Evaluation Method and Data Preprocessing

Evaluation Method

경희대학교 컴퓨터공학과
2019102191 신주영

경희대학교
KYUNG HEE UNIVERSITY

Table of Contents

경희대학교
KYUNG HEE UNIVERSITY

Ways to evaluate supervised learning

- The evaluation method of supervised learning varies depending on whether it is a classification problem or a regression problem.
- Even if evaluation indicators are appropriately selected, a good model cannot be created if they are overfitting the learning data. This should be noted.

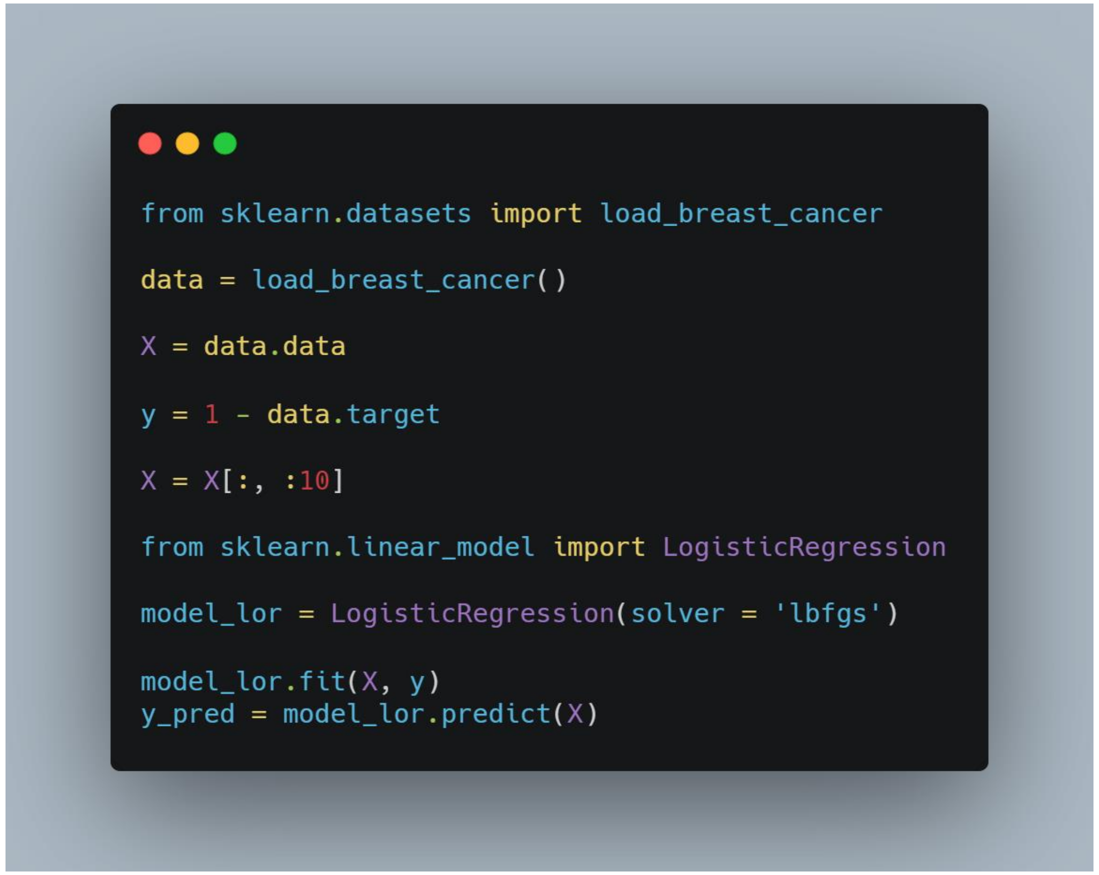
분류 문제	회귀 문제
혼동 행렬 (confusion matrix)	평균제곱오차 (mean square error)
정확도 (accuracy)	결정계수 (coefficient of determination)
정밀도 (precision)	
재현율 (recall)	
F값 (F1-score)	
곡선아래면적 (AUC)	

경희대학교
KYUNG HEE UNIVERSITY

How to evaluate classification problems

Data preprocessing

- We're going to use dataset as a breast cancer diagnosis data

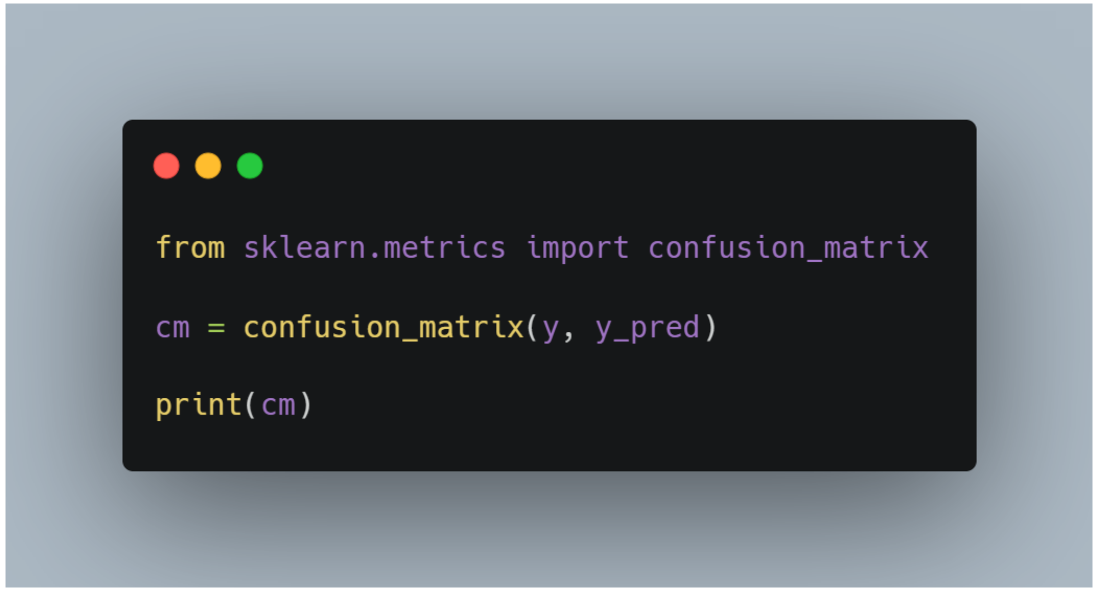


```python
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()

X = data.data

y = 1 - data.target

X = X[:, :10]

from sklearn.linear_model import LogisticRegression

model_lor = LogisticRegression(solver = 'lbfgs')

model_lor.fit(X, y)
y_pred = model_lor.predict(X)
```

How to evaluate classification problems

Confusion Matrix

- TN: Properly predict real negative data as negative
- FP: Misleading prediction of real negative data as positive
- FN: Misleading prediction of real positive data as negative
- TP: Real Positive Data Predicts Positive



```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y, y_pred)

print(cm)
```

	0	1
0	TN	FP
1	FN	TP

How to evaluate classification problems

accuracy

$$\frac{TP + TN}{TP + TN + FP + FN}$$

How to evaluate classification problems

precision

$$\frac{TP}{TP + FP}$$

How to evaluate classification problems

recall

$$\frac{TP}{TP + FN}$$

How to evaluate classification problems

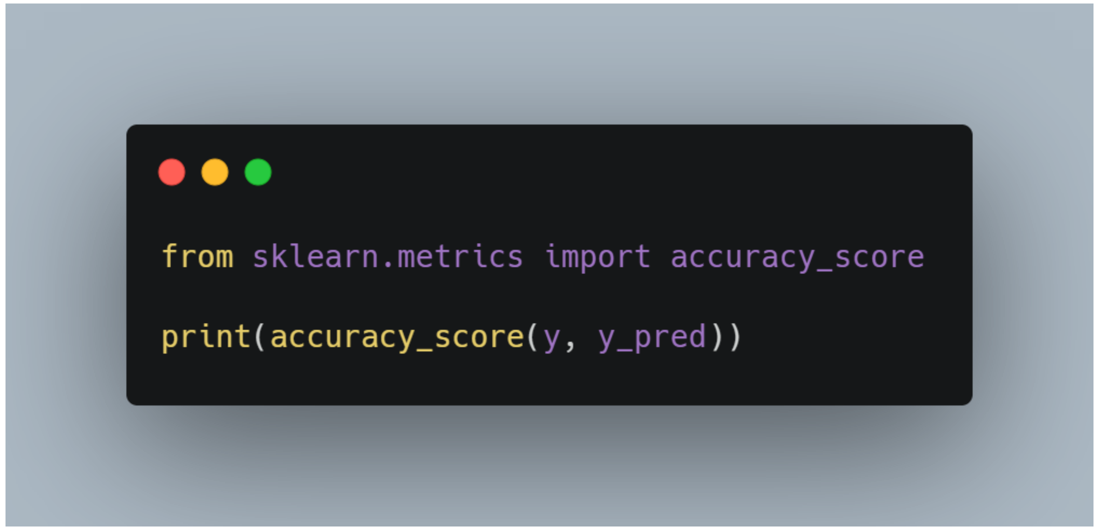
F1-score

$$\frac{(accuracy\ \times recall)}{(precision + recall)}$$

How to evaluate classification problems

accuracy

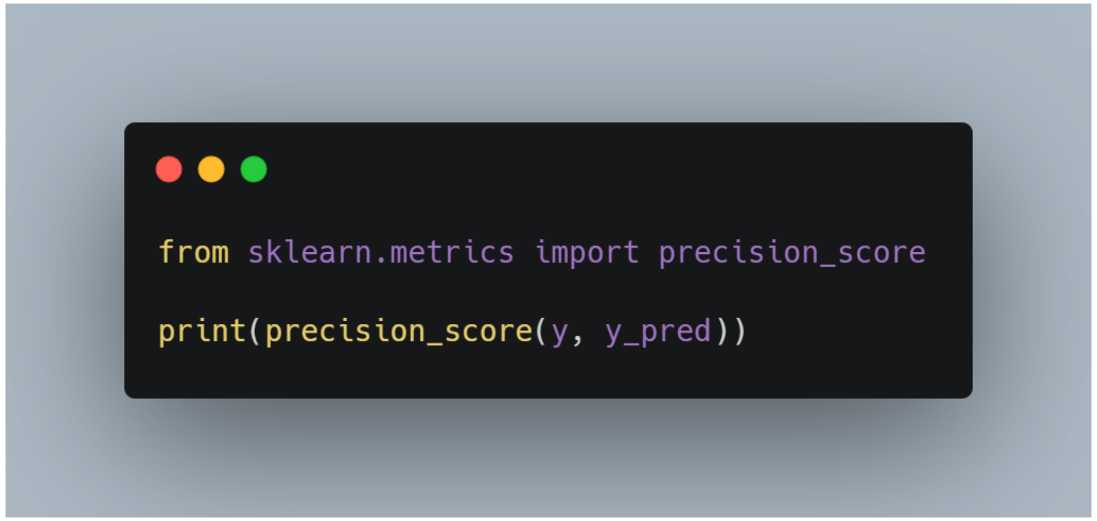
- Percentage of total forecast results correctly predicted



```
from sklearn.metrics import accuracy_score

print(accuracy_score(y, y_pred))
```

How to evaluate classification problems
precision

- The percentage of the actual positive results that you predicted was really positive

