day40-linear-svm

November 7, 2023

Day40 Linear SVM by:Loga Aswin

[21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
from sklearn.preprocessing import StandardScaler
```

[ ]:
```
df = pd.read_csv("/content/letter-recognition.csv")
```

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

[ ]:

	letter	xbox	ybox	width	height	onpix	xbar	ybar	x2bar	y2bar	\
0	T	2	8	3	5	1	8	13	0	6	
1	I	5	12	3	7	2	10	5	5	4	
2	D	4	11	6	8	6	10	6	2	6	
3	N	7	11	6	6	3	5	9	4	6	
4	G	2	1	3	1	1	8	6	6	6	

	xybar	x2ybar	xy2bar	xedge	xedgey	yedge	yedgex
0	6	10	8	0	8	0	8
1	13	3	9	2	8	4	10
2	10	3	7	3	7	3	9
3	4	4	10	6	10	2	8
4	6	5	9	1	7	5	10

[ ]:
```
df.describe()
```

[ ]:

	xbox	ybox	width	height	onpix	\
count	20000.000000	20000.000000	20000.000000	20000.00000	20000.000000	
mean	4.023550	7.035500	5.121850	5.37245	3.505850	
std	1.913212	3.304555	2.014573	2.26139	2.190458	
min	0.000000	0.000000	0.000000	0.00000	0.000000	
25%	3.000000	5.000000	4.000000	4.00000	2.000000	

```
50%        4.000000      7.000000      5.000000      6.00000      3.000000
75%        5.000000      9.000000      6.000000      7.00000      5.000000
max       15.000000     15.000000     15.000000     15.00000     15.000000

              xbar          ybar          x2bar         y2bar         xybar   \
count  20000.000000  20000.000000  20000.000000  20000.000000  20000.000000
mean       6.897600      7.500450      4.628600      5.178650      8.282050
std        2.026035      2.325354      2.699968      2.380823      2.488475
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        6.000000      6.000000      3.000000      4.000000      7.000000
50%        7.000000      7.000000      4.000000      5.000000      8.000000
75%        8.000000      9.000000      6.000000      7.000000     10.000000
max       15.000000     15.000000     15.000000     15.000000     15.000000

             x2ybar        xy2bar        xedge         xedgey        yedge   \
count  20000.00000  20000.000000  20000.000000  20000.000000  20000.000000
mean       6.45400      7.929000      3.046100      8.338850      3.691750
std        2.63107      2.080619      2.332541      1.546722      2.567073
min        0.00000      0.000000      0.000000      0.000000      0.000000
25%        5.00000      7.000000      1.000000      8.000000      2.000000
50%        6.00000      8.000000      3.000000      8.000000      3.000000
75%        8.00000      9.000000      4.000000      9.000000      5.000000
max       15.00000     15.000000     15.000000     15.000000     15.000000

             yedgex
count  20000.00000
mean       7.80120
std        1.61747
min        0.00000
25%        7.00000
50%        8.00000
75%        9.00000
max       15.00000
```

[30]:
```
df.shape
```

[30]:
```
(20000, 17)
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['letter', 'xbox ', 'ybox ', 'width ', 'height', 'onpix ', 'xbar ',
       'ybar ', 'x2bar', 'y2bar ', 'xybar ', 'x2ybar', 'xy2bar', 'xedge ',
       'xedgey', 'yedge ', 'yedgex'],
      dtype='object')
```

[ ]:
```
# let's 'reindex' the column names
df.columns = ['letter', 'xbox', 'ybox', 'width', 'height', 'onpix', 'xbar',
```

