ECS: Experiment Conduction System

Руководство по запуску обучения моделей

Разработка и поддержка:
Константин Ломотин (ke.lomotin@gmail.com)
Екатерина Козлова (hse.kozlovaes@gmail.com)

Для запуска эксперимента необходимо создать его описание, заполнив поля в файле `settings.ini` в каталоге с программой.

Затем запустить эксперимент следующей командой:

```
python ecs.py
```

Описание состоит из обязательных секций `TrainingData` и `Experiment`, а также секций гиперпараметров моделей.

Далее в рамке будут приведены примеры заполненных полей.

Секция `TrainingData`

В этой секции необходимо задать следующие настройки:

1. Название папки с датасетом для обучения. Датасеты хранятся в папке datasets.

```
dataset = ru_rgnti
```

2. Необходимо указать только одну из этих опций. Если указаны обе, тестирование будет проводиться при помощи указанного файла (приоритет опции `test_file`).
   Опция `test_percent`: доля выборки, используемой для валидации моделей.
   Опция `test_file`: название файла в каталоге выбранного датасета, который будет использоваться для валидации.

```
test_percent = 30
test_file = test_clear_annotation_balanced_w2v_mean50_48k.csv
```

Секция `Experiment`

1. Название эксперимента. Не должно включать в себя символы [\, *, ", ?, /, :, >, <, |]. Отчет по результатам эксперимента сохраняется в папке с назначенным именем в каталоге `ecs/reports/`

Если название не указано, в качестве имени назначается временна́я метка.

```
experiment_title = experiment1
```

```
experiment_title =
```

2. Идентификатор рубрикатора. Допустимо несколько значений из списка [subj, ipv, rgnti].
   Для каждого рубрикатора будет выбрана и сохранена лучшая модель.

```
rubricator = subj, rgnti
```

3. Модели для перебора. Допустимо несколько значений из списка
   - `logistic_regression`
   - `svm`
   - `perceptron`
   - `knn`
   - `decision_tree`
   - `random_forest`
   - `adaboost`

   Из перечисленных моделей будет выбрана лучшая.

   Для каждой модели в соответствующей секции необходимо указать гиперпараметры для перебора.

```
models = logistic_regression, perceptron
```

4. Бинарная классификация - 1 против всех (1 vs all). Допустимо одно значение из списка `[0, 1, true, false]`.

```
binary = 0
```

5. Количество потоков исполнения процесса. Оптимальное количество потоков зависит от количества физических ядер.

```
threads = 2
```

6. Количество фолдов кросс-валидации.
   При кросс-валидации обучающая выборка делится на `n_folds` частей (фолдов). На каждой итерации в обучении участвуют `(n_folds - 1)` частей, а оставшаяся часть используется для контроля качества.
   Таким образом, вся обучающая выборка принимает участие в обучении. В результате получается оценка эффективности выбранной модели с наиболее равномерным использованием имеющихся данных.
   Обычное значение: 3. Поле не может быть пустым.

```
n_folds = 3
```

Гиперпараметры моделей

Для выбора наиболее эффективной модели необходимо задать список моделей в поле models секции Experiment, а также создать секции с гиперпараметрами для каждого выбранного алгоритма. Названия секций должны совпадать с названиями выбранных моделей.

Используемый фреймворк *scikit-learn* позволяет настраивать модели при помощи большого числа гиперпараметров. Многие из них имеют значение по умолчанию, которое мало влияет на качество классификации. Поэтому обычно в переборе такие гиперпараметры не участвуют. Ниже будет приведено описание только тех гиперпараметров, которые используются чаще всего. **Полужирным** шрифтом будет выделено значение по умолчанию.

Для каждого гиперпараметра может быть задано несколько значений через запятую. Например:

```
activation = logistic, relu
```

```
kernel = linear, poly, rbf, sigmoid
```

Каждое добавленное значение увеличивает количество комбинаций гиперпараметров и увеличивает время работы программы, так как для каждой комбинации будет обучена и протестирована модель.

Поле гиперпараметра можно удалить или оставить пустым. Например:

```
solver =
```

В таком случае будет использоваться значение по умолчанию.

Важно! Значения некоторых гиперпараметров несовместимы между собой. Например, некоторые алгоритмы обучения не поддерживают определенные виды регуляризации. Такие комбинации гиперпараметров могут вызвать некорректную работу или падение программы.

