|  |  |
| --- | --- |
| Группа | R32811 |
| Студент | Филиппов Александр |
| Преподаватель | Евстафьев Олег Александрович |

Отчёт по практической работе №1

Классификация

# Распознавание активности человека на основе данных с мобильных сенсоров

Необходимо по данным с мобильных сенсоров при помощи прикладных алгоритмов машинного обучения предсказать активность человека по шести классам движений:

* Движется по прямой
* Движется вверх (например, движение по лестнице вверх)
* Движется вниз (например, движение по лестнице вниз)
* Сидит
* Стоит
* Лежит

# Сведения о наборе данных

Набор данных содержит записи датчиков со сматфонов (акселерометр и гироскоп с частотой дискретизации 50 Гц) от 30 участников, выполняющих следующие действия: ходьба, ходьба по лестнице, ходьба по лестнице, сидение, стояние и лежание. Данные были предварительно обработаны при помощи фильтров шума. Набор данных представлен Хорхе Л. Рейес-Ортисом.

Признаки были извлечены из 3-х осевых необработанных сигналов акселерометра и гироскопа tAcc-XYZ и tGyro-XYZ. Эти сигналы были сняты с постоянной частотой 50 Гц. Затем были отфильтрованны с помощью медианного фильтра и низкочастотного фильтра Баттерворта 3-го порядка с частотой 20 Гц для удаления шумов. Аналогичным образом сигнал ускорения был разделен на сигналы ускорения тела и гравитации (tBodyAcc-XYZ и tGravityAcc-XYZ) с помощью другого низкочастотного фильтра Баттерворта с угловой частотой 0,3 Гц. Линейное ускорение тела и угловая скорость были использованы для получения сигналов "рывка" — (tBodyAccJerk- XYZ и tBodyGyroJerk-XYZ). Также величина этих трехмерных сигналов была расчитана с использованием евклидовой нормы —

(tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag, tBodyGyroJerkMag).

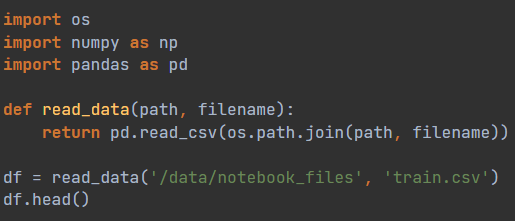
Наконец, к некоторым из этих сигналов было применено быстрое преобразование Фурье (БПФ), в результате чего получились fBodyAcc-XYZ, fBodyAccJerk-XYZ, fBodyGyro-

XYZ, fBodyAccJerkMag, fBodyGyroMag, fBodyGyroJerkMag. (Обратите внимание на "f" для обозначения сигналов в частотной области).

Набор переменных, которые были оценены по этим сигналам, следующий:

* mean(): Среднее значение
* std(): Стандартное отклонение
* mad(): Среднее абсолютное отклонение
* max(): Наибольшее значение в массиве
* min(): Наименьшее значение в массиве
* sma(): Область величины сигнала
* energy(): Мера энергии. Сумма квадратов, деленная на количество значений.
* iqr(): Интерквартильный размах
* entropy(): Энтропия сигнала
* arCoeff(): Коэффициенты авторегрессии с порядком Burg, равным 4
* correlation(): коэффициент корреляции между двумя сигналами
* maxInds(): индекс частотной составляющей с наибольшей величиной
* meanFreq(): средневзвешенное значение частотных компонент для получения средней частоты
* skewness(): перекос сигнала в частотной области
* kurtosis(): эксцесс сигнала в частотной области
* bandsEnergy(): Энергия частотного интервала в пределах 64 бинов БПФ каждого окна.
* angle(): Угол между векторами.

