Лабораторная работа №5.2

**Применение ансамблевых моделей на базе решающих деревьев в задачах классификации**

Для выполнения данной работы мы будем использовать набор данных (датасет) с информацией о медицинских параметрах больных диабетом, см. Работу 5.1.

Задачи к лабораторной работе:

1. Решить задачу классификации больных методом случайного леса.

* Провевести исследование качества модели от глубины используемых деревьев. Отрисовать зависимость на графике
* Провести исследование качества модели от количества подаваемых на дерево признаков. Отрисовать зависимость на графике
* Провести исследование качества модели от числа деревьев. Отрисовать на графике, дополнить график данными о времени обучения.

1. Решить задачу классификации с использованием XGBoost. Исследовать время обучения, качество полученных результатов. Сравнить с данными полученными в п.1 и сделать выводы (в работе). Гиперпараметры модели подбираются в этой лабораторной в ручную. Цель – попытаться найти лучшие значения и почувствовать рабочие диапазоны параметров.

**Дополнительные и вспомогательные материалы:**

Полезное чтиво по теме например тут - <https://habr.com/ru/companies/ods/articles/645887/>

**Установка и подключение**

pip install xgboost

Либо для conda

*# CPU only*

conda install -c conda-forge py-xgboost-cpu

*# Use NVIDIA GPU*

conda install -c conda-forge py-xgboost-gpu

**Внимание версия с GPU для Windows может не работать.!**

**Документация по XGBoost**

[**https://xgboost.readthedocs.io/en/latest/index.html**](https://xgboost.readthedocs.io/en/latest/index.html)

**Пример использования:**

**from** **xgboost** **import** XGBClassifier

*# read data*

**from** **sklearn.datasets** **import** load\_iris

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

data = load\_iris()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data['data'], data['target'], test\_size=.2)

*# create model instance*

bst = XGBClassifier(n\_estimators=2, max\_depth=2, learning\_rate=1, objective='binary:logistic')

*# fit model*

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

*# make predictions*

preds = bst.predict(X\_test)

### Гиперпараметры XGBoost

Какие самые важные гиперпараметры следует выбрать при реализации XGBoost и как их настроить?

#### усилитель

booster - это алгоритм повышения, для которого у вас есть 3 варианта: gbtree, gblinear или dart. Параметр по умолчанию - gbtree. dart - аналогичная версия, в которой используются методы исключения, чтобы избежать переобучения, а gblinear использует обобщенную линейную регрессию вместо деревьев решений.

#### reg\_alpha и reg\_lambda

reg\_alpha и reg\_lambda являются членами регуляризации L1 и L2 соответственно. Чем больше эти числа, тем более консервативной (менее склонной к переобучению, но может упускать актуальную информацию) становится модель. Рекомендуемые значения находятся в диапазоне 0–1000 для обоих.

#### Максимальная глубина

max\_depth устанавливает максимальную глубину деревьев решений. Чем больше это число, тем менее консервативной становится модель. Если установлено значение 0, то нет ограничения на глубину деревьев.

#### подвыборка

subsample - это размер отношения выборки, который будет использоваться при обучении предикторов. По умолчанию 1, что означает, что выборка отсутствует, и мы используем все данные. Если, например, установлено значение 0,7, то 70% наблюдений будут выбираться случайным образом для использования в каждой итерации повышения (для каждой итерации берется новая выборка). Это может помочь предотвратить переобучение.

#### num\_estimators

num\_estimators устанавливает количество раундов повышения, что равняется установке количества используемых деревьев с усилением. Чем больше это число, тем выше риск переобучения (но низкие числа также могут привести к снижению производительности).