# **Градиентный бустинг**

Мы возвращаемся на титаник, но уже с новым инструментом!

****Большая задача**** этой лабораторной - произведите исследование метода градиентного бустинга для классификации [GradientBoostingClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html), как мы изучали метод случайного леса.

1. Давайте пройдемся по основным моментам:

* начните с разработки baseline модели,
* далее проверьте влияние гиперпараметров на результат работы модели,
* проведите расширенный анализ данных (дополнительная обработка - чистка, новые признаки и т.д.) и оцените работу модели;
* поиском определите лучшую модель;

1. Далее сравниваем работу моделей (леса и бустинга) при их лучших гиперпараметрах по показателям на тестовой выборке (*hold-out*).

Помним, что поиск лучших гиперпараметров производится кросс-валидацией на обучающей выборке!

1. Напоследок, установите фреймворки [XGBoost](https://xgboost.readthedocs.io/en/latest/) и [LightGBM](https://lightgbm.readthedocs.io/en/latest/), а после этого разберитесь с тем, как создать объекты моделей и с помощью уже известных инструментов найдите лучшие гиперпараметры.

В результаты должна получиться табличка сравнения, в которой отражены показатели моделей:

* RandomForestClassifier;
* GradientBoostingClassifier;
* XGBoost;
* LightGBM.

1. импорт данных

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

titanic = pd.read\_csv('C:\ml\_edu-master\datasets\Titanic\_train.csv')

1. Выбор функции

X = titanic[['Pclass', 'Age', 'Sex']]

y = titanic['Survived']

1. Очистка данных

X['Age'].fillna(X['Age'].mean(), inplace=True)

1. Разделите тренировочный набор тестовый набор

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state = 33)

1. Функции преобразования

from sklearn.feature\_extraction import DictVectorizer

vec = DictVectorizer(sparse=False)

X\_train = vec.fit\_transform(X\_train.to\_dict(orient='record'))

print(vec.feature\_names\_)

X\_test = vec.transform(X\_test.to\_dict(orient='record'))

Результат



1. Моделирование и тестирование производительности

① Единое дерево решений

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

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

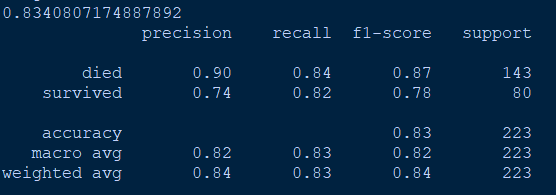
y\_predict = dtc.predict(X\_test)

from sklearn.metrics import classification\_report

print(dtc.score(X\_test, y\_test))

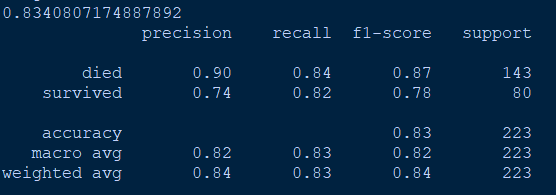
print(classification\_report(y\_predict, y\_test, target\_names = ['died', 'survived']))

Результат



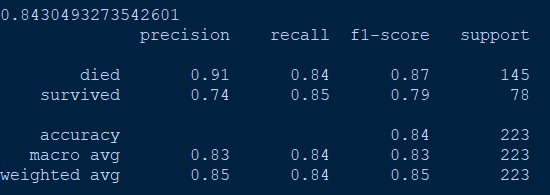
②Случайный лес

Результат



③Дерево решений градиентного подъема

Результат



Сравните лес случайных решений и способность модели XGBoost предсказать выживут ли пассажиры на Титанике：

X = titanic[['Pclass', 'Age', 'Sex']]

y = titanic['Survived']

X['Age'].fillna(X['Age'].mean(), inplace=True)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state = 33)

from sklearn.feature\_extraction import DictVectorizer

vec = DictVectorizer(sparse=False)

X\_train = vec.fit\_transform(X\_train.to\_dict(orient='record'))

X\_test = vec.transform(X\_test.to\_dict(orient='record'))

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

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

print ( "The accuracy of Random Forest Classifier on testing set:", rfc.score(X\_test, y\_test))

Результат



# Предсказать тот же набор тестов с настроенной по умолчанию моделью XGBoost

from xgboost import XGBClassifier

xgbc = XGBClassifier(learning\_rate=0.1)

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

print ('The accuracy of eXtreme Gradient Boosting Classifier on testing set:', xgbc.score(X\_test, y\_test))

