In [1]:

```
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 [2]:

```
dataset = pd.read_csv("letter-recognition.csv", sep = ",")
```

In [3]:

```
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 [4]:

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

In [5]:

```
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size = 0.20, random_state = 10
```

In [6]:

```
scaler = StandardScaler()
scaler.fit(X_train)
```

Out[6]:

```
StandardScaler()
```

In [7]:

```
X_train = scaler.transform(X_train)
X_validation = scaler.transform(X_validation)
```

In [8]:

```
mlp = MLPClassifier(hidden_layer_sizes = (250, 300), max_iter = 1000000, activation = 'logistic')
```

In [9]:

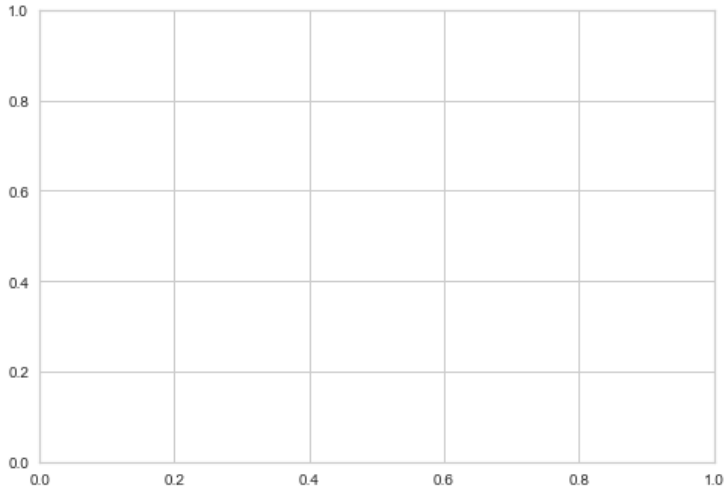
```
cm = ConfusionMatrix(mlp, classes="A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z".split(','))
```

In [10]:

```
cm.fit(X_train, Y_train.values.ravel())
```

Out[10]:

```
ConfusionMatrix(ax=<matplotlib.axes._subplots.AxesSubplot object at 0x0000019F614D4B20>,
                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'],
                cmap=<matplotlib.colors.ListedColormap object at 0x0000019F4DEC03D0>,
                estimator=MLPClassifier(activation='logistic',
                                        hidden_layer_sizes=(250, 300),
                                        max_iter=1000000))
```



In [11]:

```
cm.score(X_validation, Y_validation)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  return f(**kwargs)
```

Out[11]:

```
0.967
```

In [12]:

```
predictions = cm.predict(X_validation)
```

In [13]:

```
print("Accuracy: ", accuracy_score(Y_validation, predictions))
```

```
Accuracy:  0.967
```

In [14]:

```
print(confusion_matrix(Y_validation, predictions))
```

```
[[145   0   0   0   0   0   0   0   0   0   2   1   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0 137   0   0   0   1   0   1   0   0   0   0   0   0   0   0   0   1
    0   0   0   3   0   0   0   0]
 [  0   0 151   0   0   0   1   0   0   0   0   0   0   0   2   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0 157   0   1   0   1   0   0   1   0   0   0   0   0   0   0
    1   0   0   0   0   0   0   0]
 [  0   2   0   0 160   0   3   0   0   0   0   1   1   0   0   1   0   0
    0   0   0   0   0   0   0   2]
 [  0   0   0   0   0 140   0   0   0   0   0   0   0   0   0   0   0   0
    1   0   0   0   0   0   0   0]
 [  0   2   0   0   2   0 160   0   0   0   0   0   1   0   2   0   0   0
    0   0   0   1   0   0   0   0]
 [  0   0   0   3   0   0   2 142   0   0   2   0   0   3   0   0   1   1
    0   0   0   1   0   0   0   0]
 [  0   0   1   0   0   0   0   0 155   6   0   0   0   0   0   0   0   0
    2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   1   0   0   1 145   0   0   0   1   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   1   0   0 129   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   1   0   0   0   0 141   0   1   0   0   0   0
    0   0   0   0   0   1   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 146   0   1   0   0   0
    0   0   0   0   1   0   0   0]
 [  0   0   0   1   0   0   0   1   0   0   0   0   0 142   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   2 140   0   0   0
    0   1   0   0   2   0   0   0]
 [  0   0   0   0   0   6   0   0   0   0   0   0   1   0   0   0 153   3   0
    0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0 159   0
    0   0   0   0   0   0   0   0]
 [  0   2   1   0   0   0   0   2   0   0   3   1   0   1   0   0   1 155
    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   0   0
  135   0   0   0   0   0   0   0]
 [  0   0   0   1   0   1   0   0   0   1   0   1   0   0   0   0   0   0
    0 161   0   0   0   0   0   0]
 [  1   0   1   0   0   0   0   2   0   1   0   0   1   1   0   0   0   0
    0   0 175   0   0   0   0   0]
 [  0   3   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   2
    0   0   0 146   0   0   1   0]
 [  0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   1   0   0
    0   0   1   0 132   0   0   0]
 [  0   0   1   0   0   0   0   0   0   0   3   0   0   0   0   0   1   0
    0   0   0   0   0 153   1   0]
 [  1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    1   2   1   2   0   0 157   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   0 152]]
```

In [15]:

```
print(classification_report(Y_validation, predictions, digits=5))
```

```
              precision    recall  f1-score   support

           A    0.98639   0.97973   0.98305       148
           B    0.93836   0.95804   0.94810       143
           C    0.97419   0.98052   0.97735       154
           D    0.96319   0.97516   0.96914       161
           E    0.98160   0.94118   0.96096       170
           F    0.93333   0.99291   0.96220       141
           G    0.95238   0.95238   0.95238       168
           H    0.94667   0.91613   0.93115       155
           I    0.98726   0.94512   0.96573       164
           J    0.94771   0.97973   0.96346       148
           K    0.91489   0.99231   0.95203       130
           L    0.96575   0.97917   0.97241       144
           M    0.97987   0.98649   0.98316       148
           N    0.94040   0.98611   0.96271       144
           O    0.95890   0.96552   0.96220       145
           P    0.98710   0.93865   0.96226       163
           Q    0.96364   0.99375   0.97846       160
           R    0.97484   0.93373   0.95385       166
           S    0.96429   0.99265   0.97826       136
           T    0.98171   0.97576   0.97872       165
           U    0.98870   0.96154   0.97493       182
           V    0.95425   0.95425   0.95425       153
           W    0.97778   0.97778   0.97778       135
           X    0.99351   0.96226   0.97764       159
           Y    0.98742   0.95152   0.96914       165
           Z    0.98701   0.99346   0.99023       153

    accuracy                        0.96700      4000
   macro avg    0.96658   0.96792   0.96698      4000
weighted avg    0.96750   0.96700   0.96699      4000
```

In [16]:

```
cm.poof()
```

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

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19f614d4b20>
```

In [ ]:

```

```