Letter Recognition using Support Vector Classifier

Import libraires

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import plot_confusion_matrix,
classification_report
```

Reading data

In [2]:
```
df = pd.read_csv("letter-recognition.csv")
print(df.shape)
df
```

```
(20000, 17)
```

Out[2]:

	letter	xbox	ybox	width	height	onpix	xbar	ybar	x2bar	y2bar	xybar	x2ybar	xy2bar	x
0	T	2	8	3	5	1	8	13	0	6	6	10	8	
1	I	5	12	3	7	2	10	5	5	4	13	3	9	
2	D	4	11	6	8	6	10	6	2	6	10	3	7	
3	N	7	11	6	6	3	5	9	4	6	4	4	10	
4	G	2	1	3	1	1	8	6	6	6	6	5	9	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19995	D	2	2	3	3	2	7	7	7	6	6	6	4	
19996	C	7	10	8	8	4	4	8	6	9	12	9	13	
19997	T	6	9	6	7	5	6	11	3	7	11	9	5	
19998	S	2	3	4	2	1	8	7	2	6	10	6	8	

	letter	xbox	ybox	width	height	onpix	xbar	ybar	x2bar	y2bar	xybar	x2ybar	xy2bar	x
19999	A	4	9	6	6	2	9	5	3	1	8	1	8	

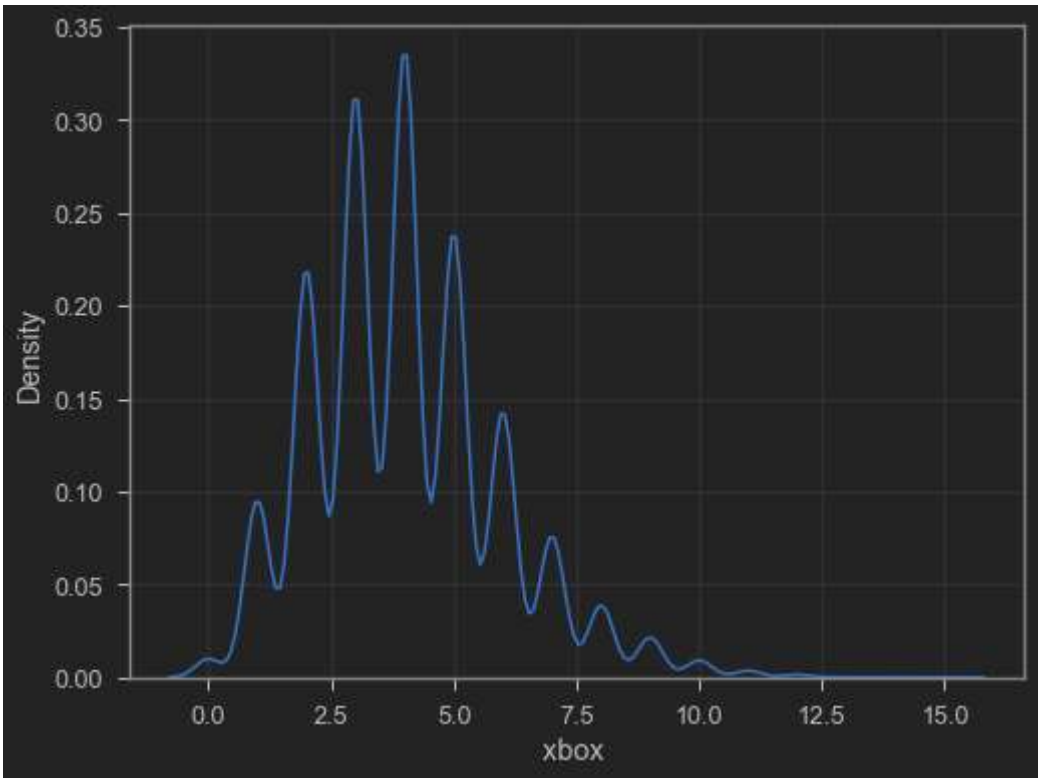
20000 rows × 17 columns

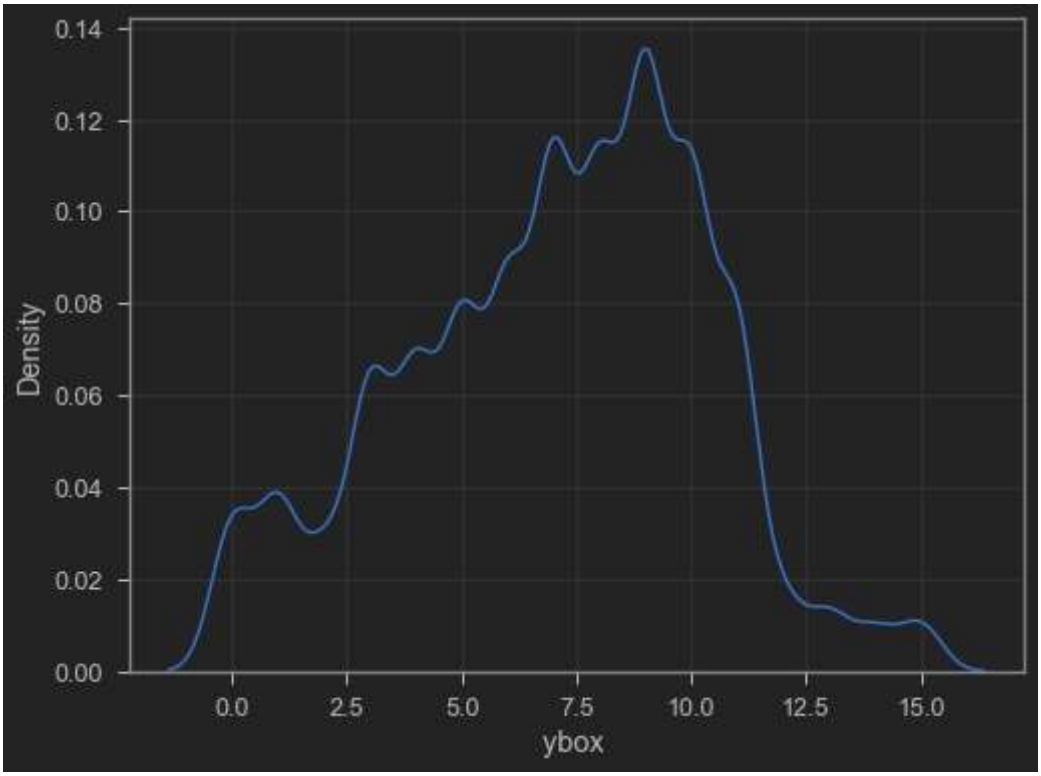
Data Analysis

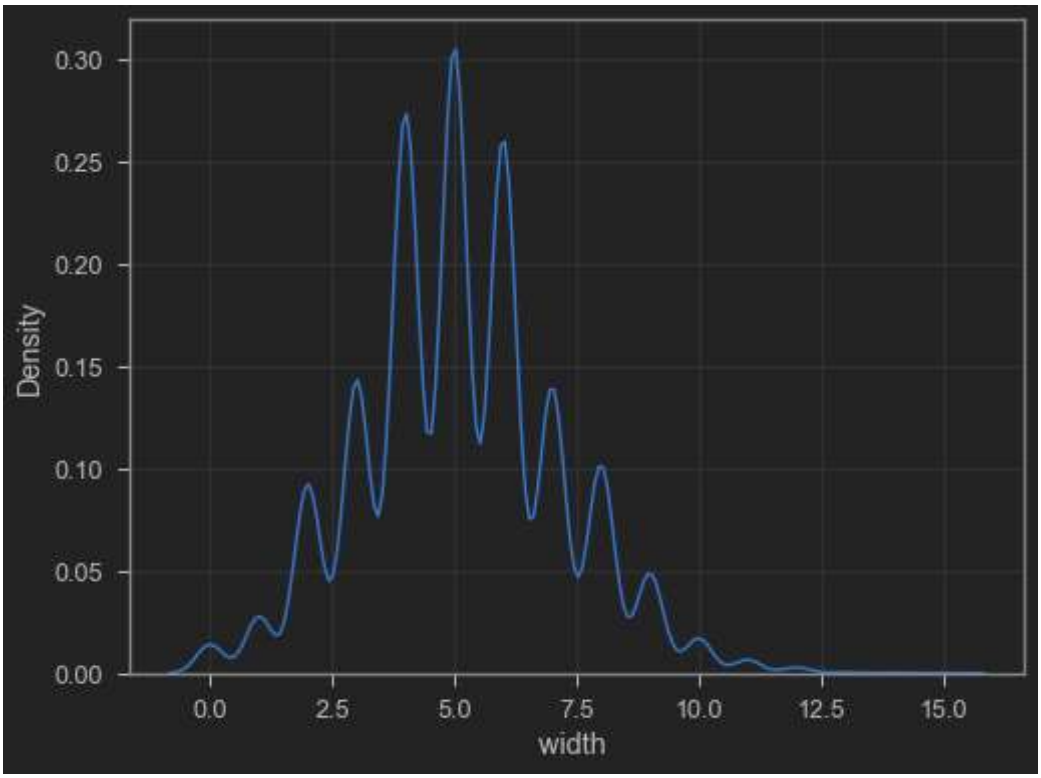
In [3]:
```
features = df.drop('letter', axis=1).columns.to_list()
```

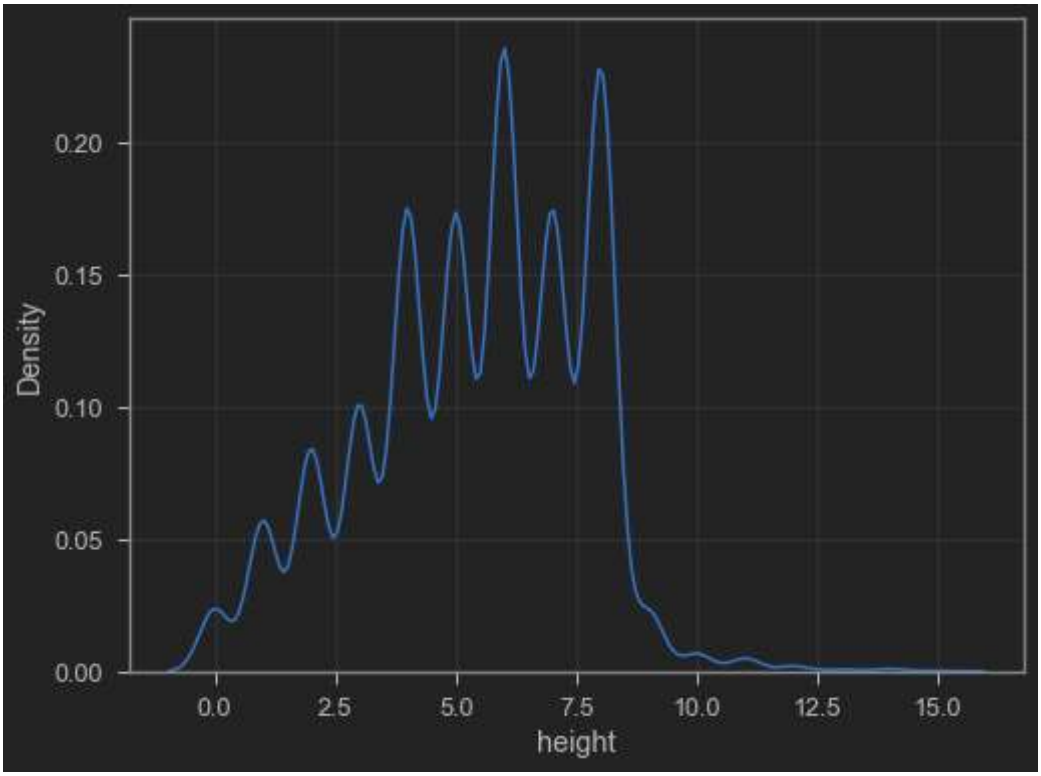
Distribution of data

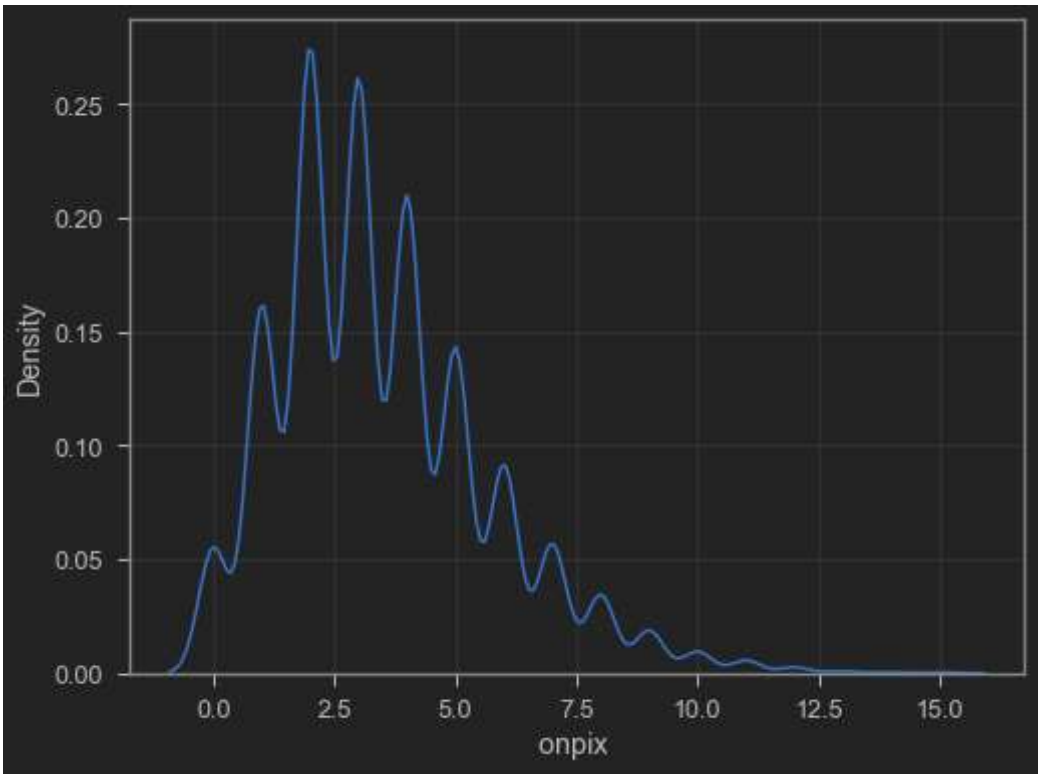
In [4]:
```
for feature in features:
    plt.figure(figsize=(8, 6))
    sns.kdeplot(x=feature, data=df)
    plt.show()
```

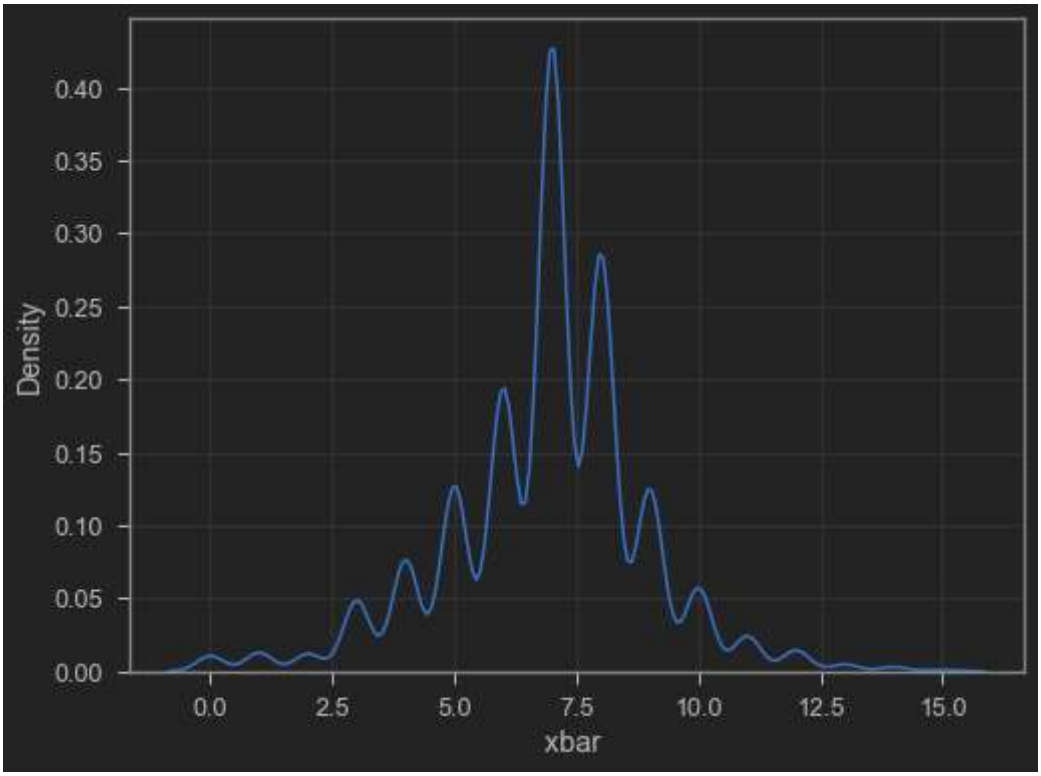


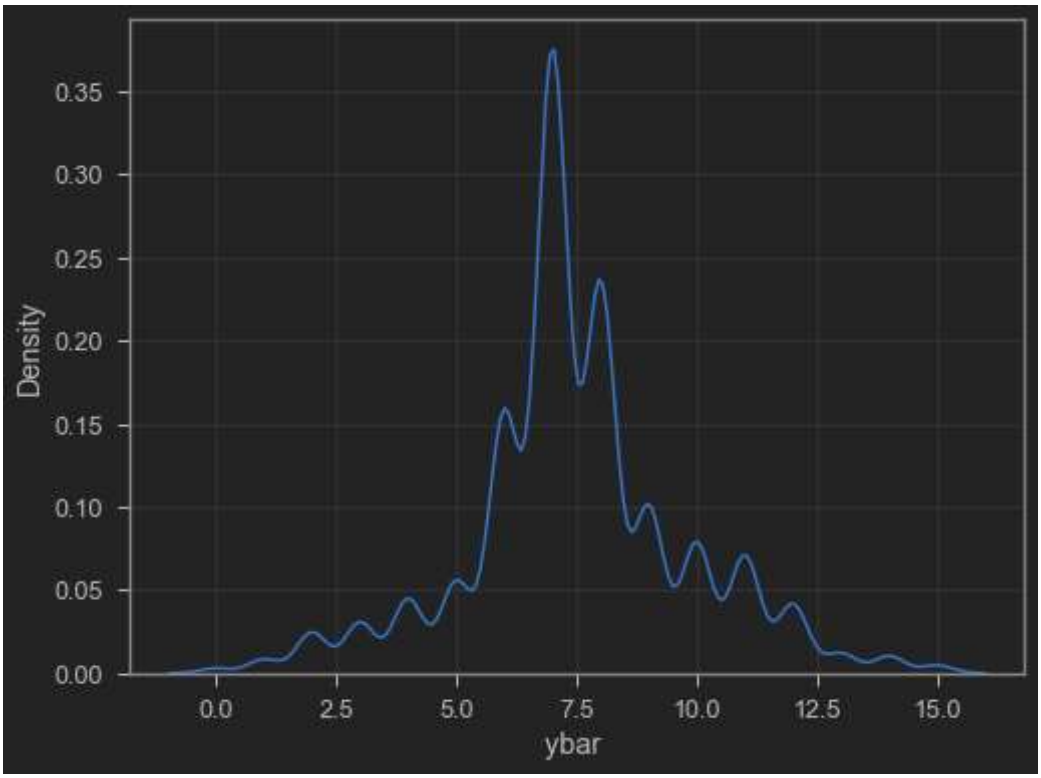

0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
ybox

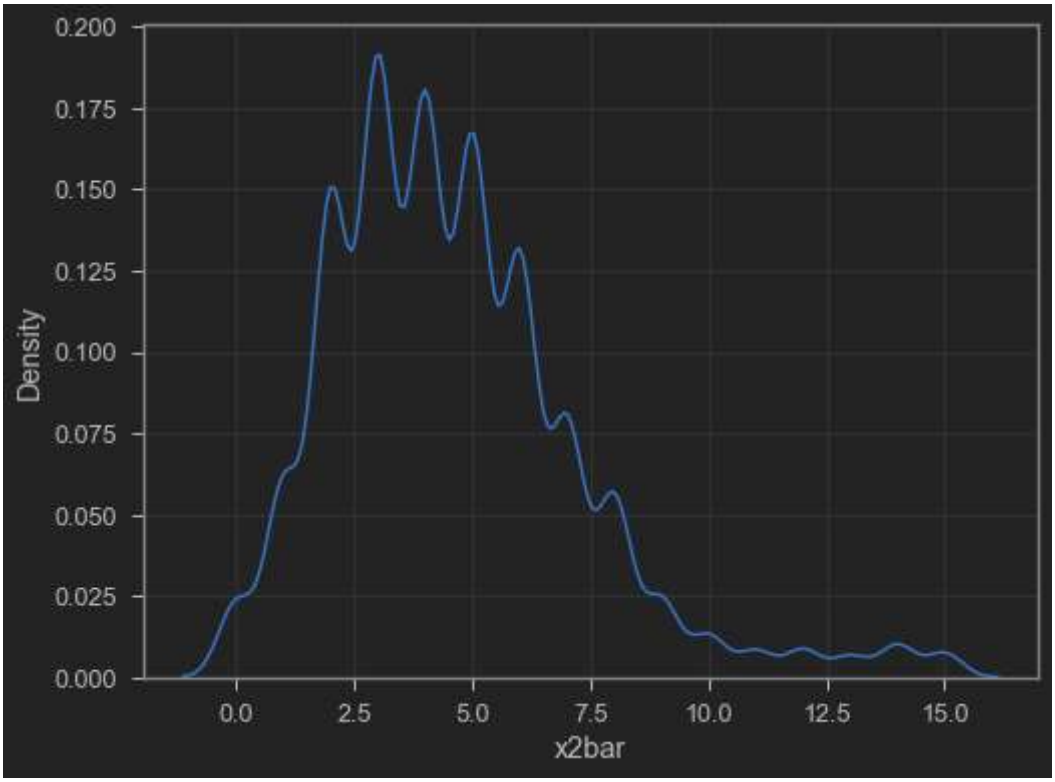