Некоторые сочетания гиперпараметров могут быть несовместимы, но не вызывать ошибки. В таком случае один из гиперпараметров игнорируется и их не нужно удалять из описания. Далее указано, какие из них не учитываются в сочетании с другими, а какие вызывают ошибку.

Секция `logistic_regression`

Логистическая регрессия – это алгоритм, позволяющий использовать линейную регрессию для классификации. Метод заключается в построении разделяющей гиперплоскости, разделяющей классы. Логистическая функция (сигмоида) позволяет отнести объект к тому или иному классу.

Документация, информация о модели и полный список гиперпараметров доступны по ссылке.

1. `penalty`
   Вид регуляризации. Регуляризация препятствует переобучению модели, накладывая штраф на слишком высокие значения весовых коэффициентов. Вид регуляризации определяет то, как эти штрафы вычисляются. Возможные значения:
   - `l1` (**несовместимо** со значениями `newton-cg`, `sag` и `lbfgs` гиперпараметра `solver`)
   - **`l2`**
2. `C`
   Ослабление регуляризации.
   Чем ниже это значение, тем сильнее будет штраф за высокие значения коэффициентов модели (регуляризация).
   Оптимальное значение может варьироваться до нескольких порядков.
   Значение по умолчанию: **`1.0`**

```
C = 0.001, 0.1, 10
```

3. `fit_intercept`

Добавлять ли в уравнение линейной модели свободное слагаемой, т.н. *смещение*. Допустимые значения (регистр имеет значение):

- **`True`**
- `False`

4. `random_state`
   Зерно для генерации случайного числа. Служит для инициализации коэффициентов модели. Задание этого числа вручную позволяет сделать эксперимент повторяемым. Допустимые значения: целые числа.
5. `solver`
   Алгоритм оптимизации, применяющийся для обучения. Допустимые значения:
   - `newton-cg`
   - `lbfgs`
   - **`liblinear`**
   - `sag`
   - `saga`

   Примечание. Алгоритм `liblinear` хорошо подходит для небольших датасетов, тогда как `sag` и `saga` работают быстрее на крупных наборах данных.
6. `max_iter`
   Ограничение на максимальное количество итераций обучения.
   Значение по умолчанию: **`100`**.

Секция `svm`

Метод опорных векторов (Support Vector Machine) – линейный алгоритм, часто применяющийся в классификации. Его сильной стороной является возможность нелинейного преобразования пространства признаков при помощи перехода от скалярных произведений к произвольным ядрам.

Документация, информация о модели и полный список гиперпараметров доступны по ссылке.

1. `C`
   Ослабление регуляризации.
   Чем ниже это значение, тем сильнее будет штраф за высокие значения коэффициентов модели (регуляризация).
   Оптимальное значение может варьироваться до нескольких порядков.
   Значение по умолчанию: **`1.0`**

```
C = 0.001, 0.1, 10
```

2. `kernel`
   Ядро, используемое для преобразования пространства признаков.
   Допустимые значения:
   - `linear`
   - `poly`
   - **`rbf`**
   - `sigmoid`
3. `degree`
   Степень полинома при использовании полиномиального ядра (значение `poly` опции `kernel`). Игнорируется при использовании других ядер.
   Значение по умолчанию: **`3`**.
4. `gamma`
   Коэффициент ядра при значениях `rbf`, `poly` и `sigmoid` опции `kernel`. Игнорируется при использовании линейного ядра (`linear`).
   Значение по умолчанию: **1/n_features**, где n_features – количество признаков (размерность вектора текста).
5. `coef0`
   Свободный член в уравнении гиперплоскости. Игнорируется при использовании ядер `linear` и `rbf`.
   Значение по умолчанию: **`0`**.
6. `max_iter`
   Ограничение на максимальное количество итераций обучения.
   Значение по умолчанию: **`-1`** (без ограничения).
7. `random_state`
   Зерно для генерации случайного числа. Служит для инициализации коэффициентов модели. Задание этого числа вручную позволяет сделать эксперимент повторяемым. Допустимые значения: целые числа.

Секция `perceptron`

Простая многослойная нейронная сеть, в которой сигнал распространяется от входов к выходам.

Документация, информация о модели и полный список гиперпараметров доступны по ссылке.