```
       'ybar', 'x2bar', 'y2bar', 'xybar', 'x2ybar', 'xy2bar', 'xedge',
       'xedgey', 'yedge', 'yedgex']
df.columns
```

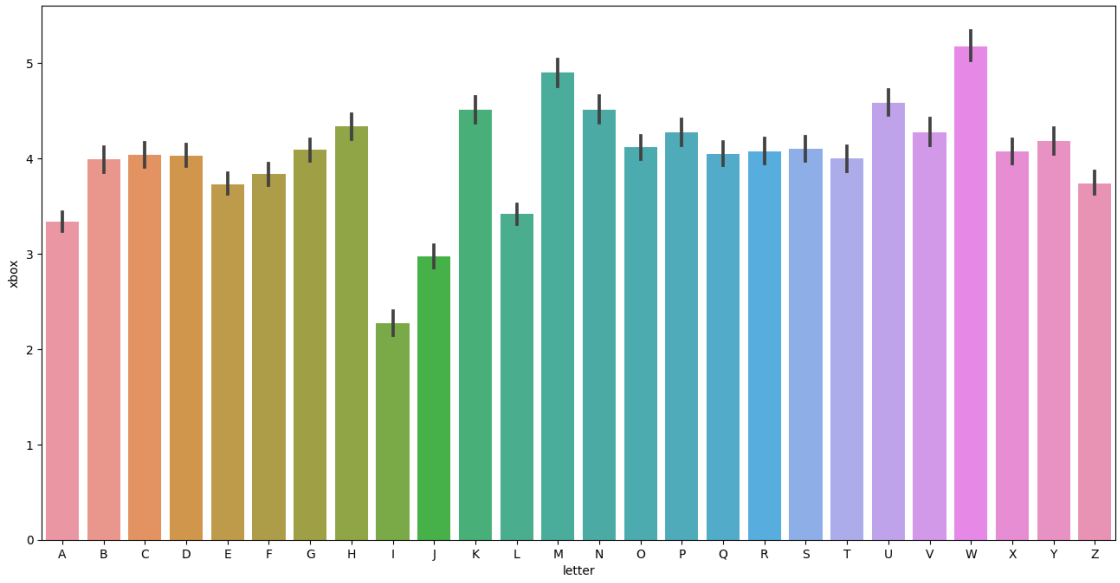
[ ]:
```
Index(['letter', 'xbox', 'ybox', 'width', 'height', 'onpix', 'xbar', 'ybar',
       'x2bar', 'y2bar', 'xybar', 'x2ybar', 'xy2bar', 'xedge', 'xedgey',
       'yedge', 'yedgex'],
      dtype='object')
```

[ ]:
```
order = list(np.sort(df['letter'].unique()))
print(order)
```