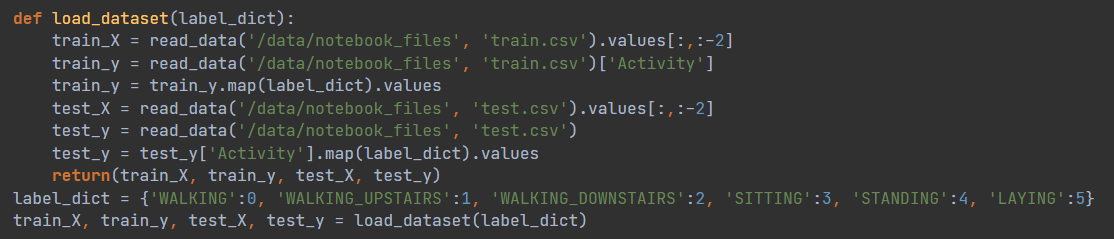
# Импорт библиотек и считывание данных

****

В прикладных задачах машинного обучения очень важен процесс извлечения признаков (feature extraction), в ходе которого данные интерпретируются в информативные признаки. Также этот процесс может называться проектирование признаков (feature engineering), это весьма трудоемкая и творческая задача. В рамках работы мы опустим эту часть и воспользуемся предобработанными данными.

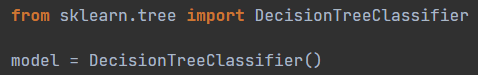
Теперь, загрузим полный набор данных и сохраним его под следующими четырьмя переменными:

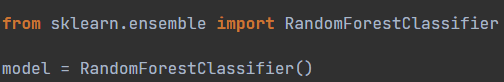
* train\_X: признаки, используемые для обучения модели
* train\_y: метки, используемые для обучения модели
* test\_X: признаки, используемые для проверки модели
* test\_y: метки, используемые для проверки модели



# Выбор модели

Импортируем выбранную нами модель из библиотеки sklearn и инициализируем её в объект model:





В качестве моделей мной были выбраны DecisionTreeClassifier и RandomForestClassifier. Интересно было протестировать, как они работают и сравнить результаты. Из группы моделей, осуществляющих классификацию, были выбраны именно они, чтобы наглядно посмотреть на работу модуля.

## Гиперпараметры DecisionTreeClassifier :

criterion = 'gini' - функция для измерения качества разбиения. Поддерживаются критерии 'gini' для неодородности Джини и 'entropy' для прироста информации.

splitter='best' - стратегия, используемая для выбора разбиения в каждом узле. Поддерживаются стратегии 'best' для выбора лучшего разбиения и 'random' для выбора лучшего рандомного разбиения.

max\_depth=None - максимальная глубина дерева Если None, то дерево строится до тех пор, пока в каждом листе не будет только один класс или пока каждый лист не будет содержать количество экземпляров, равное min\_samples\_split.

min\_samples\_split=2 - миниальное количество экземпляров, которое может содержаться в узле для его дальнейшего разбиения.

min\_samples\_leaf=1 - миниальное количество экземпляров, которое может содержаться в листе. Разбиение разрешено если в каждом листе узла будет как минимум min\_samples\_leaf экземпляров. Это может привести к сглаживанию модели, особенно в регрессии.

min\_weight\_fraction\_leaf=0.0 - минимальня взвешенная часть суммы всех весов входных экземпляров, необходимых в листовом узле. Экземпляры имеют одинаковый вес, если параметр sample\_weight не задан.

max\_features=None - максимальное количество фичей, которые рассматриваются при выборе лучшего разбиения:

1. Если 'auto', то max\_features=sqrt(n\_features).
2. Если 'sqrt', то max\_features=sqrt(n\_features).
3. Если 'log2', то max\_features=log2(n\_features).
4. Если int, то max\_features=int
5. Если None, то max\_features=n\_features. Примечание: поиск разбиения не останавливается пока не будет найдено хотя бы одно допустимое разбиение, даже если это требует эффективно проверить более чем max\_features фичей.

random\_state=None - начальное состояние генератора случайных чисел.

max\_leaf\_nodes=None - максимальное количество листьев. Дерево строится исходя из ограничения на максимальное количество листьев. Остаются только те листья, которые максимально уменьшают неоднородность.

min\_impurity\_decrease=0.0 - минимальное уменьшение неоднородности. Узел расщепляется, если неоднородность уменьшается на число больше или равное min\_impurity\_decrease.

class\_weight=None - вес класса в виде {метка класса: вес}. Если этот параметр не задан, то вес каждого класса предполагается равным 1. Если этот параметр задать равным 'balanced', то классам автоматически присвоится вес, обратнопропорциональный частоте класса в метке y входных данных по формуле n\_samples / (n\_classes \* np.bincount(y))

Примечание: веса будут умножены на коэффициенты sample\_weight (который задается при обучении модели), если он был задан.

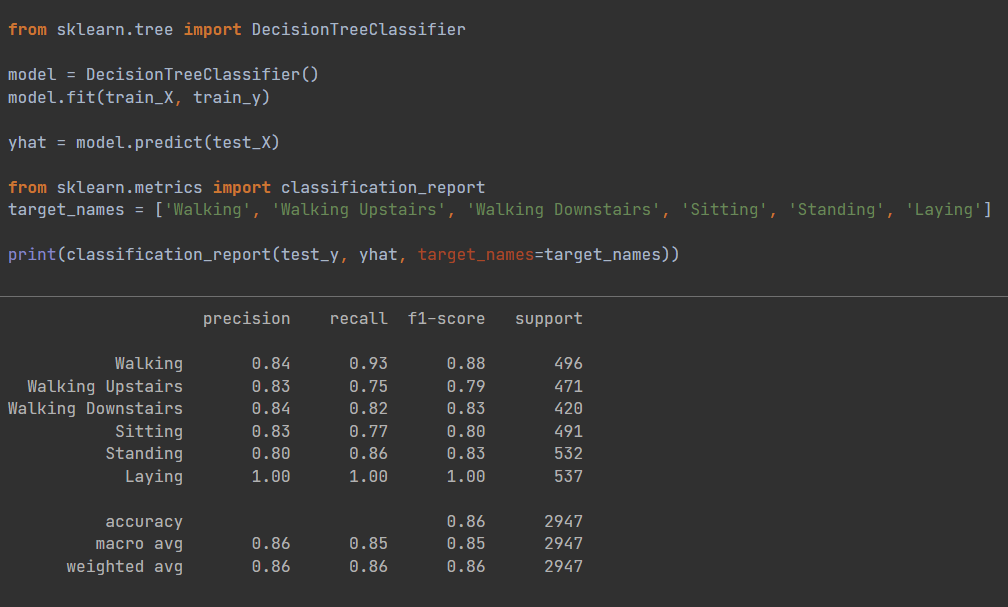
presort=False - Устанавливает сортировку данных для увеличения скорости поиска лучшего разбиения при обучении. Применяется при обучении на малых данных. Если применять при обучении на больших данных, то возможно, наоборот, замедление процесса обучения.

## Гиперпараметры RandomForestClassifier :

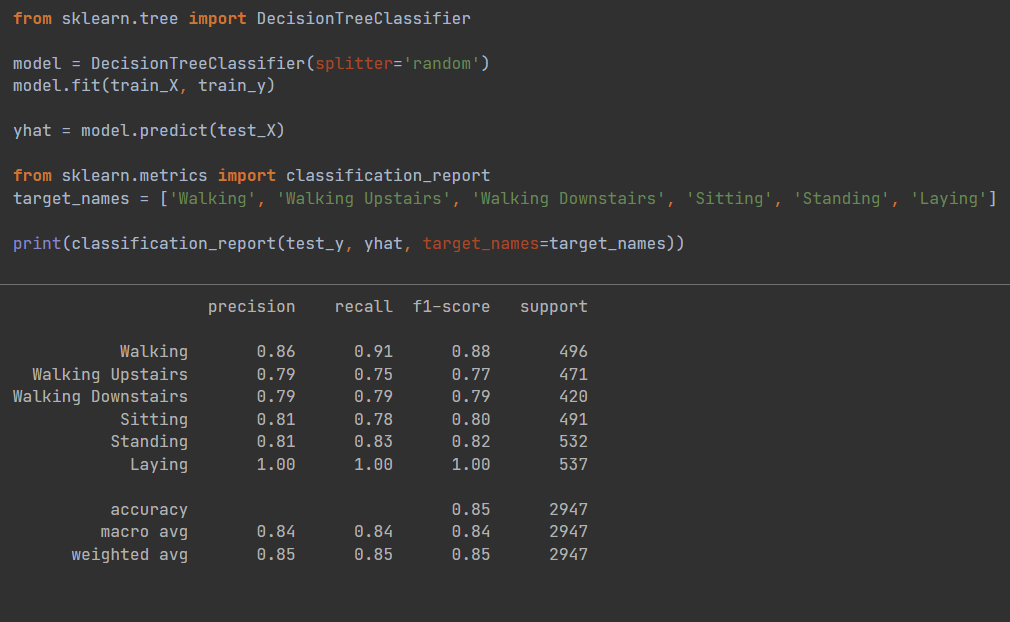
Основными параметрами для корректировки при использовании этих методов являются n\_estimators и max\_features. Первый - это количество деревьев в лесу. Чем больше, тем лучше, но и чем дольше потребуется для вычисления. **max\_depth** при увеличении глубины резко возрастает качество на обучении, но и на контроле оно, как правило, увеличивается.

# Обучение, оценка модели, результат :

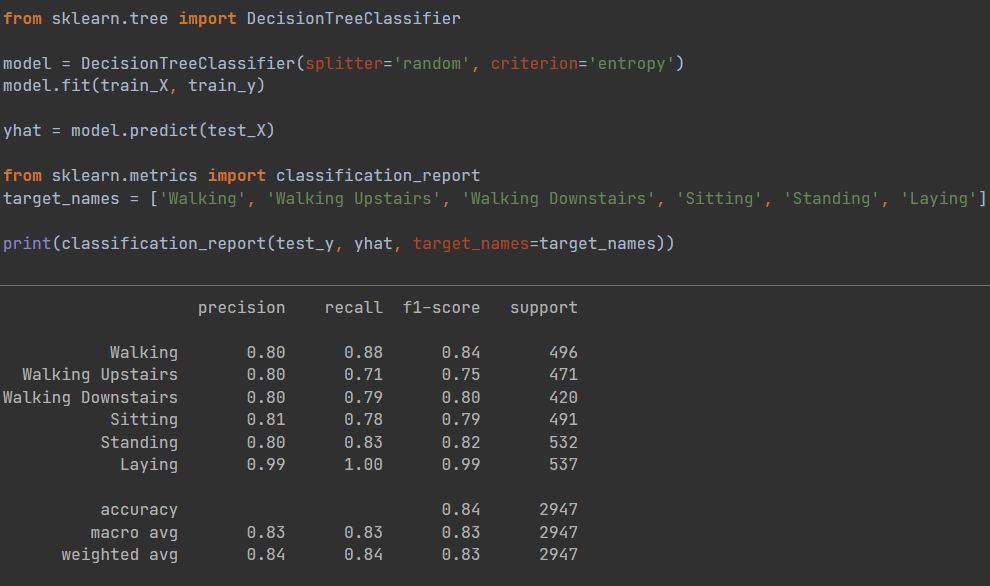
1. DecisionTreeClassifier (обычные значения)



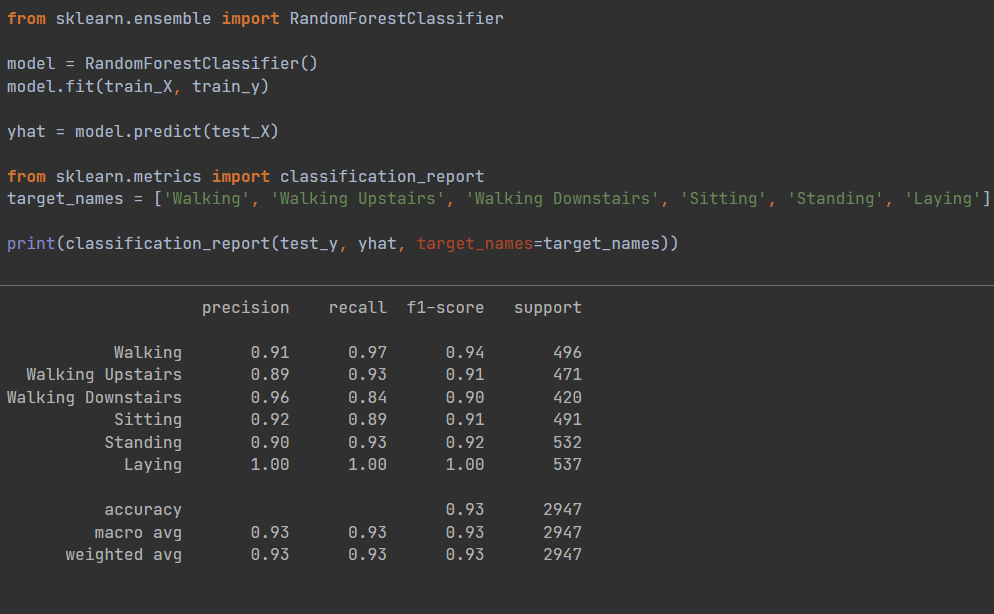
1. DecisionTreeClassifier (splitter=’random’)



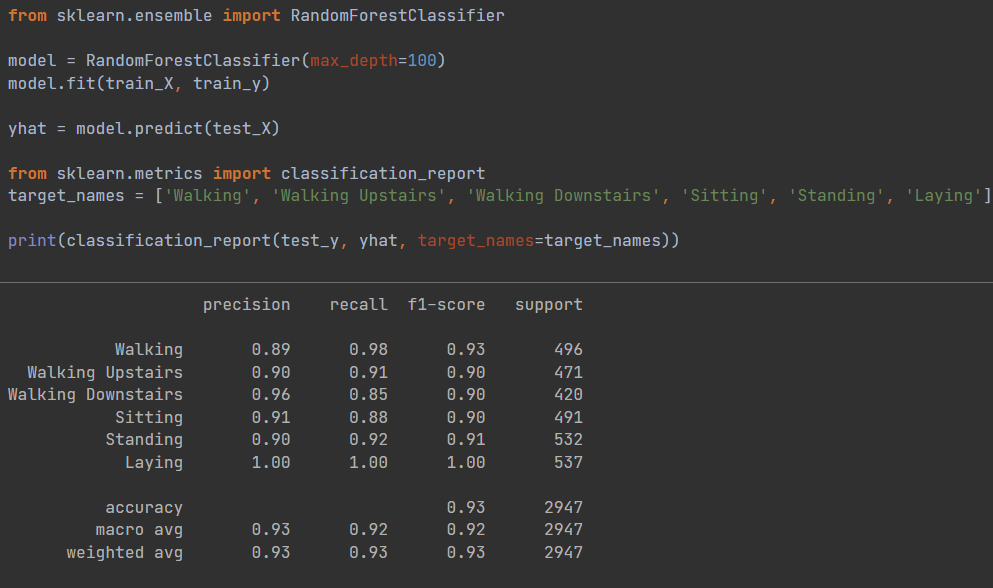
1. DecisionTreeClassifier (splitter=’random’, criterion=’entropy’)



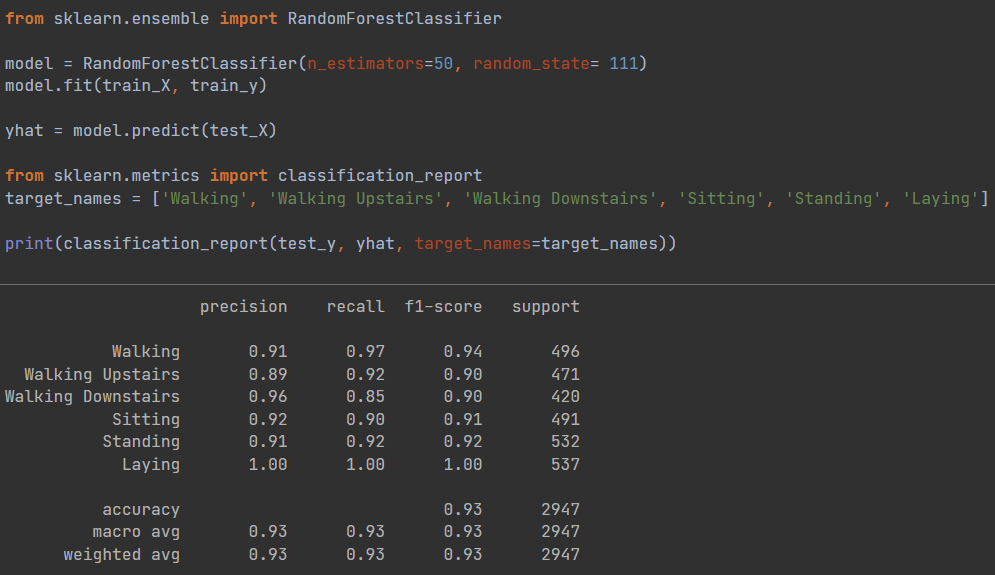
1. RandomForestClassifier (обычные значения)



1. RandomForestClassifier (max\_depth=100)



1. RandomForestClassifier (n\_estimators=50, random\_state=111)



**Вывод :** Гиперпараметры влияют на полученные результаты.

## Ответы на вопросы :

1. Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными,

а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

1. F1 - это среднее гармоническое значение между precision и recall.