**Семинар 6. Бэггинг и случайный лес.**

**ЗАДАЧА О ВИНАХ**

Реализуем бэггинг для деревьев решений. Для тренировки будем использовать датасет о винах. (winequality.csv)



Для начала подготовим данные к классификации. Условно разделим вино на хорошее и нет. Хорошим вином будем называть вино, параметр *quality*которого не менее 6.

Теперь сравним несколько методов классификации: логистическую регрессию, решающее дерево и бэггинг.

Разбейте выборку на обучающую и тренировочную с параметрами test\_size=0.30, random\_state=42.

Обучите два классификатора: логистическую регрессию (с дефолтными параметрами) и решающее дерево (random\_state=42, максимальная глубина равна 10).

Задание 5.1

2 points possible (graded)

Введите значение *f1 score* для классификатора, который показал наилучшее значение.  нет ответа

Обучите модель с использование бэггинга (функция BaggingClassifier с random\_state=42, разделение выборки на обучающую и тренировочную с параметрами test\_size=0.30, random\_state=42) для алгоритма, показавшего лучшее качество, определите количество моделей 1500. Вычислите новое значение *f1-score*.

**Случайный лес**

Теперь обучим случайный лес на простых данных и посмотрим, как можно подбирать параметры случайного леса для достижения наилучшего качества модели.

Потренируемся на данных, по которым мы будем предсказывать погоду. Датасет temps\_extended.csv

Откроем его, удалим признаки, не относящиеся к предсказанию (от дня недели, например, или от года погода не зависит), разделим на тестовую и обучающуюся выборки:

weather=pd.read\_csv('temps\_extended.csv')

y = weather['actual']

X = weather.drop(['actual','weekday','month','day','year'],axis =1)

X\_train, X\_val, Y\_train, Y\_val=train\_test\_split(X,y,test\_size=0.3, random\_state=42)

Попробуем подобрать гиперпараметры таким образом, чтобы получить оптимальный результат.

Если мы запускаем случайный лес без настройки параметров, то по умолчанию они следующие:

**from** sklearn.ensemble **import** RandomForestRegressor

**from** pprint **import** pprint

rf = RandomForestRegressor(random\_state = 42)

# Look at parameters used by our current forest

print('Параметры по умолчанию:\n')

pprint(rf.get\_params())

Параметры по умолчанию:

{'bootstrap': True,

 'ccp\_alpha': 0.0,

 'criterion': 'mse',

 'max\_depth': None,

 'max\_features': 'auto',

 'max\_leaf\_nodes': None,

 'max\_samples': None,

 'min\_impurity\_decrease': 0.0,

 'min\_impurity\_split': None,

 'min\_samples\_leaf': 1,

 'min\_samples\_split': 2,

 'min\_weight\_fraction\_leaf': 0.0,

 'n\_estimators': 100,

 'n\_jobs': None,

 'oob\_score': False,

 'random\_state': 42,

 'verbose': 0,

 'warm\_start': False}

Попробуем подбирать разные значения для некоторых параметров. Для перебора вариантов возьмем следующие параметры:

* n\_estimators
* max\_features
* max\_depth
* min\_samples\_split
* min\_samples\_leaf
* bootstrap

Мы можем сами указать, какие значения гиперпараметров надо перебрать.

Зададим сетку гиперпараметров, которые будут перебираться:

**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}

Обучим наш лес:

rf = RandomForestRegressor(random\_state=42)

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)

Давайте посмотрим, какие гиперпараметры нам предлагают как оптимальные:

rf\_random.best\_params\_

Задание 5.6.1

3 points possible (graded)

Обучите случайный лес с предустановленными параметрами и теми параметрами, которые мы отобрали как оптимальные. В обоих вариантах поставьте random\_state =42. Какое улучшение *MSE* дала подстановка отобранных гиперпараметров? Ответ округлите до одного знака после запятой.

Задание 5.6.2

Какие из утверждений о случайных деревьях верны?

* Отдельное дерево обучается на всех признаках
* Отдельное дерево обучается на подмножестве признаков
* Отдельное дерево обучается на подмножестве наблюдений
* Отдельное дерево обучается на всех наблюдениях

Задание 5.6.3

Увеличение какого (каких) из гиперпараметров может привести к переобучению?

* Количество деревьев
* Глубина дерева
* Скорость обучения