```
['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']
```

[ ]:
```
plt.figure(figsize=(16, 8))
sns.barplot(x='letter', y='xbox', data=df, order=order)
```

[ ]:
```
<Axes: xlabel='letter', ylabel='xbox'>
```



[14]:
```
df1 = df.groupby('letter').mean()
df1.head()
```

[14]:

letter	xbox	ybox	width	height	onpix	xbar	ybar
A	3.337136	6.975919	5.128010	5.178707	2.991128	8.851711	3.631179
B	3.985640	6.962141	5.088773	5.169713	4.596606	7.671018	7.062663
C	4.031250	7.063859	4.701087	5.296196	2.775815	5.437500	7.627717

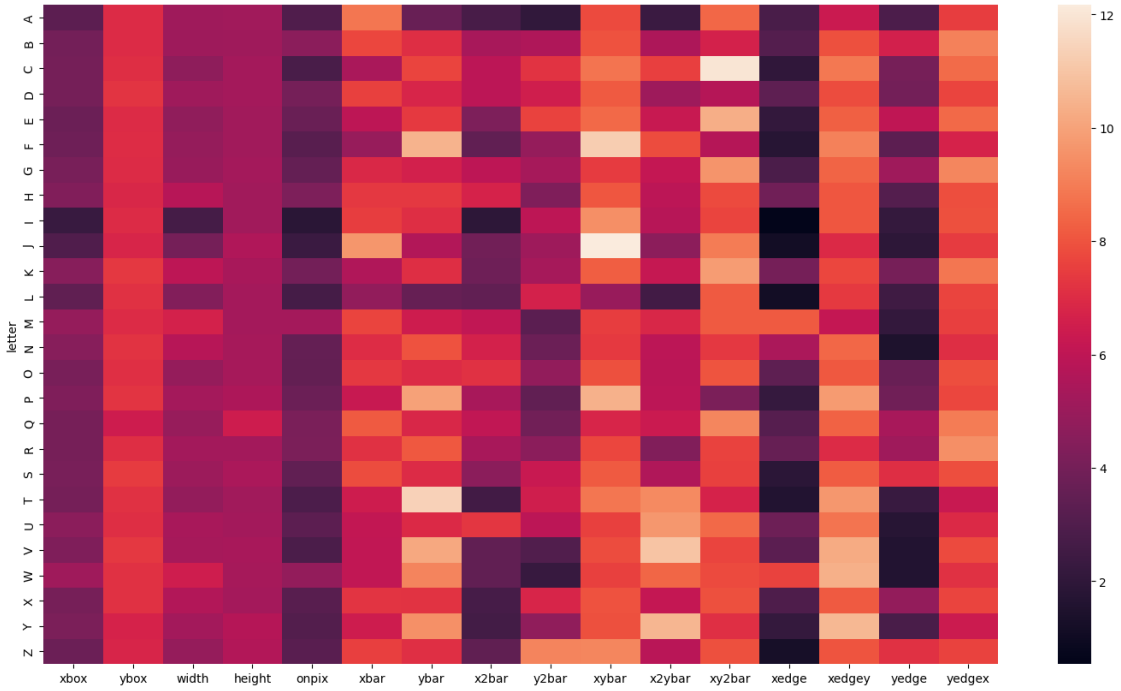
```
D       4.023602  7.244720  5.170186  5.288199  4.026087  7.539130  6.806211
E       3.727865  6.944010  4.756510  5.201823  3.679688  5.966146  7.352865

            x2bar     y2bar     xybar    x2ybar     xy2bar     xedge    xedgey  \
letter
A       2.755387  2.043093  7.802281  2.338403   8.465146  2.771863  6.321926
B       5.366841  5.571802  7.954308  5.506527   6.652742  3.117493  7.919060
C       5.927989  7.177989  8.773098  7.494565  11.947011  1.991848  8.876359
D       5.921739  6.508075  8.166460  5.111801   5.750311  3.365217  7.813665
E       4.223958  7.585938  8.507812  6.242188  10.341146  2.127604  8.298177

            yedge    yedgex
letter
A       2.875792  7.468948
B       6.612272  9.100522
C       4.080163  8.555707
D       3.971429  7.628571
E       6.022135  8.506510
```

[16]:
```
plt.figure(figsize=(18, 10))
sns.heatmap(df1)
```

[16]: <Axes: ylabel='letter'>



Data Preparation

[18]:
```
# average feature values
round(df.drop('letter', axis=1).mean(), 2)
```

[18]:
```
xbox      4.02
ybox      7.04
width     5.12
height    5.37
onpix     3.51
xbar      6.90
ybar      7.50
x2bar     4.63
y2bar     5.18
xybar     8.28
x2ybar    6.45
xy2bar    7.93
xedge     3.05
xedgey    8.34
yedge     3.69
yedgex    7.80
dtype: float64
```

[19]:
```
# splitting into X and y
X = df.drop("letter", axis = 1)
y = df['letter']
```

[22]:
```
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Spliting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
 ↪random_state=42)
```

[24]:
```
# Create and train the Linear SVM model
svm_model = SVC(kernel='linear', C=1)
svm_model.fit(X_train, y_train)

# predict
y_pred = svm_model.predict(X_test)
```

[25]:
```
# Calculate accuracy and display results
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

