Random Forest

Random Forest is an ensemble machine learning algorithm for classification and regression. It works by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. This combination of decision trees reduces overfitting compared to a single decision tree and results in improved accuracy.

In a Random Forest, the Gini index or entropy is used as the criterion to split the data at each node of each individual decision tree. The main idea is to find the split that results in the purest child nodes, as defined by the Gini index or entropy, in order to maximize the separation of classes and improve the accuracy of the model.

Once the individual trees are built, the Random Forest combines their predictions by taking a majority vote for classification or averaging for regression. The choice between using the Gini index or entropy as the split criterion does not affect the overall operation of the Random Forest, but can impact the accuracy and performance of the individual trees, and therefore the final model.

In general, the choice of split criterion, such as Gini index or entropy, is a matter of personal preference, and often depends on the specific problem and dataset being analyzed. It is common to experiment with both to determine which provides the best results for a given problem.

Gradient Boosting

Gradient Boosting is an ensemble machine learning algorithm that is used for both classification and regression problems. It builds a model by combining the predictions of multiple weaker models, usually decision trees.

Gradient Boosting works by fitting weak learners (e.g. decision trees) to the negative gradient of the loss function that is to be minimized (e.g. mean squared error for regression, cross-entropy for classification). Each iteration focuses on correcting the mistakes of the previous weak learner, with the final model being the weighted sum of all the weak learners.

Gradient Boosting is a powerful machine learning algorithm that has shown to be highly effective in many real-world applications, and has received a lot of attention in the machine learning community for its accuracy and performance. However, it can also be computationally expensive and prone to overfitting, so careful tuning of the model hyperparameters is often necessary to achieve good performance.

XGBoost classifier

XGBoost (eXtreme Gradient Boosting) is an open-source gradient boosting algorithm for machine learning and is specifically designed for large-scale, scalable and efficient implementation of gradient boosting trees. XGBoost is an optimized implementation of gradient boosting that is well known for its speed, accuracy and performance.

In XGBoost, gradient boosting is used to iteratively add trees to the model to improve its accuracy, with the prediction of each tree being added to the final prediction. XGBoost also includes a number of other techniques, such as regularization, pruning, and handling of missing values, which help to improve the performance and accuracy of the model.

XGBoost has been widely used in a variety of machine learning tasks, including classification and regression problems, and has shown to be a very effective model for many real-world datasets.

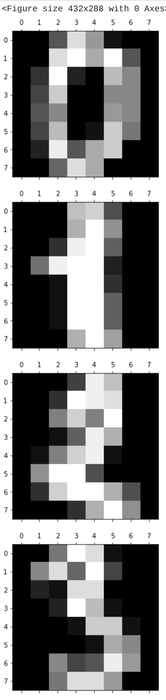
In [1]:
```
import pandas as pd
from sklearn.datasets import load_digits
```

In [2]:
```
digits = load_digits()
```

In [3]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
plt.gray()
for i in range(4):
    plt.matshow(digits.images[i])
```

```
<Figure size 432x288 with 0 Axes>
```



In [5]:
```
df = pd.DataFrame(digits.data)
```

In [6]:
```
df['target'] = digits.target
```

In [7]:
```
df
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	60	61	62	63	target
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	13.0	10.0	0.0	0.0	0.0	0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	11.0	16.0	10.0	0.0	0.0	1
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	11.0	16.0	9.0	0.0	2
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	7.0	13.0	13.0	9.0	0.0	0.0	3
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	16.0	4.0	0.0	0.0	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1792	0.0	0.0	4.0	10.0	13.0	6.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	2.0	14.0	15.0	9.0	0.0	0.0	9
1793	0.0	0.0	6.0	16.0	13.0	11.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	16.0	14.0	6.0	0.0	0.0	0
1794	0.0	0.0	1.0	11.0	15.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	2.0	9.0	13.0	6.0	0.0	0.0	8
1795	0.0	0.0	2.0	10.0	7.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	5.0	12.0	16.0	12.0	0.0	0.0	9
1796	0.0	0.0	10.0	14.0	8.0	1.0	0.0	0.0	0.0	2.0	...	0.0	0.0	1.0	8.0	12.0	14.0	12.0	1.0	0.0	8

1797 rows × 65 columns

In [8]:
```
from sklearn.model_selection import train_test_split
```

In [9]:
```
x_train, x_test, y_train, y_test = train_test_split(df.drop(['target'], axis='columns'), digits.target, test_size=0.2)
```

In [10]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [11]:
```
model = RandomForestClassifier()
```

In [12]:
```
model.fit(x_train, y_train)
```

Out[12]:
```
RandomForestClassifier()
```

In [13]:
```
model.score(x_test, y_test)
```

Out[13]:
```
0.977777777777777
```

In [ ]:
```

```

In [15]:
```
from xgboost import XGBClassifier
model = XGBClassifier()
```

In [16]:
```
model = XGBClassifier()
```

In [17]:
```
model.fit(x_train, y_train)
```

Out[17]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              objective='multi:softprob', predictor=None, ...)
```

In [18]:
```
model.score(x_test, y_test)
```

Out[18]:
```
0.9583333333333334
```

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100)
model.fit(x_train, y_train)
```

Out[19]:
```
RandomForestClassifier()
```

In [20]:
```
model.score(x_test, y_test)
```

Out[20]:
```
0.975
```

In [21]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [22]:
```
model = GradientBoostingClassifier()
model.fit(x_train, y_train)
```

Out[22]:
```
GradientBoostingClassifier()
```

In [23]:
```
model.score(x_test, y_test)
```

Out[23]:
```
0.9638888888888889
```

In [ ]:
```

```