XGBoost

- 트리 기반의 앙상블 기법
- 분류에 있어서 다른 알고리즘보다 좋은 예측 성능을 보여줌
- XGBoost는 GBM 기반이지만, GBM의 단점인 느린 수행 시간과 과적합 규제 부재 등의 문제를 해결
- 병렬 CPU 환경에서 빠르게 학습 가능

```
from sklearn.datasets import load_iris, load_wine, load_breast_cancer
from sklearn.datasets import load_boston, load_diabetes
from sklearn.model_selection import train_test_split, cross_validate
from sklearn.metrics import accuracy_score, precision_score, recall_score

import numpy as np
import xgboost as xgb
from xgboost import XGBClassifier, XGBRegressor
from xgboost import plot_importance, plot_tree

import graphviz
import matplotlib.pyplot as plt
plt.style.use(['seaborn-whitegrid'])
```

+ 코드 + 텍스트

파이썬 기반 XGBoost

```
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.2, rand
dtrain = xgb.DMatrix(data=X_train, label=y_train)
dtest = xgb.DMatrix(data=X_test, label=y_test)
```

```
params = {
    'max_depth':3,
    'eta':0.1,
    'objective':'binary:logistic',
    'eval_metric':'logloss',
    'early_stopping':100
}
num_rounds = 400
```

```
evals = [(dtrain, 'train'), (dtest, 'eval')]
xgb_model = xgb.train(params=params, dtrain=dtrain, num_boost_round=num_rounds, early_stopping_roun
```

```
[0]     train-logloss:0.609437  eval-logloss:0.6101
Multiple eval metrics have been passed: 'eval-logloss' will be used for early stopping.

Will train until eval-logloss hasn't improved in 100 rounds.
[1]     train-logloss:0.540472  eval-logloss:0.540739
[2]     train-logloss:0.482112  eval-logloss:0.484899
```

```
[3]	train-logloss:0.433652	eval-logloss:0.43922
[4]	train-logloss:0.39072	eval-logloss:0.39791
[5]	train-logloss:0.353716	eval-logloss:0.362107
[6]	train-logloss:0.322225	eval-logloss:0.329962
[7]	train-logloss:0.292921	eval-logloss:0.301792
[8]	train-logloss:0.267307	eval-logloss:0.278011
[9]	train-logloss:0.244087	eval-logloss:0.25775
[10]	train-logloss:0.22517	eval-logloss:0.240372
[11]	train-logloss:0.208072	eval-logloss:0.222709
[12]	train-logloss:0.191772	eval-logloss:0.207947
[13]	train-logloss:0.17746	eval-logloss:0.195312
[14]	train-logloss:0.164082	eval-logloss:0.184597
[15]	train-logloss:0.152497	eval-logloss:0.171834
[16]	train-logloss:0.141981	eval-logloss:0.161516
[17]	train-logloss:0.132696	eval-logloss:0.15443
[18]	train-logloss:0.123258	eval-logloss:0.149134
[19]	train-logloss:0.11591	eval-logloss:0.141752
[20]	train-logloss:0.108836	eval-logloss:0.133784
[21]	train-logloss:0.102386	eval-logloss:0.128091
[22]	train-logloss:0.096421	eval-logloss:0.123096
[23]	train-logloss:0.09119	eval-logloss:0.117352
[24]	train-logloss:0.086096	eval-logloss:0.112542
[25]	train-logloss:0.080992	eval-logloss:0.111029
[26]	train-logloss:0.076525	eval-logloss:0.107841
[27]	train-logloss:0.072183	eval-logloss:0.106375
[28]	train-logloss:0.068522	eval-logloss:0.103291
[29]	train-logloss:0.065278	eval-logloss:0.100226
[30]	train-logloss:0.062096	eval-logloss:0.097335
[31]	train-logloss:0.059067	eval-logloss:0.094945
[32]	train-logloss:0.056432	eval-logloss:0.092493
[33]	train-logloss:0.053755	eval-logloss:0.090516
[34]	train-logloss:0.051392	eval-logloss:0.088918
[35]	train-logloss:0.049336	eval-logloss:0.087151
[36]	train-logloss:0.047222	eval-logloss:0.085965
[37]	train-logloss:0.045319	eval-logloss:0.084003
[38]	train-logloss:0.04353	eval-logloss:0.083111
[39]	train-logloss:0.04177	eval-logloss:0.081183
[40]	train-logloss:0.040242	eval-logloss:0.07966
[41]	train-logloss:0.038796	eval-logloss:0.078998
[42]	train-logloss:0.037531	eval-logloss:0.078765
[43]	train-logloss:0.036269	eval-logloss:0.078487
[44]	train-logloss:0.035078	eval-logloss:0.077271
[45]	train-logloss:0.033948	eval-logloss:0.077147
[46]	train-logloss:0.032764	eval-logloss:0.076284
[47]	train-logloss:0.031765	eval-logloss:0.075907
[48]	train-logloss:0.03083	eval-logloss:0.076102
[49]	train-logloss:0.029673	eval-logloss:0.076575
[50]	train-logloss:0.02871	eval-logloss:0.076208
[51]	train-logloss:0.027801	eval-logloss:0.075562
[52]	train-logloss:0.02694	eval-logloss:0.075386
[53]	train-logloss:0.026207	eval-logloss:0.075697
[54]	train-logloss:0.025508	eval-logloss:0.075676
[55]	train-logloss:0.024828	eval-logloss:0.075366
```

```
predicts = xgb_model.predict(dtest)
print(np.round(predicts[:10], 3))
```

```
[1.    0.999 0.001 0.995 0.001 0.989 0.995 0.998 0.966 0.795]
```

```
preds = [ 1 if x > 0.5 else 0 for x in predicts ]
```

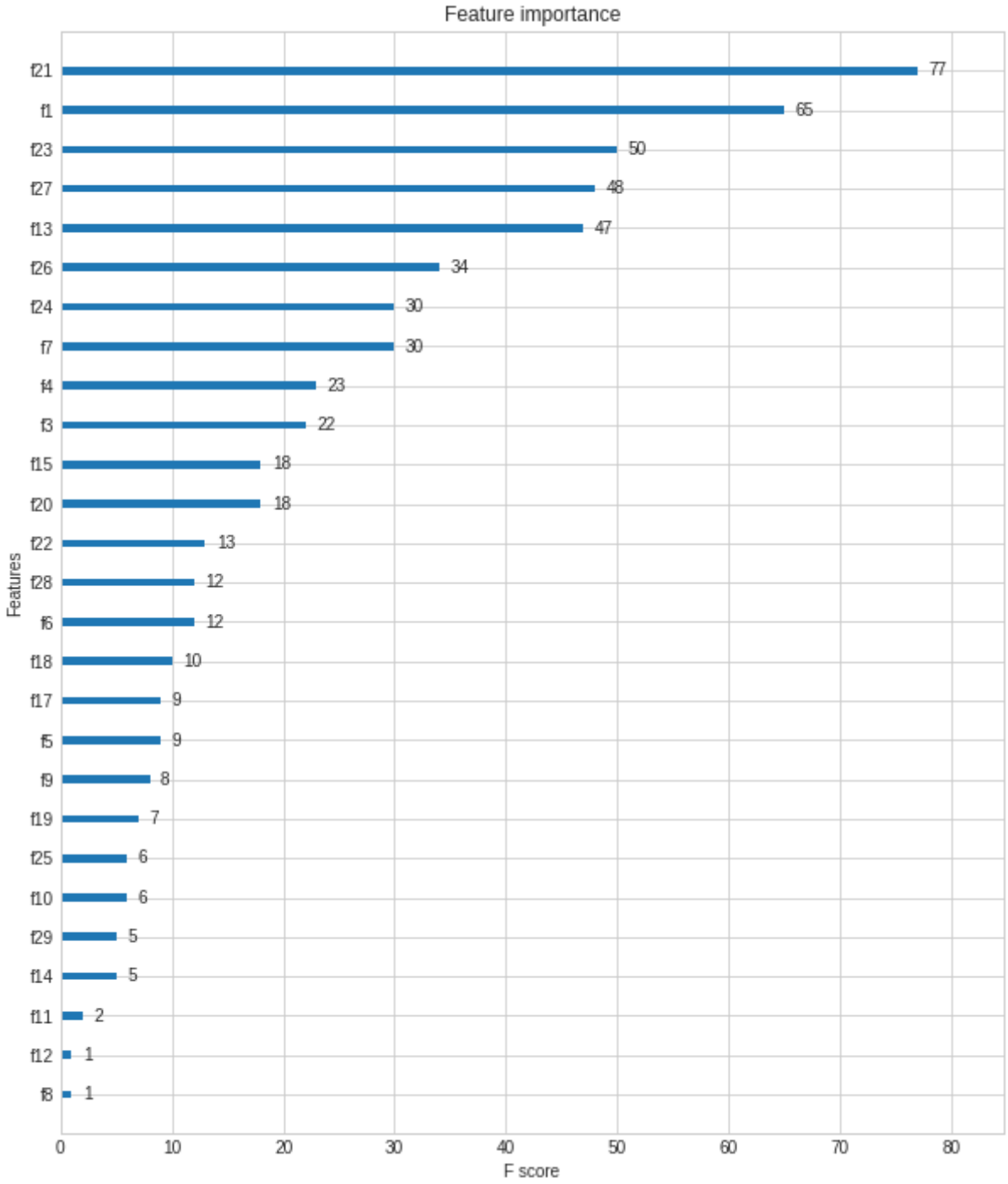
```
print(preds[:10])
```

```
[1, 1, 0, 1, 0, 1, 1, 1, 1, 1]
```

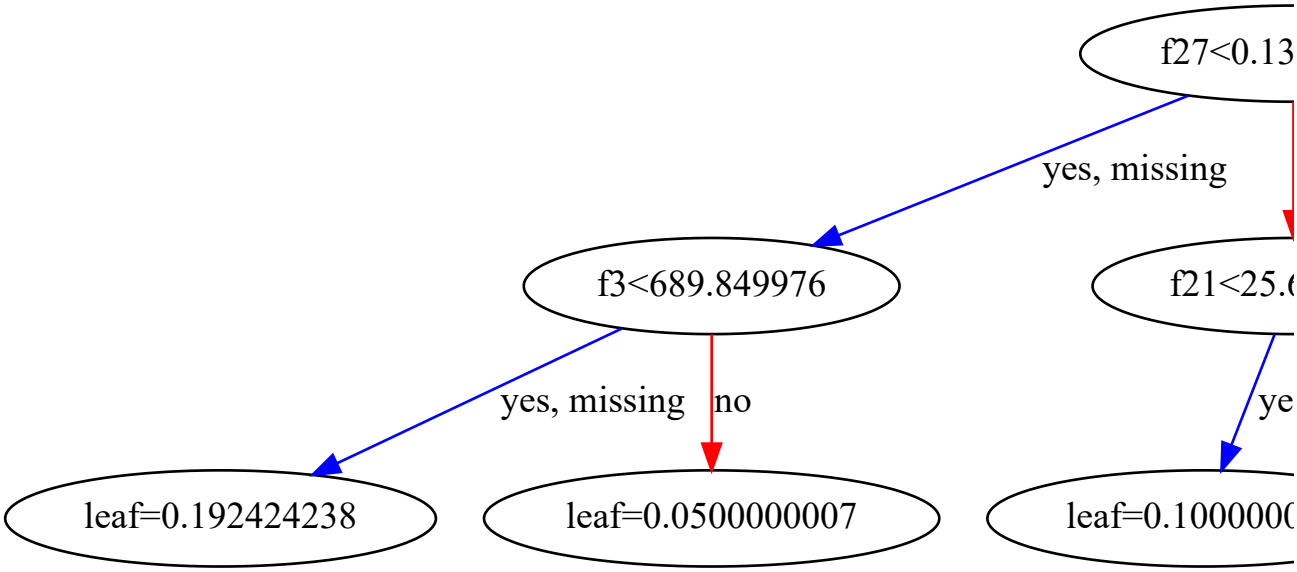
```
print("정확도: {}".format(accuracy_score(y_test, preds)))
print("정밀도: {}".format(precision_score(y_test, preds)))
print("재현율: {}".format(recall_score(y_test, preds)))
```

```
정확도: 0.9736842105263158
정밀도: 0.972972972972973
재현율: 0.9863013698630136
```

```
fig, ax = plt.subplots(figsize=(10, 12))
plot_importance(xgb_model, ax=ax);
```



```
dot_data = xgb.to_graphviz(xgb_model)
graph = graphviz.Source(dot_data)
graph
```