0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
width

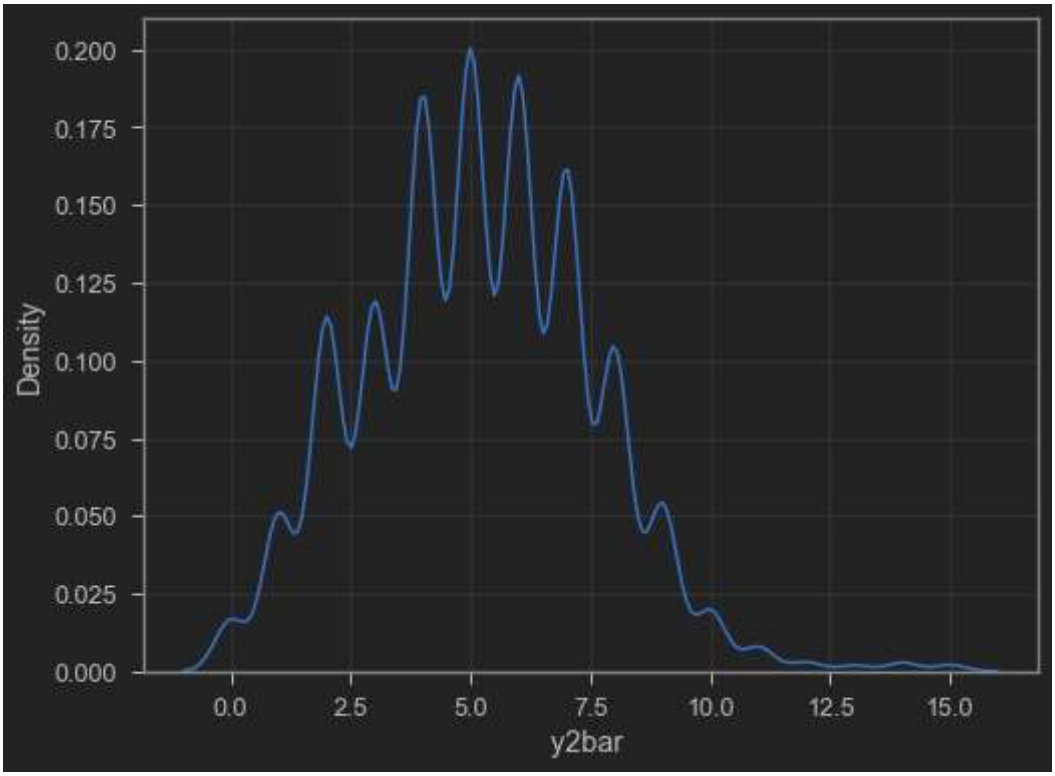

0.20
0.15
0.10
0.05
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
height

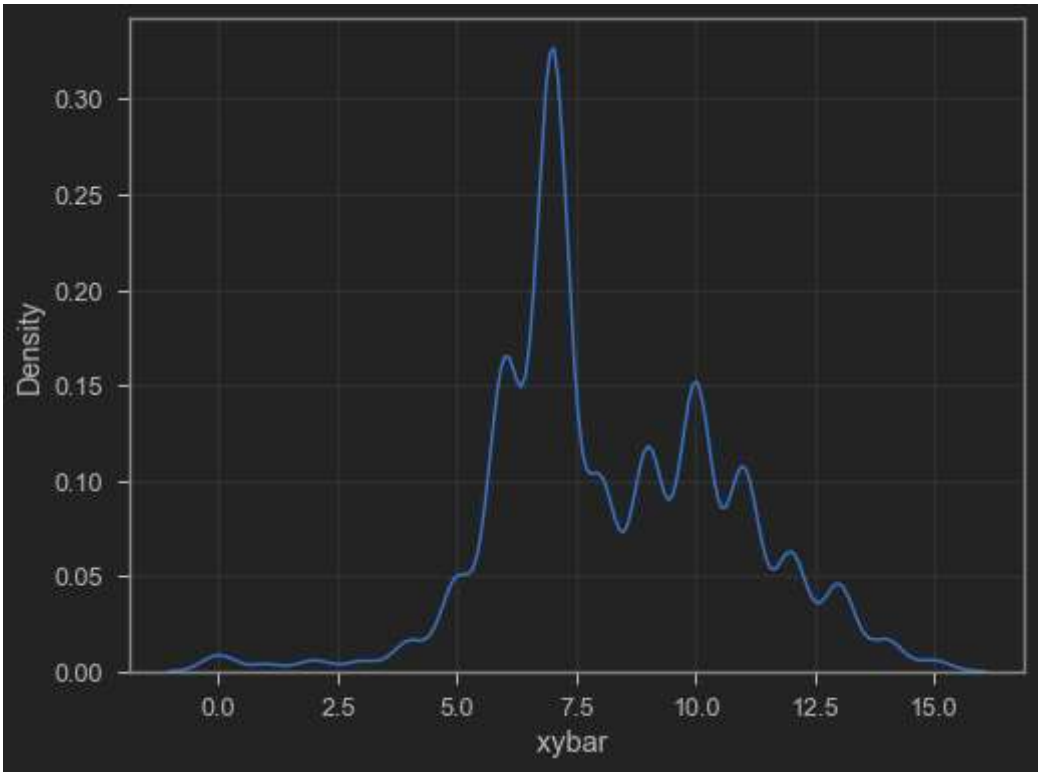

0.25
0.20
0.15
0.10
0.05
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
onpix

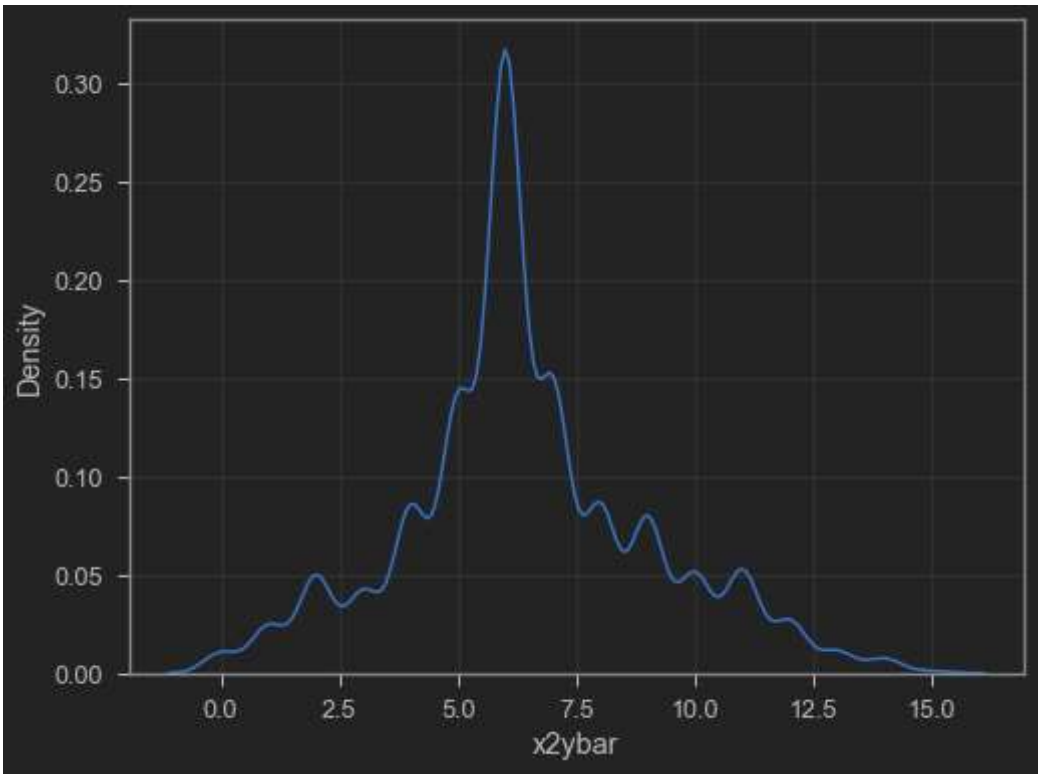

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
xbar

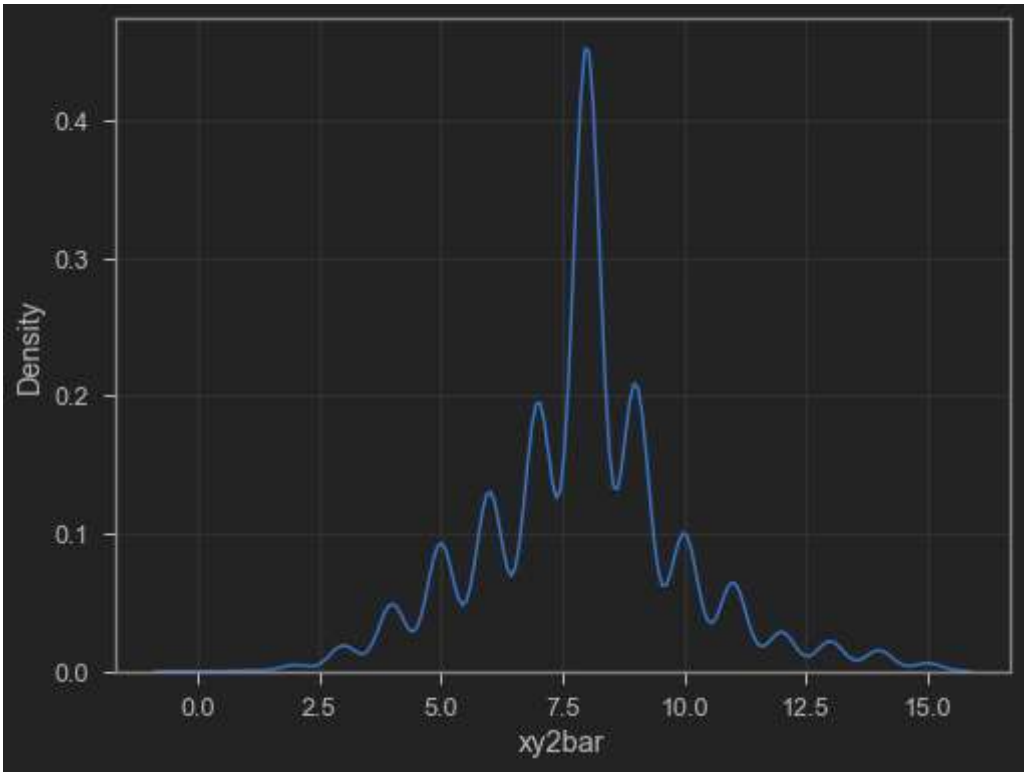

0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
ybar

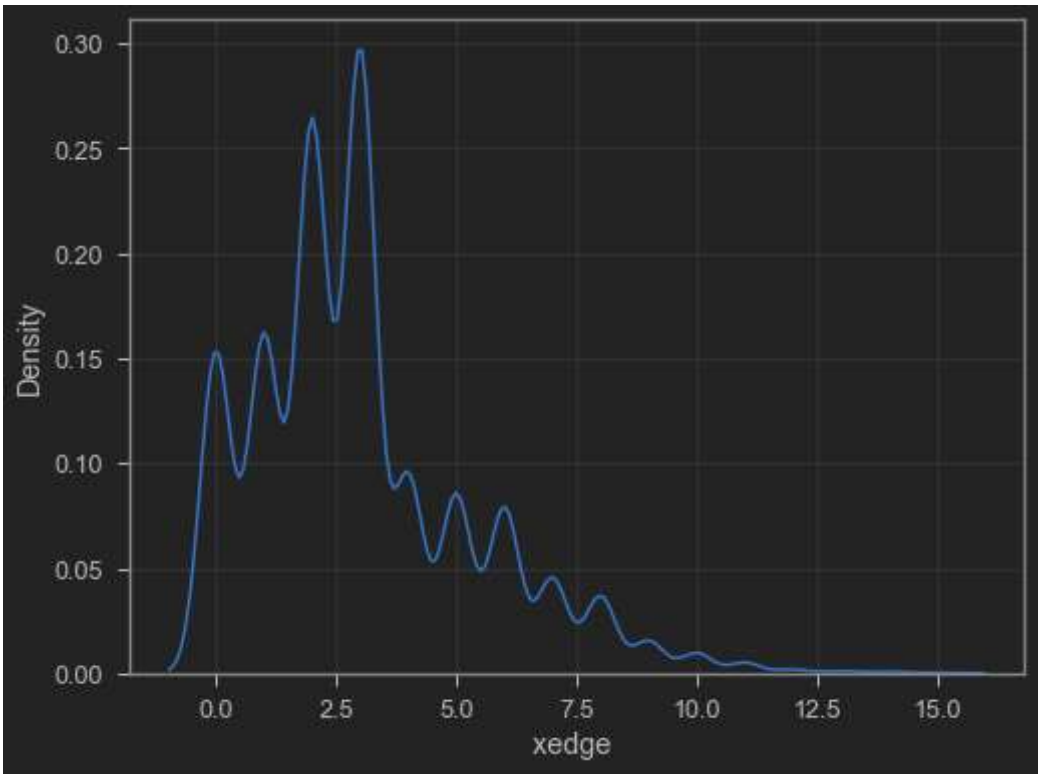

0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
x2bar

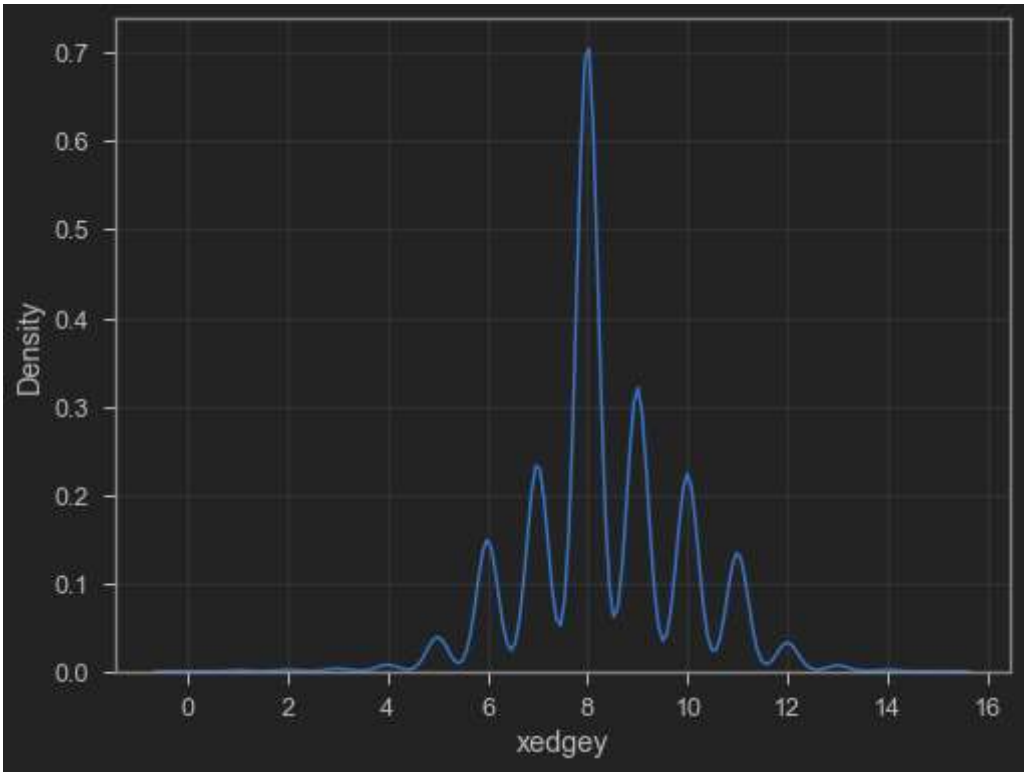

0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
y2bar

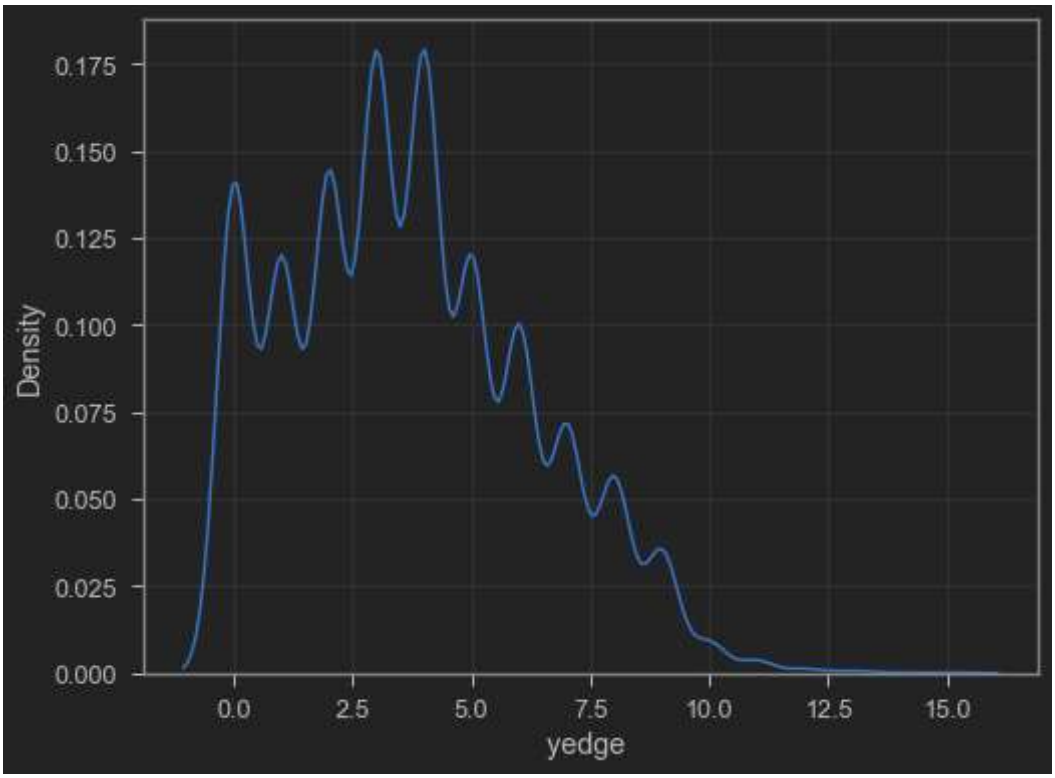