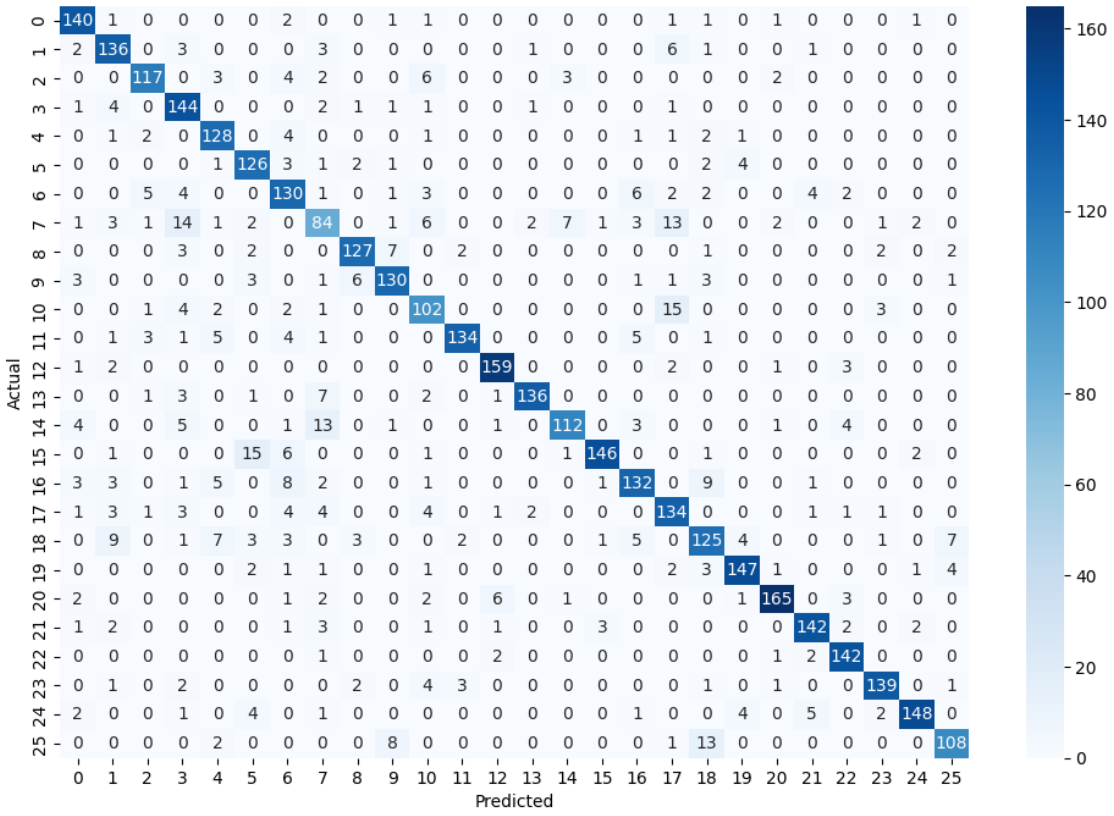
```
Accuracy: 0.86
```

[26]:
```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
A	0.87	0.94	0.90	149
B	0.81	0.89	0.85	153
C	0.89	0.85	0.87	137
D	0.76	0.92	0.83	156
E	0.83	0.91	0.87	141
F	0.80	0.90	0.85	140
G	0.75	0.81	0.78	160
H	0.65	0.58	0.61	144
I	0.90	0.87	0.89	146
J	0.86	0.87	0.87	149
K	0.75	0.78	0.77	130
L	0.95	0.86	0.91	155
M	0.93	0.95	0.94	168
N	0.96	0.90	0.93	151
O	0.90	0.77	0.83	145
P	0.96	0.84	0.90	173
Q	0.84	0.80	0.82	166
R	0.75	0.84	0.79	160
S	0.76	0.73	0.74	171
T	0.91	0.90	0.91	163
U	0.94	0.90	0.92	183
V	0.91	0.90	0.90	158
W	0.90	0.96	0.93	148
X	0.93	0.90	0.92	154
Y	0.95	0.88	0.91	168
Z	0.88	0.82	0.85	132
accuracy			0.86	4000
macro avg	0.86	0.86	0.86	4000
weighted avg	0.86	0.86	0.86	4000

[32]:
```
# Create a confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(12, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```


Actual
Predicted
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
160 140 120 100 80 60 40 20 0