Dummy Classifier

In [11]:

```
%matplotlib notebook
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_digits
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dataset = load_digits()
X, y = dataset.data, dataset.target

for class_name, class_count in zip(dataset.target_names, np.bincount(dataset.target)):
    print(class_name, class_count)
```

```
0 178
1 182
2 177
3 183
4 181
5 182
6 181
7 179
8 174
9 180
```

In [4]:

```
y_binary_imbalanced = y.copy()
y_binary_imbalanced[y_binary_imbalanced != 1] = 0
#1일때만 1 출력
print(y[1:30])
print(y_binary_imbalanced[1:30])
```

```
[1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9]
[1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
```

In [6]:

```
np.bincount(y_binary_imbalanced)
```

Out[6]:

```
array([1615,  182], dtype=int64)
```

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y_binary_imbalanced, random_state = 0)

clf = LogisticRegression().fit(X_train, y_train)
clf.score(X_test, y_test)
```

```
C:₩Users₩Eunchae₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.py:762:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[13]:

```
0.9688888888888889
```

In [14]:

```
from sklearn.dummy import DummyClassifier

dummy = DummyClassifier(strategy = 'most_frequent').fit(X_train, y_train)
y_dummy_predictioins = dummy.predict(X_test)
```

In [15]:

```
y_dummy_predictioins
```

Out[15]:

```
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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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,
       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, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [16]:

```
dummy.score(X_test, y_test)
```

Out[16]:

0.9044444444444445

Confusion matrix

In [17]:

```
from sklearn.metrics import confusion_matrix

y_majority_predicted = dummy.predict(X_test)
confusion = confusion_matrix(y_test, y_majority_predicted)
```

In [18]:

```
print(confusion)
```

```
[[407   0]
 [ 43   0]]
```

In [19]:

```
y_logreg_predicted = clf.predict(X_test)
confusion_logreg = confusion_matrix(y_test, y_logreg_predicted)
print(confusion_logreg)
```

```
[[401   6]
 [  8  35]]
```

In [23]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print('Dummy Classifier')
print('Accuracy: {:.2f}'.format(accuracy_score(y_test, y_majority_predicted)))
print('Precision: {:.2f}'.format(precision_score(y_test, y_majority_predicted)))
print('Recall: {:.2f}'.format(recall_score(y_test, y_majority_predicted)))

print('Logistic Regression based Classifier')
print('Accuracy: {:.2f}'.format(accuracy_score(y_test, y_logreg_predicted)))
print('Precision: {:.2f}'.format(precision_score(y_test, y_logreg_predicted)))
print('Recall: {:.2f}'.format(recall_score(y_test, y_logreg_predicted)))
```

```
Dummy Classifier
Accuracy: 0.90
Precision: 0.00
Recall: 0.00
Logistic Regression based Classifier
Accuracy: 0.97
Precision: 0.85
Recall: 0.81
```