1. `hidden_layer_sizes`
   Количество нейронов в скрытых слоях нейросети.

Если xn - количество нейронов в n-м скрытом слое, то запись `[x1, x2], [y1, y2, y3]` означает, что сначала будет обучена нейросеть с двумя скрытыми слоями с `x1` и `x2` нейронами, а затем модель с тремя слоями с `y1`, `y2` и `y3` нейронами.

Описание каждой нейросети обязательно заключать в квадратные скобки:

```
hidden_layer_sizes = [10, 15, 1], [6]
```

2. `activation`

   Функция активации нейронов. На данный момент возможно использование одной функции активации для всех нейронов. Список допустимых значений:
   - `identity`
   - `logistic`
   - `tanh`
   - **`relu`**

3. `solver`

   Алгоритм оптимизации, применяющийся для обучения. Допустимые значения:
   - `lbfgs`
   - `sgd`
   - **`adam`**

   Примечание. Для больших датасетов алгоритм `adam` работает быстрее на обучении и валидации на больших датасетах (тысячи и десятки тысяч записей). Алгоритм `lbfgs` сходится и работает быстрее на маленьких наборах данных.

4. `alpha`

   Коэффициент регуляризации. Регуляризация препятствует переобучению модели, накладывая штраф на слишком высокие значения весовых коэффициентов.

   Значение по умолчанию: **`0.0001`**

5. `learning_rate`

   Коэффициент скорости обучения. Слишком большое значение может привести к тому, что алгоритм обучения никогда не сойдется. Слишком маленький коэффициент приведет к слишком долгой сходимости. Используется только при алгоритме оптимизации `sgd`.

   Допустимые значения:
   - **`constant`**

- `invscaling`
- `adaptive`

6. `learning_rate_init`
   Начальное значение гиперпараметра `learning_rate`. При постоянной скорости обучения остается неизменным, а при `invscaling` и `adaptive` уменьшается со временем.
   Значение по умолчанию: **`0.001`**
7. `power_t`
   Значение по умолчанию: **`0.5`**. При использовании `invscaling` `learning_rate` скорость обучения меняется по формуле:

$$effective\_learning\_rate = learning\_rate\_init / pow(t, power\_t)$$

   где t – номер итерации,
   effective_learning_rate – скорость обучения на шаге t,
   learning_rate_init – начальная скорость обучения.

   Гиперпараметр игнорируется при использовании других видов скорости обучения.

8. `max_iter`
   Ограничение на максимальное количество итераций обучения.
   Значение по умолчанию: **`200`**.
9. `random_state`
   Зерно для генерации случайного числа. Служит для инициализации коэффициентов модели. Задание этого числа вручную позволяет сделать эксперимент повторяемым. Допустимые значения: целые числа.

Секция `knn`

Метрический алгоритм, суть которого заключается в том, что новый объект присваивается тому классу, который является наиболее распространённым среди `n_neighbors` соседей данного элемента, классы которых уже известны.

Документация, информация о модели и полный список гиперпараметров доступны по ссылке.

1. `n_neighbors`
   Количество ближайших соседей объекта. Оптимальное значение зависит от данных. Большое количество соседей часто помогает подавить шумы, но делает границу между классами менее отчетливой.
   Значение по умолчанию: **5**.
2. `weights`

Весовая функция, используемая в предсказании. Допустимые значения:
   - **`uniform`** (все соседи оцениваются одинаково)
   - `distance` (соседи поблизости вносят больший вклад, чем дальние)
3. `p`
   Степень метрики Минковского. При $p = 1$ используется Манхэттенское расстояние, при $p = 2$ используется Евклидово расстояние, при произвольном p используется метрика Минковского в общем виде.
   Значение по умолчанию: **2**.

Секция `decision_tree`

Алгоритм поддержки принятия решений, выявляющий закономерности в признаках объектов обучающей выборки.

Документация, информация о модели и полный список гиперпараметров доступны по ссылке.