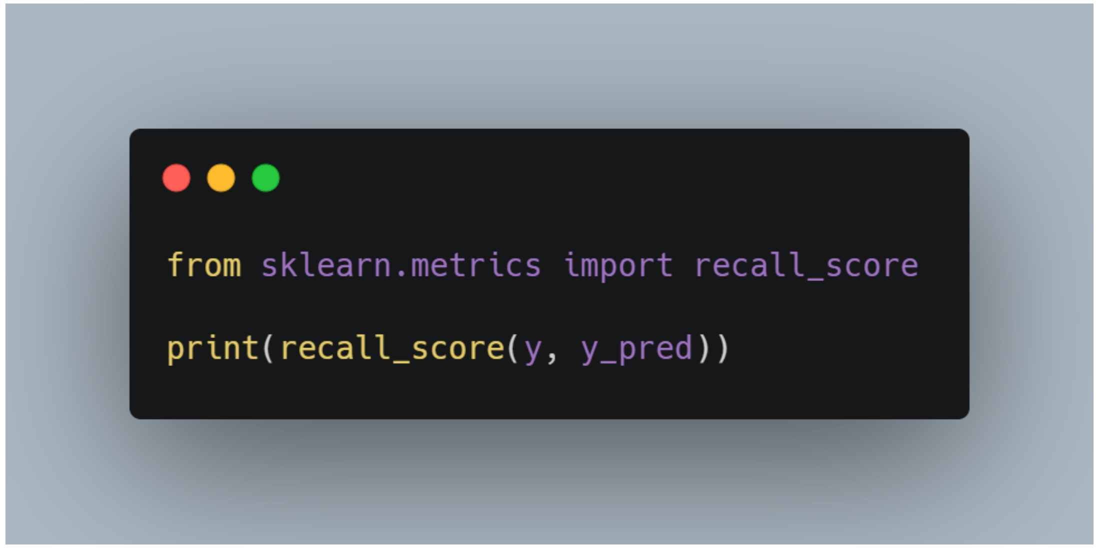


```
from sklearn.metrics import precision_score

print(precision_score(y, y_pred))
```

How to evaluate classification problems

recall

- Percentage correctly predicted as positive as actual positive



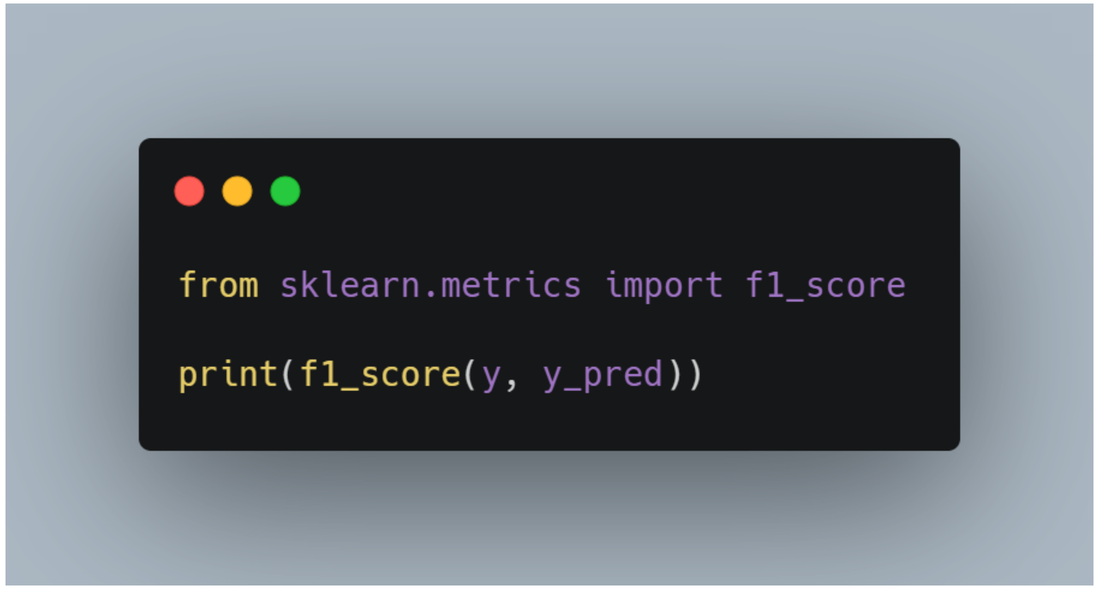
```python
from sklearn.metrics import recall_score

print(recall_score(y, y_pred))
```

경희대학교
KYUNG HEE UNIVERSITY

How to evaluate classification problems

F1-score

- reflecting both precision and recall

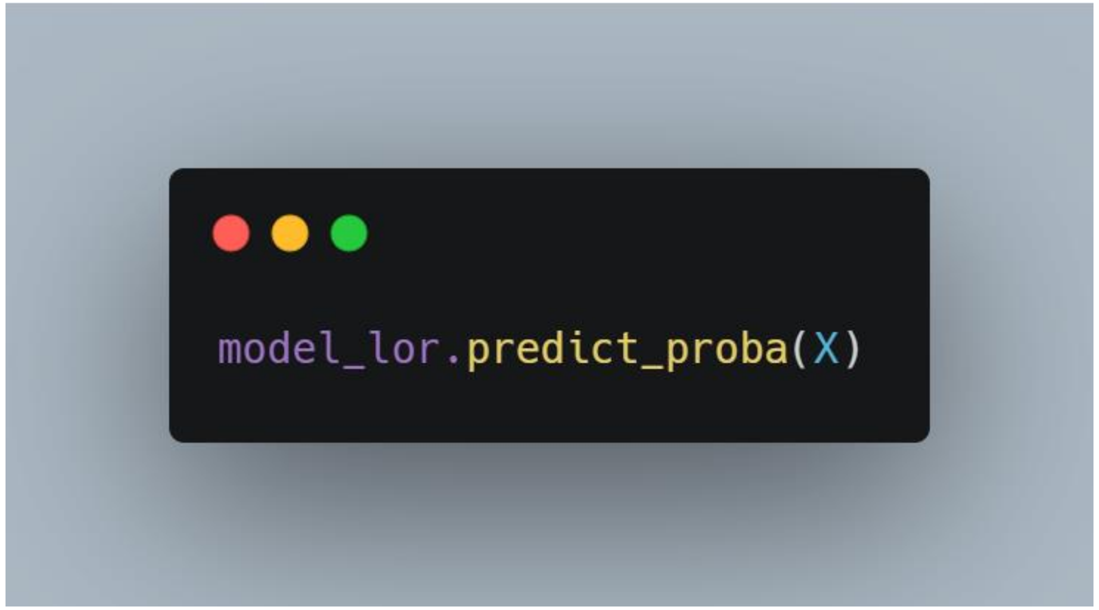


```
from sklearn.metrics import f1_score

print(f1_score(y, y_pred))
```

How to evaluate classification problems

prediction probability

- Calculate the probability of being classified as 0 and the probability of being classified as 1 respectively



```
model_lor.predict_proba(X)
```

How to evaluate classification problems

ROC curve and AUC

- The ratio of FP is set to the horizontal axis and the ratio of TP is set to the vertical axis and displayed as a graph.
- The change in the relationship between FP and TP can be seen graphically when the threshold is slightly lowered from 1 to be treated positive among the predicted probabilities.
- If the area is up to 1 and the area under the curve is close to 1, the precision is high. On the other hand, around 0.5 is not well predicted.
- **The use of ROC is recommended when dealing with unbalanced data.**

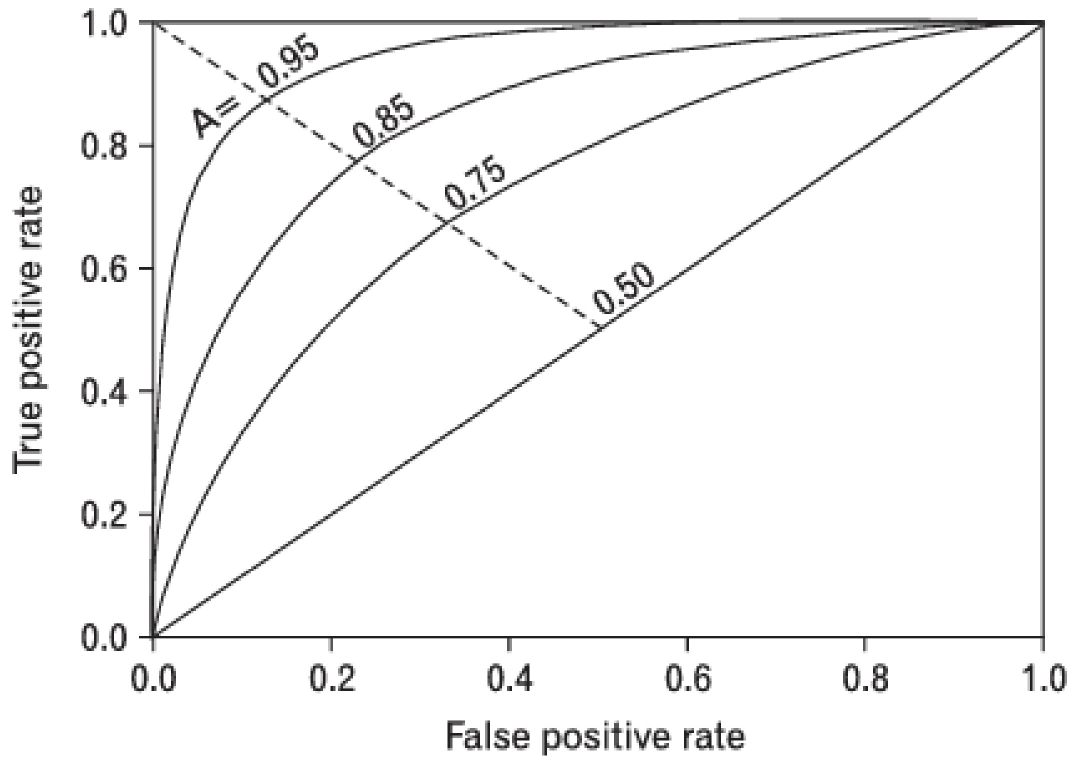
경희대학교
KYUNG HEE UNIVERSITY

How to evaluate classification problems

ROC curve and AUC

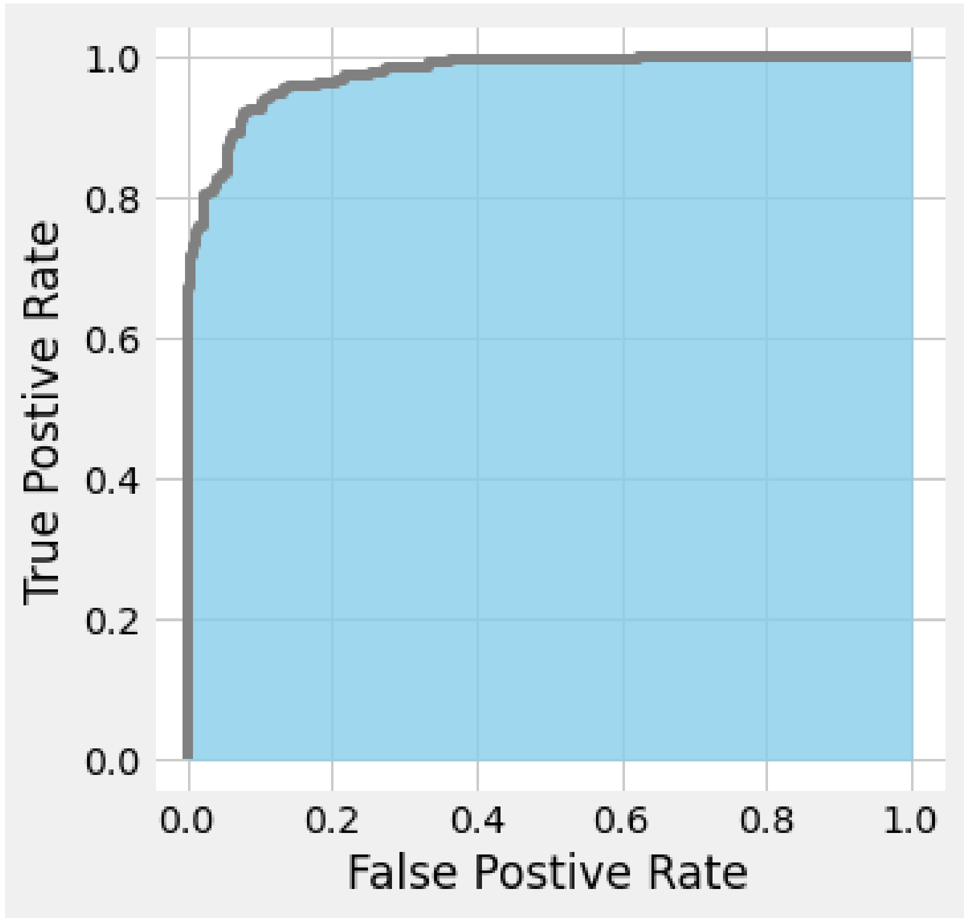
$$FPR = \frac{FP}{FP + TN}$$

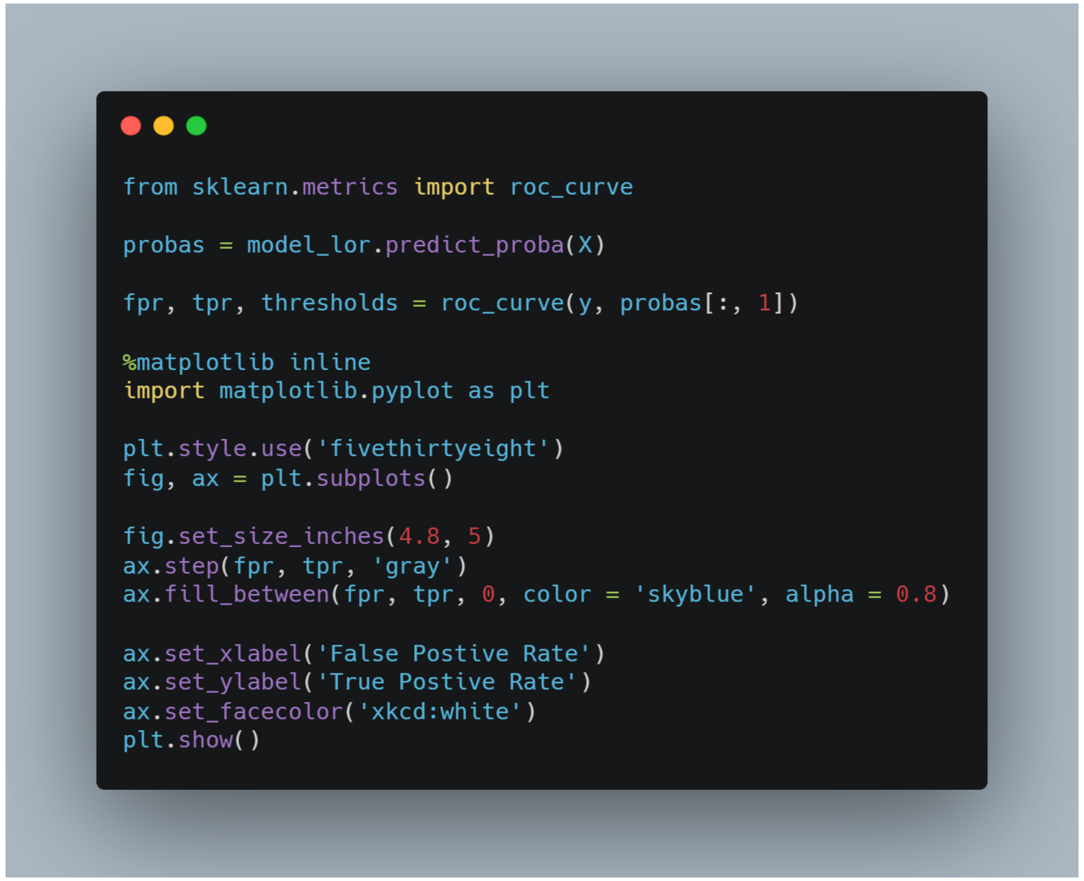
$$TPR = \frac{TP}{TP + FN}$$



How to evaluate classification problems

ROC curve and AUC

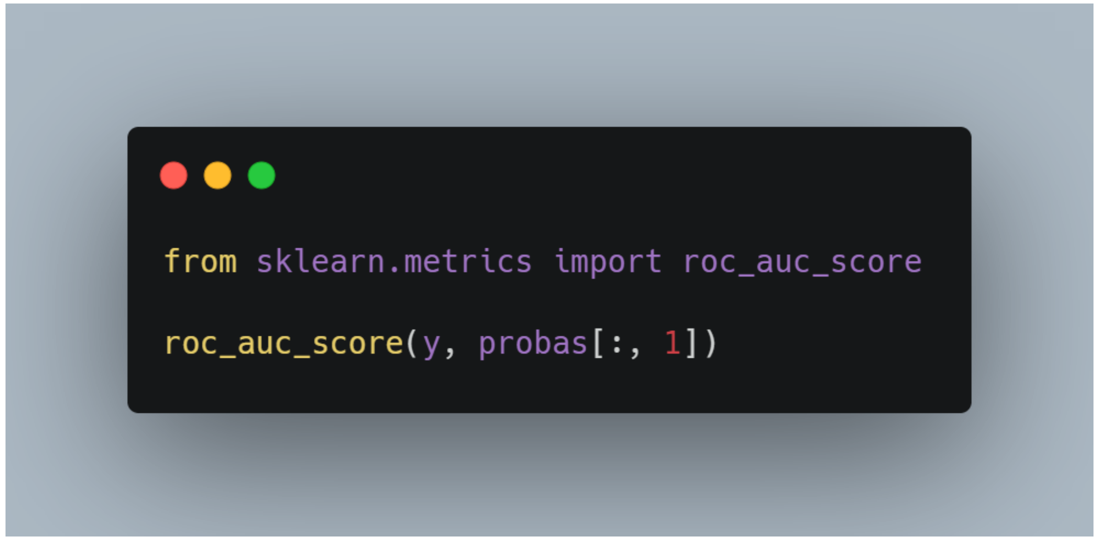




```
from sklearn.metrics import roc_curve

probas = model_lor.predict_proba(X)

fpr, tpr, thresholds = roc_curve(y, probas[:, 1])

%matplotlib inline
import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')
fig, ax = plt.subplots()

fig.set_size_inches(4.8, 5)
ax.step(fpr, tpr, 'gray')
ax.fill_between(fpr, tpr, 0, color = 'skyblue', alpha = 0.8)

ax.set_xlabel('False Postive Rate')
ax.set_ylabel('True Postive Rate')
ax.set_facecolor('xkcd:white')
plt.show()
```

How to evaluate classification problems

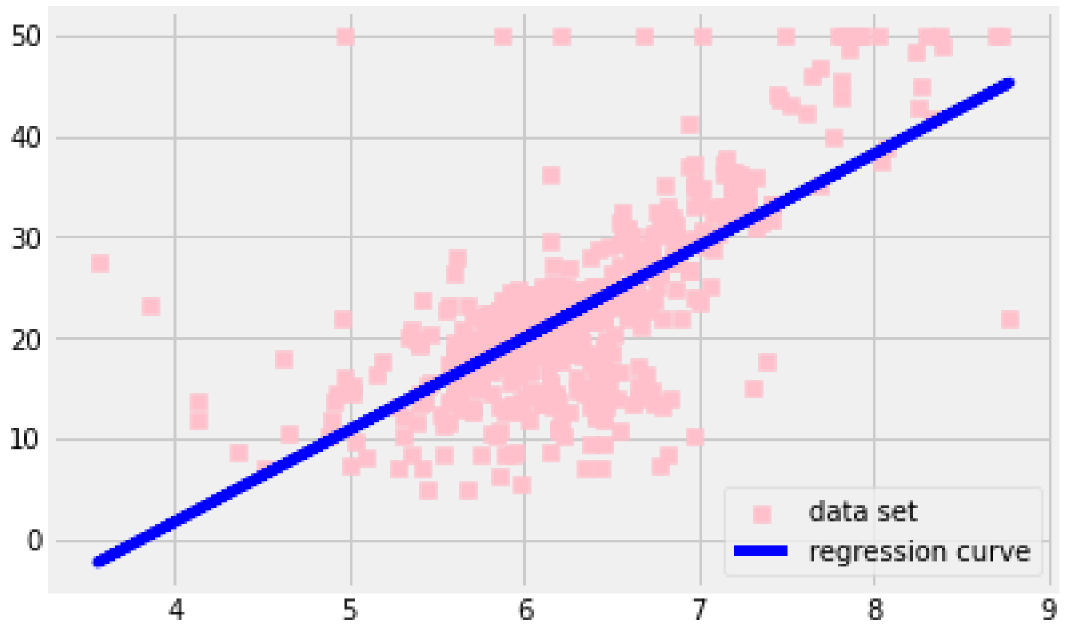
ROC curve and AUC

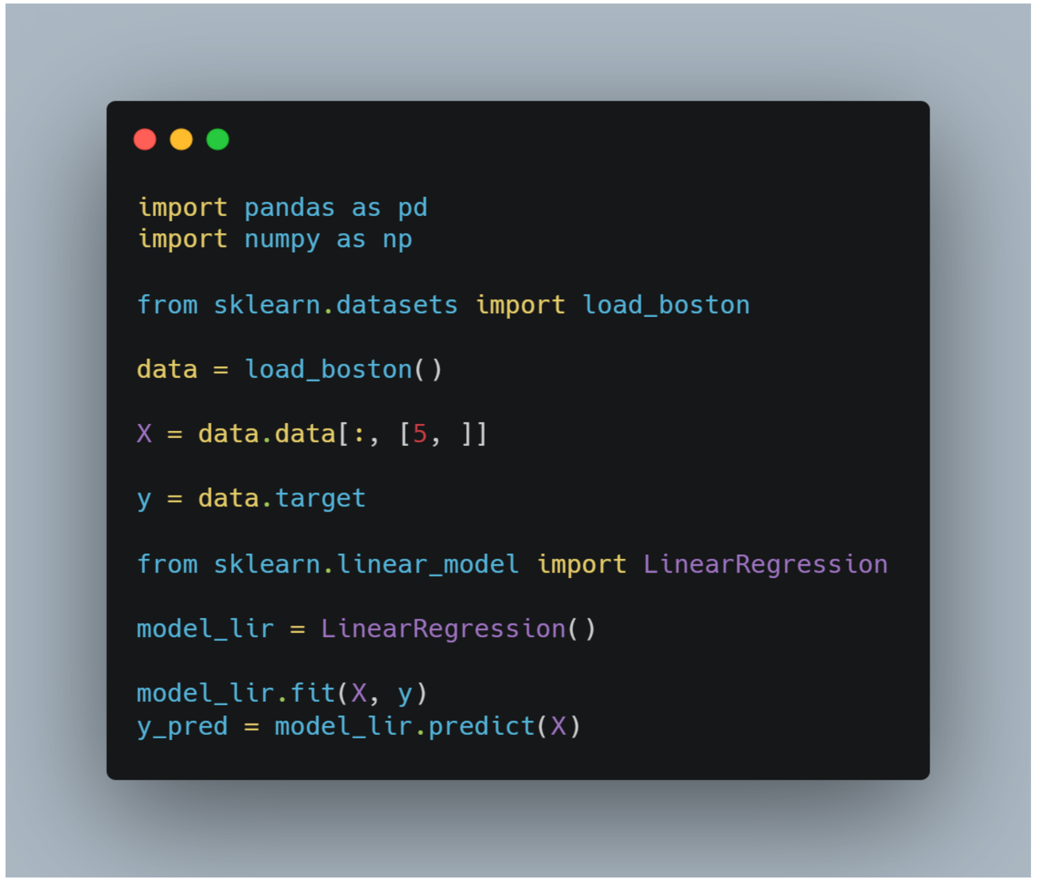


```
from sklearn.metrics import roc_auc_score

roc_auc_score(y, probas[:, 1])
```

```
Out[13]:  0.974155700158554
```

Evaluation method of Regression Problem





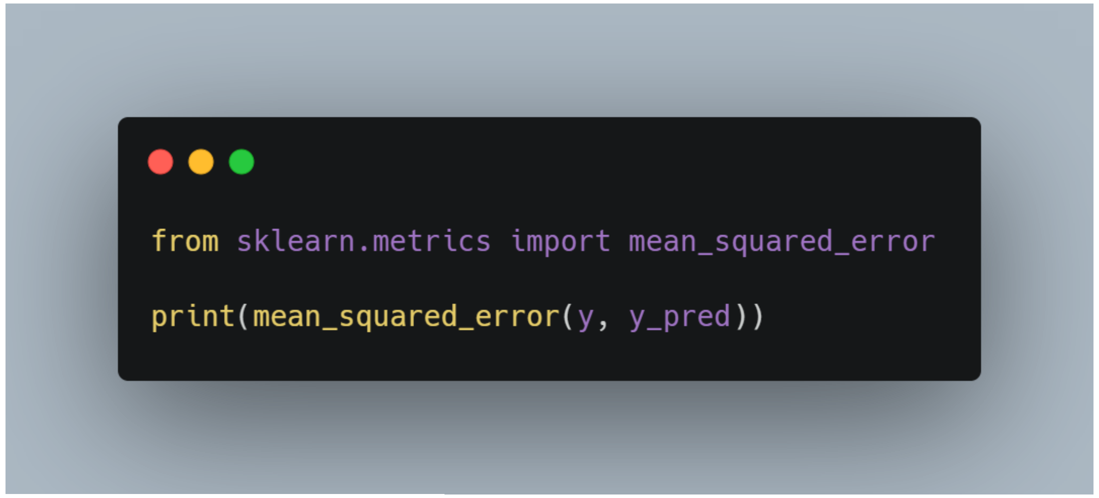
```
import pandas as pd
import numpy as np

from sklearn.datasets import load_boston

data = load_boston()

X = data.data[:, [5, ]]

y = data.target

from sklearn.linear_model import LinearRegression

model_lir = LinearRegression()

model_lir.fit(X, y)
y_pred = model_lir.predict(X)
```

경희대학교
KYUNG HEE UNIVERSITY

Evaluation method of Regression Problem

mean square error

- The mean square error is averaged after calculating both the square of error between the data to be evaluated and the predicted value.
- The smaller the mean square error, the more correct the prediction is.



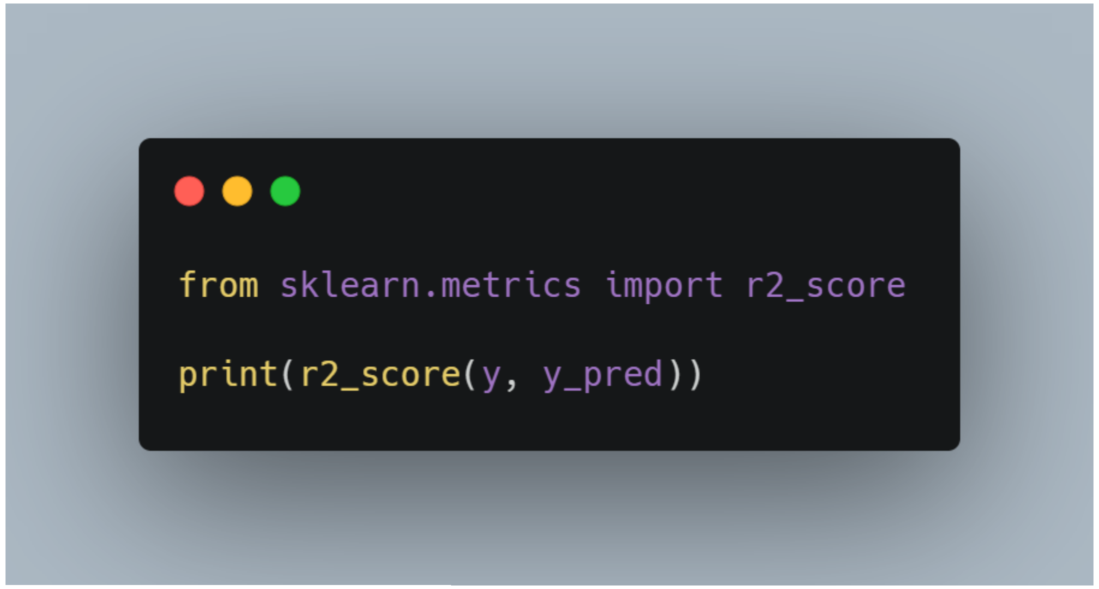
```
from sklearn.metrics import mean_squared_error

print(mean_squared_error(y, y_pred))
```

```
43.60055177116956
```

Evaluation method of Regression Problem

coefficient of determination

- It usually appears as a value between 0.0 and 1.0
- If the error between the predicted value and the actual data is too large, it may be expressed as a negative value
- The closer to 1.0, the more accurately the model represents the data point



```python
from sklearn.metrics import r2_score

print(r2_score(y, y_pred))
```

```
0.48352545599133423
```

Compared to other algorithms

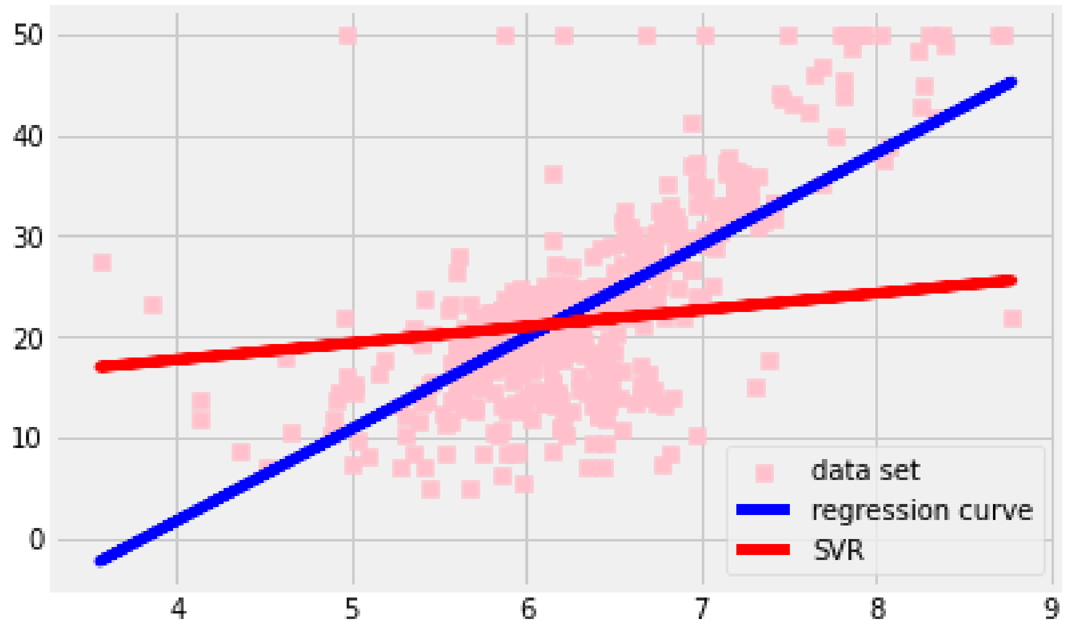
difference between mean square error and coefficient of determination

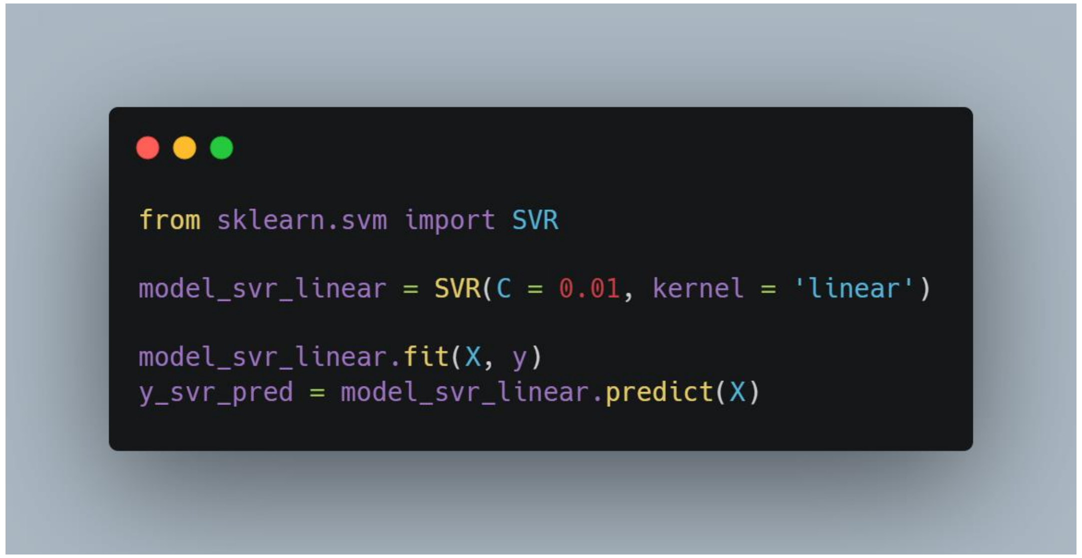
- If the variance of the dependent variable is large, the mean square error is also large.
- The coefficient of determination may be expressed as a value between 0.0 and 1.0 without depending on the variance of the dependent variable.

Compared to other algorithms

difference with SVC and SVR

- SVC shall contain as few data points as possible in margin.
- SVR should contain as many data points as possible in margin.





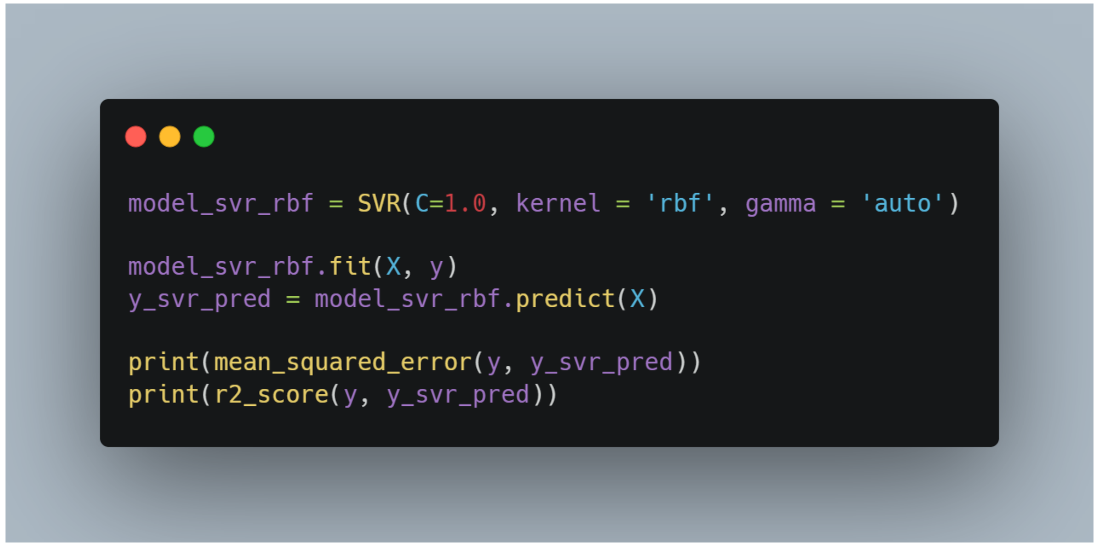
```
from sklearn.svm import SVR

model_svr_linear = SVR(C = 0.01, kernel = 'linear')

model_svr_linear.fit(X, y)
y_svr_pred = model_svr_linear.predict(X)
```

경희대학교
KYUNG HEE UNIVERSITY

Setting hyper parameter

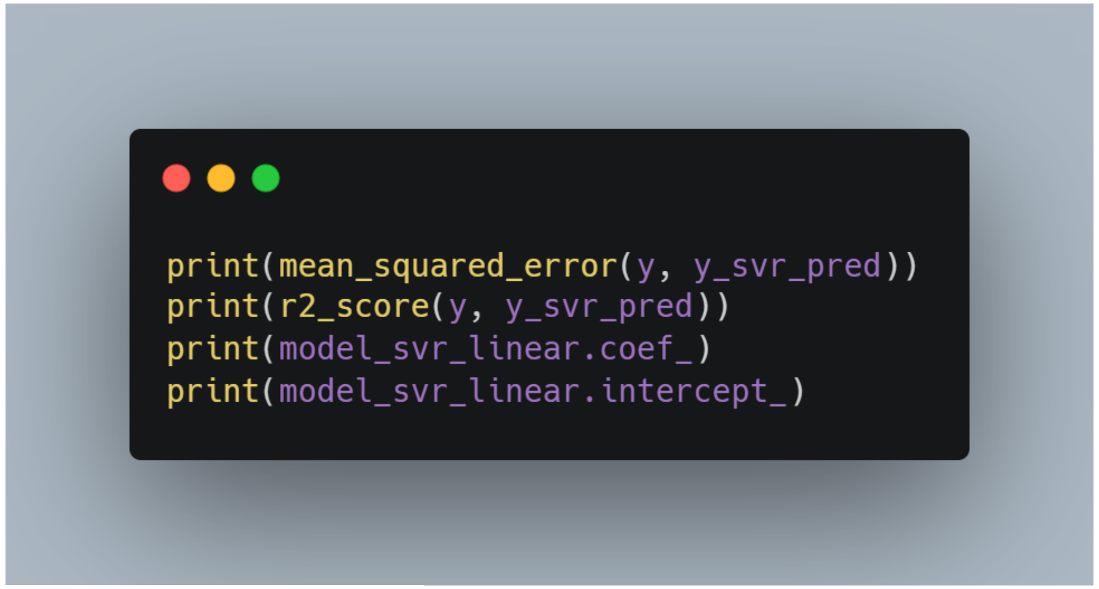
- The learning parameter is updated with the machine learning algorithm, but the hyperparameter must be set by the user before learning.
- Failure to set appropriate hyperparameters results in poor model performance.

Setting hyper parameter



```
model_svr_rbf = SVR(C=1.0, kernel = 'rbf', gamma = 'auto')

model_svr_rbf.fit(X, y)
y_svr_pred = model_svr_rbf.predict(X)

print(mean_squared_error(y, y_svr_pred))
print(r2_score(y, y_svr_pred))
```

```
36.42126375260171
0.5685684051071418
```



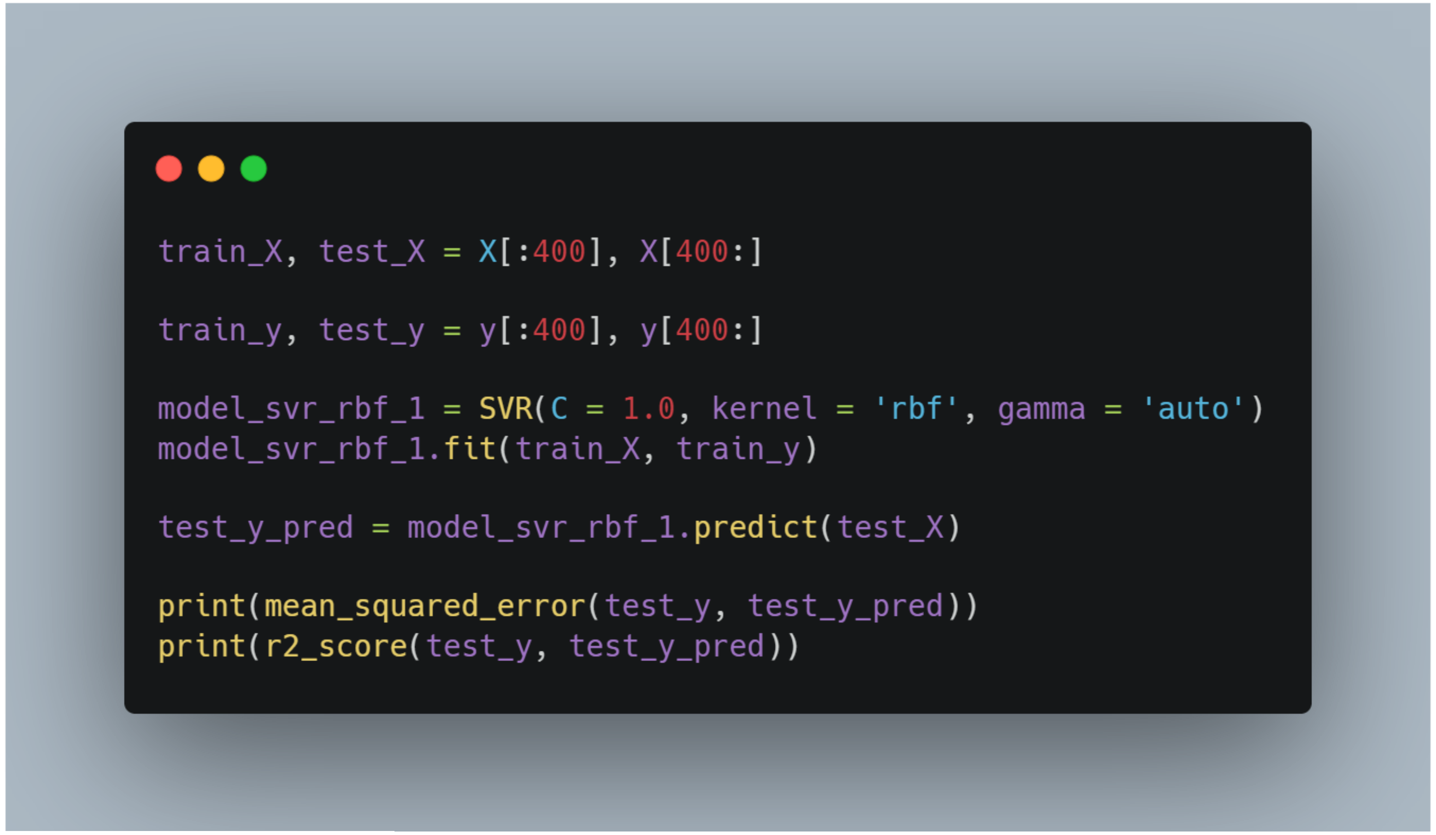
```
print(mean_squared_error(y, y_svr_pred))
print(r2_score(y, y_svr_pred))
print(model_svr_linear.coef_)
print(model_svr_linear.intercept_)
```

```
72.14197118147209
0.1454353177595659 7
[[1.64398]]
[11.13520958]
```

Over-fitting

- The phenomenon in which good prediction results are shown as learning data, but bad prediction results are shown as test data that are not used for learning is called overfitting.
- The model performance when using unknown data is called **generalization performance**.
- Even if the mean square error is small when using the learning data, the **generalization performance** is low if it is an overfitting model.

Over-fitting

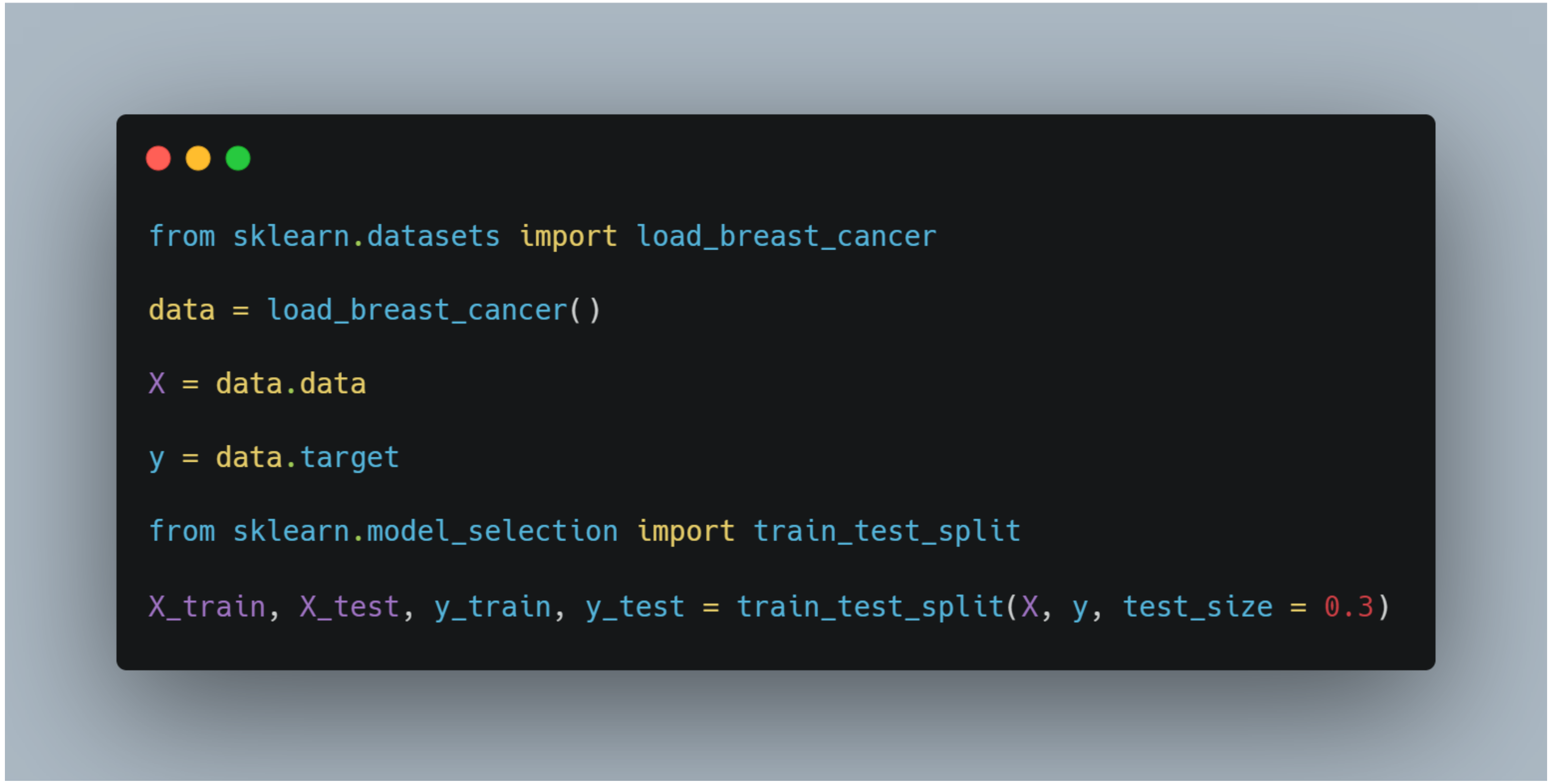


```
train_X, test_X = X[:400], X[400:]

train_y, test_y = y[:400], y[400:]

model_svr_rbf_1 = SVR(C = 1.0, kernel = 'rbf', gamma = 'auto')
model_svr_rbf_1.fit(train_X, train_y)

test_y_pred = model_svr_rbf_1.predict(test_X)

print(mean_squared_error(test_y, test_y_pred))
print(r2_score(test_y, test_y_pred))
```

```
69.16928620453004
-1.4478345530124388
```

Ways to prevent over-fitting

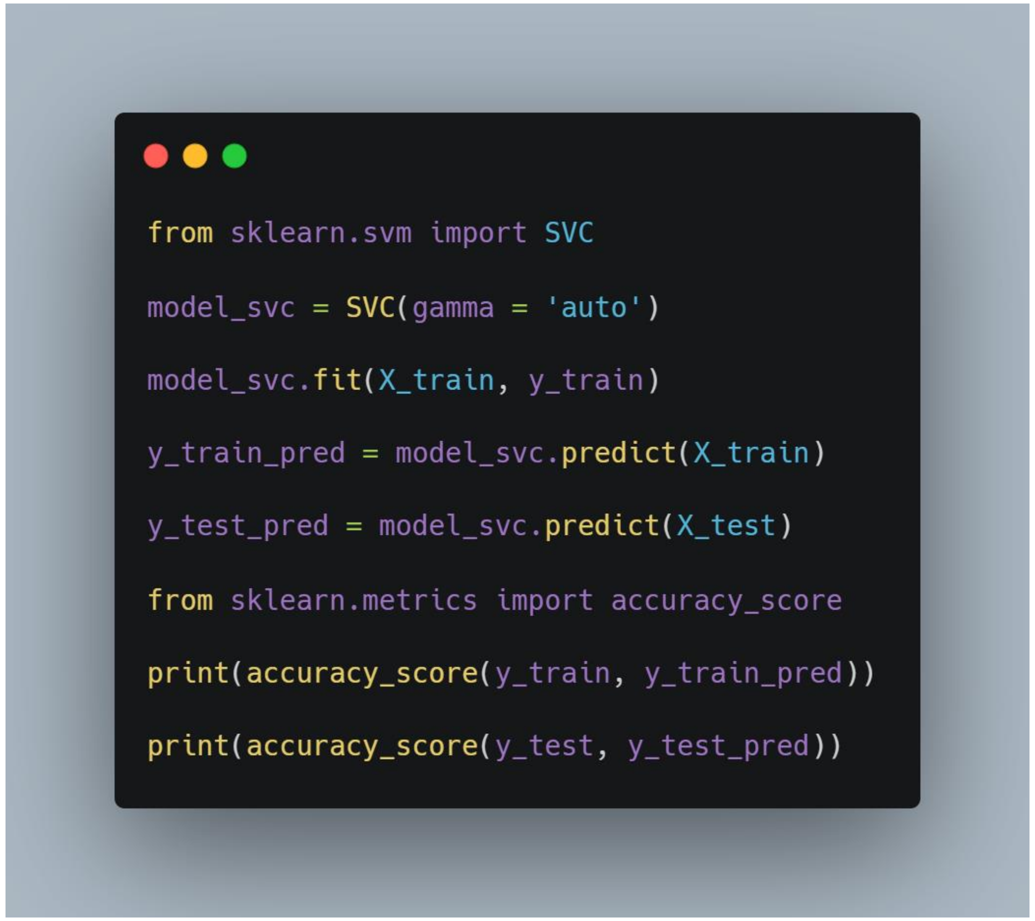
1. split training data and test data



```python
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()

X = data.data

y = data.target

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

Ways to prevent over-fitting

Models that use learning data and test data to learn with SVR

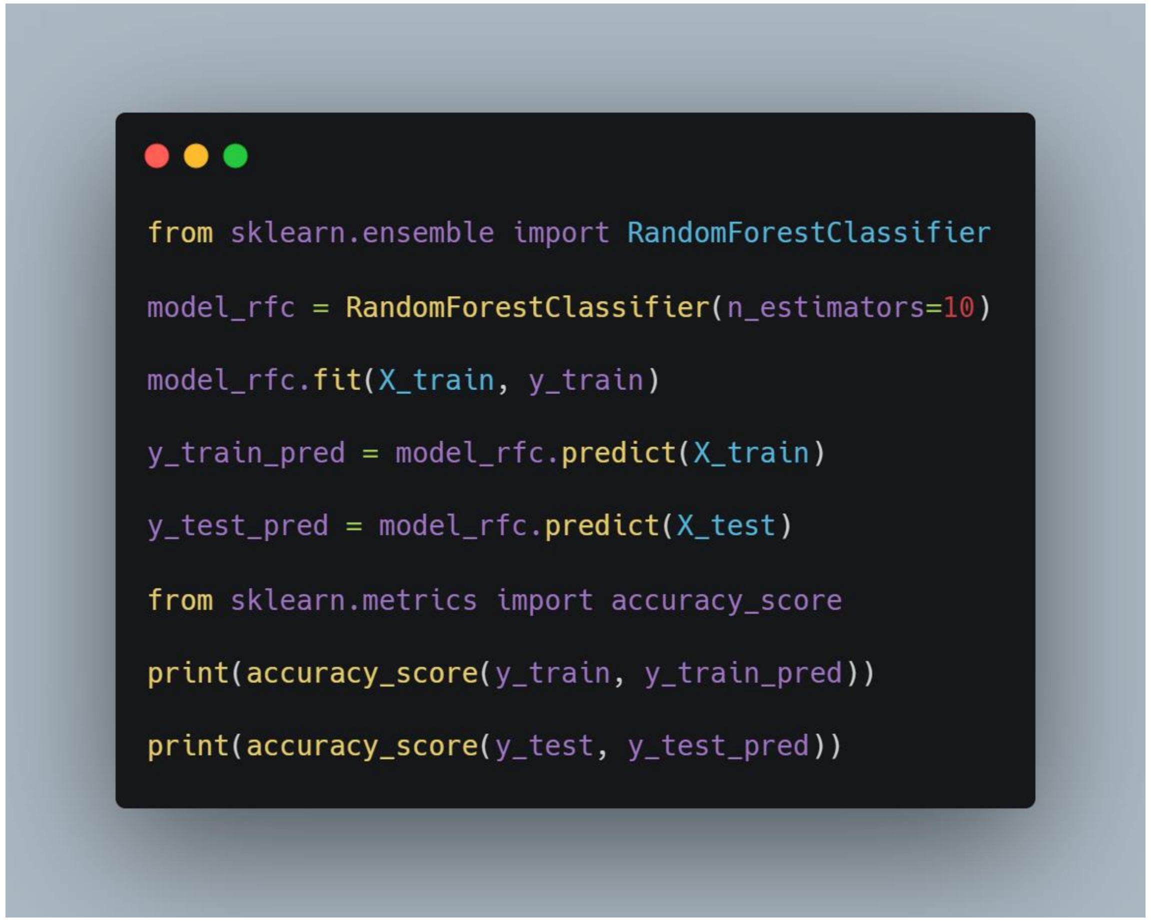


```
from sklearn.svm import SVC

model_svc = SVC(gamma = 'auto')

model_svc.fit(X_train, y_train)

y_train_pred = model_svc.predict(X_train)

y_test_pred = model_svc.predict(X_test)

from sklearn.metrics import accuracy_score

print(accuracy_score(y_train, y_train_pred))

print(accuracy_score(y_test, y_test_pred))
```

경희대학교
KYUNG HEE UNIVERSITY

Ways to prevent over-fitting

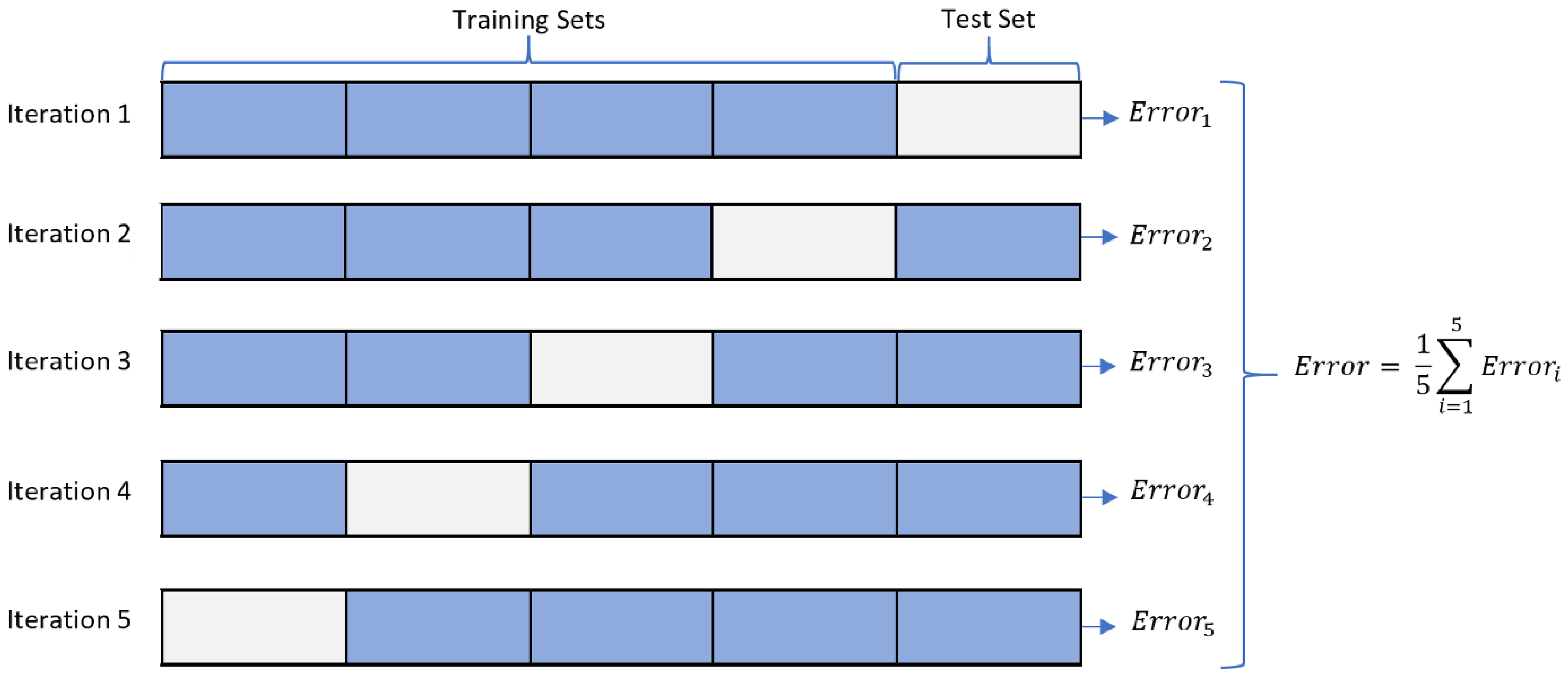
Models that use learning data and test data to learn with RFC



```
from sklearn.ensemble import RandomForestClassifier

model_rfc = RandomForestClassifier(n_estimators=10)

model_rfc.fit(X_train, y_train)

y_train_pred = model_rfc.predict(X_train)

y_test_pred = model_rfc.predict(X_test)

from sklearn.metrics import accuracy_score

print(accuracy_score(y_train, y_train_pred))

print(accuracy_score(y_test, y_test_pred))
```

Ways to prevent over-fitting

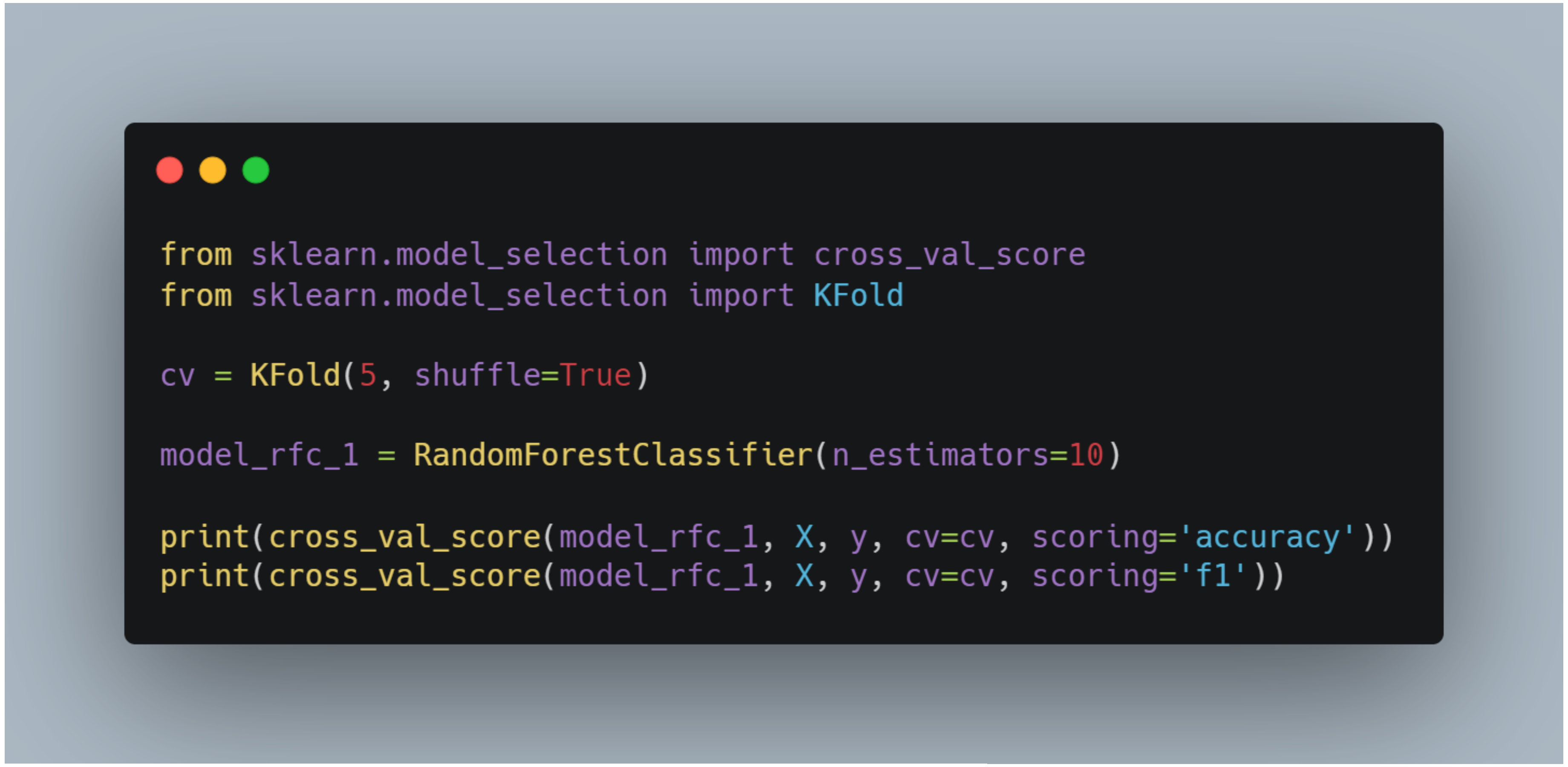
Cross Validation



경희대학교 KYUNG HEE UNIVERSITY

Ways to prevent over-fitting

Cross Validation



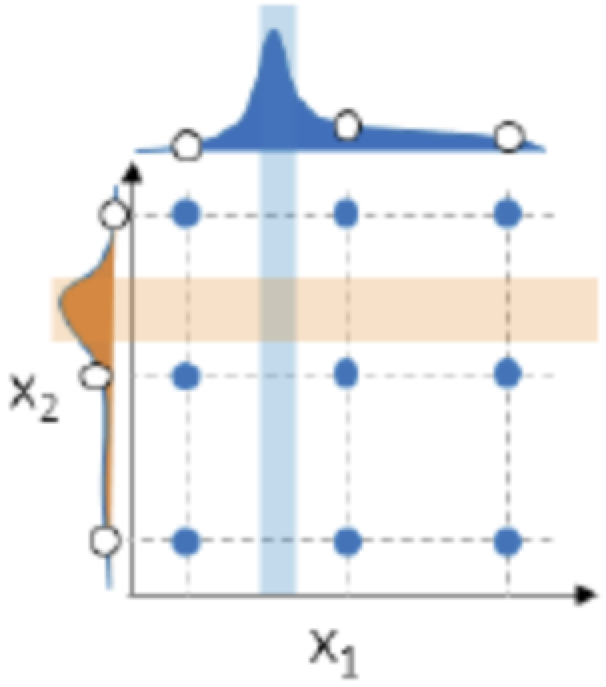
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

cv = KFold(5, shuffle=True)

model_rfc_1 = RandomForestClassifier(n_estimators=10)

print(cross_val_score(model_rfc_1, X, y, cv=cv, scoring='accuracy'))
print(cross_val_score(model_rfc_1, X, y, cv=cv, scoring='f1'))
```

```
[0.97368421 0.97368421 0.99122807 0.96491228 0.92920354]
[0.95172414 0.96341463 0.96350365 0.93959732 0.94214876]
```

경희대학교
KYUNG HEE UNIVERSITY

Searching hyper parameter

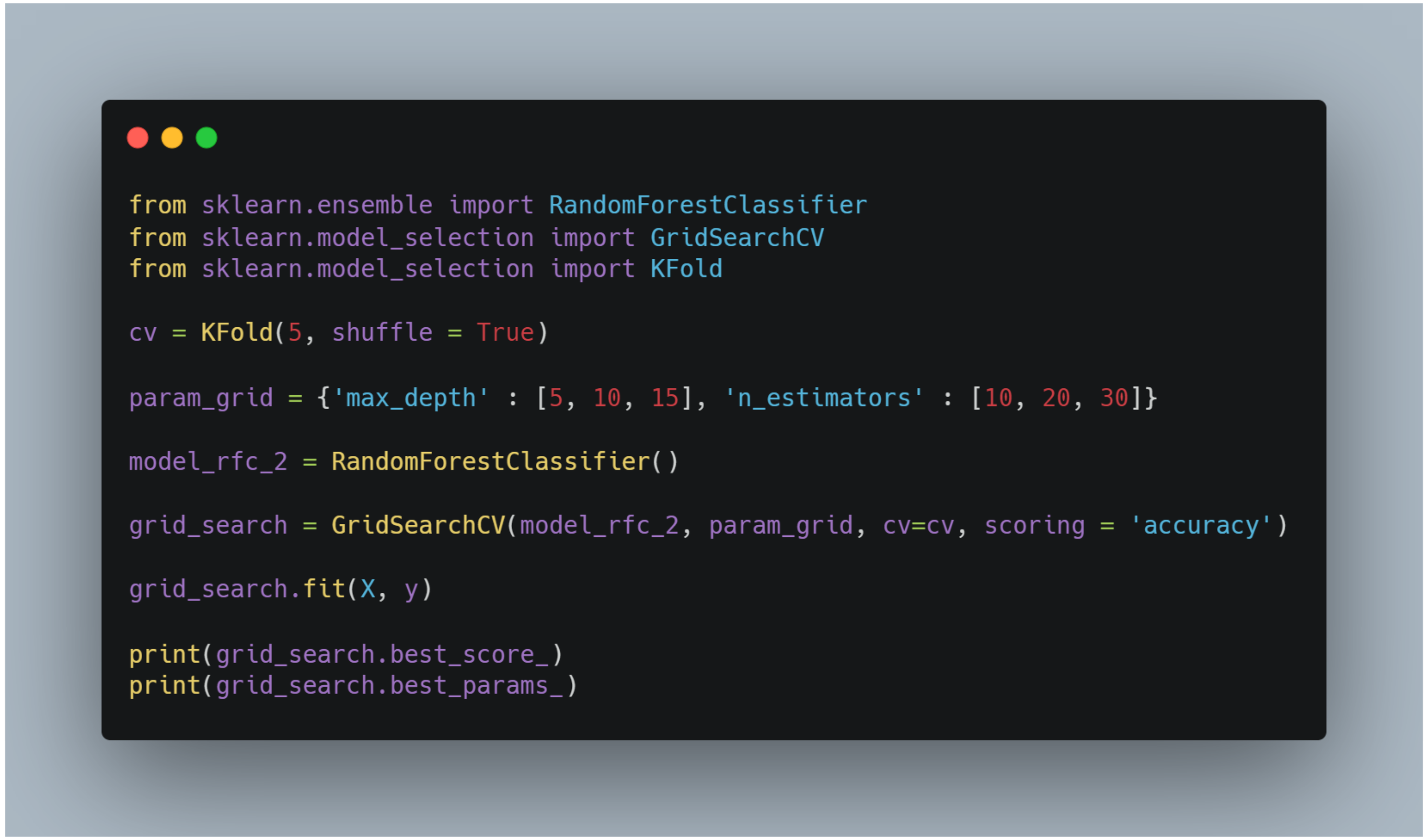
greed search



(a) Standard Grid Search

Searching hyper parameter

greed search



```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold

cv = KFold(5, shuffle = True)

param_grid = {'max_depth' : [5, 10, 15], 'n_estimators' : [10, 20, 30]}

model_rfc_2 = RandomForestClassifier()

grid_search = GridSearchCV(model_rfc_2, param_grid, cv=cv, scoring = 'accuracy')

grid_search.fit(X, y)

print(grid_search.best_score_)
print(grid_search.best_params_)
```

```
0.9420276354603322
{'max_depth': 15, 'n_estimators': 10}
```

경희대학교
KYUNG HEE UNIVERSITY

Evaluation Method and Data Preprocessing

Data Preprocessing

경희대학교 컴퓨터공학과
2019102191 신주영

경희대학교
KYUNG HEE UNIVERSITY

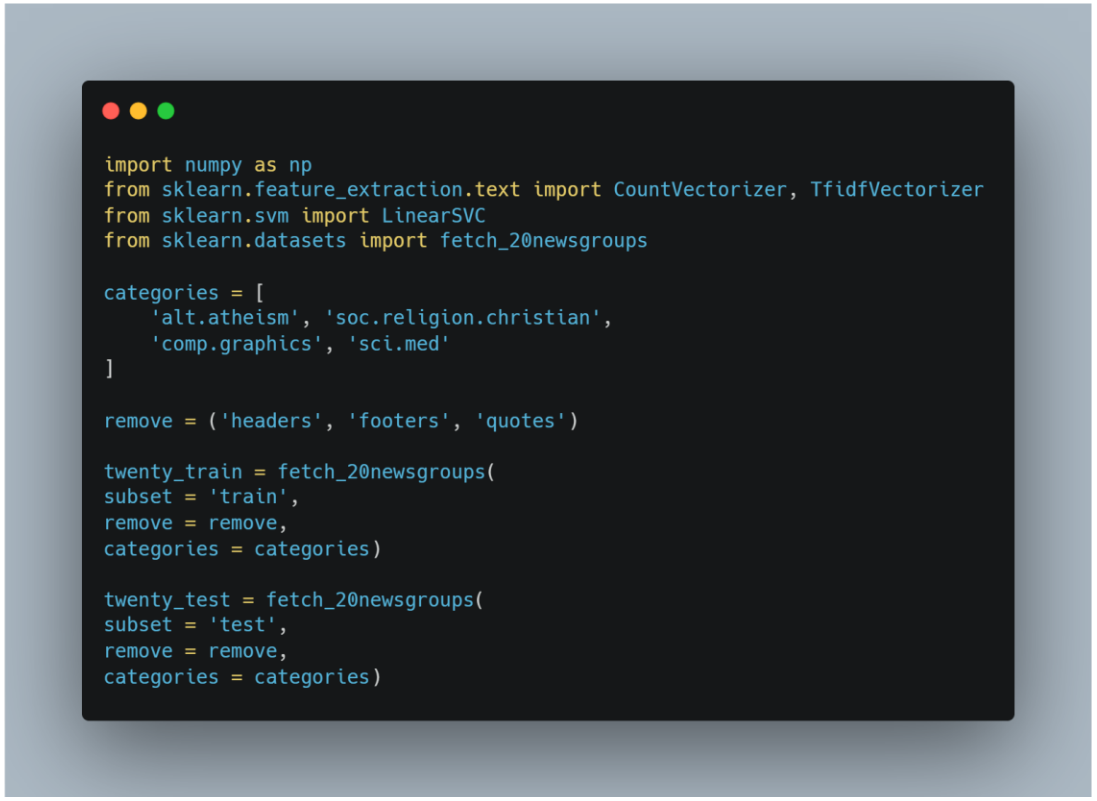
TF-IDF

Term Frequency – Inverse Document Frequency

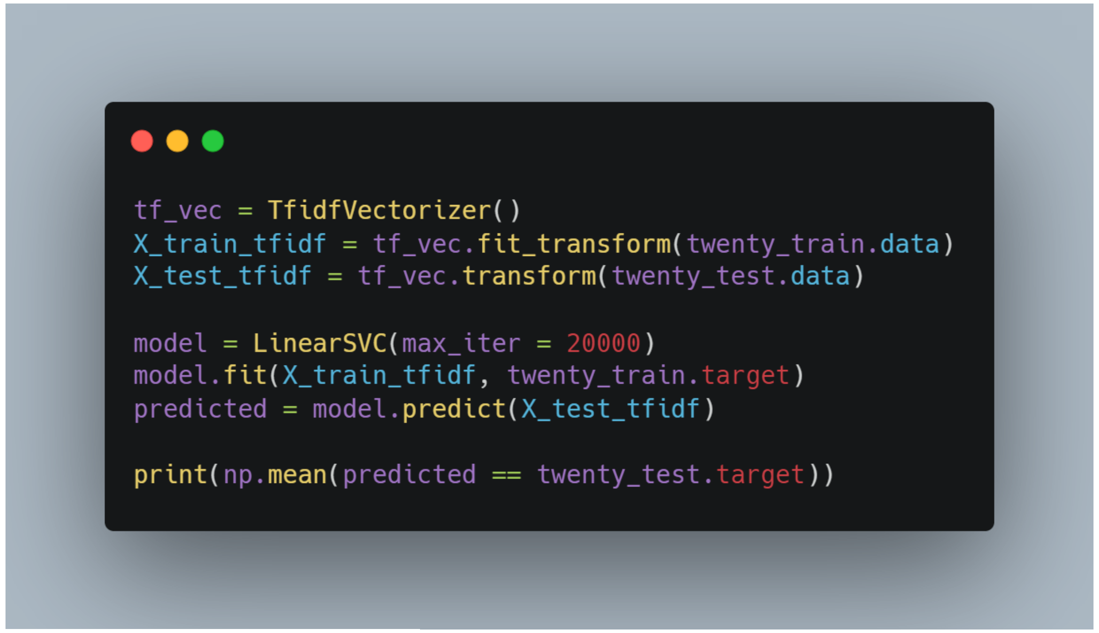
- TF-IDF is a weight used in information retrieval and text mining, and is a statistical value indicating how important a word is within a particular document when there are multiple document groups.
- TF (Term frequency) is a value that indicates how often a particular word appears within a document, and the higher this value, the more important it is in the document.
- However, if the word itself is frequently used within the document group, this means that the word appears frequently. This is called DF (document frequency), and the inverse of this value is called IDF (inverse document frequency).

경희대학교
KYUNG HEE UNIVERSITY

Sample Code



```python
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.datasets import fetch_20newsgroups

categories = [
    'alt.atheism', 'soc.religion.christian',
    'comp.graphics', 'sci.med'
]

remove = ('headers', 'footers', 'quotes')

twenty_train = fetch_20newsgroups(
subset = 'train',
remove = remove,
categories = categories)

twenty_test = fetch_20newsgroups(
subset = 'test',
remove = remove,
categories = categories)
```



```python
tf_vec = TfidfVectorizer()
X_train_tfidf = tf_vec.fit_transform(twenty_train.data)
X_test_tfidf = tf_vec.transform(twenty_test.data)

model = LinearSVC(max_iter = 20000)
model.fit(X_train_tfidf, twenty_train.target)
predicted = model.predict(X_test_tfidf)

print(np.mean(predicted == twenty_test.target))
```

```
0.8149134487350199
```

Evaluation Method and Data Preprocessing

Data Preprocessing

경희대학교 컴퓨터공학과
2019102191 신주영

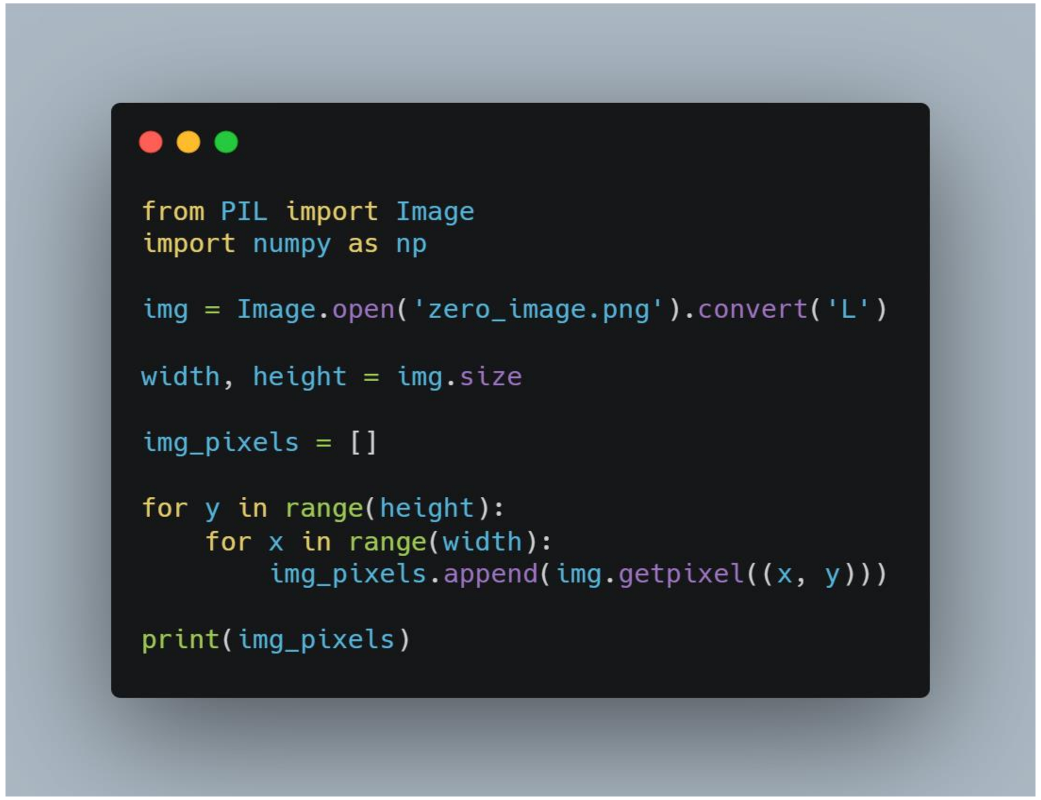
경희대학교
KYUNG HEE UNIVERSITY

Converting Image Data

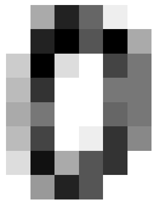
MNIST

- The brightness value of the image data pixel of each number is converted into vector data and used.

Sample Code



```python
from PIL import Image
import numpy as np

img = Image.open('zero_image.png').convert('L')

width, height = img.size

img_pixels = []

for y in range(height):
    for x in range(width):
        img_pixels.append(img.getpixel((x, y)))

print(img_pixels)
```

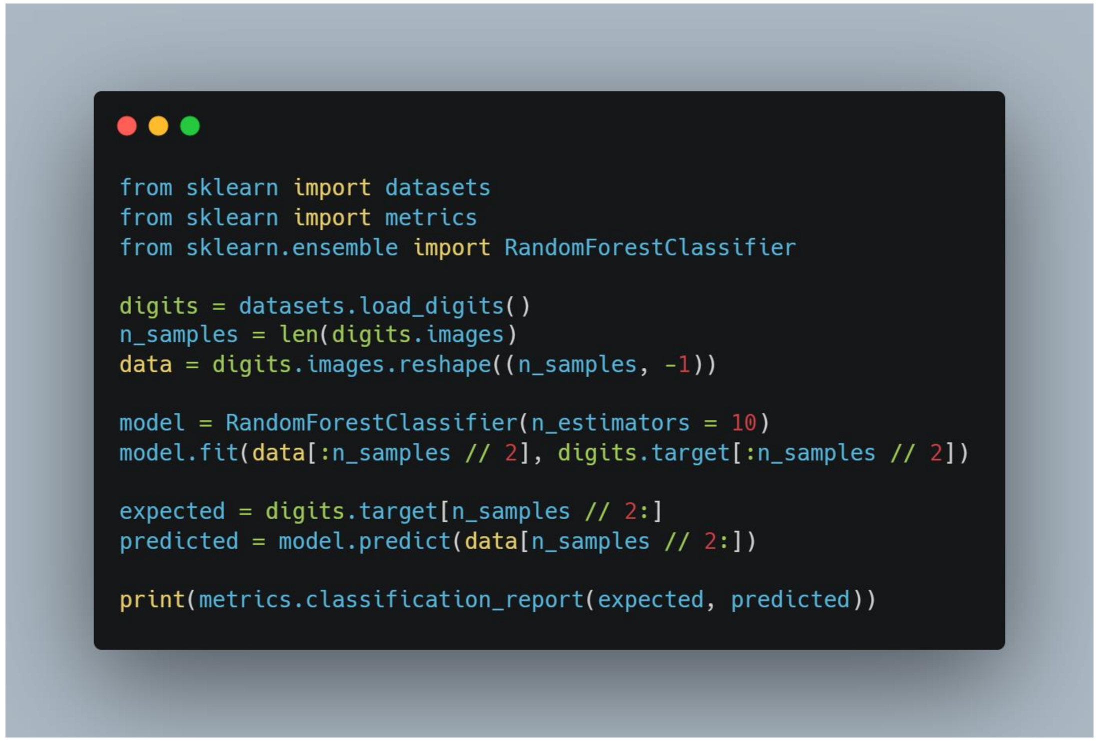


```
[255, 255, 170, 34, 102, 238, 255, 255, 255, 255, 34, 0, 85, 0, 170, 255, 255, 204, 0, 221, 255, 68, 119, 255, 255, 187, 51, 255, 255, 11
9, 119, 255, 255, 170, 119, 255, 255, 102, 119, 255, 255, 187, 68, 255, 238, 51, 136, 255, 255, 221, 17, 170, 85, 51, 255, 255, 255, 255,
153, 34, 85, 255, 255, 255]
```

경희대학교
KYUNG HEE UNIVERSITY

Sample Code

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.93        88
           1       0.84      0.88      0.86        91
           2       0.89      0.88      0.89        86
           3       0.88      0.87      0.87        91
           4       0.90      0.83      0.86        92
           5       0.78      0.86      0.82        91
           6       0.91      0.93      0.92        91
           7       0.91      0.92      0.92        89
           8       0.83      0.68      0.75        88
           9       0.81      0.83      0.82        92

    accuracy                           0.87       899
   macro avg       0.87      0.87      0.86       899
weighted avg       0.87      0.87      0.86       899
```



```python
from sklearn import datasets
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier

digits = datasets.load_digits()
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

model = RandomForestClassifier(n_estimators = 10)
model.fit(data[:n_samples // 2], digits.target[:n_samples // 2])

expected = digits.target[n_samples // 2:]
predicted = model.predict(data[n_samples // 2:])

print(metrics.classification_report(expected, predicted))
```