Submitted by Navneet Das 3433 Comp A

***'
Assignment 2-B Deep Learning
Classification using Deep neural network
Multiclass classification using Deep Neural Networks: Example: Use the handwritten OCR letter (capital & small case letters and special characters) dataset
https://archive.ics.uci.edu/ml/datasets/Letter+recognition
'***

In [1]:
```
!pip install yellowbrick
```

```
Requirement already satisfied: yellowbrick in c:\users\navne\anaconda3\lib\site-packages (1.5)
Requirement already satisfied: numpy>=1.16.0 in c:\users\navne\anaconda3\lib\site-packages (from yellow
brick) (1.21.0)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\users\navne\anaconda3\lib\site-packages
(from yellowbrick) (3.3.4)
Requirement already satisfied: scipy>=1.0.0 in c:\users\navne\anaconda3\lib\site-packages (from yellowb
rick) (1.7.1)
Requirement already satisfied: cycler>=0.10.0 in c:\users\navne\anaconda3\lib\site-packages (from yello
wbrick) (0.10.0)
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\navne\anaconda3\lib\site-packages (from
yellowbrick) (1.2.2)
Requirement already satisfied: six in c:\users\navne\anaconda3\lib\site-packages (from cycler>=0.10.0->
yellowbrick) (1.16.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\navne\anaconda3\lib
\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.0.4)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\navne\anaconda3\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\navne\anaconda3\lib\site-packages (from ma
tplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.3.1)
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import model_selection
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

from sklearn import preprocessing
from yellowbrick.classifier import ConfusionMatrix
```

In [3]:
```
df = pd.read_csv("DeepLearningData/letter-recognition.csv", sep = ",")
```

In [4]:
```
df=df.iloc[:,1:]
```

In [5]:
```
df.head()
```

Out[5]:

	T	2	8	3	5	1	8.1	13	0	6	6.1	10	8.2	0.1	8.3	0.2	8.4
0	I	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10
1	D	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9
2	N	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8
3	G	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10
4	S	4	11	5	8	3	8	8	6	9	5	6	6	0	8	9	7

In [6]:
```
names = ['Class',
         'x-box',
         'y-box',
         'width',
         'high',
         'onpix',
         'x-bar',
         'y-bar',
         'x2bar',
         'y2bar',
         'xybar',
         'x2ybr',
         'xy2br',
         'x-ege',
         'xegvy',
         'y-ege',
         'yegvx']
```

In [7]:
```
X = df.iloc[:, 1 : 17]
Y = df.select_dtypes(include = [object])
```

In [8]:
```
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size = 0.20, random_state =
```

In [9]:
```
scaler = StandardScaler()
scaler.fit(X_train)
```

Out[9]:

▾ StandardScaler

StandardScaler()

In [10]:
```
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [11]:
```
mlp = MLPClassifier(hidden_layer_sizes = (250, 300), max_iter = 1000000, activation = 'logistic')
```

In [12]:
```
cm = ConfusionMatrix(mlp, classes="A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z".split(','))
```

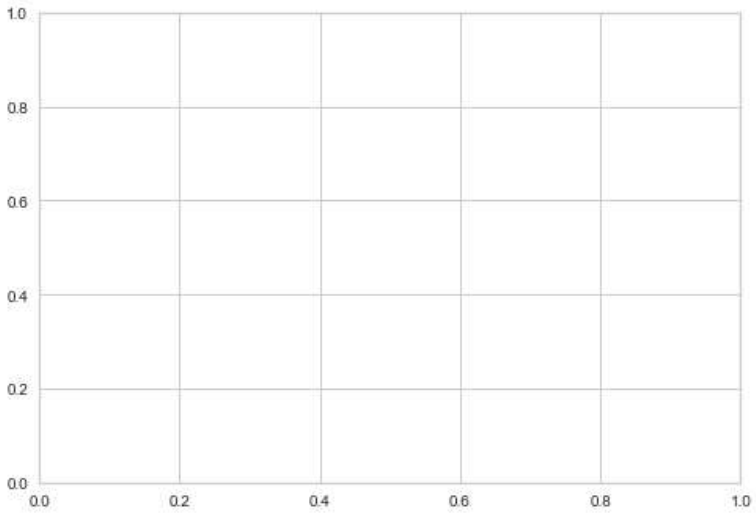
In [13]:
```
cm.fit(X_train, Y_train.values.ravel())
```

Out[13]:

▸ **ConfusionMatrix**

▸ **estimator: MLPClassifier**

▸ MLPClassifier



In [14]:
```
cm.score(X_test, Y_test)
```