0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
xybar

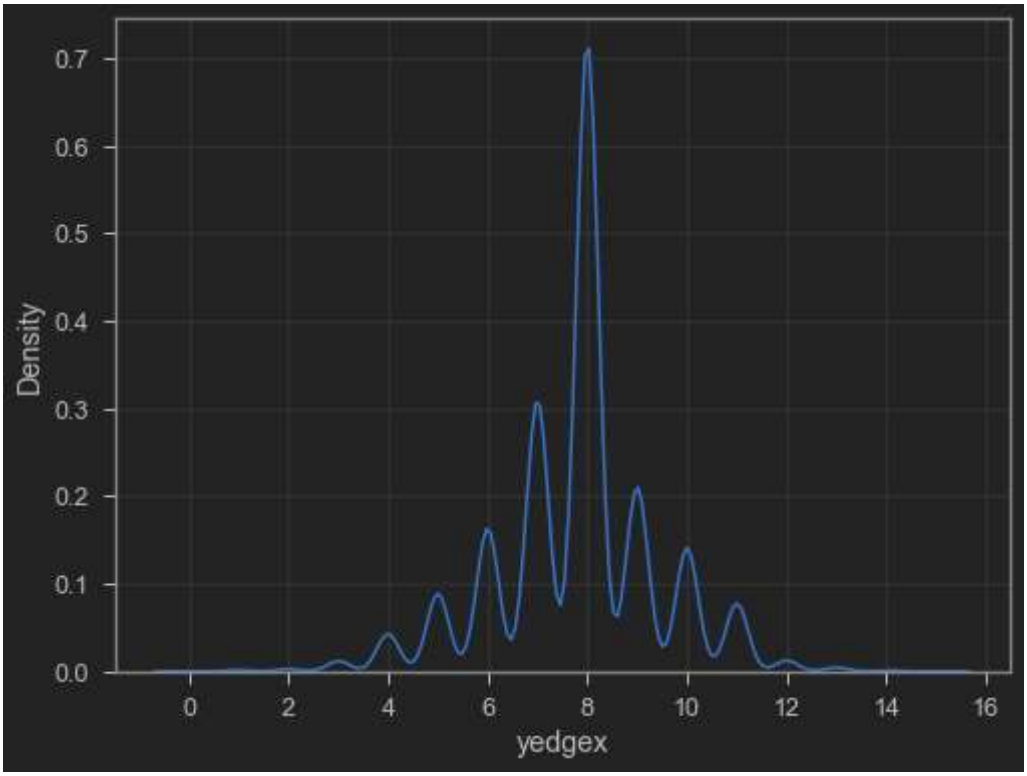

0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
x2ybar


Density
0.4
0.3
0.2
0.1
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
xy2bar


Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
xedge


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
0
2
4
6
8
10
12
14
16
xedgey

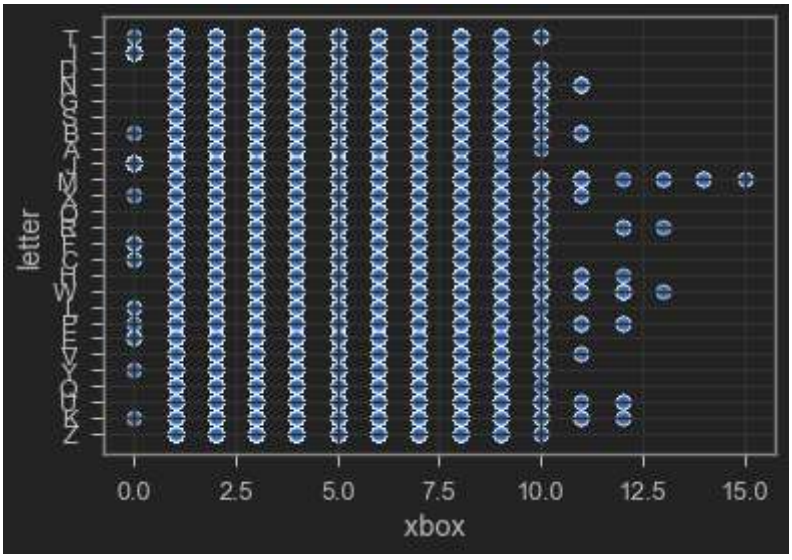

0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Density
0.0
2.5
5.0
7.5
10.0
12.5
15.0
yedge

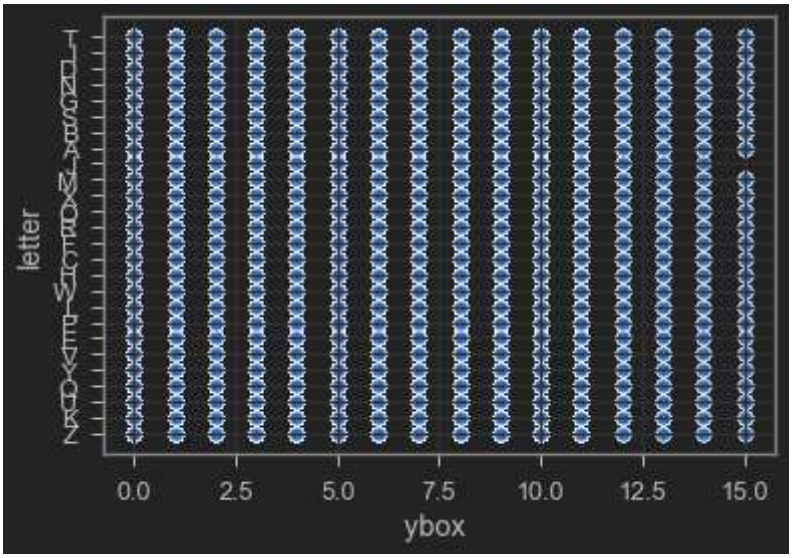


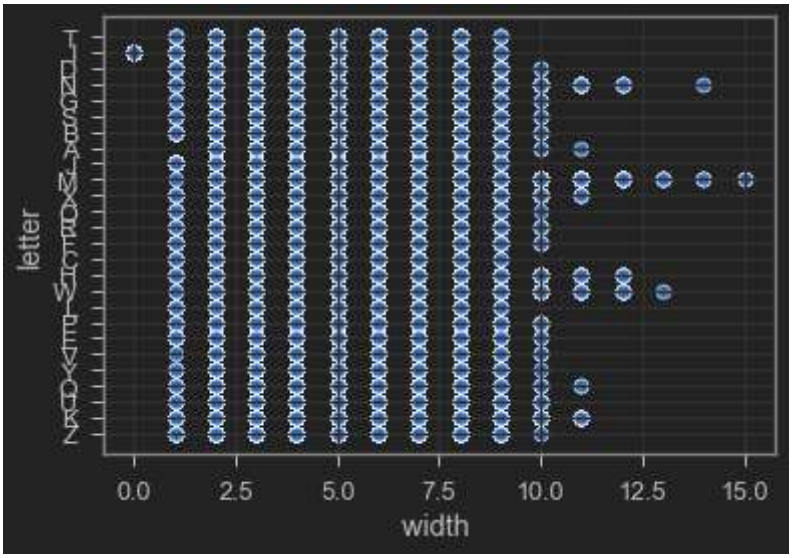
Distribution of data with respect to the arget variable

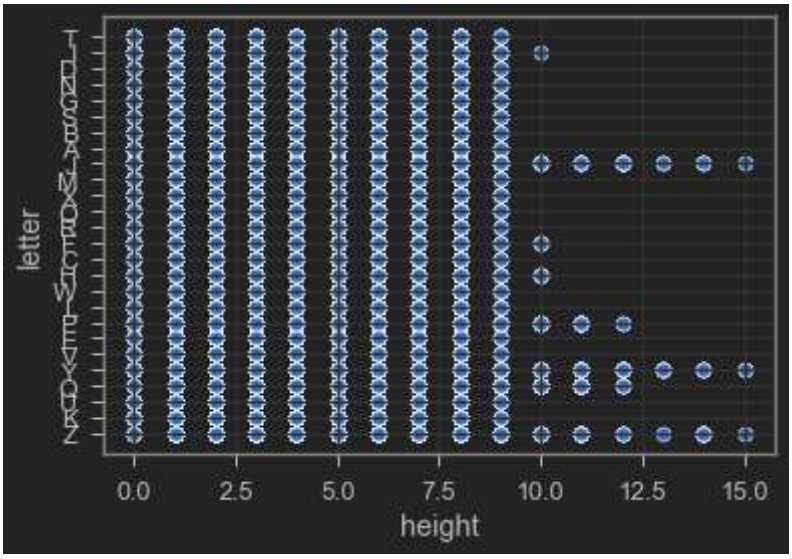
In [5]:

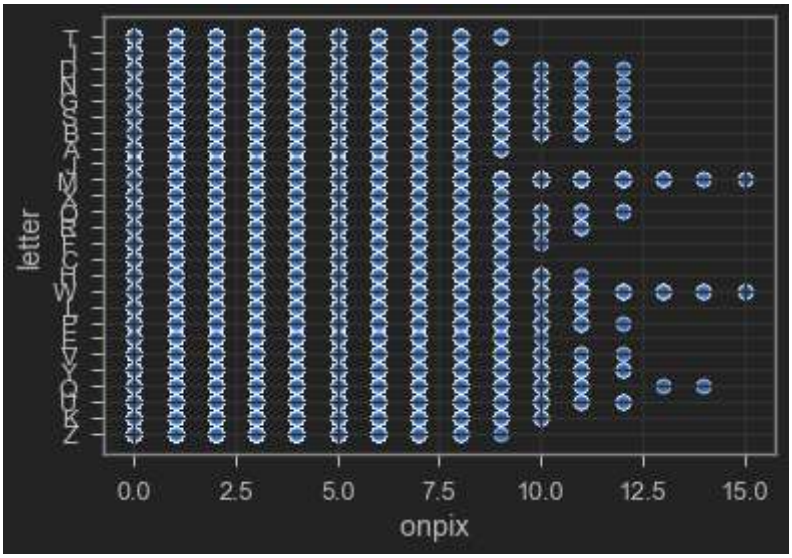
```
for feature in features:
    sns.scatterplot(x=feature, y='letter', data=df)
    plt.show()
```

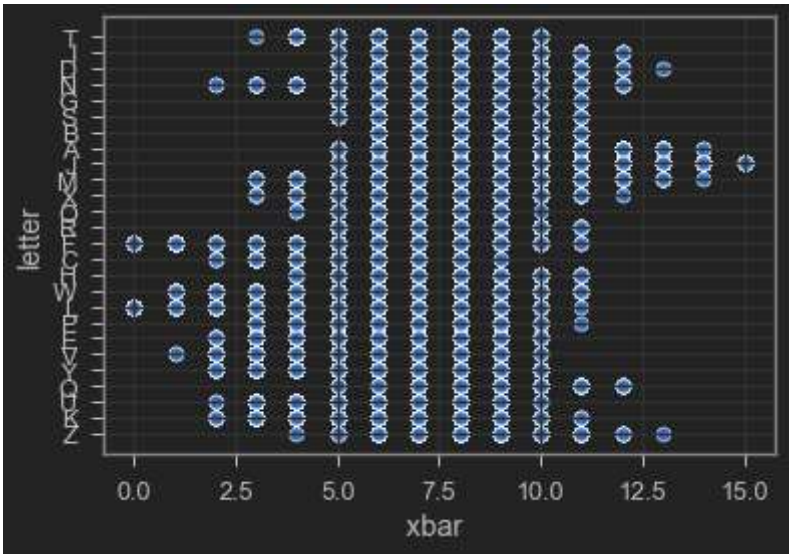


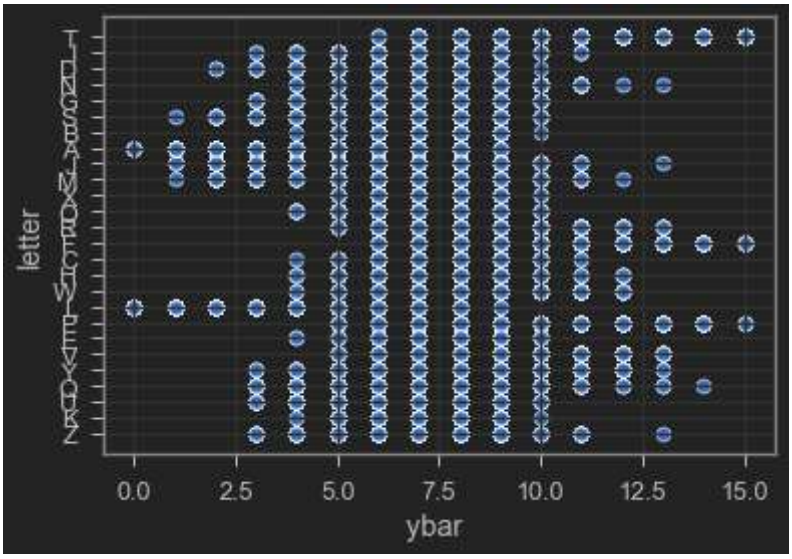

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
ybox

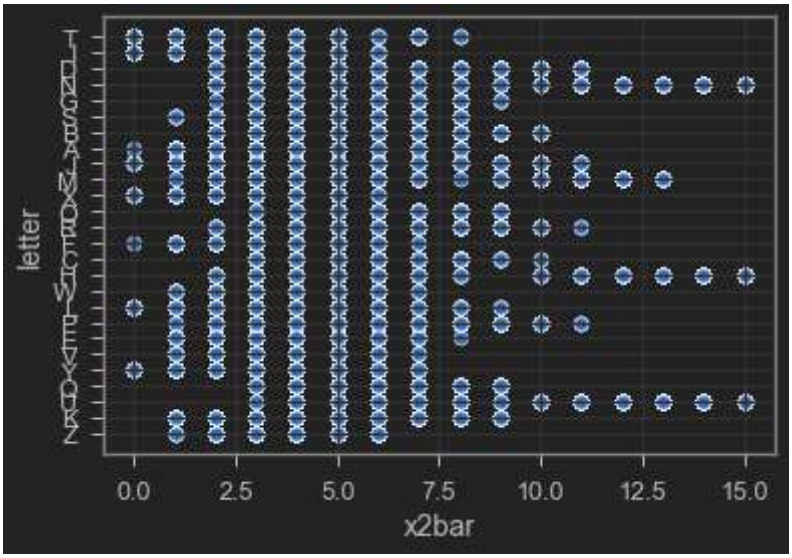

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
width

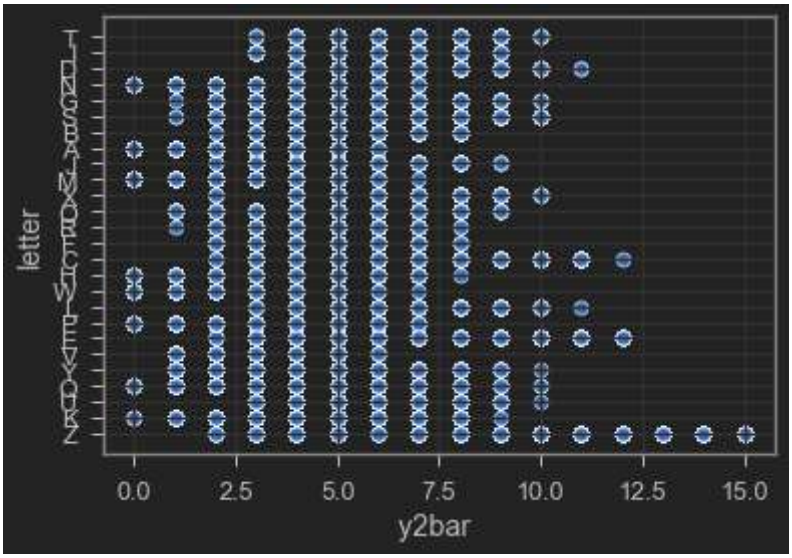

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
height

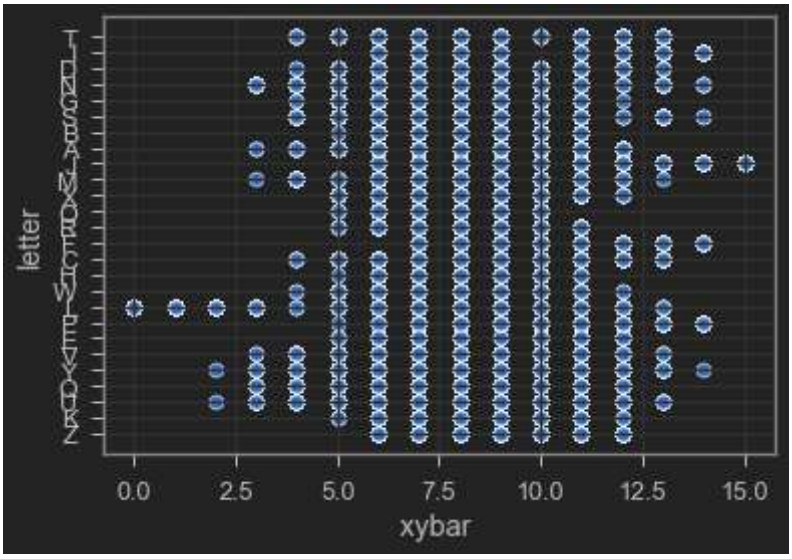

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
onpix

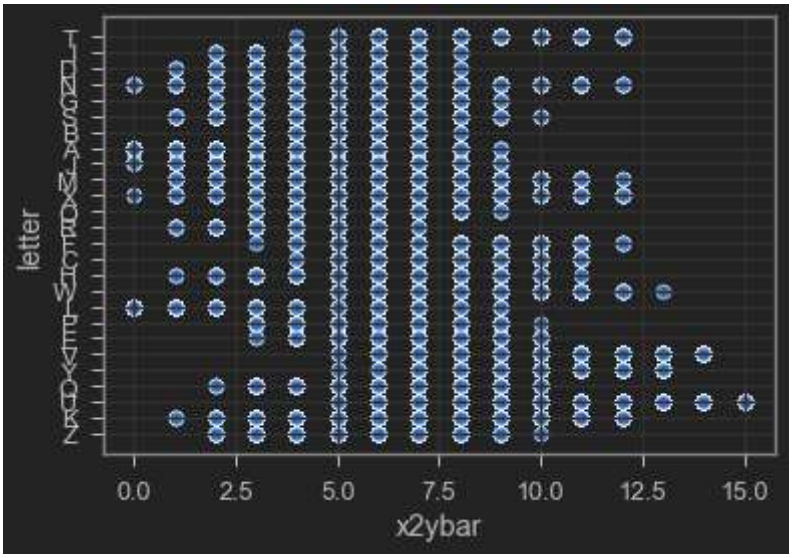

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
xbar

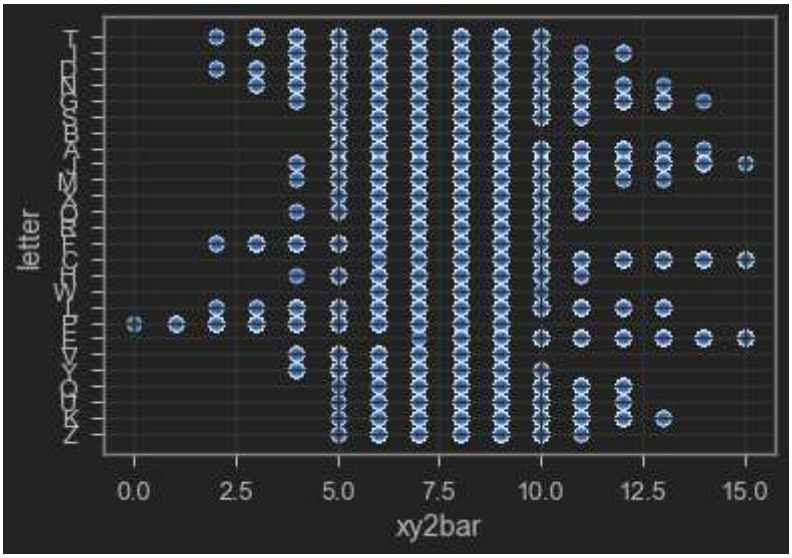

