Памятка

Для работы с программой нужно иметь в корне проекта (на одном уровне) 3 файла:

1) файл с кодом

2) файл обучающей выборки с названием 'train.csv'

3) файл тестовой выборки с названием 'test.csv'

**Требования к файлу 'train.csv'.**

В файле первый столбец — это идентификатор с названием столбца id. Далее идут столбцы-признаки в численном виде (названия столбцов могут быть любыми кроме id и target, так как мы их уже использовали). Работа с категориальными признаками в данной программе не предусмотрена. Далее должен быть столбец target с целевой переменной, т.е. в каждой строке указывается метка класса 0 или 1.

**Требования к файлу 'test.csv'.**

В файле первый столбец — это идентификатор с названием столбца id. Далее идут столбцы-признаки в численном виде (названия столбцов могут быть любыми кроме id, так как мы его уже использовали).

Весь код программы написан в формате ipynb и запускается по ячейкам (возможно в дальнейшем будет переделан в формат py, поэтому нужно иметь установленный на компьютере Jupyter.

Для корректной работы нужно иметь последнюю версию библиотеки sklearn и библиотеки python-docx. Для компьютеров на Linux и Mac это делается так: 1. Открываем командную строку (терминал) и пишем команды:

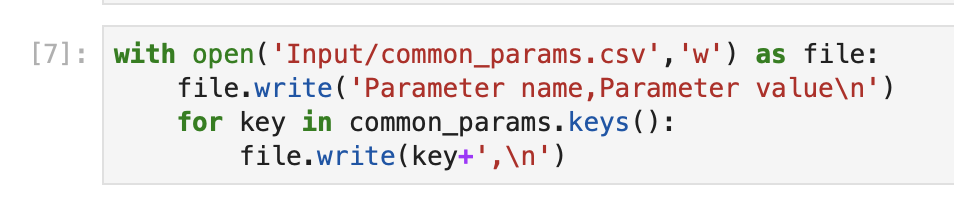
1) pip install -U scikit-learn

2) pip install python-docx

Стандартизацию признаков проводить не нужно, она делается вннутри программы.

Далее начинаем запускать по очереди ячейки (кнопка run). Создастся файловое пространство: папки Input, Output и App (внутри папки App есть папки images и models).

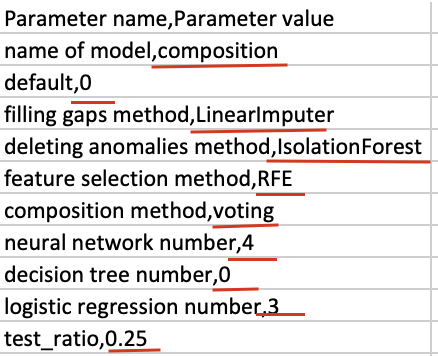
После того, как вы запустили эту ячейку (на скриншоте ниже), создаться шаблон файла common\_params.csv



Открываем документ common\_params.csv

В этом файле заполняются общие параметры. В зависимости от выбора определенного параметра будет формироваться файл methods\_params.csv с частными параметрами.

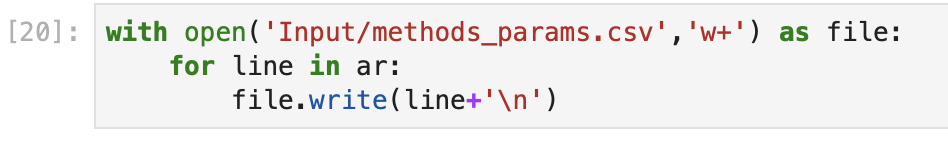
В файле common\_params.csv мы заполняем **ОБЩИЕ ПАРАМЕТРЫ** (в файле должна быть заполнена каждая строка. Пример на скриншоте ниже).



В этой памятке в приложении ниже общие параметры будут красного цвета, а частные — зеленого.

В файле common\_params.csv мы работаем только с общими параметрами, а в файле methods\_params.csv мы указываем частные параметры тех общих параметров, которые мы указали в файле common\_params.csv

После заполнения файла common\_params.csv мы сохраняем этот файл и продолжаем запускать ячейки в Jupyter далее до ячейки, показанной на скриншоте ниже



Запускаем эту ячейку. В папке Input cоздается файл methods\_params.csv. Далее переходим в этот файл.

Открываем файл methods\_params.csv и заполняем частные параметры.

Частные параметры, которые касаются базовых алгоритмов или композиций, можно оставлять не заполненными. Тогда будут устанавливаться заданные по умолчанию в sklearn значения.

Из чего состоит файл:

В каждой строке указано 3 сущности, разделенных запятой. Рассмотрим на примере строки «deleting anomalies method,IsolationForest,n\_estimators,”

Первая сущность — это этап. Например, «deleting anomalies method».

Вторая сущность — это названия метода (алгоритма), указанного в файле common\_params.csv. Например, «IsolationForest”

Третья сущность — это название параметра для метода (указанного во второй сущности). В нашем примере это «n\_estimators”

Далее после запятой нужно указать значение этого параметра. Например, в первой строке указано «deleting anomalies method,IsolationForest,n\_estimators,” и после “n\_estimators,” мы пишем “100”. Получается «deleting anomalies method,IsolationForest,n\_estimators,100”.

Таким образом мы заполняем файл и сохраняем его. После этого продолжаем запускать ячейки далее до конца.

По итогам модель композиции в формате pickle сохраняется в папке App/models в папке с названием в формате *‘название модели + дата создания модели’.* Все графики сохраняются в папке App/images в папке с названием в формате *‘название модели + дата создания модели’*. CSV-файл с предсказанными значениями и отчет сохраняются в папке Output в папке с названием в формате *‘название модели + дата создания модели’*.

**Приложение**

Описание общих и частных параметров:

1. **Name of model**. Это название нашей модели. Придумайте любое адекватное название на английском языке без использования специальных символов. Например, «composition».
2. **Default**. Это признак: модель по умолчанию (1) или нет (0). Пишите 0 (так как модель по умолчанию пока не сформирована).
3. **Filling gaps method.** Это метод заполнения пропусков. Выбираем из 3-ех вариантов:

* 'HardRemoval' — удаление всех объектов, имеющих хотя бы 1 пропуск. Частных параметров не имеет.
* 'InsertMeanMode' — заполнение пропусков средним или модой по признаку. Имеет частный параметр «threshold». Это пороговое количество различных значений в признаке, при котором признак считается категориальным (тип: целое положительное число). Если выбран 'InsertMeanMode', то данный параметр будет затребован в methods\_params.csv
* 'LinearImputer' — заполнение пропусков с помощью линейной регрессии. Частных параметров не имеет.

**Важно, чтобы регистр букв совпадал точно!**

1. **Deleting anomalies method.** Это метод удаления выбросов. Выбираем один из 8 вариантов (**но хорошо работают 'IsolationForest', 'Elliptic', 'LocalFactor'**):

**'ThreeSigma'** — нет частных параметров.

**'Grubbs'** — частный параметр: 'alpha' (число от 0.0 до 1.0)- уровень значимости для гипотезы.

**'Interquartile'** — есть частные параметры: 'low\_quant'-нижний квартиль (лучше брать 0.25), 'up\_quant'-верхний квартиль (лучше брать 0.75), 'coef'-коэффициент для расчёта расстояния;

**'IsolationForest'** — есть частные параметры: 'n\_estimators'— количество деревьев (рекомендуется значение 100), 'contamination' — доля выбросов (число от 0.0 до 1.0)

**'Elliptic'** — есть частный параметр: 'contamination' — доля выбросов (число от 0.0 до 1.0)

**'SVM'** — есть частный параметр: 'iters' — количество итераций алгоритма.

**'Approximate'** — есть частный параметр: 'deviation'- разница ранга матриц (целое число).

**'LocalFactor'** — 'neigh'- количество соседей (рекомендуем число 20), 'contamination'- доля выбросов (рекомендуем 0.1), 'algorithm' - алгоритм, используемый для вычисления ближайших соседей (рекомендуем 'auto').

1. **Feature selection method.** Это метод отбора признаков. Выбираем один из 9 вариантов:

'VarianceThreshold' — этот метод отбора признаков удаляет все признаки с низкой дисперсией. Имеет частный параметр threshold. Все признаки с дисперсией ниже значения threshold будут удалены.

'SelectKBest' — для каждого признака считаются значении функции. Выбираются наилучшие значения. Имеет частный параметр k (целое число), который означает количество отбираемых признаков.

'SelectPercentile' — для каждого признака считаются значении функции. Выбираются наилучшие значения. Имеет частный параметр percentile который означает процент отбираемых признаков от общего числа признаков (значение от 0 до 100).

'SelectFpr' — в рамках этого метода проводится False Positive Rate Test. Для каждого признака считается p-value и отбираются все признаки, у которых p-value меньше заданного значения. Это значение и есть частный параметр alpha (число от 0 до 1).

'SelectFdr' — проводится процедура Бенджамини-Хохберга. Для каждого признака считается p-value и отбираются все признаки, у которых p-value меньше заданного значения alpha. Alpha (число от 0 до 1) - это верхняя граница ожидаемой доли ложных отклонений.

'SelectFwe' — проводится оценка FWE (family-wise error rate). Для каждого признака считается p-value и отбираются все признаки, у которых p-value меньше заданного значения alpha. Alpha (число от 0 до 1) - это верхняя граница ожидаемой доли ложных отклонений.

'GenericUnivariateSelect' — это метод, который обобщает 5 предыдущих методов. У него 2 частных параметра: первый — это «mode» — режим ('percentile', 'k\_best', 'fpr', 'fdr', 'fwe'). Второй — число, соответствующее методу (например, процент, если выбран percentile).

'RFE' — проводится отбор признаков методом Recursive Feature Elimination, то есть методом интерационного рекурсивного исключения. Есть 2 частных параметра: n\_features\_to\_select (Количество оставляемых признаков) и step (шаг: если это число больше или равно 1, то это количество удаляемых на каждой итерации признаков. Если от 0.0 до 1.0, то это доля удаляемых на каждой итерации признаков.)

