Machine Learning in python

7기 김대규 deepvine@gmail.com

Essential techniques for predictive analysis

ENSEMBLE METHODS

0. Introduce scikit-learn
1. Decision Tree
2. Random Forest
3. Boosting

0. Introduce Scikit

> **Scikit-learn** (formerly **scikits.learn**) is a free software machine learning library for the Python programming language.[2] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
> Decision Tree
> (위키피디아)

installation http://scikit-learn.org/stable/install.html

documentation http://scikit-learn.org/stable/documentation.html#

더 공부하고 싶으신 분은 아래 책을 추천해드립니다

http://book.naver.com/bookdb/book_detail.nhn?bid=9566723

1. Decision Tree

앙상블을 하기 전에 의사결정트리에 대해 먼저 알고 가겠습니다.

예를 들면 의사결정트리는 아래와 같은 것이 있습니다.

source file wineTree.py

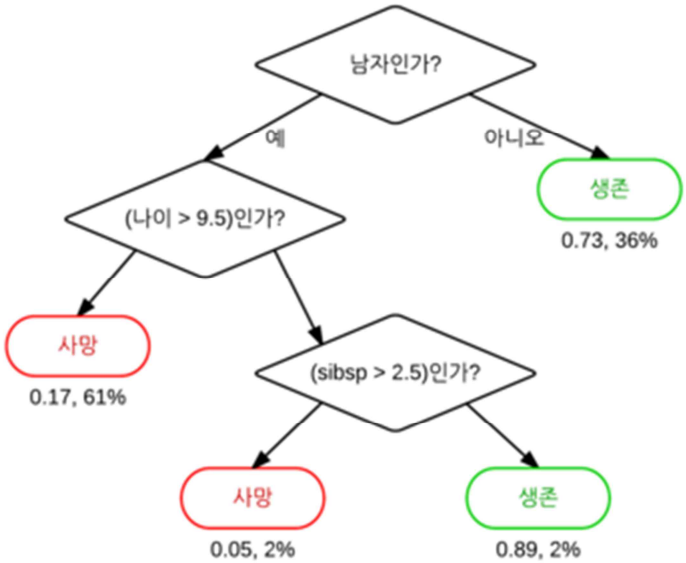
Reference source code https://github.com/andyyuan78/willey_py_ML/blob/master/06/wineTree.py

결과 (wineTree.dot)

```
digraph Tree {

node [shape=box] ;

0 [label="X[10] <= 10.525₩nmse = 0.6518₩nsamples = 1599₩nvalue = 5.636"] ;

1 [label="X[9] <= 0.575₩nmse = 0.4315₩nsamples = 983₩nvalue = 5.3662"] ;

...

14 [label="mse = 0.4297₩nsamples = 138₩nvalue = 6.6522"] ;

12 -> 14 ;

}
```

자세한 결과는 해당 소스 실행 후 wineTree.dot 파일을 확인해보세요

의사결정트리는 다음과 같은 모양으로 이루어져 있습니다.



이제 의사결정트리를 이루는데 중요한 변수 2가지를 알 수 있겠네요

첫번째는 분할점 선택! (예를 들어 위의 의사결정트리에서 나이 비교를 왜 9.5를 했는지. 10이나 9, 8도 가능하니깐 이중에 어떤걸 선택할지에 대한 문제)

두번째는 트리의 깊이(depth)(위의 의사결정트리에서 트리의 깊이는 3이라는 걸 통해 3번 의사결정 과정이 이루어진다는 걸 알 수 있습니다)

트리의 깊이와 분할점 선택이 예측에 미치는 영향

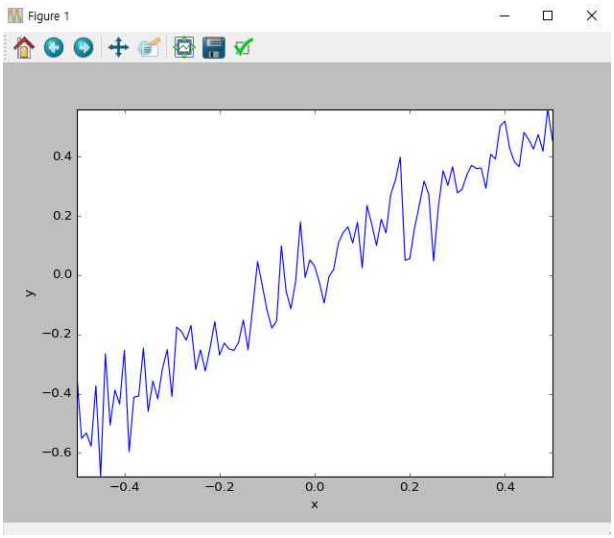
source file simpleTree.py

Reference source code

https://github.com/andyyuan78/willey_py_ML/blob/master/06/simpleTree.py

x값에 노이즈를 더한 값이 y값입니다

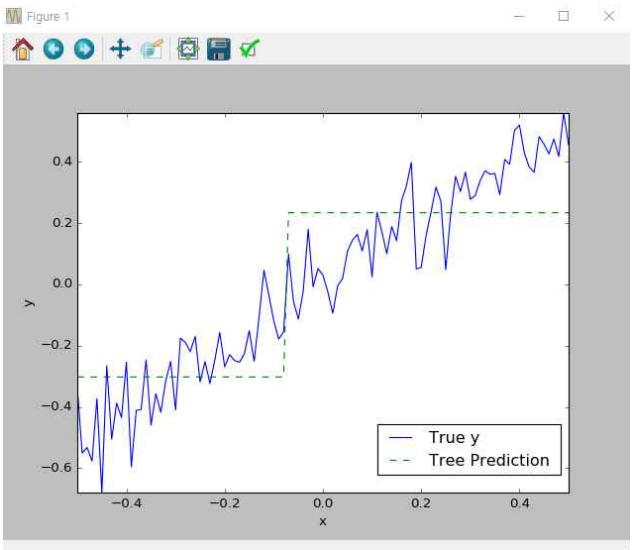
특징적으로 랜덤되는 값은 있지만 대체적으로 값이 맞아가고 있는걸 알 수 있습니다.



트리가 미치는 영향

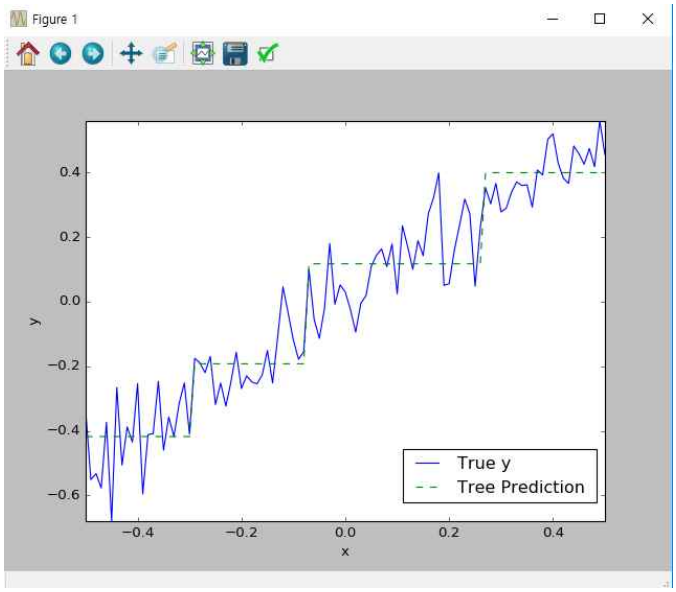
트리의 depth가 2일 때

```
simpleTree2 = DecisionTreeRegressor(max_depth=2)
simpleTree2.fit(x, y)
```



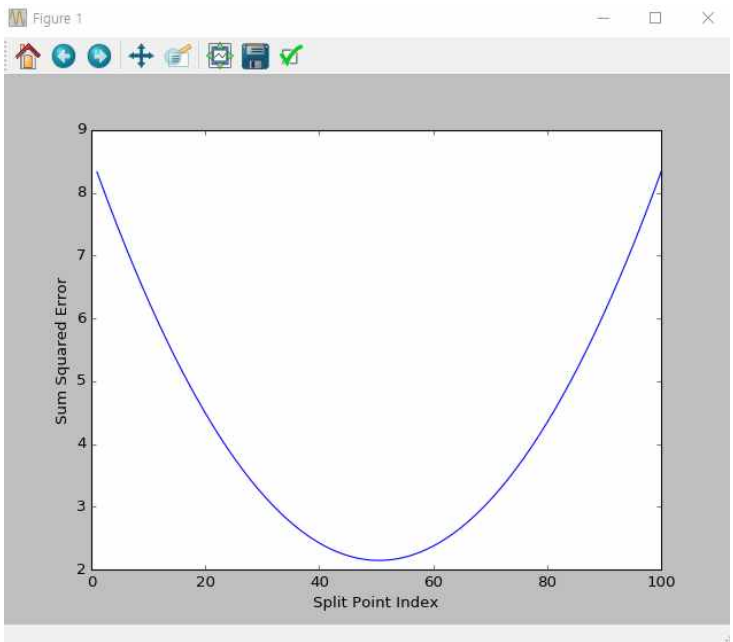
트리의 depth가 6일 때

```
simpleTree2 = DecisionTreeRegressor(max_depth=6)
simpleTree2.fit(x, y)
```