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
ybar

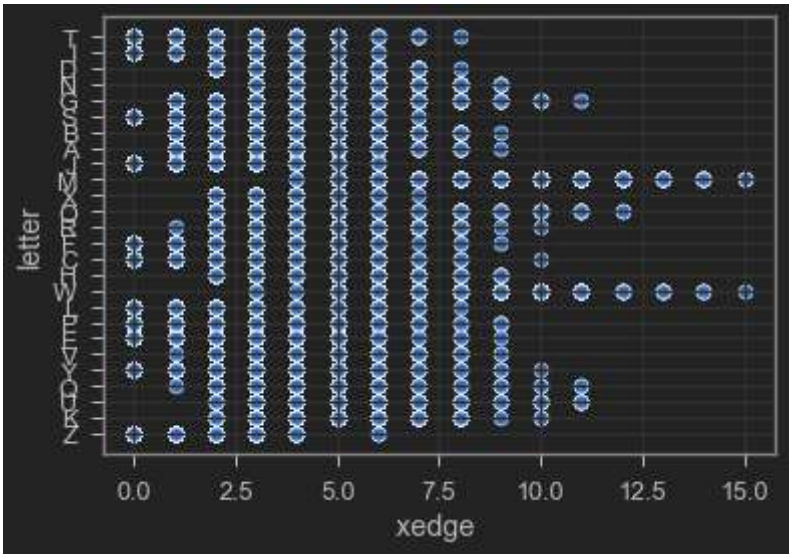

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
x2bar

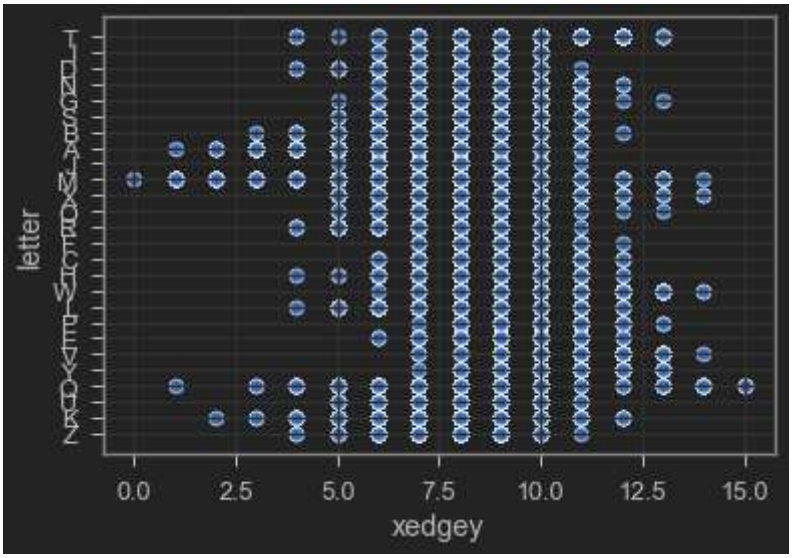

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
y2bar

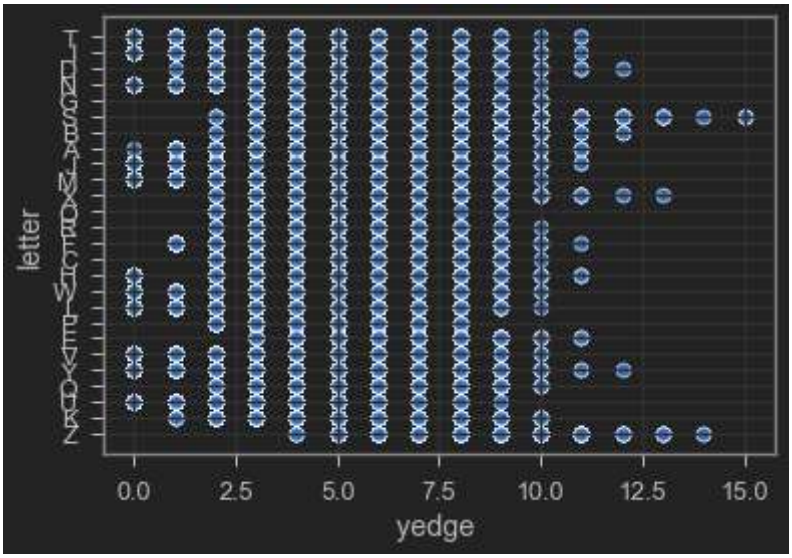

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
xybar

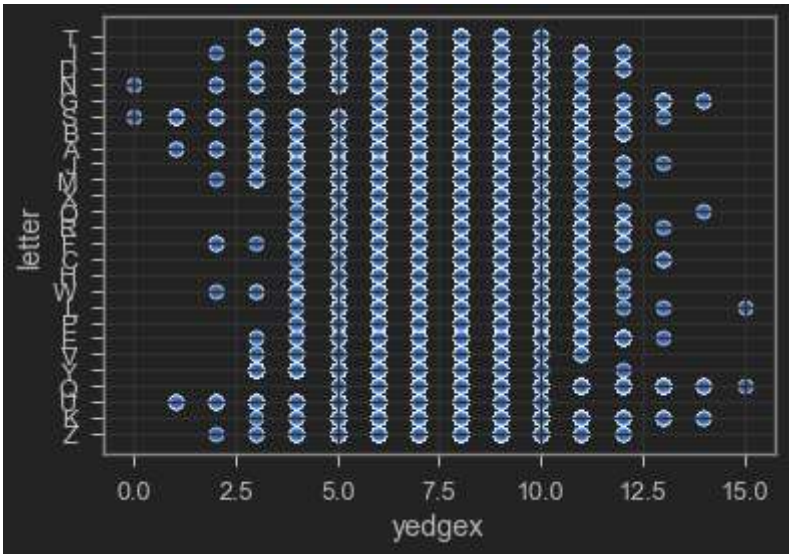

letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
x2ybar


letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
xy2bar


letter
0.0
2.5
5.0
7.5
10.0
12.5
15.0
xedge







Data Splitting and Scaling

In [6]:

```
X = df.drop('letter', axis=1)
y = df['letter']
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)
```