1. `criterion`
   Критерий меры качества разбиения образцов по ветвям. Допустимые значения:
   - **`gini`**
   - `entropy`
2. `splitter`
   Стратегия выбора разбиения. Допустимые значения:
   - **`best`** (лучшее разбиение)
   - `random` (лучшее случайное разбиение)
3. `max_depth`
   Максимальная глубина дерева.
   Значение по умолчанию: **None** (неограниченно).
   Примечание. Парсинг значения None пока не реализован, поэтому оно не может быть указано. Вместо этого допустимо оставить поле пустым.
4. `min_samples_split`
   Минимальное количество образцов, необходимое для разбиения узла.
   Значение по умолчанию: **2**.
5. `min_samples_leaf`
   Минимальное количество образцов для формирования листа дерева.
   Значение по умолчанию: **1**.
6. `max_features`
   Количество признаков, рассматриваемых при каждом разбиении.

Значение по умолчанию: **None** (все признаки).

Примечание. Парсинг значения None пока не реализован, поэтому оно не может быть указано. Вместо этого допустимо оставить поле пустым.

Допустимые значения:

- целое число признаков
- число с плавающей точкой (доля признаков от всех)
- `auto` (в этой реализации эквивалентно sqrt)
- `sqrt` (квадратный корень от всех признаков)
- `log2` (логарифм по основанию 2 от всех признаков)

7. `random_state`

   Зерно для генерации случайного числа. Служит для инициализации коэффициентов модели. Задание этого числа вручную позволяет сделать эксперимент повторяемым. Допустимые значения: целые числа.

Секция `random_forest`

Классификатор «Случайный лес» основан на бэггинге (обучении нескольких моделей на разных подвыборках с последующим голосованием), однако в дополнение к нему случайным образом выбирает подмножество признаков в каждом узле, с целью сделать деревья более независимыми;

Документация, информация о модели и полный список гиперпараметров доступны по ссылке.

1. `n_estimators`

   Количество деревьев решений в составе модели.

   Значение по умолчанию: **`10`**.

2. `criterion`

   Критерий меры качества разбиения образцов по ветвям. Допустимые значения:

   - **`gini`**
   - `entropy`

3. `max_depth`

   Максимальная глубина дерева.

   Значение по умолчанию: **None** (неограниченно).

   Примечание. Парсинг значения None пока не реализован, поэтому оно не может быть указано. Вместо этого допустимо оставить поле пустым.

4. `min_samples_split`

   Минимальное количество образцов, необходимое для разбиения узла.

   Значение по умолчанию: **2**.

5. `min_samples_leaf`
   Минимальное количество образцов для формирования листа дерева.
   Значение по умолчанию: **1**.
6. `max_features`
   Количество признаков, рассматриваемых при каждом разбиении.
   Значение по умолчанию: **None** (все признаки).
   Примечание. Парсинг значения None пока не реализован, поэтому оно не может быть указано. Вместо этого допустимо оставить поле пустым.
   Допустимые значения:
   - целое число признаков
   - число с плавающей точкой (доля признаков от всех)
   - **`auto`** (в этой реализации эквивалентно sqrt)
   - `sqrt` (квадратный корень от всех признаков)
   - `log2` (логарифм по основанию 2 от всех признаков)
7. `random_state`
   Зерно для генерации случайного числа. Служит для инициализации коэффициентов модели. Задание этого числа вручную позволяет сделать эксперимент повторяемым. Допустимые значения: целые числа.

Секция `adaboost`

Алгоритм адаптивного бустинга. Идея бустинга заключается в построении каскада моделей, где каждая последующая модель обучается на ошибках предыдущей.

Документация, информация о модели и полный список гиперпараметров доступны по ссылке.

Эта реализация ансамблевого алгоритма AdaBoost способна работать с различными классификаторами, но в ECS на текущий момент реализован адаптивный бустинг на базе деревьев решений.

1. `n_estimators`
   Максимальное количество моделей, объединяемых в бустинг. Обучение прекращается, как только завершается обучение этого количества моделей. Обучение может завершиться раньше, если ошибка станет убывать слишком медленно.
   Значение по умолчанию: **`50`**.
2. `learning_rate`

Вклад каждой модели в составе бустинга. Для оптимального результата нужно найти компромисс между количеством моделей и долей вклада каждой из них.
Значение по умолчанию: **1**.

3. `algorithm`
   Алгоритм обучения бустинга. Допустимые значения:
   - **`SAMME.R`** (как правило, сходится быстрее и достигает более низкой ошибки)
   - `SAMME`
4. `random_state`
   Зерно для генерации случайного числа. Служит для инициализации коэффициентов модели. Задание этого числа вручную позволяет сделать эксперимент повторяемым. Допустимые значения: целые числа.