In [26]:
```
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_openml
from sklearn.metrics import accuracy_score, classification_report, confusion_mat
from sklearn.multioutput import MultiOutputClassifier
from sklearn.linear_model import LogisticRegression
import seaborn as sns
import matplotlib.pyplot as plt
```

In [28]:
```
yeast = fetch_openml(name='yeast',version=4,as_frame=True,parser='auto')
X = yeast.data
y = yeast.target
```

# Preprocessing and understanding the data

In [31]:
```
X.head()
```

Out[31]:

| | Att1 | Att2 | Att3 | Att4 | Att5 | Att6 | Att7 | Att8 |
|---|---|---|---|---|---|---|---|---|
| **0** | 0.004168 | -0.170975 | -0.156748 | -0.142151 | 0.058781 | 0.026851 | 0.197719 | 0.041850 |
| **1** | -0.103956 | 0.011879 | -0.098986 | -0.054501 | -0.007970 | 0.049113 | -0.030580 | -0.077933 |
| **2** | 0.509949 | 0.401709 | 0.293799 | 0.087714 | 0.011686 | -0.006411 | -0.006255 | 0.013646 |
| **3** | 0.119092 | 0.004412 | -0.002262 | 0.072254 | 0.044512 | -0.051467 | 0.074686 | -0.007670 |
| **4** | 0.042037 | 0.007054 | -0.069483 | 0.081015 | -0.048207 | 0.089446 | -0.004947 | 0.064456 |

5 rows × 103 columns

In [33]:
```
y.head()
```

Out[33]:

| | Class1 | Class2 | Class3 | Class4 | Class5 | Class6 | Class7 | Class8 | Class9 | Class10 | Class1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | TRUE | TRUE | FALSE | FALSE | FALS |
| **1** | FALSE | FALSE | TRUE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALS |
| **2** | FALSE | TRUE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALS |
| **3** | FALSE | FALSE | TRUE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALS |
| **4** | FALSE | FALSE | TRUE | TRUE | TRUE | TRUE | FALSE | FALSE | FALSE | FALSE | FALS |

In [35]:
```
X.isnull().sum()
```

Out[35]:
```
Att1      0
Att2      0
Att3      0
Att4      0
Att5      0
         ..
Att99     0
Att100    0
Att101    0
Att102    0
Att103    0
Length: 103, dtype: int64
```

In [37]:
```
y.isnull().sum()
```

Out[37]:
```
Class1     0
Class2     0
Class3     0
Class4     0
Class5     0
Class6     0
Class7     0
Class8     0
Class9     0
Class10    0
Class11    0
Class12    0
Class13    0
Class14    0
dtype: int64
```

In [39]:
```
X.describe()
```

Out[39]:

| | Att1 | Att2 | Att3 | Att4 | Att5 | Att6 | |
|---|---|---|---|---|---|---|---|
| **count** | 2417.000000 | 2417.000000 | 2417.000000 | 2417.000000 | 2417.000000 | 2417.000000 | 2 |
| **mean** | 0.001173 | -0.000436 | -0.000257 | 0.000265 | 0.001228 | 0.000475 | |
| **std** | 0.097411 | 0.097885 | 0.097746 | 0.096969 | 0.096909 | 0.097306 | |
| **min** | -0.371146 | -0.472632 | -0.339195 | -0.467945 | -0.367044 | -0.509447 | |
| **25%** | -0.053655 | -0.058734 | -0.057526 | -0.057149 | -0.058461 | -0.060212 | |
| **50%** | 0.003649 | -0.003513 | 0.002892 | -0.000153 | 0.005565 | 0.000321 | |
| **75%** | 0.057299 | 0.048047 | 0.061007 | 0.054522 | 0.066286 | 0.059908 | |
| **max** | 0.520272 | 0.614114 | 0.353241 | 0.568960 | 0.307649 | 0.336971 | |

8 rows × 103 columns

**From the descriptive analysis, it is evident that the dataset is already scaled within the range of -1 to 1. As a result, no further normalization of the data is necessary. Additionally, there are no null values present in either the feature matrix (X) or the target variable (y).**

## Given this, we can proceed directly to implementing the multilabel classification model.

## To prepare the target variable y for multilabel classification, we will convert the True values to 1 and the False values to 0. This will create binary labels for each class, where 1 represents the presence of that class and 0 indicates its absence.