'SelectFromModel' — отбор признаков на основе оценки весовых коэффициентов модели. Частные параметры: ‘threshold’, ‘norm\_order’ и ‘max\_features’. В ходе обучения модели каждому признаку присваивается свой весовой коэффициент. Если он больше ‘threshold’, то оставляем этот признак. ‘norm\_order’ по умолчанию 1. ‘max\_features’ это количество оставляемых признаков, у которых значение весового коэффициента по модулю оказалось больше ‘threshold’. Если указать ‘threshold’ = -np.inf, то будет отбираться количество признаков равное ‘max\_features’.

1. **Composition method.** Это тип композиции. Выбираем один из 3-ех вариантов:  
   'voting','adaboost','stacking'.

Для ‘voting’ нет частных параметров.

Для 'adaboost' есть несколько частных параметров:

'n\_estimators' — количество базовых алгоритмов. Примерный порядок чисел —десятки-сотни. По умолчанию 50.

'learning\_rate' — скорость обучения. По умолчанию 1.

'algorithm' — 'SAMME' (дискретный алгоритм) или 'SAMME.R' (вещественный алгоритм бустинга). По умолчанию 'SAMME.R' (он быстрее).

'stacking' — смысл стекинга в том, что сначала обучаются базовые алгоритмы, их ответы становятся признаками в новом признаковом пространстве. Затем на основе этого признакового пространства стоится еще одна модель (метамодель). В нашем случае это логистическая регрессия.

'stacking' имеет 2 частных параметра: ‘stack\_method’ и ‘cv’.

‘stack\_method’ — имеет 4 варианта: ‘auto’ (по умолчанию), ‘predict\_proba’, ‘decision\_function’, ‘predict’. Означает методы, которые будут вызываться у каждого частного алгоритма. ‘auto’ является универсальным вариантом, который объединяет ‘predict\_proba’, ‘decision\_function’, ‘predict’. Если метод не внедрен в базовый алгоритм, то будет ошибка.

‘cv’ — количество блоков для кросс-валидации финальной модели. По умолчанию 5.

1. Следующие 3 параметра рассмотрим вместе: **neural network number, decision tree number, logistic regression number.** Тут мы указываем количество базовых алгоритмов для каждого из этих трех типов.

Если на предыдущем этапе использовался voting или stacking, то можно в каждом их этим трех типов можно указать любые числа. Если не хотим использовать какой-либо базовый алгоритм, то пишем 0.

Если использовали AdaBoost и другие алгоритмы, то выбираем нужный нам тип алгоритма и ставим 1. В остальные два типа ставим 0.

Рассмотрим частные параметры базовых алгоритмов:

**Нейронные сети**: частные параметры 'activation', 'solver', 'learning\_rate'.

'activation' — функция активации для скрытого слоя. Имеет 4 варианта: ‘identity’, ‘logistic’, ‘tanh’, ‘relu’ (по умолчанию, f(x)=max (0,x)).

'solver' — алгоритм для оптимизации весов. Имеет 3 варианта: ‘lbfgs’, ‘sgd’ (это стохастический градиентный спуск), ‘adam’ (по умолчанию). Для больших датасетов хорошо работает ‘adam’, а для маленьких ‘lbfgs’.

'learning\_rate' (имеет смысл только если 'solver' =‘sgd’) — стратегия обновления скорости обучения. Имеет 3 варианта: 'constant' (по умолчанию), 'invscaling', 'adaptive'.

'constant' — это постоянная скорость обучения.

'invscaling' — постепенно уменьшает скорость обучения на каждом временном шаге «t».

'adaptive' — поддерживает постоянную скорость обучения на уровне «learning\_rate\_init» до тех пор, пока потери от обучения продолжают уменьшаться.

**Деревья решений**: частные параметры 'criterion' и 'max\_depth'.

'criterion' — функция для измерения качества разбиения. Имеет варианта: 'gini' (по умолчанию), 'entropy'. .

'max\_depth' — максимальная глубина дерева.

**Логистическая регрессия**: имеет два частных параметра 'solver' и 'penalty'.

'solver' — алгоритм для решения проблем оптимизации. Имеет варианты: 'newton-cg', 'lbfgs' (по умолчанию), 'liblinear', 'sag', 'saga'.

'penalty' — норма, используемая для штрафа. Имеет варианты: ‘l1’ («эль» 1), ‘l2’ («эль» 2) (по умолчанию, советуем брать его), ‘elasticnet’, ‘none’.

Примечания:

* Для небольших наборов данных «liblinear» - хороший выбор, тогда как «sag» и «saga» быстрее для больших.
* «Newton-cg», «lbfgs», «sag» и «saga» поддерживают 'penalty'='l2' или 'penalty'= 'none'
* «Liblinear» и «saga» поддерживают 'penalty'='l1’ («эль» 1)
* «saga» также поддерживает 'penalty'=‘elasticnet’
* ‘Liblinear’ не поддерживает настройку 'penalty'=‘none’.

1. **Test\_ratio**. Это доля объектов, не участвующих в обучении, а используемых для измерения качества. Это число от 0 до 1 (доля). Например, «0.2».