CCPS 844 Data Mining (Lab 8) Submit your solution as a pdf file

- Q-1 Select a dataset/datasets of your choice. Apply/Fit MLP Classification Evaluate the results
- Q-2 Select a multi-label dataset/datasets of your choice. Apply/Fit MLP Classification Call the predict function to get a multi label value for your test data

In [2]:
```
% matplotlib inline

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split

pd.set_option('display.max_columns', 100)
```

In [4]:
```
from sklearn.neural_network import MLPClassifier
```

In [5]:
```
df = pd.read_csv('winequality-red.csv')
df['overfive'] = (df.quality > 5).astype(int)

df['alcohol_overten'] = (df.alcohol > 10).astype(int)
```

In [6]:
```
df.sample(5)
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alc
1512	6.4	0.790	0.04	2.2	0.061	11.0	17.0	0.99588	3.53	0.65	10.4
105	8.1	0.575	0.22	2.1	0.077	12.0	65.0	0.99670	3.29	0.51	9.2
169	7.5	0.705	0.24	1.8	0.360	15.0	63.0	0.99640	3.00	1.59	9.5
1447	6.8	0.670	0.00	1.9	0.080	22.0	39.0	0.99701	3.40	0.74	9.7
1490	7.1	0.220	0.49	1.8	0.039	8.0	18.0	0.99344	3.39	0.56	12.4

In [21]:
```
scX = StandardScaler()
scy = StandardScaler()

clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(10,), random_sta
te=1)

X=df.iloc[:,0:10].values.tolist()
y=list(map(list,zip(df.alcohol_overten, df.overfive)))
#y=list(map(list,zip(df.alcohol.values, df.quality.values)))
```

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_s
tate = 0)

X_train = scX.fit_transform(X_train)
X_test = scX.transform(X_test)
```

In [28]:
```
clf.fit(X_train,y_train)
```

Out[28]:
```
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(10,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)
```

In [29]:
```
y_pred = clf.predict(X_test)
```

In [30]:
```
pd.Series(list(map(lambda x: all(x),(y_test == y_pred)))).value_counts()
```

Out[30]:
```
True     265
False    135
dtype: int64
```