SGDClassificationDataset

March 27, 2023

[1]:
```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1)
mnist.keys()
```

[1]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'details', 'categories',
'url'])
```

[4]:
```
mnist['data'][:5]
```

[4]:
```
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.]])
```

[5]:
```
X,y = mnist["data"],mnist["target"]
```

[6]:
```
X.shape
```

[6]:
```
(70000, 784)
```

[7]:
```
y.shape
```

[7]:
```
(70000,)
```

[8]:
```
#Declare Library plotting
%matplotlib inline
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
```

[12]:
```
#Generate one example data
some_digit = X[0]
len(some_digit) #total features fron 28x28
```

[12]:
```
784
```

[14]:
```
#reshape it become 28x28
some_digit_image = some_digit.reshape(28,28)
```

[16]:
```
plt.imshow(some_digit_image,cmap=mpl.cm.binary)
plt.axis("off")
plt.show()
```

[17]:
```
y[0]
```

[17]:
```
'5'
```

[18]:
```
type(y[0])
```

[18]:
```
str
```

[20]:
```
y = y.astype(np.uint8)
type(y[0])
```

[20]:
```
numpy.uint8
```

[21]:
```
X_train, X_test,y_train,y_test = X[:60000],X[60000:],y[:60000],y[60000:]
```

[23]:
```
X_train
```

[23]:
```
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.]])
```

[24]:
```
y_train
```

[24]:
```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

[25]:
```
#Untuk sementara kita akan melakukan simplifikasi masalah dengan hanya␣
 ↪mempertimbangkan satu angka saja, misalkan nomor 5.
#Sistem ini dirancang hanya untuk membedakan angka 5 dan bukan 5. Sehingga kita␣
 ↪akan melakukan modifikasi dari vektor target menjadi 2 class,
#yaitu class angka 5 dan bukan class angka lima dengan cara sebagai berikut:
```

[29]:
```
y_train_5 = (y_train == 5)
y_train_5[0:2]
```

[29]:
```
array([ True, False])
```

[28]:
```
y_test_5 = (y_test == 5)
y_test_5[0:2]
```

[28]:
```
array([False, False])
```

[31]:
```
#SGD model
from sklearn.linear_model import SGDClassifier
```

[32]:
```
sgd_clf = SGDClassifier(max_iter=1000, tol=0.0001, random_state=42)
```

[33]:
```
sgd_clf.fit(X_train,y_train_5)
```

[33]:
```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',
              max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=42, shuffle=True, tol=0.0001,
              validation_fraction=0.1, verbose=0, warm_start=False)
```

[34]:
```
#test previose data into the model
sgd_clf.predict([some_digit])
```

[34]:
```
array([ True])
```

[35]:
```
some_data_test = X_test[0]
```

[38]:
```
label = y_test_5[0]
label
```

[38]:
```
False
```

[41]:
```
#test data test
sgd_clf.predict([some_data_test]) #hasil prediksi benar
```

[41]:
```
array([False])
```

[43]:
```
y_prediksi = sgd_clf.predict(X_test)
```

[46]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test_5,y_prediksi) #nilai accuracy sama dengan 94%
```

[46]:
```
0.9492
```

[47]:
```
#cross validation prediction
from sklearn.model_selection import cross_val_predict
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

[48]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_train_5, y_train_pred)
#Setiap baris pada confusion matrix menyatakan actual class, sementara setiap␣
 ↪kolom menyatakan predicted class.
```

[48]:
```
array([[53892,   687],
       [ 1891,  3530]])
```

[49]:
```
from sklearn.metrics import precision_score, recall_score
precision_score(y_train_5, y_train_pred)
#Precision score 83 %
```

[49]:
```
0.8370879772350012
```

[50]:
```
recall_score(y_train_5, y_train_pred)
#recall score 65%
```

[50]:
```
0.6511713705958311
```

[51]:
```
#f1 score semakin besar semakin besar juga nilai precision dan recall
from sklearn.metrics import f1_score
f1_score(y_train_5, y_train_pred)
```

[51]:
```
0.7325171197343846
```

[53]:
```
#Trade off antara Precision and recall using thresholds
y_scores = sgd_clf.decision_function([some_digit])#get decision function for␣
 ↪classification previous image "5"
y_scores
```

[53]:
```
array([2164.22030239])
```

[54]:
```
threshold = 0
y_some_digit_pred = (y_scores > threshold) #compare threshold with decision␣
 ↪score
#if y_score more than threshold than positive detection / clasify as 5
y_some_digit_pred # y_score (2164) >  threshold (0) positive detection
```

[54]:
```
array([ True])
```

[55]:
```
threshold = 8000
y_some_digit_pred = (y_scores > threshold) #compare threshold with decision␣
 ↪score
#if y_score more than threshold than positive detection / clasify as 5
y_some_digit_pred # y_score (2164) <  threshold (8000) negative detection/ not␣
 ↪"5" miss clasification
```

[55]:
```
array([False])
```

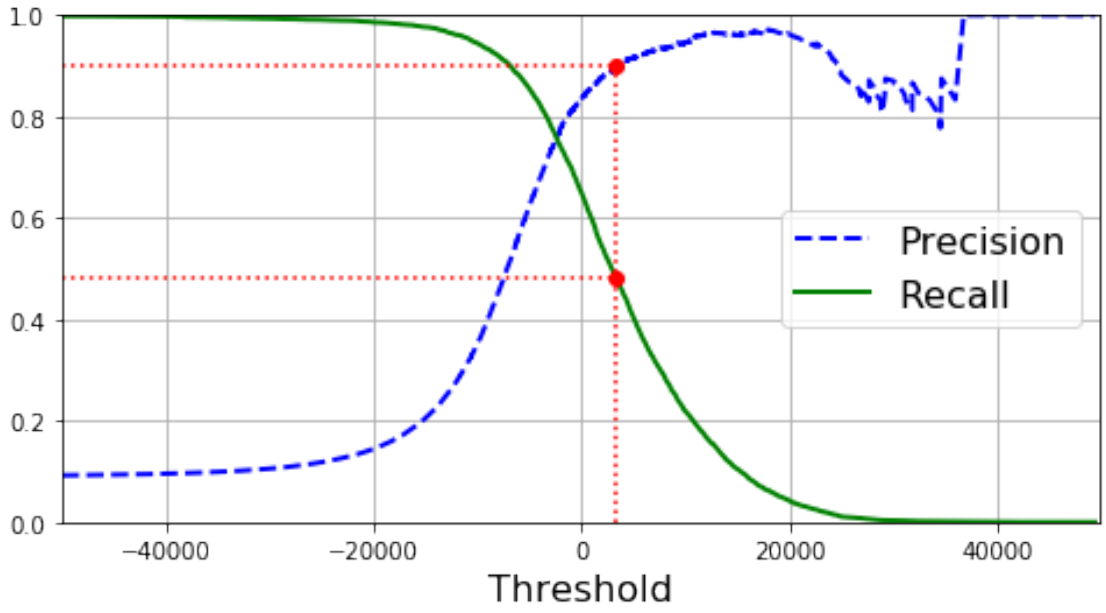
[56]:
```
 y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,
                              method="decision_function")
```

[57]:
```
from sklearn.metrics import precision_recall_curve
precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

[58]:
```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision", linewidth=2)
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall", linewidth=2)
    plt.legend(loc="center right", fontsize=16)
    plt.xlabel("Threshold", fontsize=16)
    plt.grid(True)
    plt.axis([-50000, 50000, 0, 1])
```

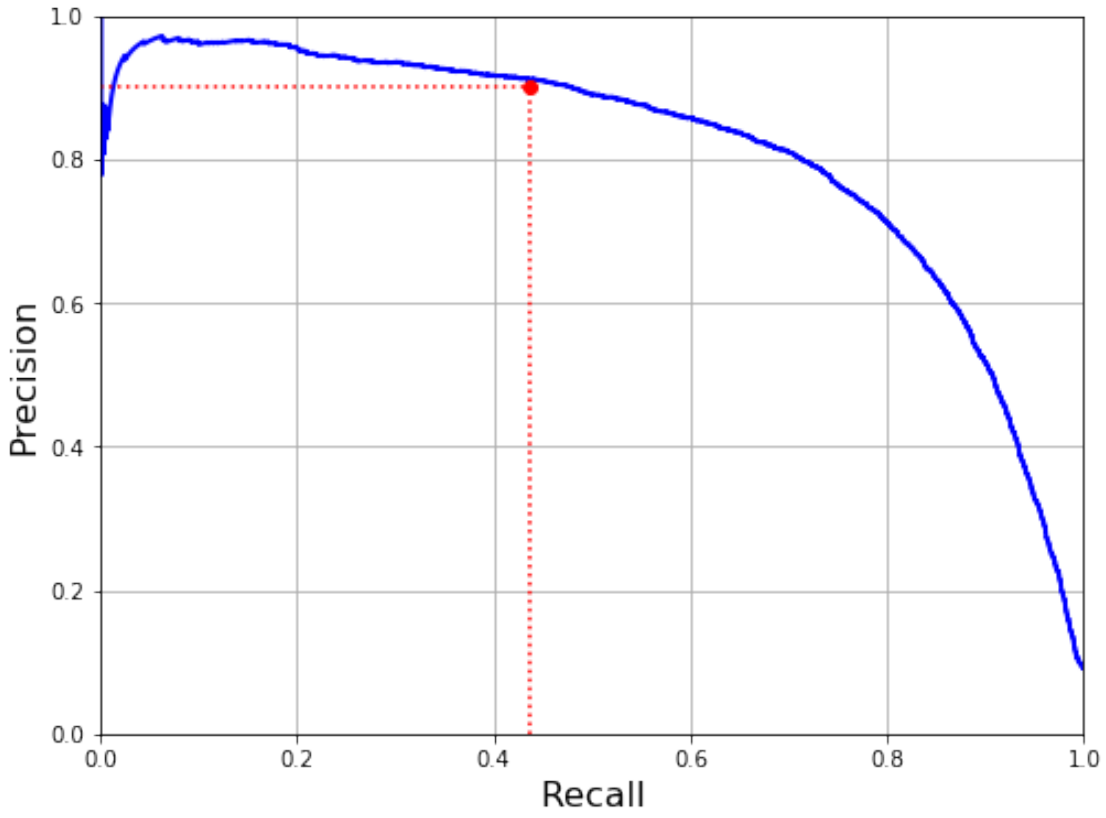
[59]:
```
recall_90_precision = recalls[np.argmax(precisions >= 0.90)]
threshold_90_precision = thresholds[np.argmax(precisions >= 0.90)]
```

[60]:
```
plt.figure(figsize=(8, 4))
plot_precision_recall_vs_threshold(precisions, recalls, thresholds)
plt.plot([threshold_90_precision, threshold_90_precision], [0., 0.9], "r:")
plt.plot([-50000, threshold_90_precision], [0.9, 0.9], "r:")
plt.plot([-50000, threshold_90_precision],␣
 ↪[recall_90_precision,recall_90_precision], "r:")
plt.plot([threshold_90_precision], [0.9], "ro")
plt.plot([threshold_90_precision], [recall_90_precision], "ro")
plt.show()
```



[61]:
```
def plot_precision_vs_recall(precisions, recalls):
    plt.plot(recalls, precisions, "b-", linewidth=2)
    plt.xlabel("Recall", fontsize=16)
    plt.ylabel("Precision", fontsize=16)
    plt.axis([0, 1, 0, 1])
    plt.grid(True)
```

[62]:
```
plt.figure(figsize=(8, 6))
plot_precision_vs_recall(precisions, recalls)
plt.plot([0.4368, 0.4368], [0., 0.9], "r:")
plt.plot([0.0, 0.4368], [0.9, 0.9], "r:")
plt.plot([0.4368], [0.9], "ro")
plt.show()
```



[63]:
```
threshold_90_precision = thresholds[np.argmax(precisions >= 0.90)]
threshold_90_precision
```

[63]:
```
3370.0194991439594
```

[64]:
```
y_train_pred_90 = (y_scores >= threshold_90_precision)
y_train_pred_90
```

[64]:
```
array([False, False, False, ...,  True, False, False])
```

[65]:
```
precision_score(y_train_5, y_train_pred_90)
```

[65]:
```
0.9000345901072293
```

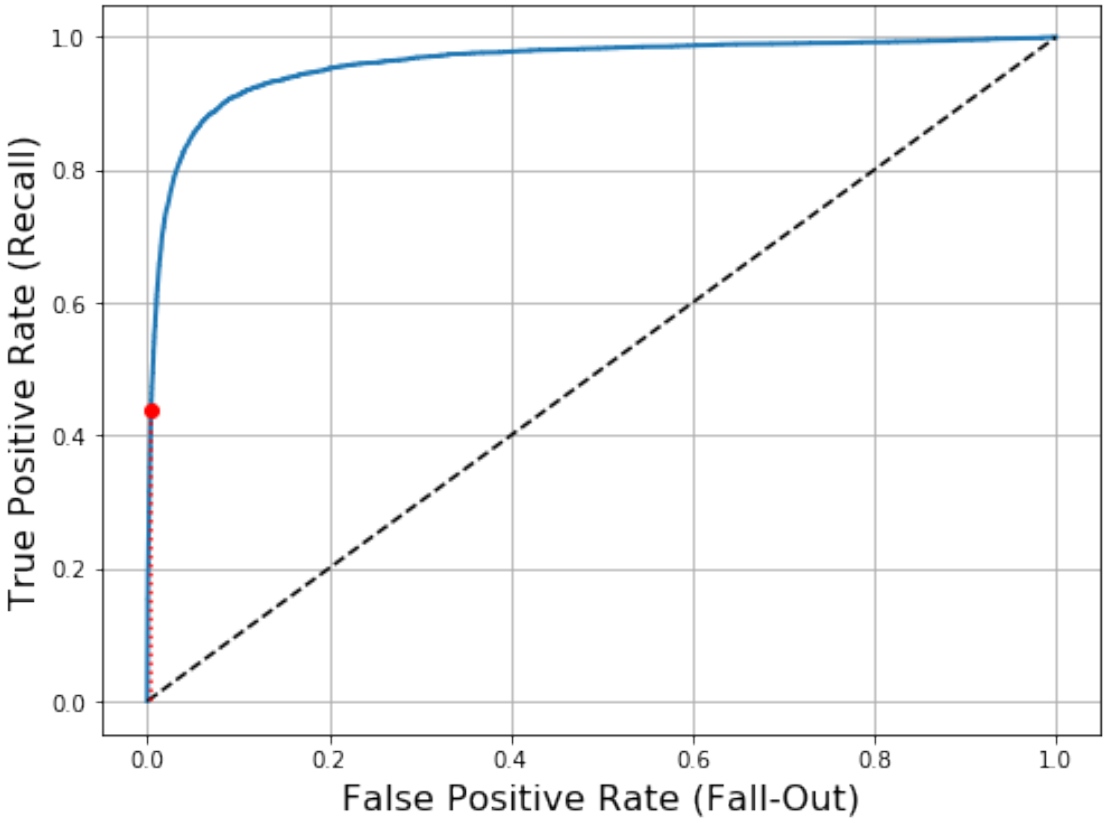
[66]:
```
recall_score(y_train_5, y_train_pred_90)
```

[66]:
```
0.4799852425751706
```

[67]:
```
#Receiver Operating Characterisitic (ROC)
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_train_5, y_scores)
```

[68]:
```
def plot_roc_curve(fpr, tpr, label=None):
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--') # dashed diagonal plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate (Fall-Out)', fontsize=16)
    plt.ylabel('True Positive Rate (Recall)', fontsize=16)
    plt.grid(True)
```

[69]:
```
plt.figure(figsize=(8, 6))
plot_roc_curve(fpr, tpr)
plt.plot([4.837e-3, 4.837e-3], [0., 0.4368], "r:")
plt.plot([0.0, 4.837e-3], [0.4368, 0.4368], "r:")
plt.plot([4.837e-3], [0.4368], "ro")
plt.show()
```



[70]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_train_5, y_scores)
```

[70]: 0.9604938554008616

[71]:
```
#performace yang bagus karena nilainya mendekati angka 1
```

[ ]: