In [1]:

```
import pandas as pd
import numpy as np
pd.set_option('display.max_rows',200)
pd.set_option('display.max_columns',200)
df=pd.read_csv('hcvdat0.csv').dropna()
ab=df.loc[:,['ALB', 'ALP','ALT', 'AST','BIL', 'CHE', 'CHOL', 'CREA',
             'GGT', 'PROT']].apply(lambda x: (x-np.mean(x))/np.std(x))
df[['ALB', 'ALP','ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT']]=ab
df
```

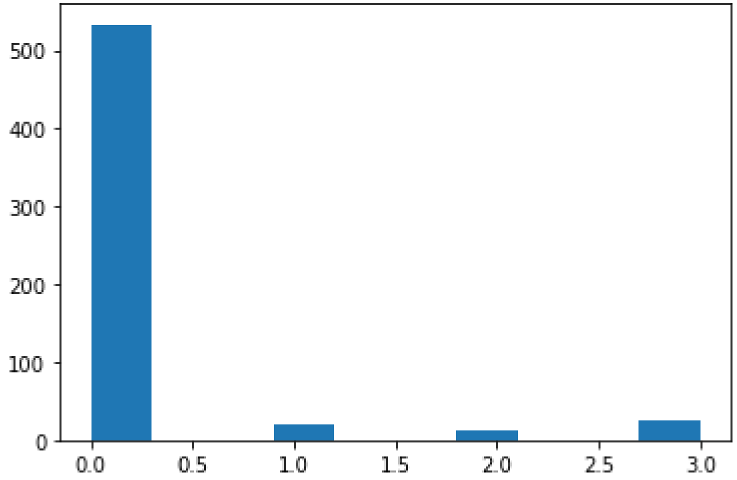
Out[1]:

	Unnamed: 0	Category_num	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL
0	1	0	0=Blood Donor	32	m	-0.542701	-0.603230	-0.905494	-0.355457	-0.202289	-0.581777	-1.916091
1	2	0	0=Blood Donor	32	m	-0.542701	0.084054	-0.411380	-0.276283	-0.409283	1.354993	-0.524241
2	3	0	0=Blood Donor	32	m	0.916417	0.253944	0.461714	0.573318	-0.282787	0.290683	-0.169629
3	4	0	0=Blood Donor	32	m	0.273710	-0.622536	0.193070	-0.340231	0.453193	-0.399063	-0.577433
4	5	0	0=Blood Donor	32	m	-0.421108	0.230777	0.289014	-0.273238	-0.081542	0.432286	-0.949775
...	...	...	...	...	...	...	...	...	...	...	...	...
608	609	3	3=Cirrhosis	58	f	-1.324372	-0.838760	-0.555297	3.539307	-0.173540	-0.887823	-1.251194
609	610	3	3=Cirrhosis	59	f	-0.455849	-0.649564	-0.334624	7.674639	1.666409	-1.111648	-0.781334
610	611	3	3=Cirrhosis	62	f	-1.671781	13.455196	-0.991844	2.330377	2.241393	-1.203005	0.805552
611	612	3	3=Cirrhosis	64	f	-3.061418	1.338926	-1.135760	0.323615	0.516441	-3.043850	-2.102261
612	613	3	3=Cirrhosis	64	f	-2.192895	0.740448	-1.106977	1.986274	2.126396	-2.989036	-1.561479

589 rows × 15 columns

```
In [30]: import matplotlib.pyplot as plt
         plt.hist(df.loc[:,'Category_num'])
```

```
Out[30]: (array([533.,   0.,   0.,  20.,   0.,   0.,  12.,   0.,   0.,  24.]),
          array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3. ]),
          <BarContainer object of 10 artists>)
```



In [29]:

```
from scipy.stats import chi2_contingency
cont_table=pd.crosstab(df.Category,df.Sex)
stat,p,dof,expected=chi2_contingency(cont_table)
print('The expected frequency table:',expected)
print('The p-value for chi2 test:',p)
if p>0.05:
    print('''The p value is more than 0.05 therefore we accept the null hypothesis''')
    print('No relation betweeen variables')
elif p<=0.05:
    print('''The p value is less than 0.05 therefore we reject the null hypothesis''')
    print('Variables are related')
```

```
The expected frequency table: [[201.82682513 324.17317487]
 [  2.68590832   4.31409168]
 [  7.67402377  12.32597623]
 [  4.60441426   7.39558574]
 [  9.20882852  14.79117148]]
The p-value for chi2 test: 0.2512997391345069
The p value is more than 0.05
    therefore we accept the null hypothesis
No relation betweeen variables
```

In [4]:

```
df.columns
```

Out[4]:

```
Index(['Unnamed: 0', 'Category_num', 'Category', 'Age', 'Sex', 'ALB', 'ALP',
       'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT'],
      dtype='object')
```

In [5]:

```
from sklearn.model_selection import train_test_split
Y=df.Category_num
X=df.loc[:,['ALB', 'ALP','ALT', 'AST','BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT']]
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2)
Y.unique()
```

Out[5]:

```
array([0, 1, 2, 3], dtype=int64)
```

In [10]:

```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression(multi_class='multinomial',random_state=0)
```