XGBClassifier

붓꽃 데이터

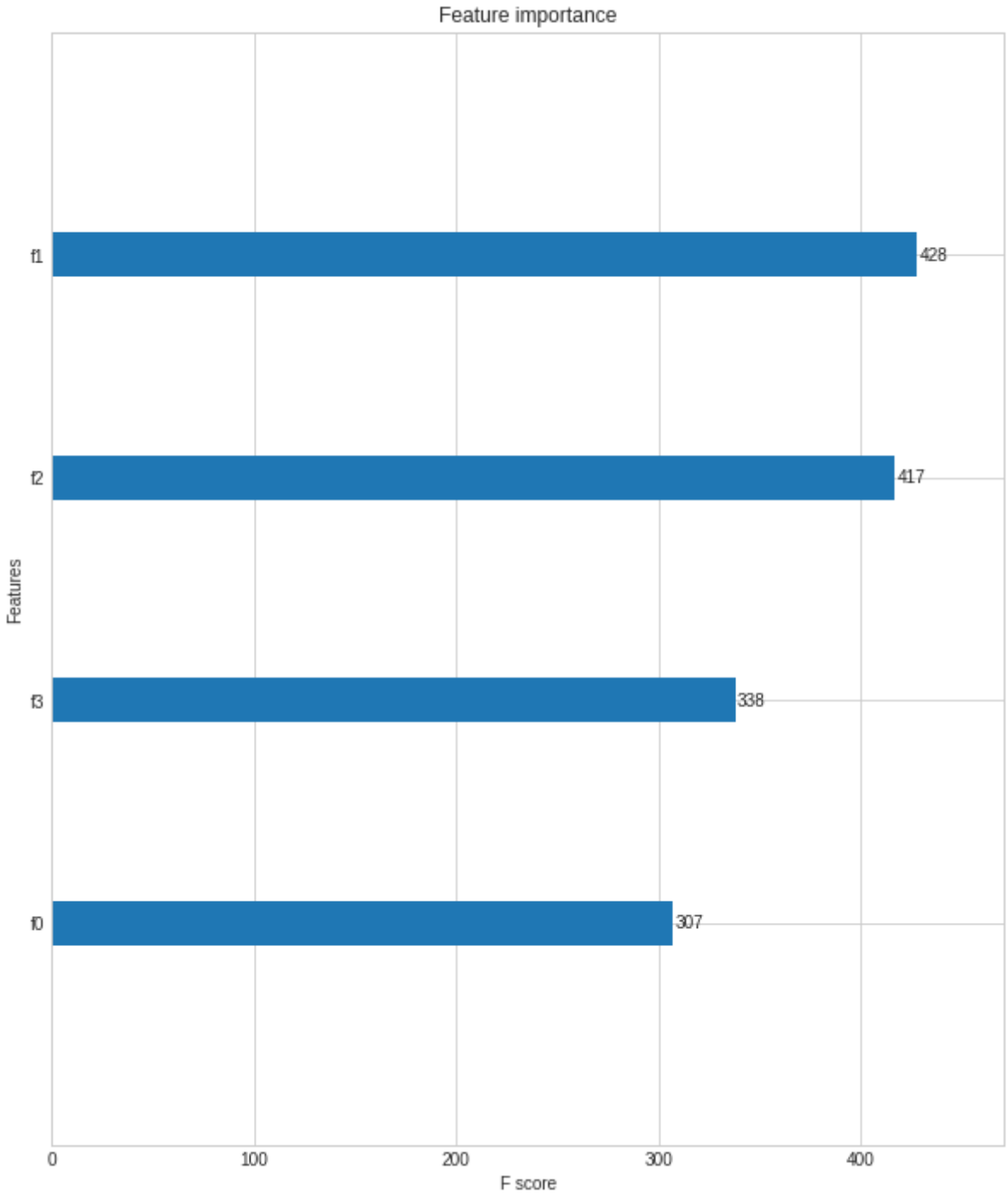
```
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2, random_s
```

```
xgbc = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)
xgbc.fit(X_train, y_train)
preds = xgbc.predict(X_test)
preds_proba = xgbc.predict_proba(X_test)[:, 1]
```

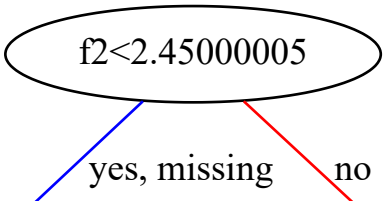
```
cross_val = cross_validate(
    estimator=xgbc,
    X=iris.data, y=iris.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.05291595458984375 (+/- 0.0026833660876727093)
avg score time: 0.0010408878326416016 (+/- 6.877231750645918e-05)
avg test score: 0.96 (+/- 0.024944382578492935)
```

```
fig, ax = plt.subplots(figsize=(10, 12))
plot_importance(xgbc, ax=ax);
```




```
dot_data = xgb.to_graphviz(xgbc)
graph = graphviz.Source(dot_data)
graph
```



와인 데이터

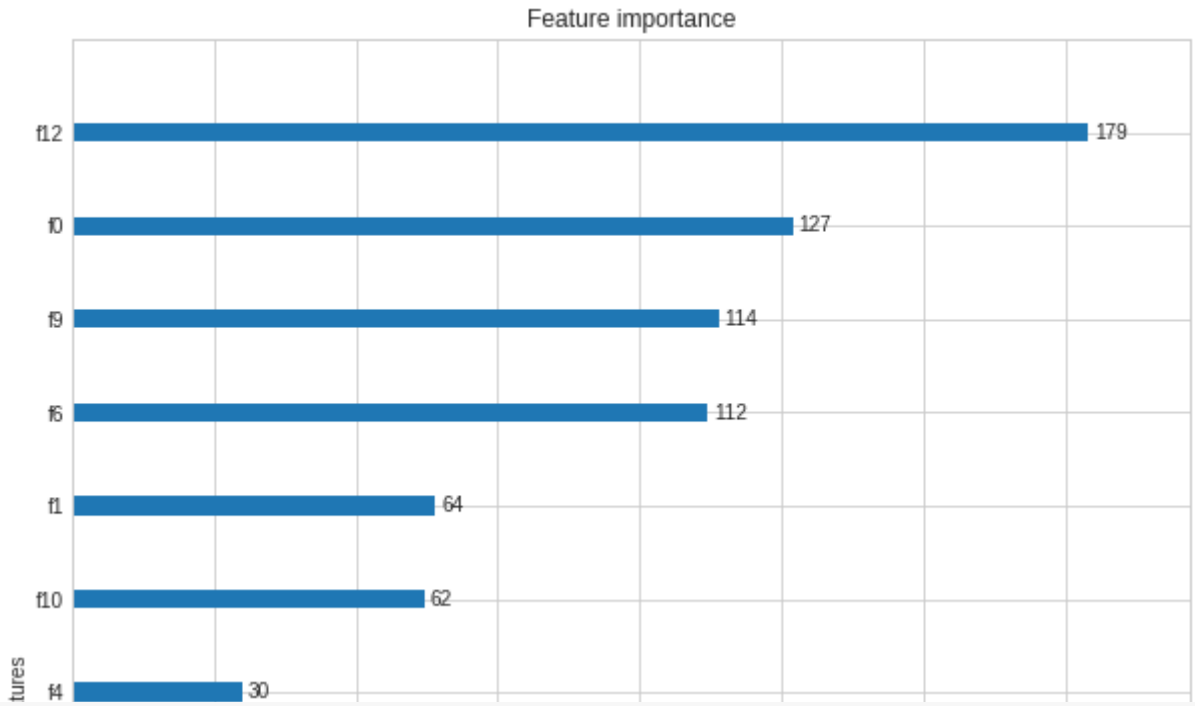
```
wine = load_wine()
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.2, random_s
```

```
xgbc = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)
xgbc.fit(X_train, y_train)
preds = xgbc.predict(X_test)
preds_proba = xgbc.predict_proba(X_test)[:, 1]
```

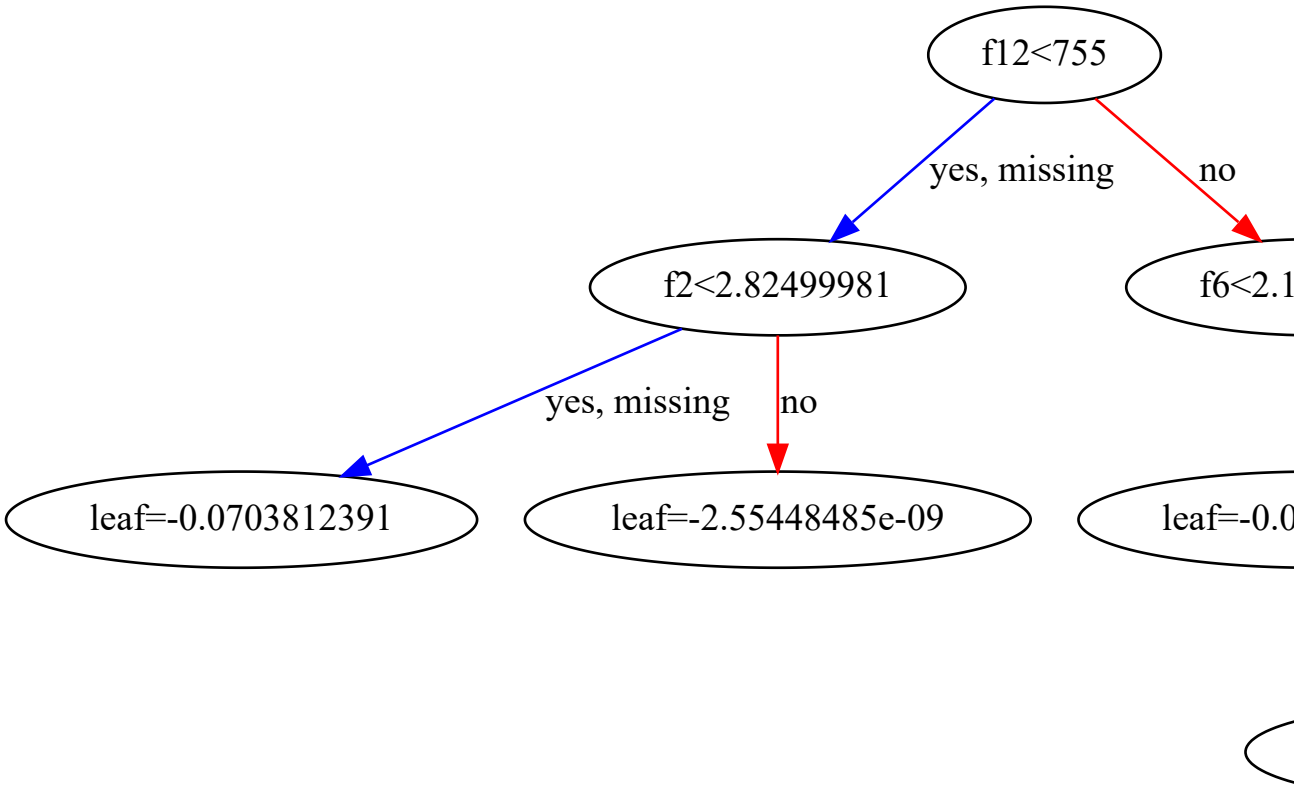
```
cross_val = cross_validate(
    estimator=xgbc,
    X=wine.data, y=wine.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.0889460563659668 (+/- 0.0029411654618832633)
avg score time: 0.0009948253631591798 (+/- 3.610498111208646e-05)
avg test score: 0.9609523809523809 (+/- 0.028267341226138717)
```

```
fig, ax = plt.subplots(figsize=(10, 12))
plot_importance(xgbc, ax=ax);
```




```
dot_data = xgb.to_graphviz(xgbc)
graph = graphviz.Source(dot_data)
graph
```



