%D1%82%D0%B8%D1%82%D1%83%D0%BB-1.png

%D1%82%D0%B8%D1%82%D1%83%D0%BB-1.png

## Задание:

* Выберите набор данных (датасет) для решения задачи классификации или регресии.
* В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
* С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.
* Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.
* Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости от используемой библиотеки можно применять \* функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.
* Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor  
from sklearn.metrics import mean\_absolute\_error,accuracy\_score, r2\_score

data = pd.read\_csv("data/Admission\_Predict\_Ver1.1.csv")

data.head()

Serial No.

GRE Score

TOEFL Score

University Rating

SOP

LOR

CGPA

Research

Chance of Admit

0

1

337

118

4

4.5

4.5

9.65

1

0.92

1

2

324

107

4

4.0

4.5

8.87

1

0.76

2

3

316

104

3

3.0

3.5

8.00

1

0.72

3

4

322

110

3

3.5

2.5

8.67

1

0.80

4

5

314

103

2

2.0

3.0

8.21

0

0.65

data\_X = data[["CGPA", "TOEFL Score", "University Rating"]]

data\_X

CGPA

TOEFL Score

University Rating

0

9.65

118

4

1

8.87

107

4

2

8.00

104

3

3

8.67

110

3

4

8.21

103

2

...

...

...

...

495

9.02

108

5

496

9.87

117

5

497

9.56

120

5

498

8.43

103

4

499

9.04

113

4

500 rows × 3 columns

data\_Y = data[["Chance of Admit "]]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 data\_X, data\_Y, test\_size=0.25, random\_state=1)

# Качество отдельных моделей  
def val\_mae(model):  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
 plt.plot(X\_test, y\_test, 'g.')  
 plt.plot(X\_test, y\_pred, 'ro')  
 plt.show()  
 result = mean\_absolute\_error(y\_test, y\_pred)  
 r2 = r2\_score(y\_test, y\_pred)  
 print(model)  
 print('MAE={}'.format(result))  
 print('R2={}'.format(r2))

for model in [  
 GradientBoostingRegressor(),  
 RandomForestRegressor(n\_estimators=50)  
]:  
 val\_mae(model)  
 print('==========================\n\n')

c:\users\ncher\appdata\local\programs\python\python36\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().  
 return f(\*\*kwargs)

png

png

GradientBoostingRegressor()  
MAE=0.05018946886061301  
R2=0.7288949295990995  
==========================  
  
  
  
  
c:\users\ncher\appdata\local\programs\python\python36\lib\site-packages\ipykernel\_launcher.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().  
 This is separate from the ipykernel package so we can avoid doing imports until

png

png

RandomForestRegressor(n\_estimators=50)  
MAE=0.05431536000000001  
R2=0.6934410032452829  
==========================

## Модель градиентного бустинга показала лучший результат на тестовой выборке

from sklearn.model\_selection import RandomizedSearchCV  
  
n\_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]  
  
max\_features = ['auto', 'sqrt']  
  
max\_depth = [int(x) for x in np.linspace(10, 110, num = 11)]  
max\_depth.append(None)  
  
min\_samples\_split = [2, 5, 10]  
  
min\_samples\_leaf = [1, 2, 4]  
  
bootstrap = [True, False]  
  
random\_grid = {'n\_estimators': n\_estimators,  
 'max\_features': max\_features,  
 'max\_depth': max\_depth,  
 'min\_samples\_split': min\_samples\_split,  
 'min\_samples\_leaf': min\_samples\_leaf,  
 'bootstrap': bootstrap}  
random\_grid

{'n\_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000],  
 'max\_features': ['auto', 'sqrt'],  
 'max\_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],  
 'min\_samples\_split': [2, 5, 10],  
 'min\_samples\_leaf': [1, 2, 4],  
 'bootstrap': [True, False]}

rf = RandomForestRegressor()  
  
rf\_random = RandomizedSearchCV(estimator = rf, param\_distributions = random\_grid, n\_iter = 100, cv = 3, verbose=2, random\_state=42, n\_jobs = -1)  
  
rf\_random.fit(X\_train, y\_train)

Fitting 3 folds for each of 100 candidates, totalling 300 fits  
  
  
[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.  
[Parallel(n\_jobs=-1)]: Done 33 tasks | elapsed: 35.9s  
[Parallel(n\_jobs=-1)]: Done 154 tasks | elapsed: 2.4min  
[Parallel(n\_jobs=-1)]: Done 300 out of 300 | elapsed: 4.7min finished  
c:\users\ncher\appdata\local\programs\python\python36\lib\site-packages\sklearn\model\_selection\\_search.py:765: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().  
 self.best\_estimator\_.fit(X, y, \*\*fit\_params)  
  
  
  
  
  
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n\_iter=100,  
 n\_jobs=-1,  
 param\_distributions={'bootstrap': [True, False],  
 'max\_depth': [10, 20, 30, 40, 50, 60,  
 70, 80, 90, 100, 110,  
 None],  
 'max\_features': ['auto', 'sqrt'],  
 'min\_samples\_leaf': [1, 2, 4],  
 'min\_samples\_split': [2, 5, 10],  
 'n\_estimators': [200, 400, 600, 800,  
 1000, 1200, 1400, 1600,  
 1800, 2000]},  
 random\_state=42, verbose=2)

rf\_random.best\_params\_

{'n\_estimators': 1600,  
 'min\_samples\_split': 2,  
 'min\_samples\_leaf': 4,  
 'max\_features': 'sqrt',  
 'max\_depth': 10,  
 'bootstrap': True}

def evaluate(model, test\_features, test\_labels):  
 predictions = model.predict(test\_features)  
 error = mean\_absolute\_error(y\_test, predictions)  
 r2 = r2\_score(y\_test, predictions)  
 print('Model Performance')  
 print('MAE: {:0.4f}'.format(error))  
 print('R2 score: {:0.4f}'.format(r2))  
 print('======================\n\n')  
  
base\_model = RandomForestRegressor(n\_estimators = 10, random\_state = 42)  
base\_model.fit(X\_train, y\_train)  
evaluate(base\_model, X\_test, y\_test)

Model Performance  
MAE: 0.0534  
R2 score: 0.7026  
======================  
  
  
  
  
c:\users\ncher\appdata\local\programs\python\python36\lib\site-packages\ipykernel\_launcher.py:11: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().  
 # This is added back by InteractiveShellApp.init\_path()

best\_random = rf\_random.best\_estimator\_  
evaluate(best\_random, X\_test, y\_test)

Model Performance  
MAE: 0.0509  
R2 score: 0.7375  
======================

Видно, что подбор гиперпараметров улучшил нашу модель, уменьшив ошибку на 0.0025

n\_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]  
  
max\_features = ['auto', 'sqrt']  
  
max\_depth = [int(x) for x in np.linspace(10, 110, num = 11)]  
max\_depth.append(None)  
  
min\_samples\_split = [2, 5, 10]  
  
min\_samples\_leaf = [1, 2, 4]  
  
bootstrap = [True, False]  
  
random\_grid\_Booster = {'n\_estimators': n\_estimators,  
 'max\_features': max\_features,  
 'max\_depth': max\_depth,  
 'min\_samples\_split': min\_samples\_split,  
 'min\_samples\_leaf': min\_samples\_leaf,  
 }

gb = GradientBoostingRegressor()  
  
gb\_random = RandomizedSearchCV(estimator = gb, param\_distributions = random\_grid\_Booster, n\_iter = 100, cv = 3, verbose=2, random\_state=42, n\_jobs = -1)  
  
gb\_random.fit(X\_train, y\_train)

Fitting 3 folds for each of 100 candidates, totalling 300 fits  
  
  
[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.  
[Parallel(n\_jobs=-1)]: Done 33 tasks | elapsed: 14.7s  
[Parallel(n\_jobs=-1)]: Done 154 tasks | elapsed: 55.5s  
[Parallel(n\_jobs=-1)]: Done 300 out of 300 | elapsed: 1.8min finished  
c:\users\ncher\appdata\local\programs\python\python36\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().  
 return f(\*\*kwargs)  
  
  
  
  
  
RandomizedSearchCV(cv=3, estimator=GradientBoostingRegressor(), n\_iter=100,  
 n\_jobs=-1,  
 param\_distributions={'max\_depth': [10, 20, 30, 40, 50, 60,  
 70, 80, 90, 100, 110,  
 None],  
 'max\_features': ['auto', 'sqrt'],  
 'min\_samples\_leaf': [1, 2, 4],  
 'min\_samples\_split': [2, 5, 10],  
 'n\_estimators': [200, 400, 600, 800,  
 1000, 1200, 1400, 1600,  
 1800, 2000]},  
 random\_state=42, verbose=2)

gb\_random.best\_params\_

{'n\_estimators': 1400,  
 'min\_samples\_split': 2,  
 'min\_samples\_leaf': 1,  
 'max\_features': 'sqrt',  
 'max\_depth': 100}

def evaluate(model, test\_features, test\_labels):  
 predictions = model.predict(test\_features)  
 error = mean\_absolute\_error(y\_test, predictions)  
 r2 = r2\_score(y\_test, predictions)  
 print('Model Performance')  
 print('MAE: {:0.4f}'.format(error))  
 print('R2 score: {:0.4f}'.format(r2))  
 print('======================\n\n')  
  
base\_model = GradientBoostingRegressor()  
base\_model.fit(X\_train, y\_train)  
evaluate(base\_model, X\_test, y\_test)

Model Performance  
MAE: 0.0505  
R2 score: 0.7287  
======================  
  
  
  
  
c:\users\ncher\appdata\local\programs\python\python36\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().  
 return f(\*\*kwargs)

best\_random = gb\_random.best\_estimator\_  
evaluate(best\_random, X\_test, y\_test)

Model Performance  
MAE: 0.0598  
R2 score: 0.6624  
======================

## Подбор параметров в градиентном бустинге не дал прироста качества