```
C:\Users\navne\anaconda3\lib\site-packages\sklearn\preprocessing\_label.py:116: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for
example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[14]:
```
0.89
```

In [15]:
```
predictions = cm.predict(X_test)
```

In [16]:
```
print("Accuracy: ", accuracy_score(Y_test, predictions))
```

```
Accuracy:  0.89
```

In [17]:
```
print(confusion_matrix(Y_test, predictions))
```

```
[[12  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
   0  0]
 [ 0 16  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0]
 [ 0  0 16  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
   0  0]
 [ 0  1  0 25  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0
   0  1]
 [ 0  0  0  0 13  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1
   0  2]
 [ 0  0  0  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
   0  0]
 [ 0  0  0  0  0  1 15  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
   0  0]
 [ 0  0  0  0  0  0  0 13  0  0  1  0  0  0  0  0  1  1  0  0  1  0  0  0
   0  0]
 [ 0  0  0  1  0  0  0  0 11  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
   1  0]
 [ 0  0  0  0  0  0  0  0  0 13  0  0  0  0  0  0  0  0  0  0  0  0  0  1
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  8  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 18  0  0  0  0  0  0  0  0  0  0  0  0
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0
   0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0  0  0  0  0  0  0  0  0
   0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  1 17  1  0  1  0  0  0  0  0  0
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 12  0  0  0  0  0  0  0  0
   0  0]
 [ 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 14  0  0  0  0  0  1  1
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0  0  0  0  0  0
   0  0]
 [ 0  0  0  0  0  2  0  0  0  0  0  1  0  0  0  0  0  0 13  0  0  0  0  0
   1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0  0  0
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  8  0  0  0
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0 12  0  0
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  1  2  0  0  0  0  0  0  0 12  0
   0  0]
 [ 0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9
   0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  11  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0
   0 16]]
```

In [18]:
```
print(classification_report(Y_test, predictions, digits=5))
```

```
              precision    recall  f1-score   support

           A    0.92308   0.92308   0.92308        13
           B    0.84211   0.94118   0.88889        17
           C    1.00000   0.88889   0.94118        18
           D    0.96154   0.86207   0.90909        29
           E    1.00000   0.72222   0.83871        18
           F    0.83333   0.93750   0.88235        16
           G    0.88235   0.88235   0.88235        17
           H    0.81250   0.76471   0.78788        17
           I    1.00000   0.78571   0.88000        14
           J    0.92857   0.92857   0.92857        14
           K    0.80000   1.00000   0.88889         8
           L    0.94737   1.00000   0.97297        18
           M    0.90909   1.00000   0.95238        20
           N    0.83333   0.93750   0.88235        16
           O    0.85000   0.80952   0.82927        21
           P    0.92308   1.00000   0.96000        12
           Q    0.93333   0.82353   0.87500        17
           R    0.77778   1.00000   0.87500         7
           S    0.81250   0.76471   0.78788        17
           T    0.93750   1.00000   0.96774        15
           U    0.80000   1.00000   0.88889         8
           V    1.00000   0.92308   0.96000        13
           W    0.92308   0.80000   0.85714        15
           X    0.75000   0.81818   0.78261        11
           Y    0.84615   1.00000   0.91667        11
           Z    0.84211   0.88889   0.86486        18

    accuracy                        0.89000       400
   macro avg    0.88726   0.90006   0.88938       400
weighted avg    0.89597   0.89000   0.88922       400
```

In [19]:
```
cm.poof()
```

```
<Figure size 576x396 with 0 Axes>
```

Out[19]:
```
<AxesSubplot:title={'center':'MLPClassifier Confusion Matrix'}, xlabel='Predicted Class', ylabel='True Cla
ss'>
```