유방암 데이터

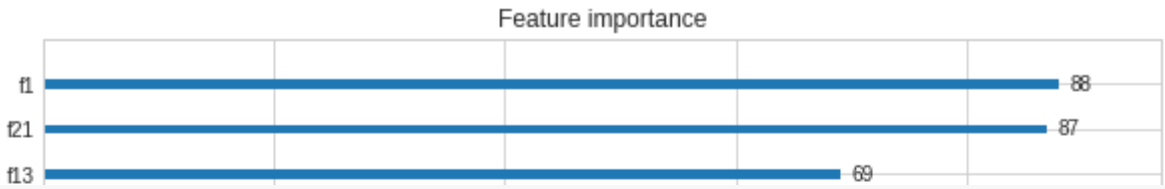
```
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.2, rand
```

```
xgbc = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)
xgbc.fit(X_train, y_train)
preds = xgbc.predict(X_test)
preds_proba = xgbc.predict_proba(X_test)[:, 1]
```

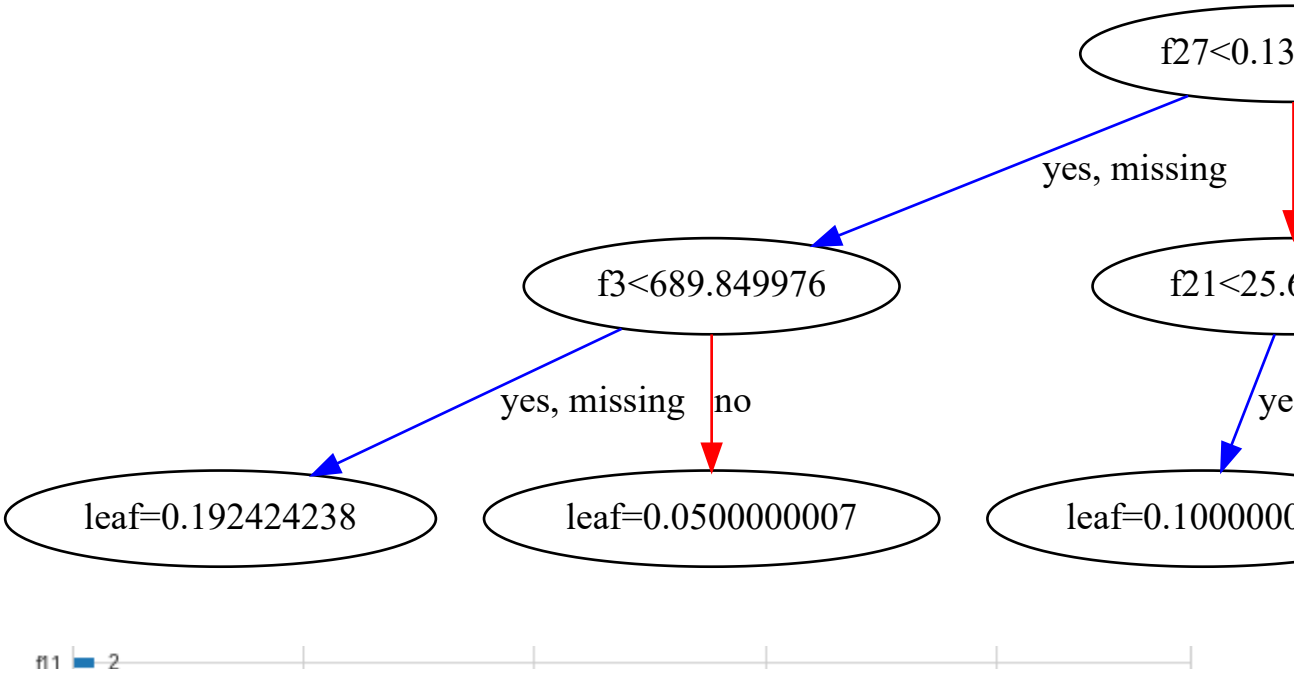
```
cross_val = cross_validate(
    estimator=xgbc,
    X=cancer.data, y=cancer.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.22794322967529296 (+/- 0.01149011045814873)
avg score time: 0.0014575958251953126 (+/- 9.16602824251225e-05)
avg test score: 0.9736376339077782 (+/- 0.009609619188189153)
```

```
fig, ax = plt.subplots(figsize=(10, 12))
plot_importance(xgbc, ax=ax);
```




```
dot_data = xgb.to_graphviz(xgbc)
graph = graphviz.Source(dot_data)
graph
```



XGBRegressor

F score

보스턴 데이터

```
boston = load_boston()
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target, test_size=0.2, rand
```

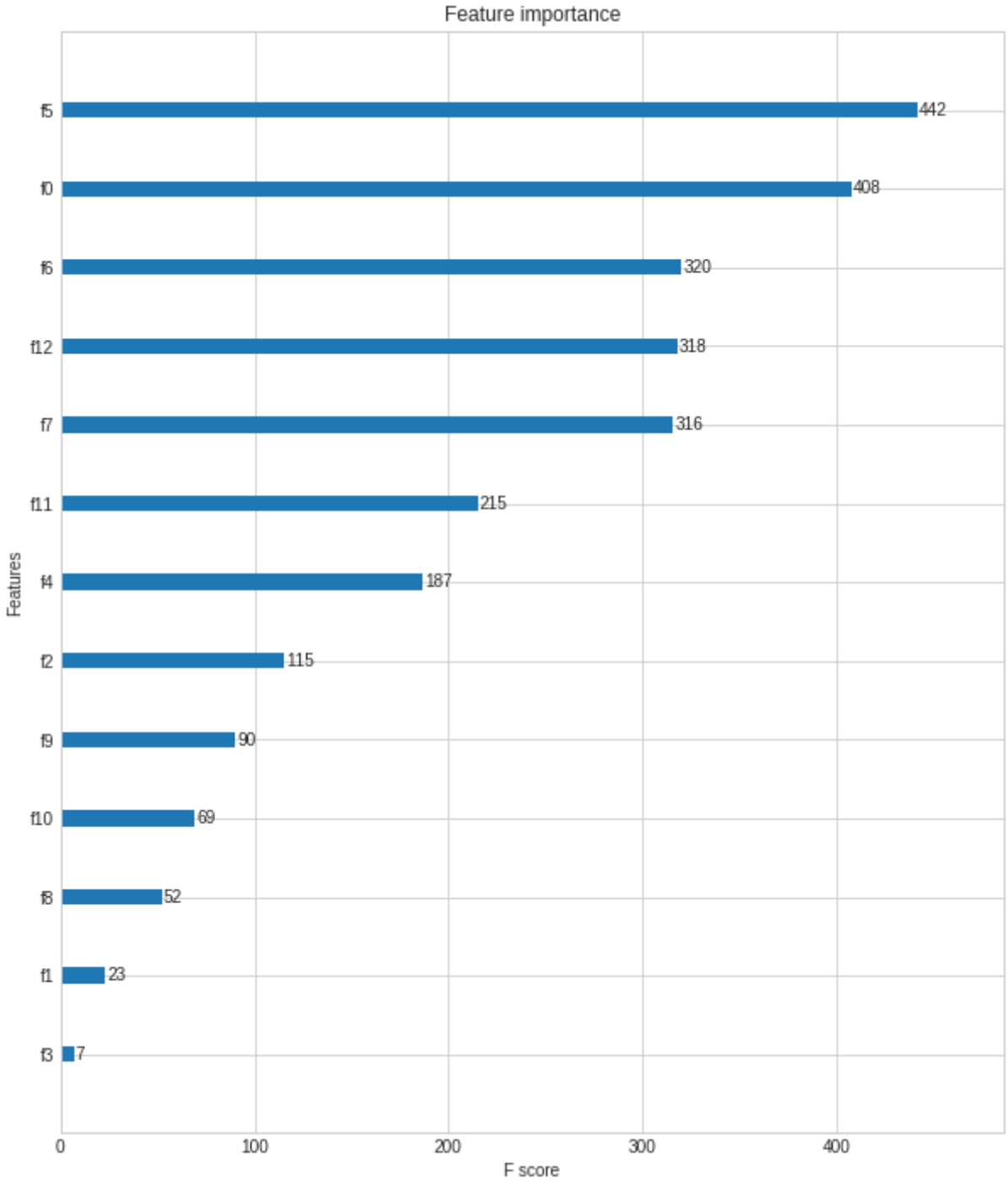
```
xgbr = XGBRegressor(n_estimators=400, learning_rate=0.1, max_depth=3, objective ='reg:squarederror'
xgbr.fit(X_train, y_train)
preds = xgbr.predict(X_test)
```

```
cross_val = cross_validate(
    estimator=xgbr,
    X=boston.data, y=boston.target,
    cv=5
)
```

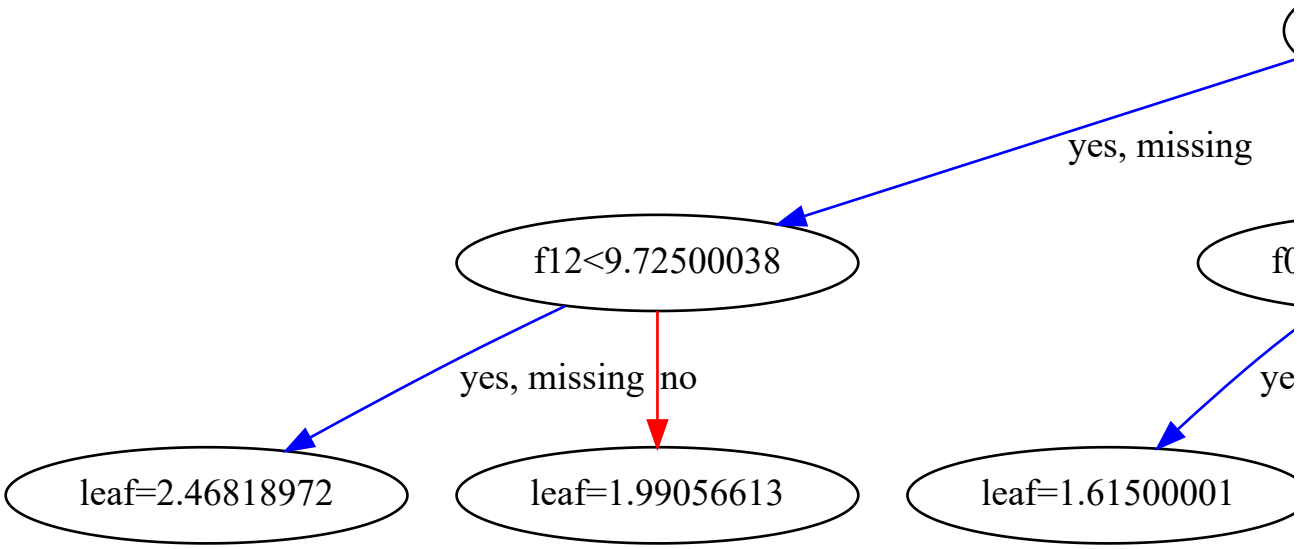
```
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.15248451232910157 (+/- 0.001125980790363763)
avg score time: 0.002347373962402344 (+/- 0.0001847086204840063)
avg test score: 0.6884390572208088 (+/- 0.164997474845101)
```

```
fig, ax = plt.subplots(figsize=(10, 12))
plot_importance(xgbr, ax=ax);
```




```
dot_data = xgb.to_graphviz(xgbr)
graph = graphviz.Source(dot_data)
graph
```



