In [1]:
```
import pandas as pd
import numpy as np

from sklearn.datasets import fetch_openml
import matplotlib.pyplot as plt #used to display the data

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
# import picklewith
# open('mnist.pickle',"rb") as f:mnist=pickle.load(f)
```

In [2]:
```
mnist = fetch_openml("mnist_784", as_frame=False)
```

In [3]:
```
print(mnist.keys())
```

```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names', 'target_name
s', 'DESCR', 'details', 'url'])
```

In [4]:
```
display(mnist.data)
```

```
array([[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., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [5]:
```
print(mnist.data.shape)
```

```
(70000, 784)
```

In [6]:
```
display(mnist.target)
```

```
array(['5', '0', '4', ..., '4', '5', '6'], dtype=object)
```

In [7]:
```
for idx, image_data in enumerate(mnist.data[:100]):
    plt.subplot(10, 10, idx + 1)
    plt.imshow(image_data.reshape(28, 28), cmap="binary") #cmap="binary": This spec
    plt.axis("off")

plt.show()
```

In [8]:
```
# display(mnist.target[:10])
# # Now, just prints first 10 labels from data set target property to check matchin
```

```
array(['5', '0', '4', '1', '9', '2', '1', '3', '1', '4'], dtype=object)
```

data preparation

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(
    mnist.data, mnist.target, test_size=15, stratify=mnist.target, random_state=42)
```

In [10]:
```
print("X_train: {}, y_train: {}".format(X_train.shape, y_train.shape))
print("X_test: {}, y_test: {}".format(X_test.shape, y_test.shape))
```

```
X_train: (59500, 784), y_train: (59500,)
X_test: (10500, 784), y_test: (10500,)
```

Modeling

In [11]:
```
# Initializes Random Forest classifier (with default hyperparameters)
rf_clf = RandomForestClassifier(
    n_estimators=100, max_leaf_nodes=16, n_jobs=-1)
#n_job=-1 means
#means that the computation will be done using all available processors.
#It can speed up the training process for large datasets
```

In [12]:
```
# Measuring accuracy over Cross-Validation

# NOTE: This step may take several minutes to complete
rf_clf_cv_accuracy = cross_val_score(
    rf_clf, X_train, y_train, scoring="accuracy", cv=5)
```

In [13]:
```
print("Random Forest CV Error: {} (mean) [Standard Deviation (STD): {}]".format(
    np.mean(rf_clf_cv_accuracy), np.std(rf_clf_cv_accuracy)))
```

```
Random Forest CV Error: 0.8178991596638655 (mean) [Standard Deviation (STD): 0.002
6403142032478187]
```

Implementing Ensembling with Bagging & DecisionTree Classifiers

In [14]:
```
# Initializes Bagging Classifier with Decision Tree as base model
bag_clf_50 = BaggingClassifier(
    DecisionTreeClassifier(splitter="random", max_leaf_nodes=16),
    n_estimators=50,
    n_jobs=-1)
```

In [15]:
```
bag_clf_50_cv_accuracy = cross_val_score(
    bag_clf_50, X_train, y_train, scoring="accuracy", cv=5)
```

In [16]:
```
print("Bagging Classifier (with 50 estimators) CV Error: {} (mean) [Standard Deviat
    np.mean(bag_clf_50_cv_accuracy), np.std(bag_clf_50_cv_accuracy)))
```

```
Bagging Classifier (with 50 estimators) CV Error: 0.7238991596638655 (mean) [Stand
ard Deviation (STD): 0.006356763778152658]
```

In [17]:
```
# Initializes Bagging Classifier with Decision Tree with 100 estimators as base mod
bag_clf_100 = BaggingClassifier(
    DecisionTreeClassifier(splitter="random", max_leaf_nodes=16),
    n_estimators=100,
    n_jobs=-1)
```

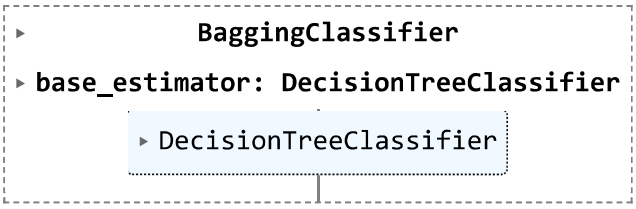
In [18]:
```
bag_clf_100_cv_accuracy = cross_val_score(
    bag_clf_100, X_train, y_train, scoring="accuracy", cv=5)
```

In [19]:
```
print("Bagging Classifier (with 100 estimators) CV Error: {} (mean) [Standard Devia
    np.mean(bag_clf_100_cv_accuracy), np.std(bag_clf_100_cv_accuracy)))
```

```
Bagging Classifier (with 100 estimators) CV Error: 0.7283697478991595 (mean) [Stan
dard Deviation (STD): 0.004230689774135436]
```

In [20]:
```
# Now, with this Bagging Classifier with 100 estimators, let's build model on full
bag_clf_100.fit(X_train, y_train)
```

Out[20]:



In [21]:
```
test_predictions = bag_clf_100.predict(X_test)
```

In [22]:
```
print("Accuracy on Test Data:", accuracy_score(y_test, test_predictions))
```

```
Accuracy on Test Data: 0.7293333333333333
```

```
In [24]: ?cross_val_score
```

viva questions Cross-validation is a technique in machine learning that splits the data into multiple subsets to assess model performance, helping to evaluate and select the best model.