분할점 선택이 미치는 영향

분할점 선택에 따른 MSE(오차제곱합:작을수록 좋습니다)



따라서 50정도에 분할하는게 가장 예측이 우수하다는 걸 알 수있습니다

하지만 트리의 depth가 무조건 높다고해서 예측이 우수한 것이 아닙니다.

depth를 무작정 높임으로 쓸데없는(예측 값을 오히려 깎는) 값이 나올 수 도 있습니다.

자세한 내용은 아래 실습을 해보시면서 참고해보세요

트리의 깊이가 무조건 클수록 좋은건 아니다

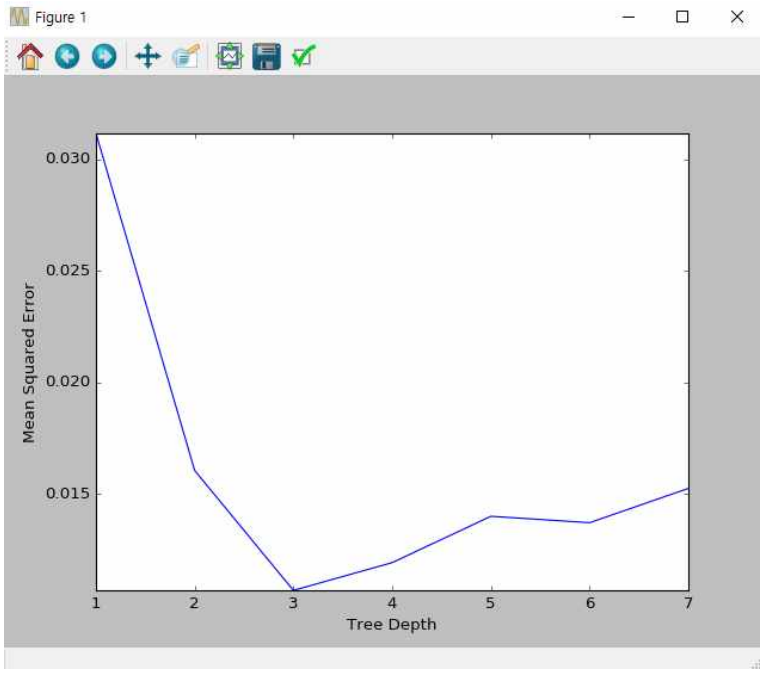
source file simpleTreeCV.py

Reference source code

https://github.com/andyyuan78/willey_py_ML/blob/master/06/simpleTreeCV.py

트리의 깊이가 너무 커버리면 overfitting 문제가 초래합니다

overfitting 참고 http://www.aistudy.co.kr/learning/mining/technique_jang.htm



위의 예제에서는 트리 깊이가 3이 가장 적합한걸 알 수 있습니다

이렇게 트리깊이와 분할점 선택이 주요하지만 다른 기준도 여러가지 있습니다.

이런 기준여러가지를 종합하여 앙상블 실습을 해보겠습니다

실습data http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

2. Random Forest

Original source code http://media.wiley.com/product_ancillary/49/11189617/DOWNLOAD/06.zip

앙상블 패키지를 사용하기 위해 아래 모듈을 import합니다

sklearn import ensemble

랜덤 포레스트 클래스 생성자

```
class sklearn.ensemble.RandomForestRegressor (n_estimators=10,
criterion='mse', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',
max_leaf_nodes=None, min_impurity_split=1e-07, bootstrap=True,
oob_score=False, n_jobs=1, random_state=None, verbose=0,
warm_start=False)
```

train_test_split()함수

배열 또는 매트릭스를 랜덤하게 train과 test set으로 나눕니다.

ex) training과 test set을 80대 20으로 나눈다.