당뇨병 데이터

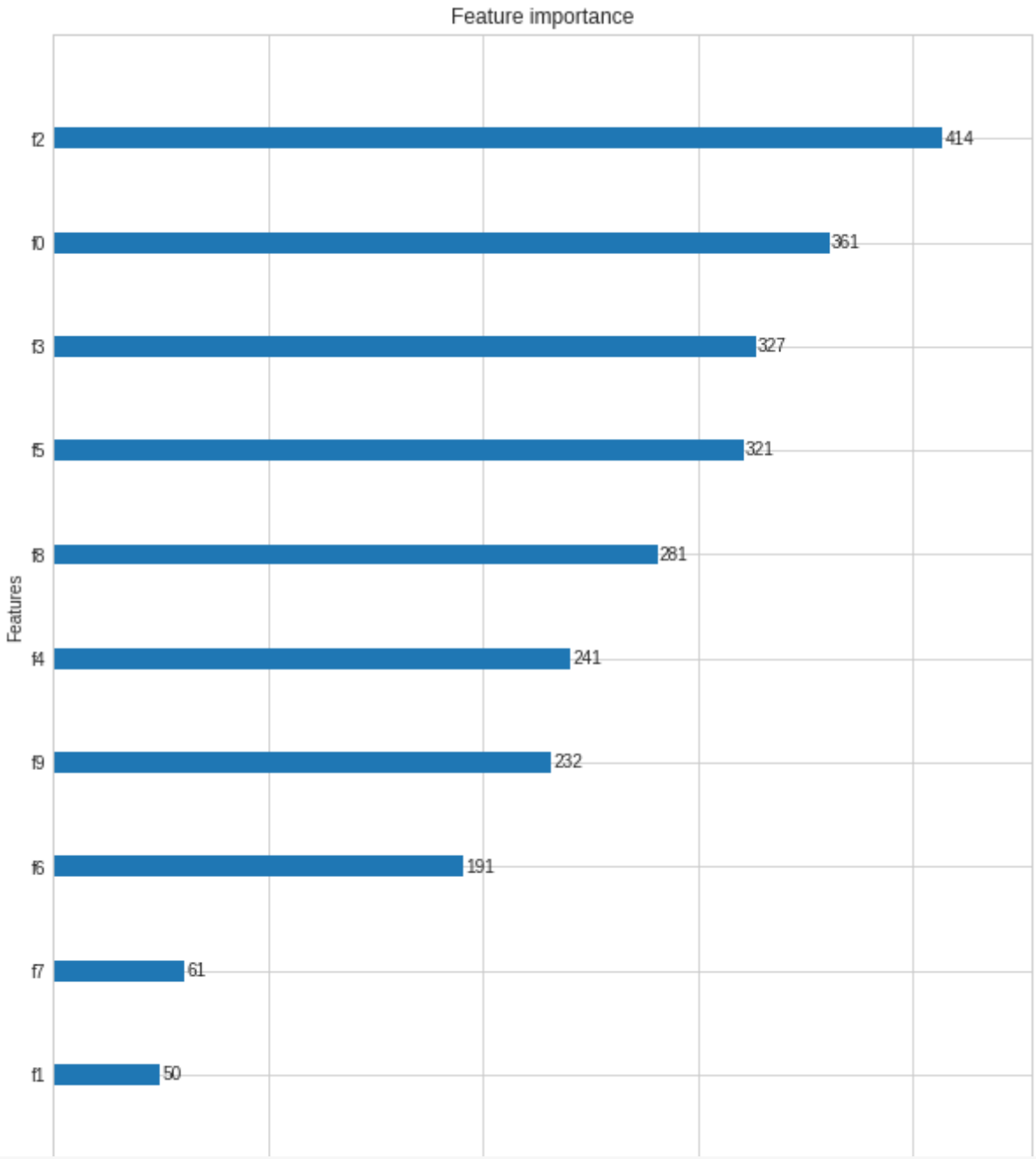
```
diabetes = load_diabetes()
X_train, X_test, y_train, y_test = train_test_split(diabetes.data, diabetes.target, test_size=0.2,
```

```
xgbr = XGBRegressor(n_estimators=400, learning_rate=0.1, max_depth=3, objective ='reg:squarederror'
xgbr.fit(X_train, y_train)
preds = xgbr.predict(X_test)
```

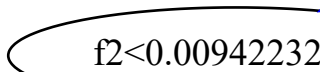
```
cross_val = cross_validate(
    estimator=xgbr,
    X=diabetes.data, y=diabetes.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.12624716758728027 (+/- 0.005968848696494289)
avg score time: 0.0020700931549072266 (+/- 6.22577702004638e-05)
avg test score: 0.3000529025802777 (+/- 0.07589311710543882)
```

```
fig, ax = plt.subplots(figsize=(10, 12))
plot_importance(xgbr, ax=ax);
```

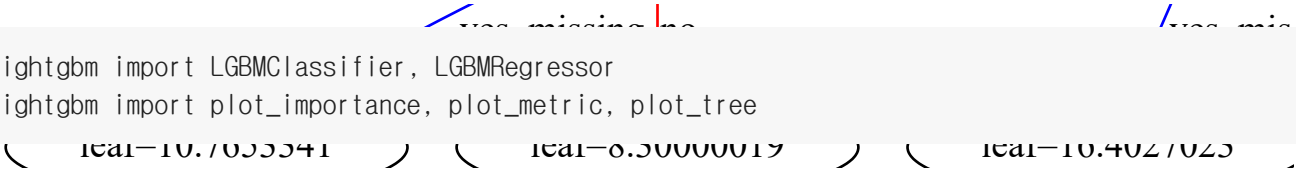



```
dot_data = xgb.to_graphviz(xgbr)
graph = graphviz.Source(dot_data)
graph
```



LightGBM

- 빠른 학습과 예측 시간
- 더 적은 메모리 사용
- 범주형 특징의 자동 변환과 최적 분할




```
from lightgbm import LGBMClassifier, LGBMRegressor
from lightgbm import plot_importance, plot_metric, plot_tree
```

LGBMClassifier

붓꽃 데이터

```
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2, random_s
```

```
lgbmc = LGBMClassifier(n_estimators=400)
evals = [(X_test, y_test)]
lgbmc.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_set=evals, verbo
preds = lgbmc.predict(X_test)
```

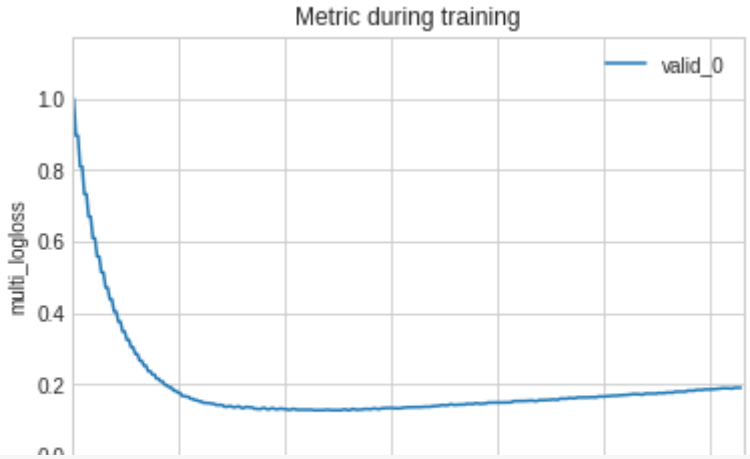
```
[1]     valid_0's multi_logloss: 0.997777       valid_0's multi_logloss: 0.997777
Training until validation scores don't improve for 100 rounds.
[2]     valid_0's multi_logloss: 0.895442       valid_0's multi_logloss: 0.895442
[3]     valid_0's multi_logloss: 0.809103       valid_0's multi_logloss: 0.809103
[4]     valid_0's multi_logloss: 0.732391       valid_0's multi_logloss: 0.732391
[5]     valid_0's multi_logloss: 0.669225       valid_0's multi_logloss: 0.669225
[6]     valid_0's multi_logloss: 0.608976       valid_0's multi_logloss: 0.608976
[7]     valid_0's multi_logloss: 0.557876       valid_0's multi_logloss: 0.557876
[8]     valid_0's multi_logloss: 0.513242       valid_0's multi_logloss: 0.513242
[9]     valid_0's multi_logloss: 0.470866       valid_0's multi_logloss: 0.470866
[10]    valid_0's multi_logloss: 0.437898       valid_0's multi_logloss: 0.437898
[11]    valid_0's multi_logloss: 0.403873       valid_0's multi_logloss: 0.403873
[12]    valid_0's multi_logloss: 0.375711       valid_0's multi_logloss: 0.375711
[13]    valid_0's multi_logloss: 0.348203       valid_0's multi_logloss: 0.348203
[14]    valid_0's multi_logloss: 0.324794       valid_0's multi_logloss: 0.324794
[15]    valid_0's multi_logloss: 0.303965       valid_0's multi_logloss: 0.303965
[16]    valid_0's multi_logloss: 0.285136       valid_0's multi_logloss: 0.285136
[17]    valid_0's multi_logloss: 0.266944       valid_0's multi_logloss: 0.266944
[18]    valid_0's multi_logloss: 0.253009       valid_0's multi_logloss: 0.253009
```

```
[19]    valid_0's multi_logloss: 0.237524       valid_0's multi_logloss: 0.237524
[20]    valid_0's multi_logloss: 0.227865       valid_0's multi_logloss: 0.227865
[21]    valid_0's multi_logloss: 0.216152       valid_0's multi_logloss: 0.216152
[22]    valid_0's multi_logloss: 0.208233       valid_0's multi_logloss: 0.208233
[23]    valid_0's multi_logloss: 0.197981       valid_0's multi_logloss: 0.197981
[24]    valid_0's multi_logloss: 0.191621       valid_0's multi_logloss: 0.191621
[25]    valid_0's multi_logloss: 0.182997       valid_0's multi_logloss: 0.182997
[26]    valid_0's multi_logloss: 0.17707        valid_0's multi_logloss: 0.17707
[27]    valid_0's multi_logloss: 0.168032       valid_0's multi_logloss: 0.168032
[28]    valid_0's multi_logloss: 0.165841       valid_0's multi_logloss: 0.165841
[29]    valid_0's multi_logloss: 0.160077       valid_0's multi_logloss: 0.160077
[30]    valid_0's multi_logloss: 0.156214       valid_0's multi_logloss: 0.156214
[31]    valid_0's multi_logloss: 0.15288        valid_0's multi_logloss: 0.15288
[32]    valid_0's multi_logloss: 0.148715       valid_0's multi_logloss: 0.148715
[33]    valid_0's multi_logloss: 0.148072       valid_0's multi_logloss: 0.148072
[34]    valid_0's multi_logloss: 0.146258       valid_0's multi_logloss: 0.146258
[35]    valid_0's multi_logloss: 0.142771       valid_0's multi_logloss: 0.142771
[36]    valid_0's multi_logloss: 0.142732       valid_0's multi_logloss: 0.142732
[37]    valid_0's multi_logloss: 0.137857       valid_0's multi_logloss: 0.137857
[38]    valid_0's multi_logloss: 0.139432       valid_0's multi_logloss: 0.139432
[39]    valid_0's multi_logloss: 0.136587       valid_0's multi_logloss: 0.136587
[40]    valid_0's multi_logloss: 0.138459       valid_0's multi_logloss: 0.138459
[41]    valid_0's multi_logloss: 0.1344 valid_0's multi_logloss: 0.1344
[42]    valid_0's multi_logloss: 0.137125       valid_0's multi_logloss: 0.137125
[43]    valid_0's multi_logloss: 0.136025       valid_0's multi_logloss: 0.136025
[44]    valid_0's multi_logloss: 0.132475       valid_0's multi_logloss: 0.132475
[45]    valid_0's multi_logloss: 0.131464       valid_0's multi_logloss: 0.131464
[46]    valid_0's multi_logloss: 0.134381       valid_0's multi_logloss: 0.134381
[47]    valid_0's multi_logloss: 0.130702       valid_0's multi_logloss: 0.130702
[48]    valid_0's multi_logloss: 0.133597       valid_0's multi_logloss: 0.133597
[49]    valid_0's multi_logloss: 0.130966       valid_0's multi_logloss: 0.130966
[50]    valid_0's multi_logloss: 0.132577       valid_0's multi_logloss: 0.132577
[51]    valid_0's multi_logloss: 0.131264       valid_0's multi_logloss: 0.131264
[52]    valid_0's multi_logloss: 0.129436       valid_0's multi_logloss: 0.129436
[53]    valid_0's multi_logloss: 0.131788       valid_0's multi_logloss: 0.131788
[54]    valid_0's multi_logloss: 0.129431       valid_0's multi_logloss: 0.129431
[55]    valid_0's multi_logloss: 0.129606       valid_0's multi_logloss: 0.129606
[56]    valid_0's multi_logloss: 0.129778       valid_0's multi_logloss: 0.129778
[57]    valid_0's multi_logloss: 0.129615       valid_0's multi_logloss: 0.129615
[58]    valid_0's multi_logloss: 0.128187       valid_0's multi_logloss: 0.128187
```

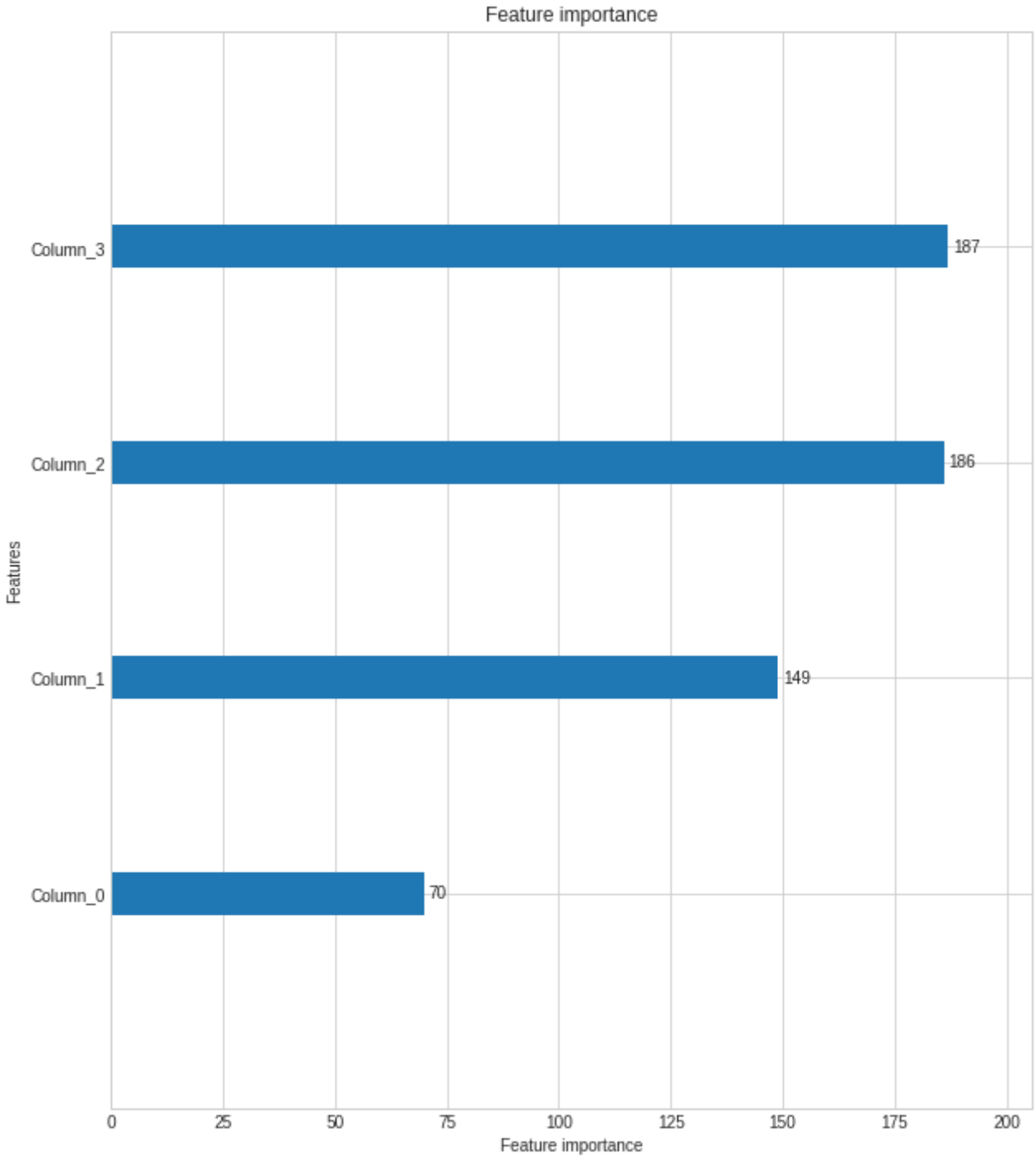
```
cross_val = cross_validate(
    estimator=lgbmc,
    X=iris.data, y=iris.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.05982894897460937 (+/- 0.005069872199959179)
avg score time: 0.0014867782592773438 (+/- 0.00014086047294427018)
avg test score: 0.9533333333333335 (+/- 0.06182412330330468)
```

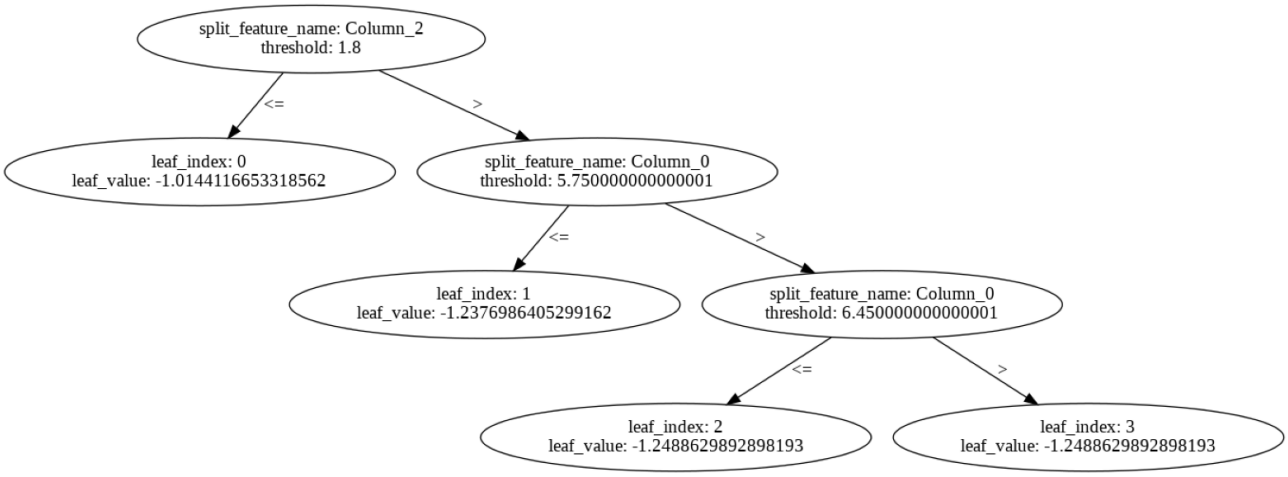
```
plot_metric(lgbmc);
```




```
plot_importance(lgbmc, figsize=(10, 12));
```




```
plot_tree(lgbmc, figsize=(28, 24));
```



와인 데이터

```
wine = load_wine()
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.2, random_s
```

```
lgbmc = LGBMClassifier(n_estimators=400)
evals = [(X_test, y_test)]
lgbmc.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_set=evals, verbo
preds = lgbmc.predict(X_test)
```

```
[1]	valid_0's multi_logloss: 1.09587	valid_0's multi_logloss: 1.09587
Training until validation scores don't improve for 100 rounds.
[2]	valid_0's multi_logloss: 1.00013	valid_0's multi_logloss: 1.00013
[3]	valid_0's multi_logloss: 0.913552	valid_0's multi_logloss: 0.913552
[4]	valid_0's multi_logloss: 0.843464	valid_0's multi_logloss: 0.843464
[5]	valid_0's multi_logloss: 0.781558	valid_0's multi_logloss: 0.781558
[6]	valid_0's multi_logloss: 0.716933	valid_0's multi_logloss: 0.716933
[7]	valid_0's multi_logloss: 0.656888	valid_0's multi_logloss: 0.656888
[8]	valid_0's multi_logloss: 0.61532	valid_0's multi_logloss: 0.61532
[9]	valid_0's multi_logloss: 0.574294	valid_0's multi_logloss: 0.574294
[10]	valid_0's multi_logloss: 0.533566	valid_0's multi_logloss: 0.533566
[11]	valid_0's multi_logloss: 0.496561	valid_0's multi_logloss: 0.496561
[12]	valid_0's multi_logloss: 0.464202	valid_0's multi_logloss: 0.464202
[13]	valid_0's multi_logloss: 0.430112	valid_0's multi_logloss: 0.430112
[14]	valid_0's multi_logloss: 0.402736	valid_0's multi_logloss: 0.402736
[15]	valid_0's multi_logloss: 0.380988	valid_0's multi_logloss: 0.380988
[16]	valid_0's multi_logloss: 0.355402	valid_0's multi_logloss: 0.355402
[17]	valid_0's multi_logloss: 0.335661	valid_0's multi_logloss: 0.335661
[18]	valid_0's multi_logloss: 0.315222	valid_0's multi_logloss: 0.315222
[19]	valid_0's multi_logloss: 0.296297	valid_0's multi_logloss: 0.296297
[20]	valid_0's multi_logloss: 0.27677	valid_0's multi_logloss: 0.27677
[21]	valid_0's multi_logloss: 0.263501	valid_0's multi_logloss: 0.263501
[22]	valid_0's multi_logloss: 0.247876	valid_0's multi_logloss: 0.247876
```

