**Инструкция к сдаче**

1. Настоятельно рекомендуем сдавать практическое задание в виде ссылки на личный репозиторий на github.
2. Рекомендуемый способ организации данных в репозитории: создать отдельные папки по темам и помещать в них отдельные файлы для каждой задачи с правильным расширением.

Ссылка на инструкцию по работе с git и сдачу практики:

<https://docs.google.com/document/d/1RAT_ukE39iOfbz1xa39QXae2hBUEZ4U6Fko_wFDdrsM/edit>

Ссылка на видеокурс по Git:

<https://geekbrains.ru/courses/66>

Если остались сложности с системой git, то обратитесь к преподавателю или наставнику.

Тональность текста

## В блокноте text\_classification.ipynb разобрали анализ тональности используя полносвязанную сеть сделать на тех же данных

1. tf-idf/count vectorizer + логистическую регрессию

2. Обучить вашу архитектуре сети возможно туже что и была на занятии проверить

* взять предобученный эмбединг(к примеру word2vec) и загрузить в слой Embedding
* взять слой Embedding без предобученных весов

Сравнить все подходы в том числе и полносвязанную сеть что лучше отработало

## Тема “Классификация текста”

В качестве заготовки для задания прогоним часть 2ого домашнего задания. Нам необходимо получить разреженные матрицы, используя CountVectorizer, TfidfVectorizer для 'tweet\_stemmed' и 'tweet\_lemmatized' столбцов (4 матрицы).

***Задание 1.***

Построим модель LogisticRegression, используя Bag-of-Words признаки для столбца combine\_df['tweet\_stemmed'].

- Поделим Bag-of-Words признаки на train, test (train заканчивается на 31962 строке combine\_df)

- Ответами является столбец train\_df['label']

- Рассчитаем predict\_proba, приведем prediction в в бинарный вид: если предсказание >= 0.3 то 1, иначе 0, тип заменим на int

- Рассчитаем f1\_score

Повторим аналогично для столбца combine\_df['tweet\_lemmatized'].

***Задание 2.***

Построим модель LogisticRegression, используя TF-IDF признаки для столбца combine\_df['tweet\_stemmed'].

- Поделим TF-IDF признаки на train, test (train заканчивается на 31962 строке combine\_df)

- Ответами является столбец train\_df['label']

- Рассчитаем predict\_proba, приведем prediction в в бинарный вид: если предсказание >= 0.3 то 1, иначе 0, тип заменим на int

- Рассчитаем f1\_score

Повторим аналогично для столбца combine\_df['tweet\_lemmatized'].

***Задание 3.***

Выведите результаты f1-score всех моделей, сделайте вывод.

***Задание 4.***

Теперь перейдем к визуализации. Посмотрим, какие слова являются наиболее популярные в датасете с помощью облака слов (WordCloud).

Облако слов - это визуализация, в которой наиболее частые слова большого размера, а менее частые слова меньшего размера.

- объединим слова в одну строку

- создадим словарь частот слов с помощью collections.Counter

- нарисуем облако слов с частотами слов с помощью WordCloud.generate\_from\_frequencies()

- используем nltk.corpus.stopwords как параметр stopwords, чтобы убрать "мусорные" частотные слова

***Задание 5.***

Теперь отобразим облако слов для отзывов, не содержащих токсичных комментариев (combine\_df['label'] == 0).

***Задание 6.***

Теперь отобразим облако слов для отзывов, содержащих токсичные комментарии (combine\_df['label'] == 1).