In [17]:

```
#doing feature selection using recursive feature elimantion RFE
from sklearn.feature_selection import RFE
rfe=RFE(logreg, n_features_to_select=6)
rfe.fit(x_train,y_train)
bo=rfe.support_
x_train2=x_train.loc[:,bo]
x_test2=x_test.loc[:,bo]
```

In [19]:

```
#fitting the logistic regrssion model
logregf=logreg.fit(x_train2,y_train)
#fitting the predicted y values based on testing set of x
y_pred=logregf.predict(x_test2)
```

In [23]:

```
#printing the coeffiecients of the model
logregf.coef_
```

Out[23]:

```
array([[ 1.33615388,  2.00953005, -1.63228573, -0.79185824, -0.00327701,
        -1.10374112],
       [-1.1723826 , -0.07398408,  0.480232  ,  0.4120778 ,  0.90836252,
         0.81490406],
       [-0.95735938, -0.26004419,  0.53478011,  0.14819144,  0.33499215,
         0.26915206],
       [ 0.7935881 , -1.67550179,  0.61727362,  0.23158899, -1.24007765,
         0.01968501]])
```

In [24]:

```
#printing intercepts of the model
logregf.intercept_
```

Out[24]:

```
array([ 4.55175685, -1.41029173, -1.33684463, -1.80462049])
```

In [22]:

```
#calculating accuracy of fitted model based on difference
#between actual y values and predicted y values from our model
print('The accuracy of our fitted model is:',logregf.score(x_test2,y_test))
```

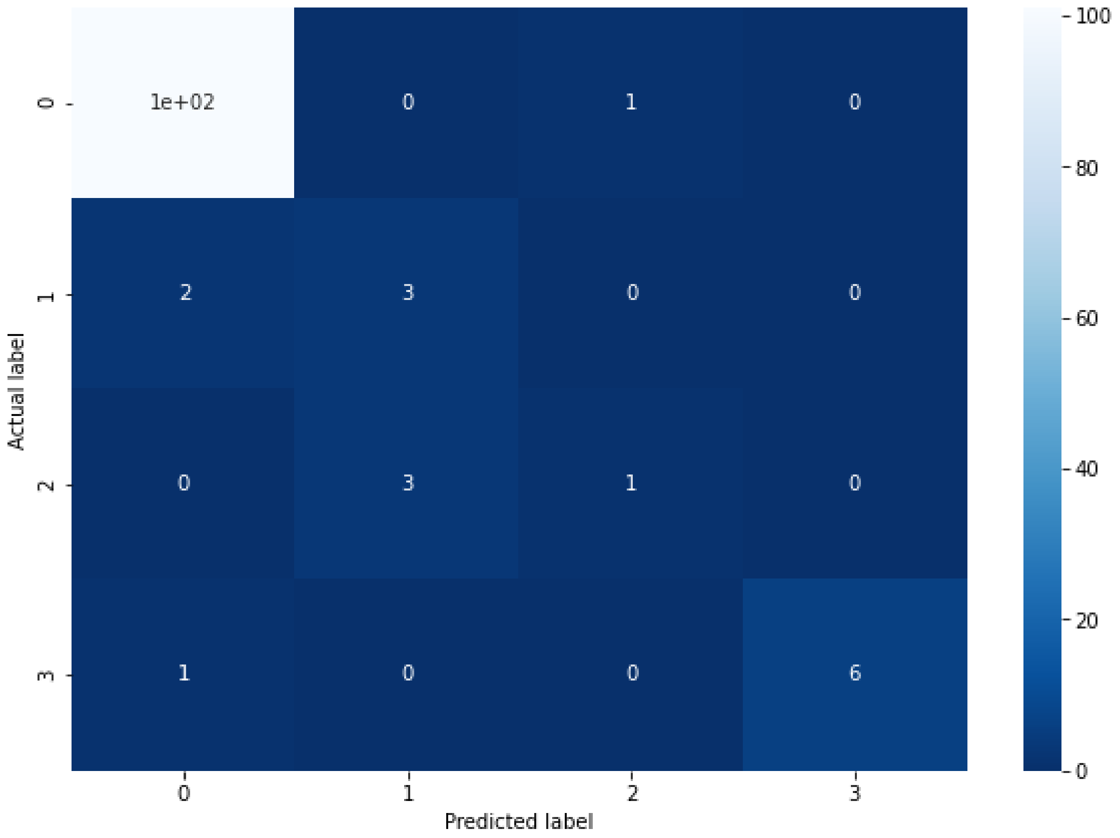
```
The accuracy of our fitted model is: 0.940677966101695
```

In [40]:

```
#printing confusion matrix to visually see accuracy of model
from sklearn import metrics
import seaborn as sns
cm=metrics.confusion_matrix(y_test,y_pred)
plt.figure(figsize=(10,7))
sns.heatmap(cm,annot=True,cmap='Blues_r')
plt.ylabel('Actual label');
plt.xlabel('Predicted label')
```

Out[40]:

```
Text(0.5, 42.0, 'Predicted label')
```



In [ ]: