hello.edu

참고 드라이브

https://drive.google.com/drive/folders/1YNRsniidbQCIF87ZGV5ViFX4dm9Eo5O8

<https://github.com/vinta/awesome-python>

주목받는 파이썬 깃허브

<https://github.com/trending/python>

파이썬 명령어

pdf 52page 참고

싸이킥 런(SCIKIT - LEARN)

https://scikit-learn.org/stable/user\_guide.html

K-ICT 빅데이터 센터

에서 원격리모트 ID를 개인적으로 따로 받을 수 있다.

우리 회사는

----데이터를 활용하여

---모델을 만들고

이 모델을 적용한 ---기능을 만드는 회사입니다.

<https://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_regression.html#sphx-glr-auto-examples-ensemble-plot-gradient-boosting-regression-py>

print(\_\_doc\_\_)

# Author: Peter Prettenhofer <peter.prettenhofer@gmail.com>

## License: BSD 3 clause

import numpy as np

import matplotlib.pyplot as plt

from sklearn import ensemble

from sklearn import datasets

from sklearn.utils import shuffle

from sklearn.metrics import mean\_squared\_error

# #############################################################################

# Load data

boston = datasets.load\_boston()

X, y = shuffle(boston.data, boston.target, random\_state=13)

X = X.astype(np.float32)

offset = int(X.shape[0] \* 0.9)

X\_train, y\_train = X[:offset], y[:offset]

X\_test, y\_test = X[offset:], y[offset:]

# #############################################################################

# Fit regression model

params = {'n\_estimators': 500, 'max\_depth': 4, 'min\_samples\_split': 2,

'learning\_rate': 0.01, 'loss': 'ls'}

clf = ensemble.GradientBoostingRegressor(\*\*params)

clf.fit(X\_train, y\_train)

mse = mean\_squared\_error(y\_test, clf.predict(X\_test))

print("MSE: %.4f" % mse)

# #############################################################################

# Plot training deviance

# compute test set deviance

test\_score = np.zeros((params['n\_estimators'],), dtype=np.float64)

for i, y\_pred in enumerate(clf.staged\_predict(X\_test)):

test\_score[i] = clf.loss\_(y\_test, y\_pred)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.title('Deviance')

plt.plot(np.arange(params['n\_estimators']) + 1, clf.train\_score\_, 'b-',

label='Training Set Deviance')

plt.plot(np.arange(params['n\_estimators']) + 1, test\_score, 'r-',

label='Test Set Deviance')

plt.legend(loc='upper right')

plt.xlabel('Boosting Iterations')

plt.ylabel('Deviance')

# #############################################################################

# Plot feature importance

feature\_importance = clf.feature\_importances\_

# make importances relative to max importance

feature\_importance = 100.0 \* (feature\_importance / feature\_importance.max())

sorted\_idx = np.argsort(feature\_importance)

pos = np.arange(sorted\_idx.shape[0]) + .5

plt.subplot(1, 2, 2)

plt.barh(pos, feature\_importance[sorted\_idx], align='center')

plt.yticks(pos, boston.feature\_names[sorted\_idx])

plt.xlabel('Relative Importance')

plt.title('Variable Importance')

plt.show()