Изображение выглядит как текст

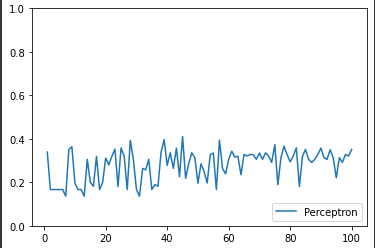
Автоматически созданное описание

**Задание 1**

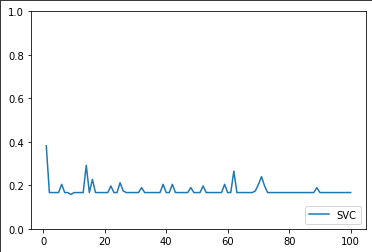
Используем BaggingClassifier из библиотеки sklearn в качестве алгоритма бэггинга.

В качестве базовых классификаторов используем классификаторы Perceptron, SVC и DecisionTreeClassifier. В качестве метрики качества классификации используем метрику balanced\_accuracy\_score из библиотеки sklearn.

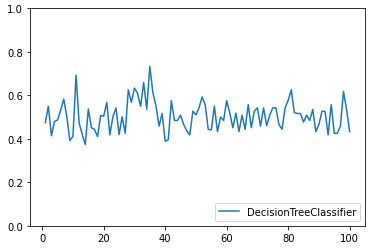
Начнем с базового классификатора Perceptron. Для него были получены следующие значения метрики balanced\_accuracy\_score в зависимости от количества классификаторов:



Теперь перейдем к SVC. Построим график зависимости для balanced\_accuracy\_score в зависимости от времени выглядит так:



Теперь перейдем к DecisionTreeClassifier. Построим график зависимости для balanced\_accuracy\_score в зависимости от времени выглядит так:



**Вывод:**

Мы можем наблюдать, что среди этих 3 классификаторов наилучшие значения метрики balanced\_accuracy\_score имеет классификатор DecisionTreeClassifier, т.к. его качество классификации в среднем равна 0.5.

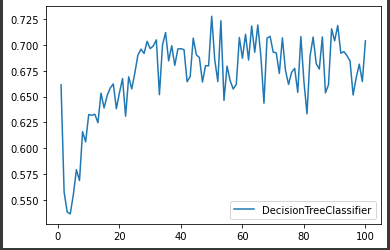
При использовании алгоритмов бэггинга с ростом числа классификаторов видим несущественный рост качества классификации для базового классификатора Perceptron. Для DecisionTreeClassifier наблюдаем периодический рост качества классификации в виде часто встречающихся пиков. У SVC не видим существенного роста качества классификации, а лишь довольно редкие пики.

**Задание 2**

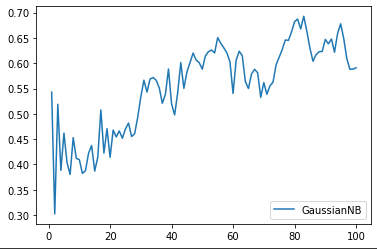
Будем использовать AdaBoostClassifier из библиотеки sklearn в качестве алгоритма бустинга.

В качестве базовых классификаторов будем использовать DecisionTreeClassifier и GaussianNB. В качестве метрики качества классификации будем использовать метрику balanced\_accuracy\_score из библиотеки sklearn.

Начнем с базового классификатора DecisionTreeClassifier. Для него были получены следующие значения метрики balanced\_accuracy\_score в зависимости от количества классификаторов:



Перейдем к базовому классификатору GaussianNB. Для него график balanced\_accuracy\_score в зависимости от времени выглядит следующим образом:



**Вывод:**

В обоих случаях видим сначала резкий спад качества классификации, а затем значительный рост. Базовый классификатор DecisionTreeClassifier оказался немного лучше базового классификатора GaussianNB, т.к. качество классификации у него выше (в первом случае больше 0.7, во втором меньше 0.7). Оба случая показали достаточно большие значения метрики balanced\_accuracy\_score.

**Задание 3**

Для стэкинга мы будем использовать следующие классификаторы: DecisionTreeClassifier, GaussianNB, SVC и KNeighborsClassifier.

Оценим качество классификации следующим образом: сначала посмотрим, как с задачей справится каждый классификатор по отдельности, а затем построим мета-классификатор и сравним его результаты с результатами отдельных классификаторов. В качестве метрики качества классификации опять будем использовать balanced\_accuracy\_score.

При использовании одиночного классификатора DecisionTreeClassifier мы получили следующее значение метрики balanced\_accuracy\_score:



При использовании одиночного классификатора GaussianNB мы получили следующее значение метрики balanced\_accuracy\_score:



При использовании одиночного классификатора SVC мы получили следующее значение метрики balanced\_accuracy\_score:



При использовании одиночного классификатора KNeighborsClassifier мы получили следующее значение метрики balanced\_accuracy\_score:



Теперь узнаем значение balanced accuracy score для мета-классификатора:



**Вывод:**

Видим, что при построении мета-классификатора, используя стекинг, качество нашей классификации возросло. При использовании отдельных классификаторов среднее значение метрики balanced\_accuracy\_score составляло 0.7565, а для мета-классификатора оно составило 0.797. Таким образом, качество нашей классификации возросло на 5%.