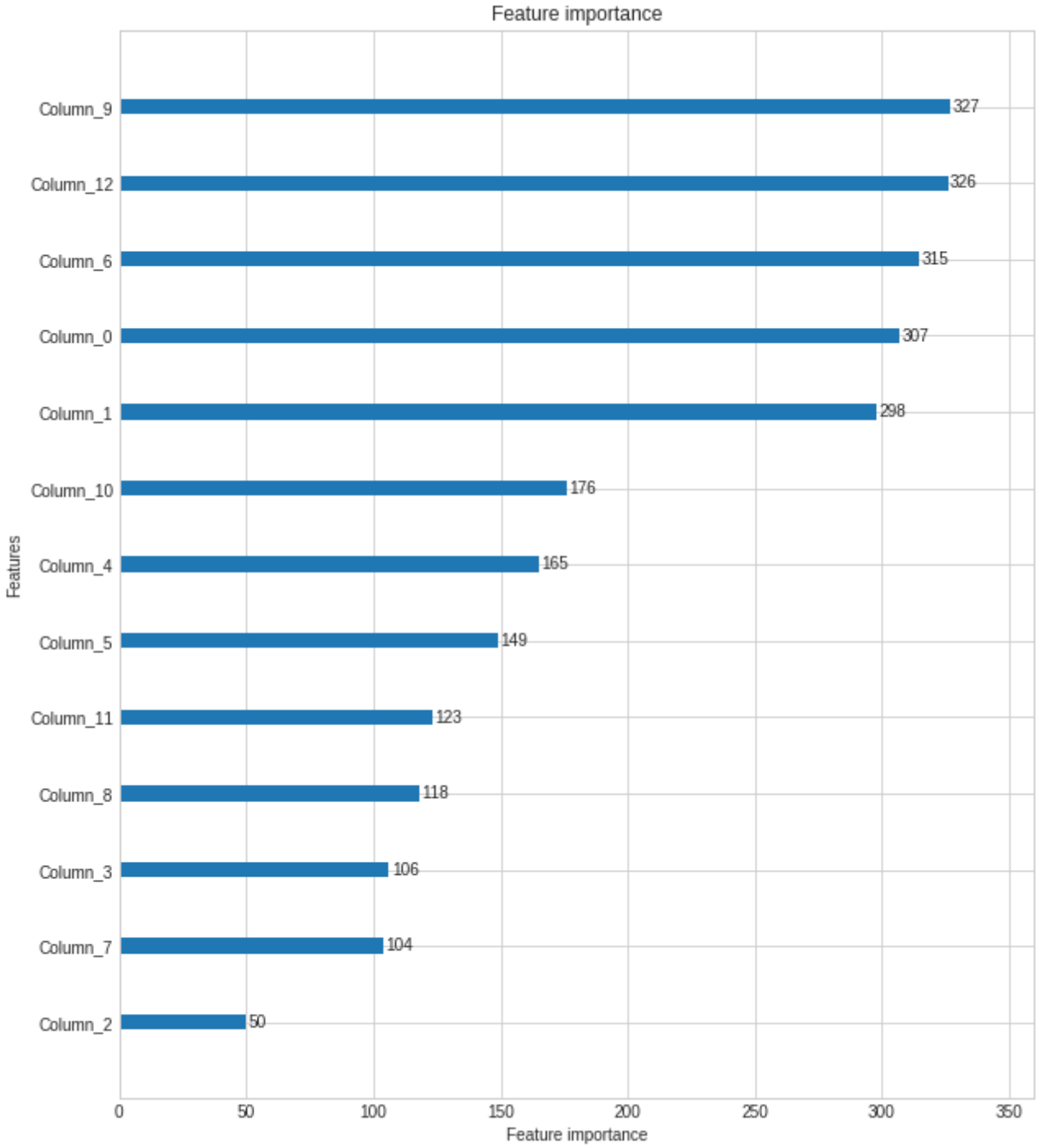
```
[23]    valid_0's multi_logloss: 0.232118       valid_0's multi_logloss: 0.232118
[24]    valid_0's multi_logloss: 0.220981       valid_0's multi_logloss: 0.220981
[25]    valid_0's multi_logloss: 0.210079       valid_0's multi_logloss: 0.210079
[26]    valid_0's multi_logloss: 0.201183       valid_0's multi_logloss: 0.201183
[27]    valid_0's multi_logloss: 0.189384       valid_0's multi_logloss: 0.189384
[28]    valid_0's multi_logloss: 0.180099       valid_0's multi_logloss: 0.180099
[29]    valid_0's multi_logloss: 0.173239       valid_0's multi_logloss: 0.173239
[30]    valid_0's multi_logloss: 0.164453       valid_0's multi_logloss: 0.164453
[31]    valid_0's multi_logloss: 0.156619       valid_0's multi_logloss: 0.156619
[32]    valid_0's multi_logloss: 0.150649       valid_0's multi_logloss: 0.150649
[33]    valid_0's multi_logloss: 0.142388       valid_0's multi_logloss: 0.142388
[34]    valid_0's multi_logloss: 0.136239       valid_0's multi_logloss: 0.136239
[35]    valid_0's multi_logloss: 0.132299       valid_0's multi_logloss: 0.132299
[36]    valid_0's multi_logloss: 0.126211       valid_0's multi_logloss: 0.126211
[37]    valid_0's multi_logloss: 0.118597       valid_0's multi_logloss: 0.118597
[38]    valid_0's multi_logloss: 0.115265       valid_0's multi_logloss: 0.115265
[39]    valid_0's multi_logloss: 0.108623       valid_0's multi_logloss: 0.108623
[40]    valid_0's multi_logloss: 0.102767       valid_0's multi_logloss: 0.102767
[41]    valid_0's multi_logloss: 0.0988445      valid_0's multi_logloss: 0.0988445
[42]    valid_0's multi_logloss: 0.0936784      valid_0's multi_logloss: 0.0936784
[43]    valid_0's multi_logloss: 0.0897994      valid_0's multi_logloss: 0.0897994
[44]    valid_0's multi_logloss: 0.0887182      valid_0's multi_logloss: 0.0887182
[45]    valid_0's multi_logloss: 0.083452       valid_0's multi_logloss: 0.083452
[46]    valid_0's multi_logloss: 0.0800268      valid_0's multi_logloss: 0.0800268
[47]    valid_0's multi_logloss: 0.0770026      valid_0's multi_logloss: 0.0770026
[48]    valid_0's multi_logloss: 0.0738369      valid_0's multi_logloss: 0.0738369
[49]    valid_0's multi_logloss: 0.0714092      valid_0's multi_logloss: 0.0714092
[50]    valid_0's multi_logloss: 0.0681705      valid_0's multi_logloss: 0.0681705
[51]    valid_0's multi_logloss: 0.0652509      valid_0's multi_logloss: 0.0652509
[52]    valid_0's multi_logloss: 0.0631447      valid_0's multi_logloss: 0.0631447
[53]    valid_0's multi_logloss: 0.0604264      valid_0's multi_logloss: 0.0604264
[54]    valid_0's multi_logloss: 0.0576571      valid_0's multi_logloss: 0.0576571
[55]    valid_0's multi_logloss: 0.0571832      valid_0's multi_logloss: 0.0571832
[56]    valid_0's multi_logloss: 0.0533282      valid_0's multi_logloss: 0.0533282
[57]    valid_0's multi_logloss: 0.0517798      valid_0's multi_logloss: 0.0517798
[58]    valid_0's multi_logloss: 0.0498855      valid_0's multi_logloss: 0.0498855
```

```
cross_val = cross_validate(
    estimator=lgbmc,
    X=wine.data, y=wine.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

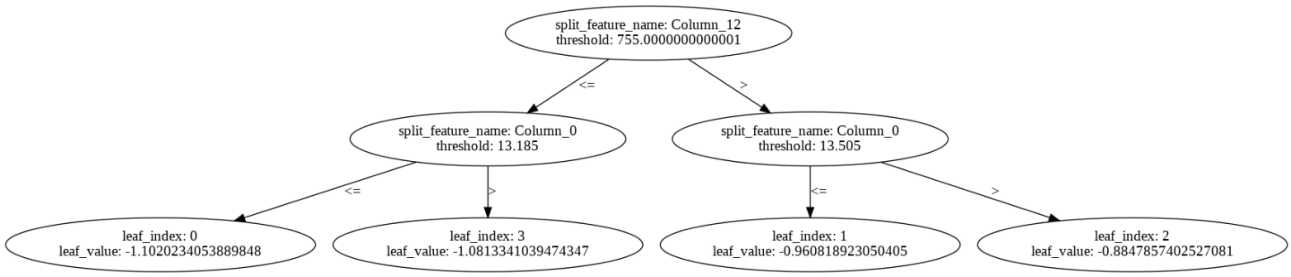
```
avg fit time: 0.07416844367980957 (+/- 0.007060089115089727)
avg score time: 0.001535654067993164 (+/- 0.00023176478647171997)
avg test score: 0.9720634920634922 (+/- 0.030430686929136006)
```

```
plot_metric(lgbmc);
```


```
plot_importance(lgbmc, figsize=(10, 12));
```




```
plot_tree(lgbmc, figsize=(28, 24));
```



유방암 데이터

```
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.2, rand
```

```
lgbmc = LGBMClassifier(n_estimators=400)
evals = [(X_test, y_test)]
lgbmc.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_set=evals, verbo
preds = lgbmc.predict(X_test)
```

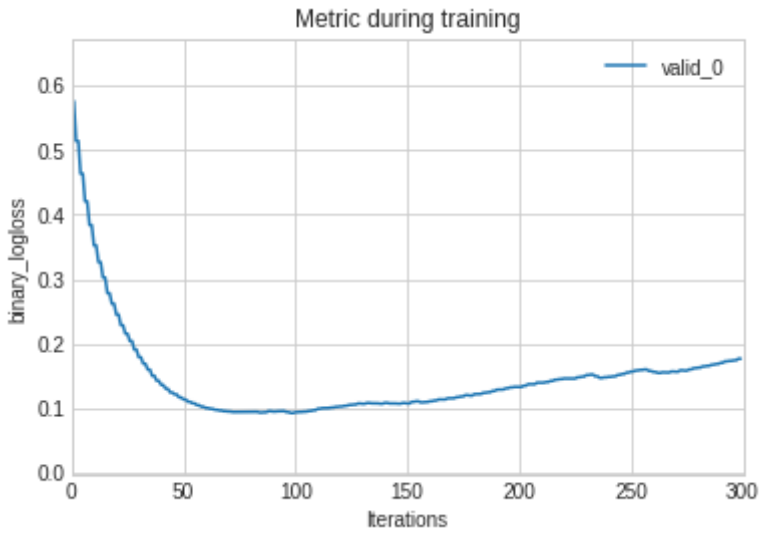
```
[1]	valid_0's binary_logloss: 0.575665	valid_0's binary_logloss: 0.575665
Training until validation scores don't improve for 100 rounds.
[2]	valid_0's binary_logloss: 0.513919	valid_0's binary_logloss: 0.513919
[3]	valid_0's binary_logloss: 0.463418	valid_0's binary_logloss: 0.463418
[4]	valid_0's binary_logloss: 0.420506	valid_0's binary_logloss: 0.420506
[5]	valid_0's binary_logloss: 0.383753	valid_0's binary_logloss: 0.383753
[6]	valid_0's binary_logloss: 0.352228	valid_0's binary_logloss: 0.352228
[7]	valid_0's binary_logloss: 0.326437	valid_0's binary_logloss: 0.326437
[8]	valid_0's binary_logloss: 0.302562	valid_0's binary_logloss: 0.302562
[9]	valid_0's binary_logloss: 0.278182	valid_0's binary_logloss: 0.278182
[10]	valid_0's binary_logloss: 0.261933	valid_0's binary_logloss: 0.261933
[11]	valid_0's binary_logloss: 0.245115	valid_0's binary_logloss: 0.245115
[12]	valid_0's binary_logloss: 0.228615	valid_0's binary_logloss: 0.228615
[13]	valid_0's binary_logloss: 0.215537	valid_0's binary_logloss: 0.215537
[14]	valid_0's binary_logloss: 0.204222	valid_0's binary_logloss: 0.204222
[15]	valid_0's binary_logloss: 0.190866	valid_0's binary_logloss: 0.190866
[16]	valid_0's binary_logloss: 0.179044	valid_0's binary_logloss: 0.179044
[17]	valid_0's binary_logloss: 0.169045	valid_0's binary_logloss: 0.169045
[18]	valid_0's binary_logloss: 0.159975	valid_0's binary_logloss: 0.159975
[19]	valid_0's binary_logloss: 0.15035	valid_0's binary_logloss: 0.15035
[20]	valid_0's binary_logloss: 0.142652	valid_0's binary_logloss: 0.142652
[21]	valid_0's binary_logloss: 0.136131	valid_0's binary_logloss: 0.136131
[22]	valid_0's binary_logloss: 0.130356	valid_0's binary_logloss: 0.130356
[23]	valid_0's binary_logloss: 0.124556	valid_0's binary_logloss: 0.124556
[24]	valid_0's binary_logloss: 0.12149	valid_0's binary_logloss: 0.12149
[25]	valid_0's binary_logloss: 0.116817	valid_0's binary_logloss: 0.116817
[26]	valid_0's binary_logloss: 0.113633	valid_0's binary_logloss: 0.113633
[27]	valid_0's binary_logloss: 0.110017	valid_0's binary_logloss: 0.110017
[28]	valid_0's binary_logloss: 0.10766	valid_0's binary_logloss: 0.10766
[29]	valid_0's binary_logloss: 0.104768	valid_0's binary_logloss: 0.104768
[30]	valid_0's binary_logloss: 0.102319	valid_0's binary_logloss: 0.102319
[31]	valid_0's binary_logloss: 0.100305	valid_0's binary_logloss: 0.100305
[32]	valid_0's binary_logloss: 0.0991288	valid_0's binary_logloss: 0.0991288
[33]	valid_0's binary_logloss: 0.097612	valid_0's binary_logloss: 0.097612
[34]	valid_0's binary_logloss: 0.0965774	valid_0's binary_logloss: 0.0965774
```

