0.0	1.0	0.0	1	0.0	

5 rows × 23 columns

In [23]:
```
imputer = Imputer(missing_values = "NaN" ,strategy = 'most_frequent', axis = 0)
```

In [24]:
```
imputer = imputer.fit(nom_x)
```

In [25]:
```
nom_x = imputer.transform(nom_x)
```

In [26]:
```
nom_x = pd.DataFrame(data = nom_x)
```

In [27]:
```
nom_x.columns = ['aa','ab','ac','ad','ae','af','ag','ah','ai','aj','ak','al','am'
```

In [28]:
```
nom_x.head()
```

Out[28]:

	aa	ab	ac	ad	ae	af	ag	ah	ai	aj	...	an	ao	ap	aq	ar	as	at	au	av	aw
0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
1	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	...	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
2	1.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	...	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0
3	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
4	1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	...	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

5 rows × 23 columns

In [29]:
```
frames = [nom_x,continuous_x]
```

In [30]:
```
#x_data = pd.concat(frames, axis=1)
x_data = continuous_x
```

In [31]:
```
x_data.head()
```

Out[31]:

	0	1	2	3	4	5	6	7	8	9	10	
0	0.000000	0.196860	0.0	0.0	0.0	0.0	0.173367	0.044776	0.151954	0.013699	0.000000	0.6350
1	0.639456	0.196860	0.0	0.0	0.0	0.0	0.146194	0.069352	0.215518	0.062671	0.079365	0.5751
2	0.000000	0.007175	0.0	1.0	0.5	0.5	0.030151	0.002488	0.110062	0.000000	0.000000	0.2846
3	1.000000	0.196860	0.0	1.0	0.0	0.0	0.027638	0.002488	0.176475	0.003425	0.000000	0.6131
4	1.000000	0.009865	0.0	0.0	0.0	0.0	0.025126	0.009950	0.110062	0.062671	0.000000	0.6788

In [32]:
```
x_data.sample(frac = 1).head()
```

Out[32]:

	0	1	2	3	4	5	6	7	8	9	10	
44	0.000000	0.196860	0.0	0.0	0.00	0.0	0.218593	0.047264	0.097801	0.013699	0.079365	0.7
47	1.000000	0.398206	1.0	1.0	0.25	0.5	0.201005	0.161692	0.606629	0.136986	0.000000	0.6
158	0.639456	0.196860	0.0	0.0	0.75	0.0	0.080402	0.000000	0.267410	0.062671	0.079365	0.2
66	0.639456	0.176233	0.0	0.0	0.00	0.0	0.125628	0.027363	0.114149	0.006849	0.000000	0.7
145	1.000000	0.196860	0.0	0.0	0.00	0.0	0.040201	0.011443	0.142758	0.007877	0.000000	0.3

In [33]:
```
X_train, X_test, y_train, y_test = train_test_split(x_data,y_value,test_size=0.3,
```

In [34]:
```
#X_train.columns = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','
#x_data.columns = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p
#X_test.columns = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p
```

In [35]:
```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import confusion_matrix
```

```
Using TensorFlow backend.
```

In [36]:
```
classifier = Sequential()
accuracy = 0
```

In [37]:
```
for x in range(10):
    classifier.add(Dense(output_dim = 1024, init = 'uniform', activation = 'relu'
    classifier.add(Dense(output_dim = 512, init = 'uniform', activation = 'relu')
    classifier.add(Dense(output_dim = 256, init = 'uniform', activation = 'relu')
    classifier.add(Dense(output_dim = 128, init = 'uniform', activation = 'relu')
    classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'
    classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics
    classifier.fit(X_train, y_train, batch_size = 20, nb_epoch = 100)

    '''
    y_pred = classifier.predict(X_test)
    y_pred = (y_pred > 0.5)
    cm = confusion_matrix(y_test, y_pred)
    accuracy += ((cm[0][0]+cm[1][1])/(cm[1][0]+cm[1][1]+cm[0][0]+cm[0][1]))*100
    '''
```

```
y:3: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activat
ion="relu", units=512, kernel_initializer="uniform")`
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
C:\Users\USER\Anaconda3\envs\keras_env\lib\site-packages\ipykernel_launcher.p
y:4: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activat
ion="relu", units=256, kernel_initializer="uniform")`
  after removing the cwd from sys.path.
C:\Users\USER\Anaconda3\envs\keras_env\lib\site-packages\ipykernel_launcher.p
y:5: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activat
ion="relu", units=128, kernel_initializer="uniform")`
  """
C:\Users\USER\Anaconda3\envs\keras_env\lib\site-packages\ipykernel_launcher.p
y:6: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activat
ion="sigmoid", units=1, kernel_initializer="uniform")`

C:\Users\USER\Anaconda3\envs\keras_env\lib\site-packages\keras\models.py:944:
UserWarning: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  warnings.warn('The `nb_epoch` argument in `fit` '
```

In [ ]:

In [ ]:

Creating the confusion matrix

In [ ]:

In [38]:
```
#accuracy/10
```

In [39]:
```
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
cm = confusion_matrix(y_test, y_pred)
accuracy = ((cm[0][0]+cm[1][1])/(cm[1][0]+cm[1][1]+cm[0][0]+cm[0][1]))*100
```

In [40]:
```
accuracy
```

Out[40]:
```
88.0
```

In [ ]:

In [ ]: