In [12]:
```
%matplotlib inline
```

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

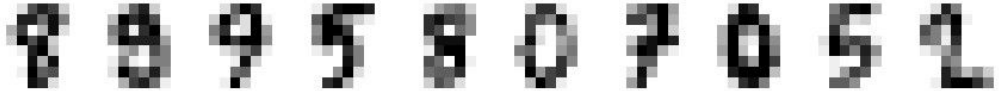
In [14]:
```
digits = datasets.load_digits()

#digits.data flattened image
#digits.image 8x8 image
#digits.target label

#print("digits.data.shape,digits.target.shape,digits.images.shape")
#print(digits.data.shape,digits.target.shape,digits.images.shape)

_, axes = plt.subplots(nrows=1, ncols=10, figsize=(10, 3))
for ax, image, label in zip(axes, digits.images, digits.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
    ax.set_title("Training: %i" % label)
```

In [15]:

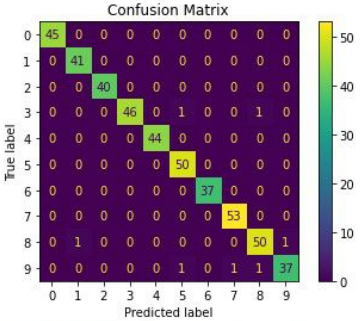
```python
from sklearn import svm, linear_model
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import precision_score, accuracy_score, recall_score
import time

#Run supervised training and classification testing using the SVM, Naïve Bayes and Logistic Regression classifiers
svc = svm.SVC()
lr = linear_model.LogisticRegression(max_iter=5000)
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()

X_train = digits.data
X_test = digits.target

# Split data into 25% test and 75% training sets
X_train, X_test, y_train, y_test = train_test_split(X_train, X_test, test_size=0.25, shuffle=True)

#Run iterations for each type of classifier
for clf in [svc, lr, gnb, mnb, bnb]:
  y_pred=clf.fit(X_train, y_train).predict(X_test)

#To display the accuracy scores for each supervised test classifiers run
  print(accuracy_score(y_pred,y_test),clf)
  _, axes = plt.subplots(nrows=1, ncols=10, figsize=(15, 3))

#Display sample image predictions
  for ax, image, prediction in zip(axes, X_test, y_pred):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
    ax.set_title(f"Prediction: {prediction}")
  plt.show()

#Display classification reports for each classifier
  print(
    f"Classification report for classifier {clf}:\n"
    f"{metrics.classification_report(y_test, y_pred)}\n"
  )
# Calculate and plot confusion matrices for each classifier
  confusion_matrices = confusion_matrix(y_test, y_pred, labels=clf.classes_)
  disp = metrics.ConfusionMatrixDisplay(confusion_matrix=confusion_matrices,display_labels=clf.classes_)
  disp.plot()
  disp.ax_.set_title("Confusion Matrix")
  print(f"Confusion matrix:\n{disp.confusion_matrix}")
  plt.show()
```

```
0.9844444444444445 SVC()
```

Prediction: 8 Prediction: 9 Prediction: 9 Prediction: 5 Prediction: 8 Prediction: 0 Prediction: 7 Prediction: 0 Prediction: 5 Prediction: 2



```
Classification report for classifier SVC():
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       0.98      1.00      0.99        41
           2       1.00      1.00      1.00        40
           3       1.00      0.96      0.98        48
           4       1.00      1.00      1.00        44
           5       0.96      1.00      0.98        50
           6       1.00      1.00      1.00        37
           7       0.98      1.00      0.99        53
           8       0.96      0.96      0.96        52
           9       0.97      0.93      0.95        40

    accuracy                           0.98       450
   macro avg       0.99      0.98      0.98       450
weighted avg       0.98      0.98      0.98       450


Confusion matrix:
[[45  0  0  0  0  0  0  0  0  0]
 [ 0 41  0  0  0  0  0  0  0  0]
 [ 0  0 40  0  0  0  0  0  0  0]
 [ 0  0  0 46  0  1  0  0  1  0]
 [ 0  0  0  0 44  0  0  0  0  0]
 [ 0  0  0  0  0 50  0  0  0  0]
 [ 0  0  0  0  0  0 37  0  0  0]
 [ 0  0  0  0  0  0  0 53  0  0]
 [ 0  1  0  0  0  0  0  0 50  1]
 [ 0  0  0  0  0  1  0  1  1 37]]
```

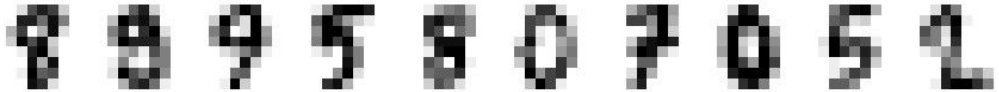



```
0.9644444444444444 LogisticRegression(max_iter=5000)
```

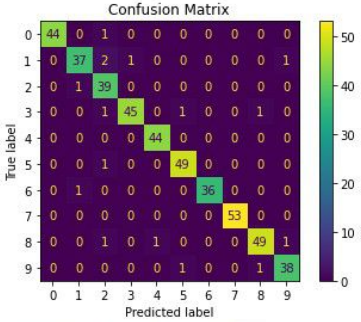
Prediction: 8 Prediction: 9 Prediction: 9 Prediction: 5 Prediction: 8 Prediction: 0 Prediction: 7 Prediction: 0 Prediction: 5 Prediction: 2

```
0.9644444444444444 LogisticRegression(max_iter=5000)
```

Prediction: 8 Prediction: 9 Prediction: 9 Prediction: 5 Prediction: 8 Prediction: 0 Prediction: 7 Prediction: 0 Prediction: 5 Prediction: 2



```
Classification report for classifier LogisticRegression(max_iter=5000):
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        45
           1       0.95      0.90      0.92        41
           2       0.87      0.97      0.92        40
           3       0.98      0.94      0.96        48
           4       0.98      1.00      0.99        44
           5       0.96      0.98      0.97        50
           6       1.00      0.97      0.99        37
           7       1.00      1.00      1.00        53
           8       0.96      0.94      0.95        52
           9       0.95      0.95      0.95        40

    accuracy                           0.96       450
   macro avg       0.96      0.96      0.96       450
weighted avg       0.97      0.96      0.96       450


Confusion matrix:
[[44  0  1  0  0  0  0  0  0  0]
 [ 0 37  2  1  0  0  0  0  0  1]
 [ 0  1 39  0  0  0  0  0  0  0]
 [ 0  0  1 45  0  1  0  0  1  0]
 [ 0  0  0  0 44  0  0  0  0  0]
 [ 0  0  1  0  0 49  0  0  0  0]
 [ 0  1  0  0  0  0 36  0  0  0]
 [ 0  0  0  0  0  0  0 53  0  0]
 [ 0  0  1  0  1  0  0  0 49  1]
 [ 0  0  0  0  0  1  0  0  1 38]]
```

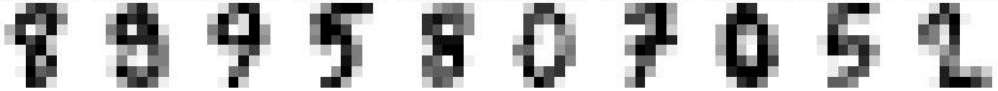



```
0.8622222222222222 GaussianNB()
```

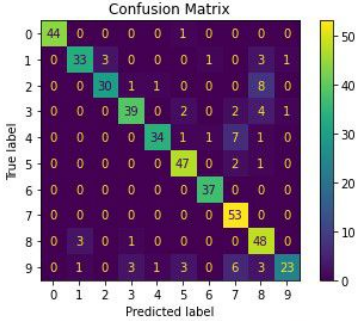
Prediction: 8 Prediction: 9 Prediction: 7 Prediction: 5 Prediction: 8 Prediction: 0 Prediction: 7 Prediction: 0 Prediction: 5 Prediction: 2

```
0.8622222222222222 GaussianNB()
```

Prediction: 8 Prediction: 9 Prediction: 7 Prediction: 5 Prediction: 8 Prediction: 0 Prediction: 7 Prediction: 0 Prediction: 5 Prediction: 2



```
Classification report for classifier GaussianNB():
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        45
           1       0.89      0.80      0.85        41
           2       0.91      0.75      0.82        40
           3       0.89      0.81      0.85        48
           4       0.94      0.77      0.85        44
           5       0.87      0.94      0.90        50
           6       0.95      1.00      0.97        37
           7       0.76      1.00      0.86        53
           8       0.71      0.92      0.80        52
           9       0.92      0.57      0.71        40

    accuracy                           0.86       450
   macro avg       0.88      0.86      0.86       450
weighted avg       0.88      0.86      0.86       450


Confusion matrix:
[[44  0  0  0  0  1  0  0  0  0]
 [ 0 33  3  0  0  0  1  0  3  1]
 [ 0  0 30  1  1  0  0  0  8  0]
 [ 0  0  0 39  0  2  0  2  4  1]
 [ 0  0  0  0 34  1  1  7  1  0]
 [ 0  0  0  0  0 47  0  2  1  0]
 [ 0  0  0  0  0  0 37  0  0  0]
 [ 0  0  0  0  0  0  0 53  0  0]
 [ 0  3  0  1  0  0  0  0 48  0]
 [ 0  1  0  3  1  3  0  6  3 23]]
```

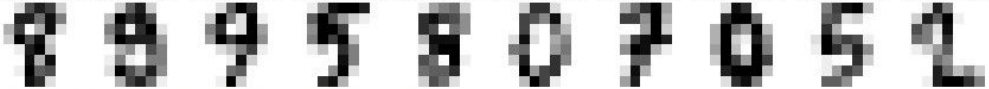



```
0.8977777777777778 MultinomialNB()
```

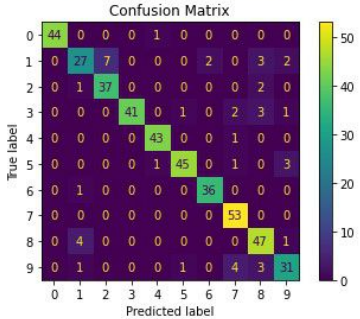
Prediction: 8 Prediction: 9 Prediction: 7 Prediction: 5 Prediction: 8 Prediction: 0 Prediction: 7 Prediction: 0 Prediction: 5 Prediction: 2

```
0.8977777777777778 MultinomialNB()
```

Prediction: 8 Prediction: 9 Prediction: 7 Prediction: 5 Prediction: 8 Prediction: 0 Prediction: 7 Prediction: 0 Prediction: 5 Prediction: 2



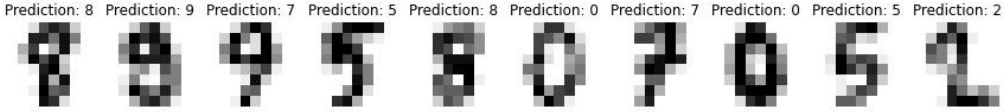
```
Classification report for classifier MultinomialNB():
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        45
           1       0.79      0.66      0.72        41
           2       0.84      0.93      0.88        40
           3       1.00      0.85      0.92        48
           4       0.96      0.98      0.97        44
           5       0.96      0.90      0.93        50
           6       0.95      0.97      0.96        37
           7       0.87      1.00      0.93        53
           8       0.81      0.90      0.85        52
           9       0.82      0.78      0.79        40

    accuracy                           0.90       450
   macro avg       0.90      0.89      0.89       450
weighted avg       0.90      0.90      0.90       450


Confusion matrix:
[[44  0  0  0  1  0  0  0  0  0]
 [ 0 27  7  0  0  0  2  0  3  2]
 [ 0  1 37  0  0  0  0  0  2  0]
 [ 0  0  0 41  0  1  0  2  3  1]
 [ 0  0  0  0 43  0  0  1  0  0]
 [ 0  0  0  0  1 45  0  1  0  3]
 [ 0  1  0  0  0  0 36  0  0  0]
 [ 0  0  0  0  0  0  0 53  0  0]
 [ 0  4  0  0  0  0  0  0 47  1]
 [ 0  1  0  0  0  1  0  4  3 31]]
```



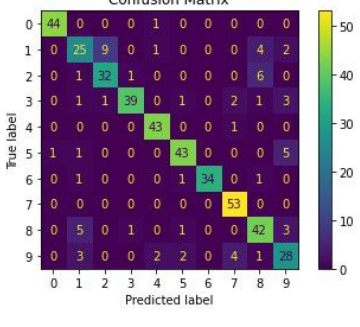

```
0.8511111111111112 BernoulliNB()
```

Prediction: 8 Prediction: 9 Prediction: 7 Prediction: 5 Prediction: 8 Prediction: 0 Prediction: 7 Prediction: 0 Prediction: 5 Prediction: 2

```
0.8511111111111112 BernoulliNB()
```




```
Classification report for classifier BernoulliNB():
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        45
           1       0.68      0.61      0.64        41
           2       0.76      0.80      0.78        40
           3       0.95      0.81      0.88        48
           4       0.91      0.98      0.95        44
           5       0.90      0.86      0.88        50
           6       1.00      0.92      0.96        37
           7       0.88      1.00      0.94        53
           8       0.76      0.81      0.79        52
           9       0.68      0.70      0.69        40

    accuracy                           0.85       450
   macro avg       0.85      0.85      0.85       450
weighted avg       0.85      0.85      0.85       450


Confusion matrix:
[[44  0  0  0  1  0  0  0  0  0]
 [ 0 25  9  0  1  0  0  0  4  2]
 [ 0  1 32  1  0  0  0  0  6  0]
 [ 0  1  1 39  0  1  0  2  1  3]
 [ 0  0  0  0 43  0  0  1  0  0]
 [ 1  1  0  0  0 43  0  0  0  5]
 [ 0  1  0  0  0  1 34  0  1  0]
 [ 0  0  0  0  0  0  0 53  0  0]
 [ 0  5  0  1  0  1  0  0 42  3]
 [ 0  3  0  0  2  2  0  4  1 28]]
```



In [16]:

```python
# Calculate all the averaged accuracy values for all classifiers used
from sklearn.metrics import precision_score, accuracy_score, recall_score

for clf in [svc, lr, gnb, mnb, bnb]:
    # Split data into 75% train and 25% test subsets
    X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.25, shuffle=True)
    average_accuracy_scores = np.empty(10)
    for i in range(10):
        y_pred = clf.fit(X_train, y_train).predict(X_test)
        average_accuracy_scores[i] = accuracy_score(y_pred,y_test)
    print(f"{clf}:{np.mean(average_accuracy_scores)}\n")
```

```
SVC():0.9822222222222223

LogisticRegression(max_iter=5000):0.9577777777777777

GaussianNB():0.8644444444444443

MultinomialNB():0.9022222222222224

BernoulliNB():0.8488888888888889
```