```
[35]    valid_0's binary_logloss: 0.0955052    valid_0's binary_logloss: 0.0955052
[36]    valid_0's binary_logloss: 0.0949672    valid_0's binary_logloss: 0.0949672
[37]    valid_0's binary_logloss: 0.094184     valid_0's binary_logloss: 0.094184
[38]    valid_0's binary_logloss: 0.0942163    valid_0's binary_logloss: 0.0942163
[39]    valid_0's binary_logloss: 0.0940062    valid_0's binary_logloss: 0.0940062
[40]    valid_0's binary_logloss: 0.0943151    valid_0's binary_logloss: 0.0943151
[41]    valid_0's binary_logloss: 0.0944416    valid_0's binary_logloss: 0.0944416
[42]    valid_0's binary_logloss: 0.0945103    valid_0's binary_logloss: 0.0945103
[43]    valid_0's binary_logloss: 0.0934388    valid_0's binary_logloss: 0.0934388
[44]    valid_0's binary_logloss: 0.0938111    valid_0's binary_logloss: 0.0938111
[45]    valid_0's binary_logloss: 0.095569     valid_0's binary_logloss: 0.095569
[46]    valid_0's binary_logloss: 0.0948761    valid_0's binary_logloss: 0.0948761
[47]    valid_0's binary_logloss: 0.0953523    valid_0's binary_logloss: 0.0953523
[48]    valid_0's binary_logloss: 0.0956434    valid_0's binary_logloss: 0.0956434
[49]    valid_0's binary_logloss: 0.0941128    valid_0's binary_logloss: 0.0941128
[50]    valid_0's binary_logloss: 0.0926712    valid_0's binary_logloss: 0.0926712
[51]    valid_0's binary_logloss: 0.0939158    valid_0's binary_logloss: 0.0939158
[52]    valid_0's binary_logloss: 0.0943063    valid_0's binary_logloss: 0.0943063
[53]    valid_0's binary_logloss: 0.0947826    valid_0's binary_logloss: 0.0947826
[54]    valid_0's binary_logloss: 0.0959015    valid_0's binary_logloss: 0.0959015
[55]    valid_0's binary_logloss: 0.0969547    valid_0's binary_logloss: 0.0969547
[56]    valid_0's binary_logloss: 0.0992705    valid_0's binary_logloss: 0.0992705
[57]    valid_0's binary_logloss: 0.100129     valid_0's binary_logloss: 0.100129
[58]    valid_0's binary_logloss: 0.100378     valid_0's binary_logloss: 0.100378
```

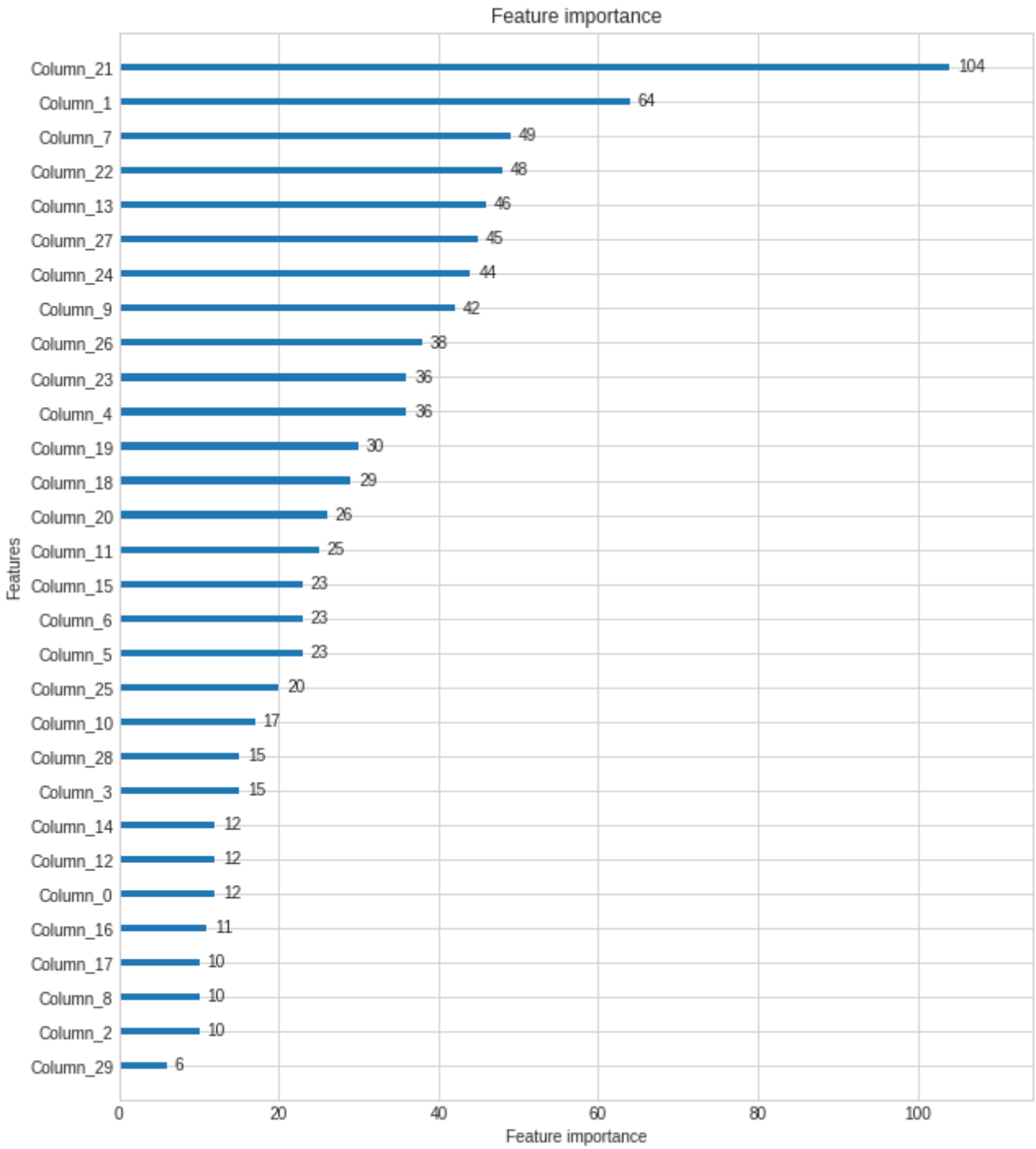
```
cross_val = cross_validate(
    estimator=lgbmc,
    X=cancer.data, y=cancer.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.23347859382629393 (+/- 0.009549878827050216)
avg score time: 0.0026559352874755858 (+/- 0.00033745595555550673)
avg test score: 0.9701288619779536 (+/- 0.0180536992368202)
```

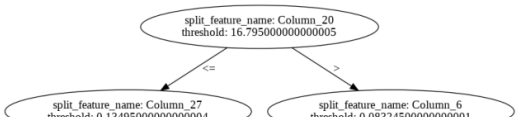
```
plot_metric(lgbmc);
```




```
plot_importance(lgbmc, figsize=(10, 12));
```




```
plot_tree(lgbmc, figsize=(28, 24));
```



LGBMRegressor

보스턴 데이터

```
boston = load_boston()
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target, test_size=0.2, rand
```

```
lgbmr = LGBMRegressor(n_estimators=400)
evals = [(X_test, y_test)]
lgbmr.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_set=evals, verbo
preds = lgbmr.predict(X_test)
```

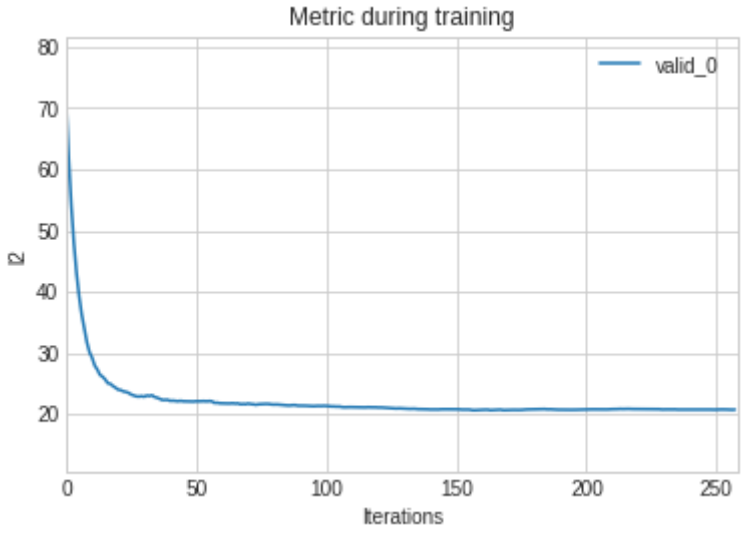
```
[1]	valid_0's l2: 71.4897
Training until validation scores don't improve for 100 rounds.
[2]	valid_0's l2: 61.8603
[3]	valid_0's l2: 54.0848
[4]	valid_0's l2: 47.9263
[5]	valid_0's l2: 43.0702
[6]	valid_0's l2: 39.268
[7]	valid_0's l2: 36.2667
[8]	valid_0's l2: 33.8931
[9]	valid_0's l2: 31.7028
[10]	valid_0's l2: 30.082
[11]	valid_0's l2: 29.1638
[12]	valid_0's l2: 27.9059
[13]	valid_0's l2: 27.2503
[14]	valid_0's l2: 26.3942
[15]	valid_0's l2: 26.0862
[16]	valid_0's l2: 25.6483
[17]	valid_0's l2: 25.0221
[18]	valid_0's l2: 24.919
[19]	valid_0's l2: 24.5501
[20]	valid_0's l2: 24.2858
[21]	valid_0's l2: 23.9636
[22]	valid_0's l2: 23.8872
[23]	valid_0's l2: 23.676
[24]	valid_0's l2: 23.5907
[25]	valid_0's l2: 23.4353
[26]	valid_0's l2: 23.1653
[27]	valid_0's l2: 23.0198
[28]	valid_0's l2: 22.8605
[29]	valid_0's l2: 22.8311
[30]	valid_0's l2: 22.8943
[31]	valid_0's l2: 22.8056
[32]	valid_0's l2: 22.9915
[33]	valid_0's l2: 22.9286
[34]	valid_0's l2: 23.0395
[35]	valid_0's l2: 22.76
[36]	valid_0's l2: 22.6387
[37]	valid_0's l2: 22.4443
```