In [43]:
```
y = y.astype(str).applymap(lambda x: 1 if x == 'TRUE' else 0) # Convert to strin
```

In [45]:
```
y.head()
```

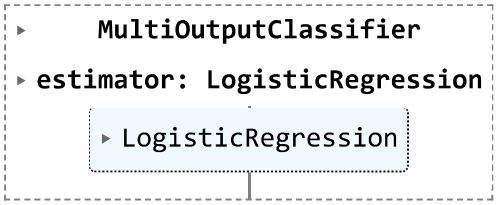
Out[45]:

| | Class1 | Class2 | Class3 | Class4 | Class5 | Class6 | Class7 | Class8 | Class9 | Class10 | Class1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | |
| **1** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **2** | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **3** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **4** | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | |

In [47]:
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [49]:
```
model = MultiOutputClassifier(LogisticRegression(solver='lbfgs', max_iter=1000))
model.fit(X_train, y_train)
```

Out[49]:




In [50]:
```
y_pred = model.predict(X_test)
```

## Accuraccy per label

In [54]:
```
print("Accuracy per label:")
for i, label in enumerate(yeast.target_names):
    print(f"{label} Accuracy: {accuracy_score(y_test.iloc[:, i], y_pred[:, i]):.
```

```
Accuracy per label:
Class1 Accuracy: 0.78
Class2 Accuracy: 0.64
Class3 Accuracy: 0.74
Class4 Accuracy: 0.74
Class5 Accuracy: 0.75
Class6 Accuracy: 0.76
Class7 Accuracy: 0.84
Class8 Accuracy: 0.83
Class9 Accuracy: 0.94
Class10 Accuracy: 0.88
Class11 Accuracy: 0.87
Class12 Accuracy: 0.76
Class13 Accuracy: 0.76
Class14 Accuracy: 0.99
```

## The classification report

In [57]:
```
print("\nClassification Report for each label:")
for i, label in enumerate(yeast.target_names):
    print(f"\n{label} Classification Report:\n", classification_report(y_test.il
```

Classification Report for each label:

Class1 Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.95 | 0.85 | 317 |
| 1 | 0.82 | 0.46 | 0.58 | 167 |
| accuracy | | | 0.78 | 484 |
| macro avg | 0.79 | 0.70 | 0.72 | 484 |
| weighted avg | 0.78 | 0.78 | 0.76 | 484 |

Class2 Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.65 | 0.77 | 0.71 | 273 |
| 1 | 0.61 | 0.47 | 0.53 | 211 |
| accuracy | | | 0.64 | 484 |
| macro avg | 0.63 | 0.62 | 0.62 | 484 |
| weighted avg | 0.63 | 0.64 | 0.63 | 484 |

Class3 Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.80 | 0.78 | 288 |
| 1 | 0.68 | 0.65 | 0.66 | 196 |
| accuracy | | | 0.74 | 484 |
| macro avg | 0.73 | 0.72 | 0.72 | 484 |
| weighted avg | 0.73 | 0.74 | 0.73 | 484 |

Class4 Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.85 | 0.81 | 313 |
| 1 | 0.67 | 0.54 | 0.60 | 171 |
| accuracy | | | 0.74 | 484 |
| macro avg | 0.72 | 0.70 | 0.70 | 484 |
| weighted avg | 0.73 | 0.74 | 0.73 | 484 |

Class5 Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.92 | 0.84 | 340 |
| 1 | 0.65 | 0.36 | 0.46 | 144 |
| accuracy | | | 0.75 | 484 |
| macro avg | 0.71 | 0.64 | 0.65 | 484 |
| weighted avg | 0.74 | 0.75 | 0.73 | 484 |

Class6 Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|

```
              0       0.76      0.97      0.86       357
              1       0.67      0.14      0.23       127

    accuracy                            0.76       484
   macro avg       0.71      0.56      0.54       484
weighted avg       0.74      0.76      0.69       484


Class7 Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.99      0.91       408
           1       0.50      0.04      0.07        76

    accuracy                           0.84       484
   macro avg       0.67      0.52      0.49       484
weighted avg       0.79      0.84      0.78       484


Class8 Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.99      0.91       401
           1       0.57      0.05      0.09        83

    accuracy                           0.83       484
   macro avg       0.70      0.52      0.50       484
weighted avg       0.79      0.83      0.77       484


Class9 Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       454
           1       1.00      0.00      0.00        30

    accuracy                           0.94       484
   macro avg       0.97      0.50      0.48       484
weighted avg       0.94      0.94      0.91       484


Class10 Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94       429
           1       0.00      0.00      1.00        55

    accuracy                           0.88       484
   macro avg       0.44      0.50      0.97       484
weighted avg       0.79      0.88      0.95       484


Class11 Classification Report:
              precision    recall  f1-score   support

           0       0.87      1.00      0.93       422
           1       0.50      0.02      0.03        62

    accuracy                           0.87       484
   macro avg       0.69      0.51      0.48       484
```

```
weighted avg       0.83      0.87      0.82       484


Class12 Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.07      0.12       118
           1       0.77      0.98      0.86       366

    accuracy                           0.76       484
   macro avg       0.67      0.53      0.49       484
weighted avg       0.72      0.76      0.68       484


Class13 Classification Report:
              precision    recall  f1-score   support

           0       0.53      0.07      0.12       119
           1       0.76      0.98      0.86       365

    accuracy                           0.76       484
   macro avg       0.65      0.52      0.49       484
weighted avg       0.71      0.76      0.68       484


Class14 Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       477
           1       1.00      0.00      0.00         7

    accuracy                           0.99       484
   macro avg       0.99      0.50      0.50       484
weighted avg       0.99      0.99      0.98       484
```

