Вопрос 1

Верно

Баллов: 1,00 из 1,00

В этом задании мы предлагаем вам собрать регулярное выражение из "деталей" так, чтобы оно **выделяло в отдельные токены знаки препинания, числа и слова**.

А именно:

- Числа с плавающей точкой вида 123.23 выделяются в один токен. Десятичным разделителем может быть точка или запятая.
- Число может быть отрицательным: иметь знак -123.4
- Целой части числа может вовсе не быть: последовательности -0.15 и $-.15$ означают одно и то же число.
- При этом числа с нулевой дробной частью не допускаются: строка "$12345.$" будет разделена на два токена "12345" и "."
- Идущие подряд знаки препинания выделяются каждый в отдельный токен.
- Наконец множество букв в словах ограничивается только кириллическим алфавитом (33 буквы, включая букву ё).

Обратите внимание, что **в результате токенизации не должно получаться пустых токенов.**

Вы можете использовать следующие тесты для отладки своего регулярного выражения:

Текст	Результат
`Контактный телефон: 123123.`	`контактный телефон : 123123 .`
`Что-нибудь надо придумать.`	`что - нибудь надо придумать .`
`Значение числа E=2.7182.`	`значение числа e = 2.7182 .`
`Демон123, как тебя зовут в реале?`	`демон 123 , как тебя зовут в реале ?`
`-1-.15=-1.15`	`-1 -.15 = -1.15`
`- 1 - .15 = -1.15`	`- 1 - .15 = -1.15`
`Какого ;%:?* тут происходит?`	`какого ; % : ? * тут происходит ?`

Детали "конструктора", из которых можно собрать решение задачи:

- [а-яё]+ // ненулевая последовательность любых букв русского алфавита.
- -?\d*[.,]?\d+ // возможно знак, возможная целая часть числа, возможно, десятичный знак и остальная часть числа.
- \S // любой символ, кроме разделителей (пробелов, переносов строк)
- | // **| отвечает за выбор из двух паттернов**, например: [а-яё]+|\d последовательность букв или одна цифра.

Хорошие ресурсы для изучения и отладки регулярных выражений:

- https://regex101.com/
- https://docs.python.org/3/howto/regex.html
- https://habr.com/ru/post/349860/

Для примера:

Ввод	Результат
`Мама мыла раму.`	`мама мыла раму .`

Ответ: (штрафной режим: 0 %)

Сброс ответа

```
import re
import sys


# модифицируйте это регулярное выражение
TOKENIZE_RE = re.compile(r'[а-яё]+|-?\d*[.,]?\d+|[\d]+|\S', re.I)


def tokenize(txt):
    return TOKENIZE_RE.findall(txt)


for line in sys.stdin:
    print(' '.join(tokenize(line.strip().lower())))

```

Вопрос 2

Верно

Баллов: 1,00 из 1,00

Дана следующая коллекция текстов. Постройте словарь (отображение из строкового представления токенов в их номера) и вектор весов (DF).

$DF(w) = \frac{DocCount(w,c)}{Size(c)}$ - частота слова w в коллекции c (отношение количества документов, в которых слово используется, к общему количеству документов).

> Казнить нельзя, помиловать. Нельзя наказывать.
>
> Казнить, нельзя помиловать. Нельзя освободить.
>
> Нельзя не помиловать.
>
> Обязательно освободить.

При токенизации используйте регулярное выражение из семинара: [\w\d]+. После токенизации все токены нужно привести к нижнему регистру. Фильтрацию по частоте не использовать.

Ответ запишите в две строки:

1. в первой строке - содержимое словаря - список уникальных токенов через пробел в порядке возрастания частоты встречаемости. При одинаковой частоте сортировать по алфавиту.
2. во второй строке - список весов (DF) токенов, округлённых до 2 знака после запятой и разделённых пробелами, в том же порядке, что и токены в первой строке.

Ответ: (штрафной режим: 0 %)

```
наказывать не обязательно казнить освободить нельзя помиловать
0.25 0.25 0.25 0.5 0.5 0.75 0.75
```

	Comment
✔	Верное решение. Так держать!

Прошли все тесты! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос 3

Нет ответа

Баллов: 0,00 из 1,00

Постройте матрицу признаков для текстов с шага 5 с использованием словаря и вектора весов, полученного на шаге 5. Используйте взвешивание $lTFIDF = \ln(TF + 1) \cdot IDF$.

Значения признаков следует отмасштабировать так, чтобы для каждого признака его среднее значение по выборке равнялось 0, а среднеквадратичное отклонение 1: $x_i^{scaled} = \frac{x_i - E(x)}{\sigma(x)}$.

В результате масштабирования **для каждого столбца** матрицы признаков среднее должно равняться 0, а среднеквадратичное отклонение 1.

При расчёте среднеквадратического отклонения необходимо использовать скорректированную оценку $\sigma = \sqrt{\frac{\sum_{i-1}^{n}(x_i - E(x))^2}{n-1}}$. Чтобы получить такую оценку с помощью numpy, необходимо передать параметр ddof=1:

```
feature_matrix = np.zeros((num_docs, num_feats))
feats_std = feature_matrix.std(0, ddof=1)
```

Ответ отформатируйте так, чтобы на каждой строке были признаки одного документа. Порядок столбцов должен соответствовать порядку слов в словаре (как в ответе на шаге 5, **по возрастанию df**). Столбцы разделяйте одним пробелом. В качестве разделителя целой и дробной части используйте точку или запятую. Округлять значения не обязательно. Решение, при проверке, автоматически округлится до двух знаков. Метод округления - либо "математический", либо свойственный Python rounding half to even strategy, если интересно, посмотрите IEEE 754.

Пример ответа для первых двух документов (до полного ответа не хватает ещё двух строк):

```
1.5  -0.5 -0.5 0.87 -0.76 0.60 0.16
-0.5 -0.5 -0.5 0.87 0.18  0.60 0.16
```

Ответ: (штрафной режим: 0 %)

Вопрос Инфо

Вопрос 4

Выполнен

Баллов: 1,00 из 1,00

Дана выборка с распределением классов $p(0) = 0.6, p(1) = 0.4$. Выберите решающее правило, которое позволит получить наивысшую accuracy.

Выберите один ответ:

a. Всегда предсказывать 1

b. Всегда предсказывать 0

c. Случайное угадывание (монетка, распределение Бернулли) с $p(1) = 0.4$

d. Случайное угадывание (монетка, распределение Бернулли) с $p(1) = 0.6$

e. Случайное угадывание (монетка, распределение Бернулли) с $p(1) = 0.5$

Ваш ответ верный.

Вопрос 5

Выполнен

Баллов: 1,00 из 1,00

Вы занимаетесь тематической классификацией текстов (например, сортируете объявления пользователей по категориям). У вас есть коллекция из 100000 текстов, в которой содержится 10000 уникальных токенов (это размер словаря). Вы хотите построить матрицу признаков для текстов в вашем датасете, чтобы затем обучить логистическую регрессию.

Построив матрицу, вы обнаруживаете, что 99.5% значений матрицы - нулевые.

Если бы вы хранили датасет в плотной матрице (например, np.array), то вам бы потребовалось достаточно много памяти (каждое значение признака занимает 4 байта, тип np.float32):

$dense = |texts| \times |vocab| \times 4bytes = 4 \cdot 10^9 bytes \approx 3814.7 Megabytes$ - здесь мы считаем, что в мегабайте 1024 килобайт, в килобайте - 1024 байт.

Вместо этого вы решаете хранить датасет в разреженной матрице в формате COO (coordinate, scipy.sparse.coo_matrix). В этом формате для каждого ненулевого элемента хранится три числа: значение элемента, номер столбца и номер строки. Для хранения координат используется тип np.uint32, 4 байта на каждое значение.

Оцените количество памяти, которое экономится при использовании разреженной матрицы для хранения датасета.

Выберите один ответ:

a. sparse = |texts| x |vocab| x 4bytes x 0.005 =2 * 10^7 bytes
SavedMemory = (dense - sparse)/bytes_in_megabyte ~ 3795.6 Megabytes

b. sparse = |texts| x |vocab| x (4bytes+4bytes+4bytes) x 0.995 = 1.194 * 10^{10} bytes
SavedMemory = (sparse - dense)/bytes_in_megabyte ~ 7572.1Megabytes

c. sparse = |texts| x |vocab| x (4bytes) x 0.995 = 3.98 * 10^9 bytes
SavedMemory = (sparse - dense)/bytes_in_megabyte ~ 19Megabytes

d. sparse = |texts| x |vocab| x (4bytes+4bytes+4bytes) x 0.005 = 6 * 10^7 bytes
SavedMemory = (dense- sparse)/bytes_in_megabyte ~ 3757.5 Megabytes

Ваш ответ верный.

Вопрос Инфо

Вопрос 6

Выполнен

Баллов: 1,00 из 1,00

Выберите правильные характеристики для эффекта переобучения

Выберите один или несколько ответов:

- [x] a. Модель очень хорошо работает на обучающей выборке и гораздо хуже - на валидационной.
- [ ] b. Модель плохо работает на обучающей выборке.
- [x] c. Модель выделила случайные закономерности, не существующие в процессах реального мира, породивших обучающую выборку.
- [ ] d. Модель очень хорошо работает и на обучающей выборке и на валидационной.

Ваш ответ верный.