```
[38]    valid_0's l2: 22.2704
[39]    valid_0's l2: 22.3006
[40]    valid_0's l2: 22.2952
[41]    valid_0's l2: 22.1307
[42]    valid_0's l2: 22.1636
[43]    valid_0's l2: 22.1141
[44]    valid_0's l2: 22.0672
[45]    valid_0's l2: 22.1321
[46]    valid_0's l2: 22.0597
[47]    valid_0's l2: 22.0474
[48]    valid_0's l2: 22.0256
[49]    valid_0's l2: 22.035
[50]    valid_0's l2: 22.0159
[51]    valid_0's l2: 22.065
[52]    valid_0's l2: 22.0541
[53]    valid_0's l2: 22.0814
[54]    valid_0's l2: 22.0893
[55]    valid_0's l2: 22.0479
[56]    valid_0's l2: 22.1003
[57]    valid_0's l2: 22.0557
[58]    valid_0's l2: 21.7922
```

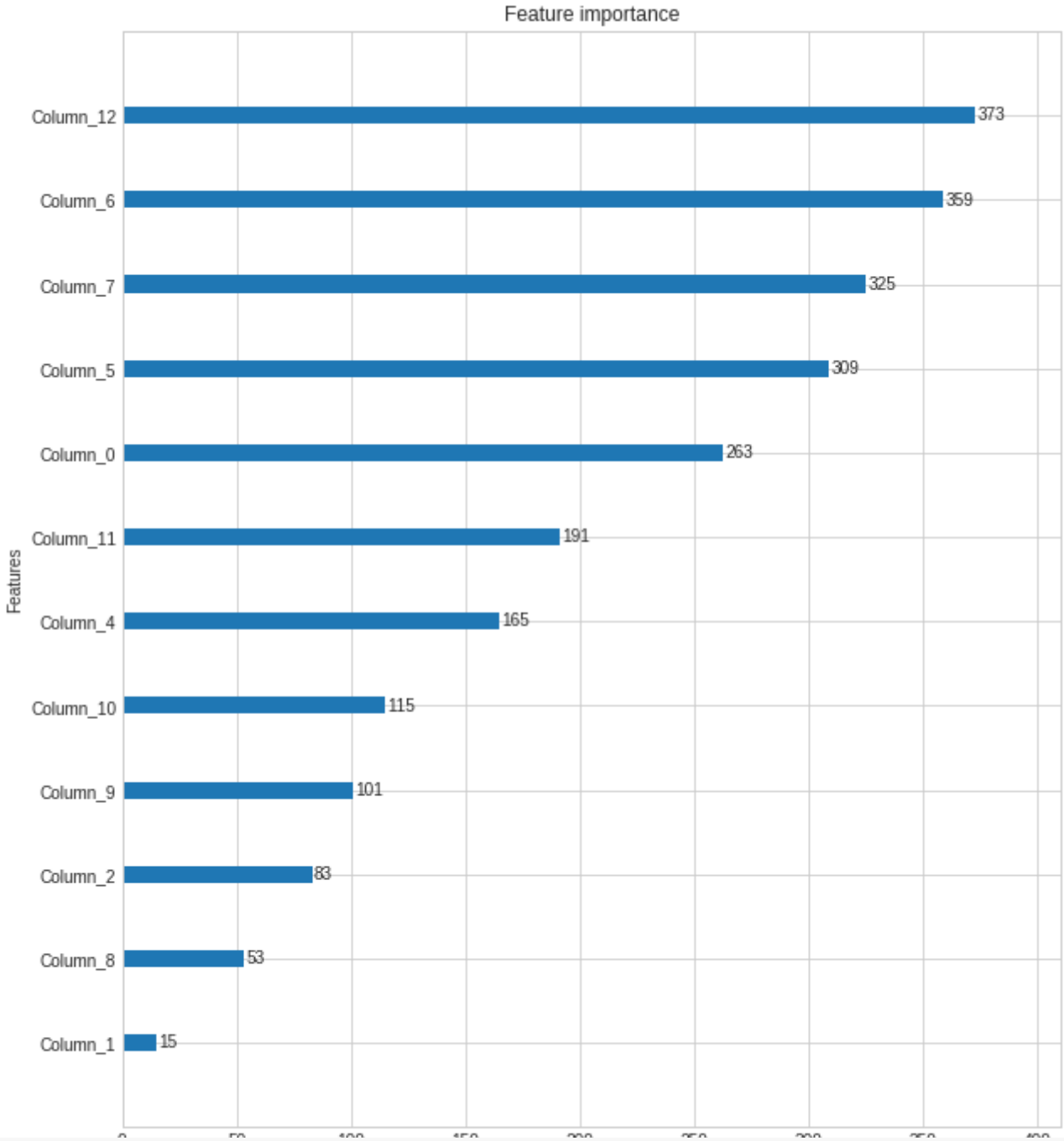
```
cross_val = cross_validate(
    estimator=lgbmr,
    X=boston.data, y=boston.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.180123233795166 (+/- 0.012205847208333558)
avg score time: 0.003948497772216797 (+/- 0.0006217747813506834)
avg test score: 0.5692468252571979 (+/- 0.2956636613238221)
```

```
plot_metric(lgbmr);
```




```
plot_importance(lgbmr, figsize=(10, 12));
```




```
plot_tree(lgbmr, figsize=(28, 24));
```

당뇨병 데이터

```
diabetes = load_diabetes()
X_train, X_test, y_train, y_test = train_test_split(diabetes.data, diabetes.target, test_size=0.2,
```

```
lgbmr = LGBMRegressor(n_estimators=400)
evals = [(X_test, y_test)]
lgbmr.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_set=evals, verbo
preds = lgbmr.predict(X_test)
```

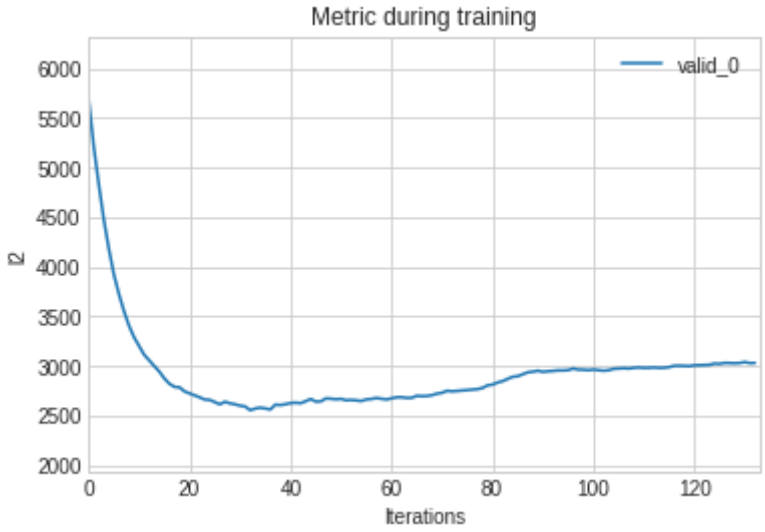
```
[1]     valid_0's l2: 5692.17
Training until validation scores don't improve for 100 rounds.
[2]     valid_0's l2: 5218.79
[3]     valid_0's l2: 4835.34
[4]     valid_0's l2: 4479.84
[5]     valid_0's l2: 4183.42
[6]     valid_0's l2: 3924.35
[7]     valid_0's l2: 3726.96
[8]     valid_0's l2: 3555.3
[9]     valid_0's l2: 3407.95
[10]    valid_0's l2: 3291.33
[11]    valid_0's l2: 3201.78
[12]    valid_0's l2: 3114.69
[13]    valid_0's l2: 3059.18
[14]    valid_0's l2: 3005.79
[15]    valid_0's l2: 2945.18
[16]    valid_0's l2: 2876.37
[17]    valid_0's l2: 2822.71
[18]    valid_0's l2: 2791.31
[19]    valid_0's l2: 2786.39
[20]    valid_0's l2: 2745.25
[21]    valid_0's l2: 2724.07
[22]    valid_0's l2: 2705.17
[23]    valid_0's l2: 2685.41
[24]    valid_0's l2: 2663.2
[25]    valid_0's l2: 2659.75
[26]    valid_0's l2: 2636.48
[27]    valid_0's l2: 2615.15
[28]    valid_0's l2: 2640.63
[29]    valid_0's l2: 2624.73
[30]    valid_0's l2: 2615.14
[31]    valid_0's l2: 2601.41
[32]    valid_0's l2: 2593.79
[33]    valid_0's l2: 2555.64
[34]    valid_0's l2: 2572.12
[35]    valid_0's l2: 2581.22
[36]    valid_0's l2: 2573.21
[37]    valid_0's l2: 2562.82
[38]    valid_0's l2: 2611.74
[39]    valid_0's l2: 2607.7
[40]    valid_0's l2: 2616.08
[41]    valid_0's l2: 2627.31
[42]    valid_0's l2: 2631.77
[43]    valid_0's l2: 2626.38
[44]    valid_0's l2: 2646.27
[45]    valid_0's l2: 2668.98
[46]    valid_0's l2: 2642.24
[47]    valid_0's l2: 2644.99
[48]    valid_0's l2: 2674.41
```

```
[49]    valid_0's l2: 2671.46
[50]    valid_0's l2: 2665.95
[51]    valid_0's l2: 2669.9
[52]    valid_0's l2: 2656.4
[53]    valid_0's l2: 2659.14
[54]    valid_0's l2: 2655.23
[55]    valid_0's l2: 2648.59
[56]    valid_0's l2: 2663.95
[57]    valid_0's l2: 2668.91
[58]    valid_0's l2: 2678.76
```

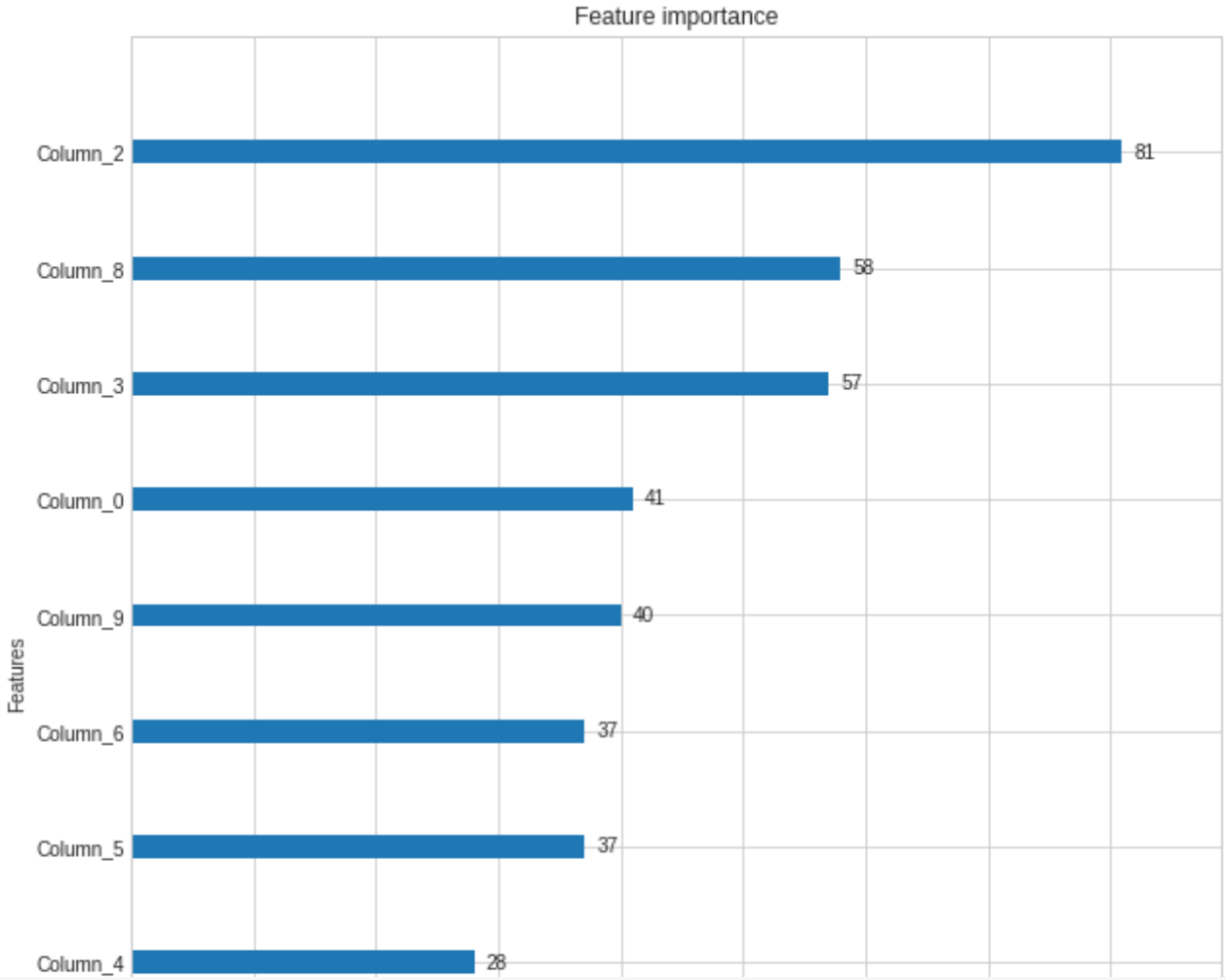
```
cross_val = cross_validate(
    estimator=lgbmr,
    X=diabetes.data, y=diabetes.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std())
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].
```

```
avg fit time: 0.11682257652282715 (+/- 0.0023398513562390352)
avg score time: 0.0035699844360351563 (+/- 8.115350979245287e-05)
avg test score: 0.30867643947179507 (+/- 0.07010708786960605)
```

```
plot_metric(lgbmr);
```




```
plot_importance(lgbmr, figsize=(10, 12));
```




```
plot_tree(lgbmr, figsize=(28, 24));
```