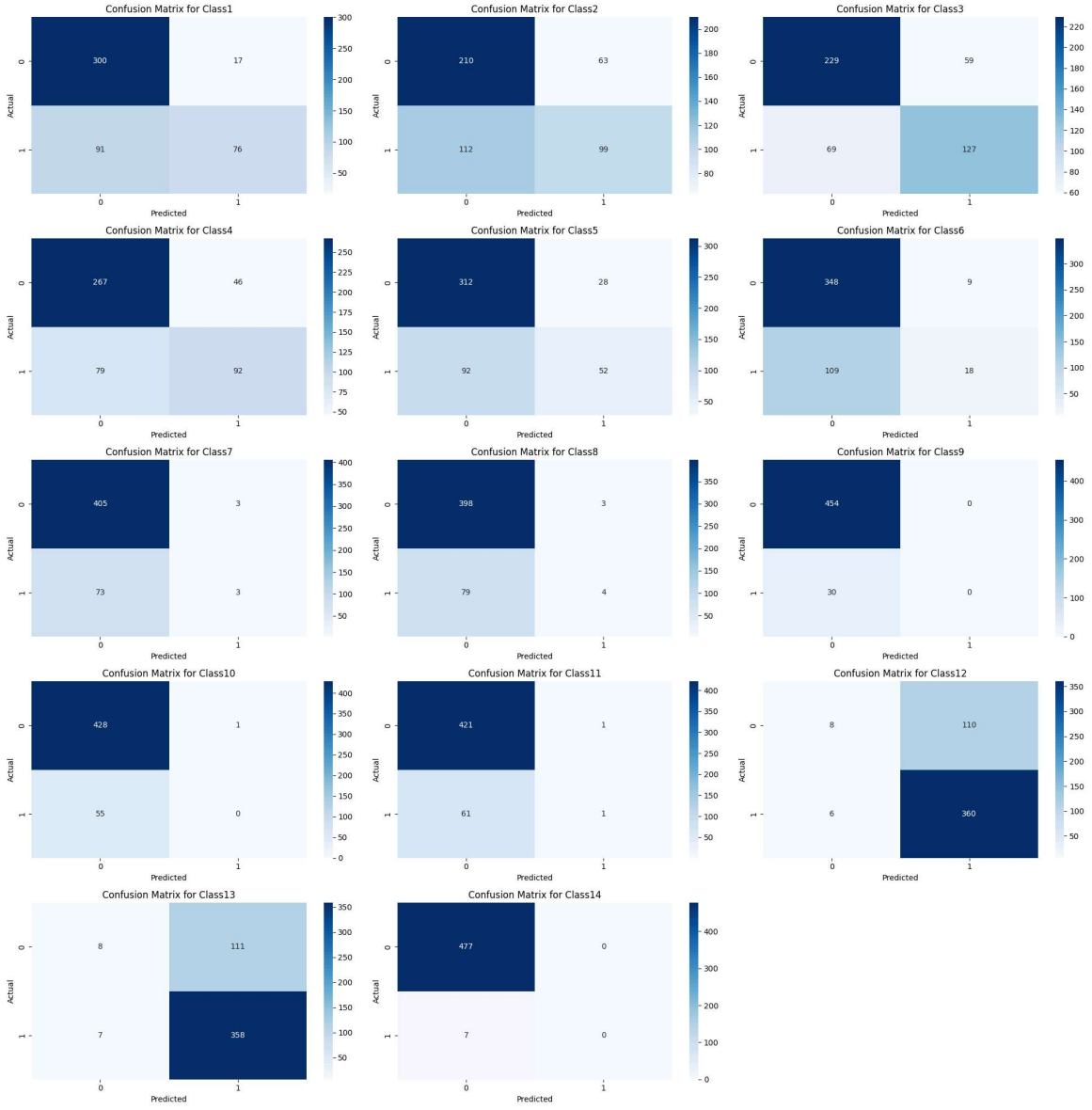
## The confusion matrix

In [62]:

```
fig, axes = plt.subplots(5, 3, figsize=(20, 20))

for i, ax in enumerate(axes.flat):
    if i < 14:
        sns.heatmap(confusion_matrix(y_test.iloc[:, i], y_pred[:, i]), ax=ax, an
        ax.set_xlabel("Predicted")
        ax.set_ylabel("Actual")
        ax.set_title(f"Confusion Matrix for {y_test.columns[i]}")
    else:
        ax.axis('off')

plt.tight_layout()
plt.show()
```


Confusion Matrix for Class1
300
17
91
76
Confusion Matrix for Class2
210
63
112
99
Confusion Matrix for Class3
229
59
69
127
Confusion Matrix for Class4
267
46
79
92
Confusion Matrix for Class5
312
28
92
52
Confusion Matrix for Class6
348
9
109
18
Confusion Matrix for Class7
405
3
73
3
Confusion Matrix for Class8
398
3
79
4
Confusion Matrix for Class9
454
0
30
0
Confusion Matrix for Class10
428
1
55
0
Confusion Matrix for Class11
421
1
61
1
Confusion Matrix for Class12
8
110
6
360
Confusion Matrix for Class13
8
111
7
358
Confusion Matrix for Class14
477
0
7
0
Actual
Predicted

In [ ]: