Студент: Симонов Кирилл
Дата: 7 ноября 2021 г.

Отчёт об использовании CodeStyle инструментов

1) **Выбор инструментов**

В качетсве инструментов для эффективного анализа и рефакторинга кода были выбраны:

- `pycodestyle`

  Данный стилистический линтер был выбран вследствие того, что текущий проект написан на языке программирования Python и для читаемости кода и удобства последующей с ним работы необходимо привести его к соответствующему виду, описанному в стандарте PEP 8.

- `pyflakes`

  Данный логический линтер был выбран для проверки проекта на наличие паттернов опасного кода и его возможное неоднозначное поведение.

- `mccabe`

  Данная утилита была выбрана для анализа и оценки цикломатической сложности программного кода, что позволит понять в каких местах он переусложнен, и что необходимо поправить.

Можно было взять просто flake8 с соответствующими плагинами, но тогда не получилось бы набрать 3 инструмента, а был бы один универсальный.

2) **Конфигурация инструментов**

- `pycodestyle`

  Данный стилистический линетер был установлен с помощью пакетного менеджера pip согласно инструкции с GitHub репозитория проекта и был использован в соответствии с инструкцией.

```
C:\Windows\system32>python -m pip install pycodestyle
Collecting pycodestyle
  Downloading pycodestyle-2.8.0-py2.py3-none-any.whl (42 kB)
     |████████████████████████████████| 42 kB 812 kB/s
Installing collected packages: pycodestyle
Successfully installed pycodestyle-2.8.0
```

```
pycodestyle --show-source --show-pep8 src > reports/pycodestyle_report
```

  Для того, чтобы при работе с утилитой каждый раз отдельно не прописывать параметры её вызова, был создан конфигурационный файл ***setup.cfg***, в котором были указаны настройки для некоторых параметров:

```
[pycodestyle]
show-source = True
max-line-length = 100
count = False
```

  * увеличина максимально-допустимая длина строки;
  * включена функция отображения исходного кода, в котором найдена ошибка;
  * отключен вывод счётчика ошибок.

– `pyflakes`

Данный логический линтер был установлен аналогичным образом и использован согласно соответствующей инструкции из GitHub репозитория; никаких дополнительных настроек не производилось.

```
C:\Windows\system32>python -m pip install --upgrade pyflakes
Collecting pyflakes
  Downloading pyflakes-2.4.0-py2.py3-none-any.whl (69 kB)
     |████████████████████████████████| 69 kB 596 kB/s
Installing collected packages: pyflakes
```

```
pyflakes src > reports/pyflakes_report
```

```
[pyflakes]
# Nothing
```

– `mccabe`

Аналогично предыдущим пунктам пакет был установлен использован в соответствии с инструкцией из GitHub репозитория проекта.

```
C:\Windows\system32>python -m pip install --upgrade mccabe
Collecting mccabe
  Downloading mccabe-0.6.1-py2.py3-none-any.whl (8.6 kB)
Installing collected packages: mccabe
Successfully installed mccabe-0.6.1
```

```
python -m mccabe src\pipeline.py > mccabe_report

python -m mccabe src\data\load_and_prepare.py >> reports/mccabe_report

python -m mccabe src\models\create train predict.py >> reports/mccabe report
```

```
[mccabe]
# Nothing
```

Стоит отметить, что `mccabe` *из коробки обладает довольно небольшим функционалом и умеет просто оценивать цикломатическую сложность функций из строго переданного файла.*

В большей степени эта утилита раскрывается, когда используется как плагин к flake8, о котором упоминалось ранее.

3) Результаты

– `pycodestyle`

Так как код был предварительно отформатирован встроенными утилитами IDE PyCharm, анализ на соответствие кода стандарту PEP8 показал, что критических ошибок не было обнаружено.

Однако, в нескольких местах исполняемого кода рекомендовалось сократить длинну строки, так как их количество превышает установленное нами максимально допустимое, а значит на некоторых компьютерах код может некорректно (некрасиво) отображаться.

Отчёт можно посмотреть в файле pycodestyle_report

– `pyflakes`

Стандартно настроенный логический линтер обнаружил в проекте две некритические ошибки: импортирование функционала сторонней библиотеки, который не был использован, а также отсутствие в конструкции f-строки каких-либо плейсхолдеров.

Первая ошибка может указывать на то, что был реализован не весь функционал из изначально планируемого, а вторая – на то, что функционал f-строки был излишен и можно обойтись обычной строкой.

В любом случае обе эти ошибки не являются критическими и легко поправимы.

Отчёт можно посмотреть в файле pyflakes_report

– `mccabe`

Оценка цикломатической сложности кода показала, что максимальное значение этой величины во всём проекте не более 5.

Согласно википедии (на которую в том числе ссылаются авторы проекта) код считается перегруженным и неоптимальным, если его цикламотическая сложность равна 10 и более.

Следовательно, можно сделать вывод, что наш код выдерживает проверку на сложность и перегруженность.

Отчёт можно посмотреть в файле mccabe_report