In [3]:

```
##Dummy classifier
```

In [4]:

```
from sklearn.datasets import load_digits
import numpy as np
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 [5]:

```
y_binary_imbalanced=y.copy()
y_binary_imbalanced[y_binary_imbalanced!=1]=0
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]:

```
X[1:30]
```

Out[6]:

```
array([[ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       [ 0.,  0.,  7., ...,  9.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  1.,  0.,  0.],
       [ 0.,  0., 10., ...,  1.,  0.,  0.],
       [ 0.,  0.,  9., ..., 12., 11.,  0.]])
```

In [7]:

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

Out[7]:

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

In [11]:

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y_binary_imbalanced,
                                                  random_state=0)
svm=SVC().fit(X_train,y_train)
svm.score(X_test,y_test)
```

Out[11]:

0.9955555555555555

In [13]:

```
from sklearn.linear_model import LogisticRegression

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₩donghyunkim₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.p
y:940: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[13]:

0.9688888888888889

In [15]:

```
from sklearn.dummy import DummyClassifier

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

In [16]:

```
y_dummy_predictions
```

Out[16]:

```
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 [17]:

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

Out[17]:

```
0.9044444444444445
```

In [18]:

```
##Confusion Matrix
```

In [20]:

```
from sklearn.metrics import confusion_matrix

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

In [21]:

```
print(confusion)
```

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

In [22]:

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

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

In [26]:

```
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

C:₩Users₩donghyunkim₩anaconda3₩lib₩site-packages₩sklearn₩metrics₩_classification.p
y:1272: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due
to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

```

```