In [8]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

Modeling using SVC

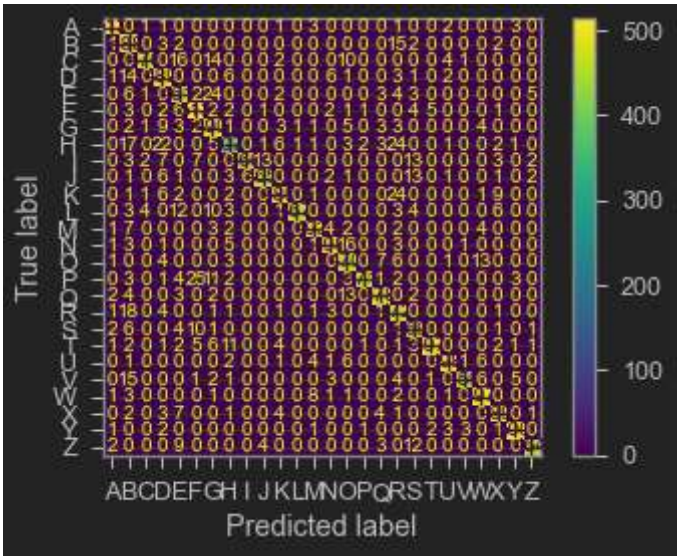
In [9]:
```
svc = SVC()


svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)


score = svc.score(X_test, y_test)
score
```

Out[9]: 0.9078787878787878

Model Evaluation

In [10]:
```
plot_confusion_matrix(svc, X_train, y_train)
```

Out[10]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x25260adab50>



In [11]:
```
print(classification_report(y_test, y_pred))
```

```
svc.get_params()
```

	precision	recall	f1-score	support
A	0.98	0.97	0.98	264
B	0.75	0.94	0.84	253
C	0.96	0.83	0.89	213
D	0.79	0.96	0.87	282
E	0.91	0.91	0.91	257
F	0.89	0.92	0.91	232
G	0.84	0.91	0.87	245
H	0.92	0.67	0.77	238
I	0.97	0.91	0.94	239
J	0.93	0.89	0.91	245
K	0.81	0.87	0.84	203
L	1.00	0.92	0.96	248
M	0.93	0.97	0.95	274
N	0.97	0.91	0.94	254
O	0.86	0.89	0.87	248
P	0.95	0.88	0.92	277
Q	0.93	0.92	0.93	279
R	0.72	0.92	0.81	254
S	0.91	0.89	0.90	253
T	0.97	0.89	0.93	258
U	0.97	0.90	0.93	293
V	0.96	0.94	0.95	267
W	0.92	0.97	0.95	237
X	0.95	0.95	0.95	275
Y	0.98	0.94	0.96	272
Z	0.96	0.88	0.92	240
accuracy			0.91	6600
macro avg	0.91	0.91	0.91	6600
weighted avg	0.91	0.91	0.91	6600

Out[11]:
```
{'C': 1.0,
 'break_ties': False,
 'cache_size': 200,
 'class_weight': None,
 'coef0': 0.0,
 'decision_function_shape': 'ovr',
 'degree': 3,
 'gamma': 'scale',
 'kernel': 'rbf',
 'max_iter': -1,
 'probability': False,
 'random_state': None,
 'shrinking': True,
 'tol': 0.001,
 'verbose': False}
```