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

따라서 아래 소스는 training, test set을 7:3으로 나눕니다.

```
xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size=0.30,
random_state=531)
```

```
nTreeList = range(50, 500, 10)
for iTrees in nTreeList:
    depth = None
    maxFeat  = 4 #try tweaking
    wineRFModel = ensemble.RandomForestRegressor(n_estimators=iTrees,
max_depth=depth, max_features=maxFeat,
oob_score=False, random_state=531)
```

아래는 주요 파라미터에 대한 설명입니다

자세한 파라미터 설명은 http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

참고바랍니다

n_estimators 앙상블 트리의 개수, default=10

max_depth (아까 논의한거네요) 최대 트리의 depth 크기 default=None

min_samples_split 최소 샘플 표본의 수 default=2

min_samples_leaf 만약 분기 때문에 노드의 샘플 개수가 min_samples_leaf보다 작아진다면 분기하지 않는다. default=1

max_features 문제에 있는 피처의 개수에 의존적인 최적의 분기를 찾기 위하여 고려할 피처의 개수 default=None

random_state 만약 정수 형식이라면, 정수를 랜덤 숫자를 생성하는 시드로 사용
RandomState의 인스턴스라면, 인스턴스를 랜덤 숫자 생성기로 사용
None이라면, numpy.random에서 사용하는 RandomState의 인스턴스를 랜덤 숫자 생성기로 사용
default=None

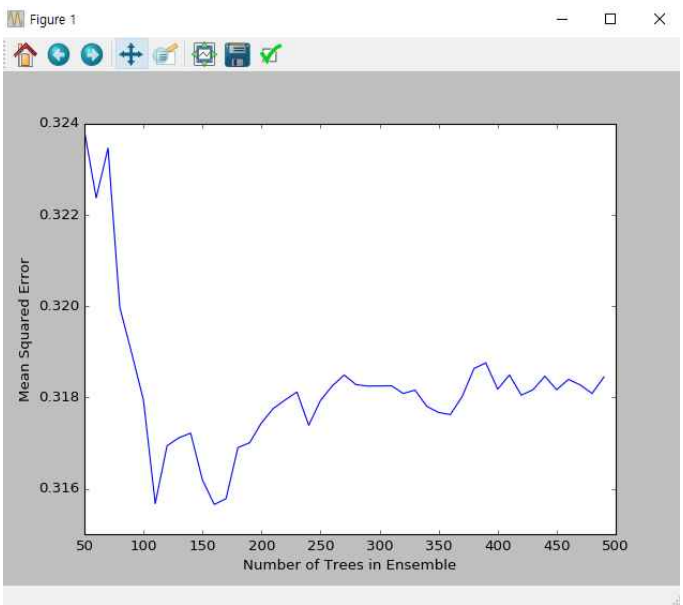
xTrain, yTrain을 통해 의사결정트리를 생성합니다

```
wineRFModel.fit(xTrain,yTrain)
```

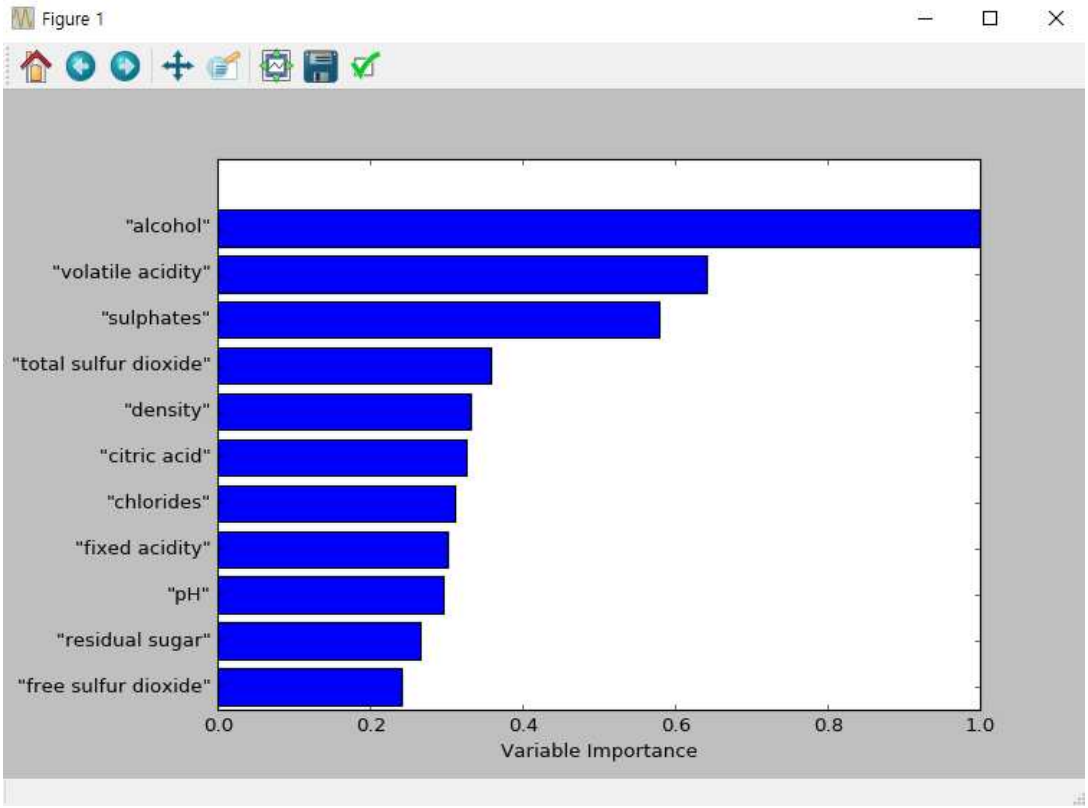
예측 오차제곱합

```
MSE
0.318446272387
```

(MSE가 작을수록 추정의 정확성이 높아짐)



트리의 개수가 100개 정도일 때 최고의 예측이 나오는걸 알 수있습니다



알코올 도수가 와인의 맛에 가장 큰 영향을 미친다는 걸 알 수 있습니다

3. Gradient Boosting

클래스 생성자

```
class sklearn.ensemble.GradientBoostingRegressor(loss='ls',
learning_rate=0.1, n_estimators=100, subsample=1.0,
criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_split=1e-07,
init=None, random_state=None, max_features=None, alpha=0.9, verbose=0,
max_leaf_nodes=None, warm_start=False, presort='auto')
```

```
nEst = 2000
depth = 7
learnRate = 0.01
subSamp = 0.5
wineGBMModel = ensemble.GradientBoostingRegressor(n_estimators=nEst,
                                                  max_depth=depth,
                                                  learning_rate=learnRate,
                                                  subsample = subSamp,
                                                  loss='ls')
```

아래는 주요 파라미터에 대한 설명입니다

자세한 파라미터 설명은 http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html#sklearn.ensemble.GradientBoostingRegressor

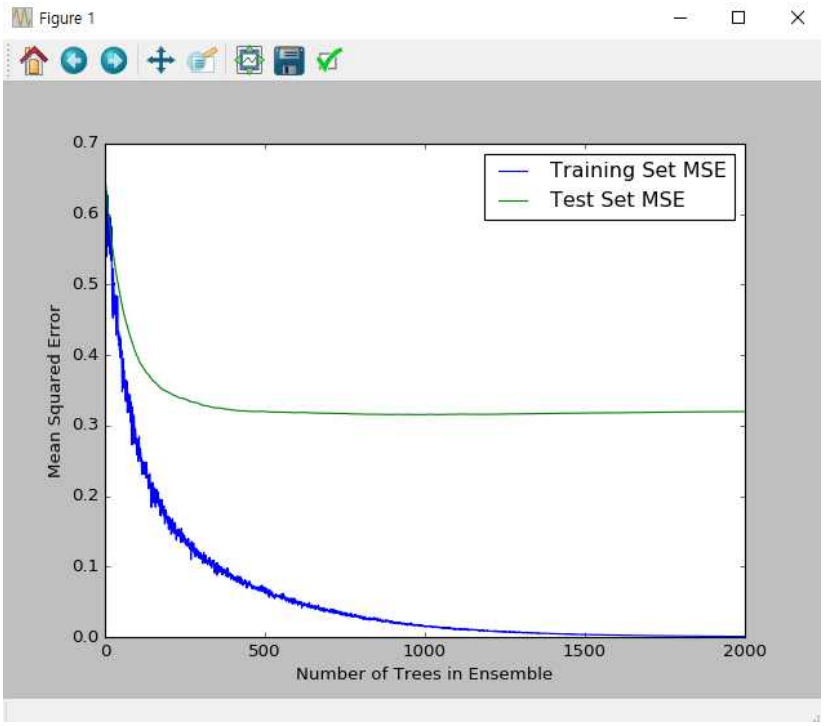
참고바랍니다

xTrain, yTrain을 통해 의사결정트리를 생성합니다

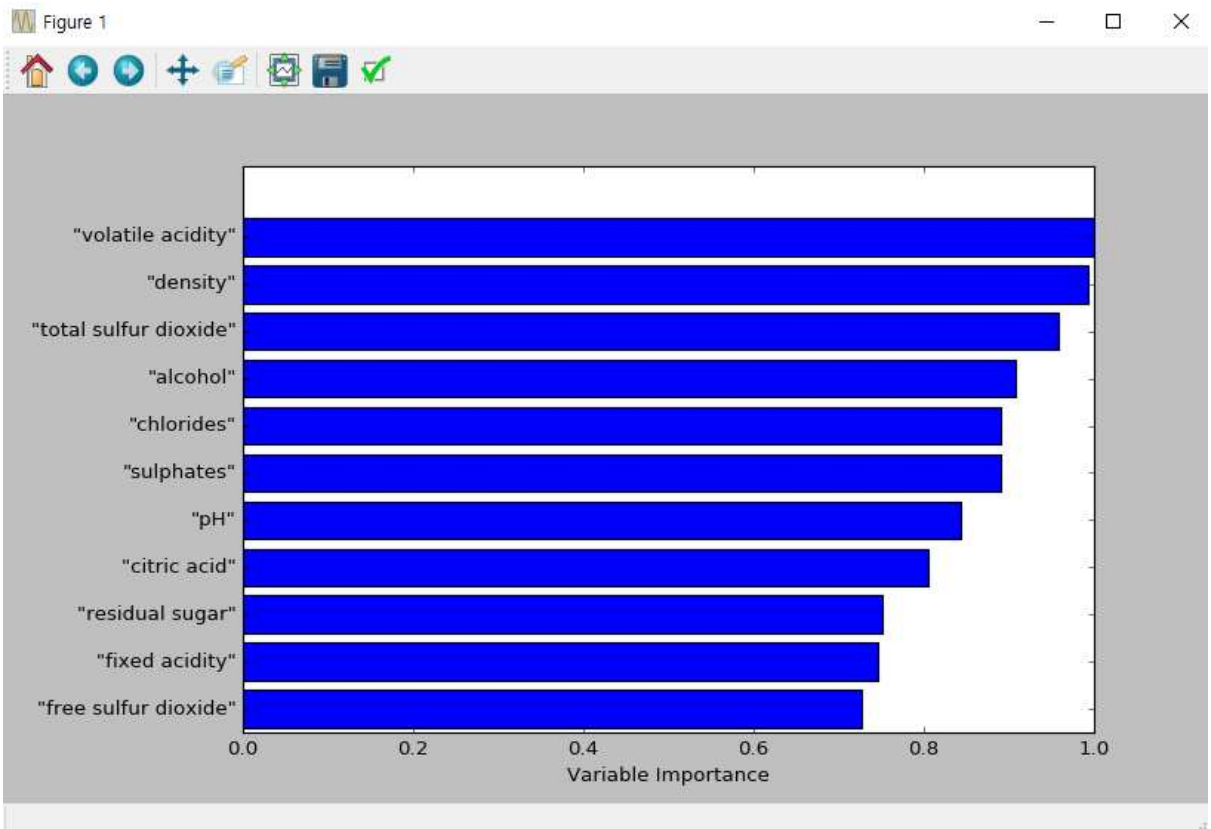
```
wineGBMModel.fit(xTrain, yTrain)
```

예측 오차제곱합

```
MSE
0.315427753309
986
```



파란색의 그래디언트 부스팅 모델이 랜덤포레스트 보다 성능이 더 좋은걸 알 수 있습니다



휘발성 산도가 와인의 맛에 가장 큰 영향을 미친다는 걸 알 수 있습니다