Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»
Отчёт по лабораторной работе №5
«Ансамбли моделей машинного обучения»

Выполнила:
Шимолина П.К.,
группа ИУ5-61Б

Проверил:
Нардид А.Н.,
каф. ИУ5

Дата:

Подпись:

Дата:

Подпись:

2023
Москва

lab-5-2

June 14, 2023

[1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
import category_encoders as ce
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from mlxtend.regressor import StackingCVRegressor
from sklearn.datasets import make_regression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
```

[2]:
```
df = pd.read_csv("Life Expectancy Data.csv")
```

[3]:
```
y = df["Life expectancy "]
X =  df.drop(["Life expectancy "], axis=1)
```

[4]:
```
y.fillna(y.median(), inplace=True)
```

[5]:
```
X.Year = pd.to_datetime(X.Year).dt.year
```

[6]:
```
bin_enc = ce.BinaryEncoder(drop_invariant=True)
X = bin_enc.fit_transform(X)
```

[7]:
```
X.fillna(X.mean(), inplace=True)
```

[8]:
```
X_train, X_test, y_train, y_test= train_test_split(X, y, test_size= 0.30,␣
↪random_state=9)
```

0.1 ( )

, , .

[9]:
```
rfr = RandomForestRegressor()
rfr.fit(X_train, y_train)
prediction = rfr.predict(X_test)
```

[10]:
```
params = {
 'max_depth' : [10, 15, 20],
}

grid = GridSearchCV(estimator=RandomForestRegressor(), param_grid=params, cv= 5)
grid.fit(X_train, y_train)
```

[10]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'max_depth': [10, 15, 20]})
```

[11]:
```
grid.best_score_, grid.best_params_
```

[11]:
```
(0.9560259403650033, {'max_depth': 20})
```

[12]:
```
rfr = RandomForestRegressor(max_depth=15, random_state=0)
rfr.fit(X_train, y_train)
prediction = rfr.predict(X_test)
mean_squared_error(y_test, prediction)
```

[12]:
```
3.780427824778443
```

0.2

. ,
. , ,
, . , ,
.

[13]:
```
params = {
    'n_estimators': [500, 800],
    'max_depth': [5, 8],
    'min_samples_split': [2, 5],
    'learning_rate': [0.01, 0.1]
}
grid = GridSearchCV(GradientBoostingRegressor(), param_grid=params, cv=5,␣
 ↪n_jobs=-1)
grid.fit(X_train, y_train)
```

[13]:
```
GridSearchCV(cv=5, estimator=GradientBoostingRegressor(), n_jobs=-1,
             param_grid={'learning_rate': [0.01, 0.1], 'max_depth': [5, 8],
                         'min_samples_split': [2, 5],
                         'n_estimators': [500, 800]})
```

[14]:
```
grid.best_score_, grid.best_params_
```

[14]:
```
(0.9572816094106351,
 {'learning_rate': 0.1,
  'max_depth': 5,
  'min_samples_split': 5,
  'n_estimators': 800})
```

[15]:
```
gbr = GradientBoostingRegressor(**grid.best_params_)
gbr.fit(X_train, y_train)
prediction = gbr.predict(X_test)
```

[16]:
```
mean_squared_error(y_test, prediction)
```

[16]:
```
3.428038590524013
```

0.3

, .

. ,

.

[17]:
```
reg1 = RandomForestRegressor(random_state=42)
reg2 = GradientBoostingRegressor(random_state=42)
reg3 = LinearRegression()
meta_learner = LinearRegression()
sr = StackingCVRegressor(regressors=[reg1, reg2, reg3],␣
 ↪meta_regressor=meta_learner)
params = {
    'randomforestregressor__max_depth': [5, 8]
}

grid = GridSearchCV(estimator=sr, param_grid=params, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)
```

[17]:
```
GridSearchCV(cv=5,
             estimator=StackingCVRegressor(meta_regressor=LinearRegression(),
regressors=[RandomForestRegressor(random_state=42),
GradientBoostingRegressor(random_state=42),
                                                LinearRegression()]),
             n_jobs=-1,
             param_grid={'randomforestregressor__max_depth': [5, 8]})
```

[18]:
```
sr.get_params().keys()
```

[18]:
```
dict_keys(['cv', 'meta_regressor__copy_X', 'meta_regressor__fit_intercept',
'meta_regressor__n_jobs', 'meta_regressor__normalize',
```

```
'meta_regressor__positive', 'meta_regressor', 'multi_output', 'n_jobs',
'pre_dispatch', 'random_state', 'refit', 'regressors', 'shuffle',
'store_train_meta_features', 'use_features_in_secondary', 'verbose',
'randomforestregressor', 'gradientboostingregressor', 'linearregression',
'randomforestregressor__bootstrap', 'randomforestregressor__ccp_alpha',
'randomforestregressor__criterion', 'randomforestregressor__max_depth',
'randomforestregressor__max_features', 'randomforestregressor__max_leaf_nodes',
'randomforestregressor__max_samples',
'randomforestregressor__min_impurity_decrease',
'randomforestregressor__min_samples_leaf',
'randomforestregressor__min_samples_split',
'randomforestregressor__min_weight_fraction_leaf',
'randomforestregressor__n_estimators', 'randomforestregressor__n_jobs',
'randomforestregressor__oob_score', 'randomforestregressor__random_state',
'randomforestregressor__verbose', 'randomforestregressor__warm_start',
'gradientboostingregressor__alpha', 'gradientboostingregressor__ccp_alpha',
'gradientboostingregressor__criterion', 'gradientboostingregressor__init',
'gradientboostingregressor__learning_rate', 'gradientboostingregressor__loss',
'gradientboostingregressor__max_depth',
'gradientboostingregressor__max_features',
'gradientboostingregressor__max_leaf_nodes',
'gradientboostingregressor__min_impurity_decrease',
'gradientboostingregressor__min_samples_leaf',
'gradientboostingregressor__min_samples_split',
'gradientboostingregressor__min_weight_fraction_leaf',
'gradientboostingregressor__n_estimators',
'gradientboostingregressor__n_iter_no_change',
'gradientboostingregressor__random_state',
'gradientboostingregressor__subsample', 'gradientboostingregressor__tol',
'gradientboostingregressor__validation_fraction',
'gradientboostingregressor__verbose', 'gradientboostingregressor__warm_start',
'linearregression__copy_X', 'linearregression__fit_intercept',
'linearregression__n_jobs', 'linearregression__normalize',
'linearregression__positive'])
```

[19]:
```
grid.best_score_, grid.best_params_
```

[19]:
```
(0.9501129846734061, {'randomforestregressor__max_depth': 8})
```

[20]:
```
reg1 = RandomForestRegressor(random_state=42, max_depth=8)
```

[21]:
```
sr = StackingCVRegressor(regressors=[reg1, reg2, reg3],␣
 ↪meta_regressor=meta_learner)
sr.fit(X_train, y_train)
prediction = sr.predict(X_test)
```

```
C:\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature
names, but RandomForestRegressor was fitted without feature names
  warnings.warn(
C:\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature
names, but GradientBoostingRegressor was fitted without feature names
  warnings.warn(
C:\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature
names, but LinearRegression was fitted without feature names
  warnings.warn(
```

[22]:
```
mean_squared_error(y_test, prediction)
```

[22]: 4.43231527722557