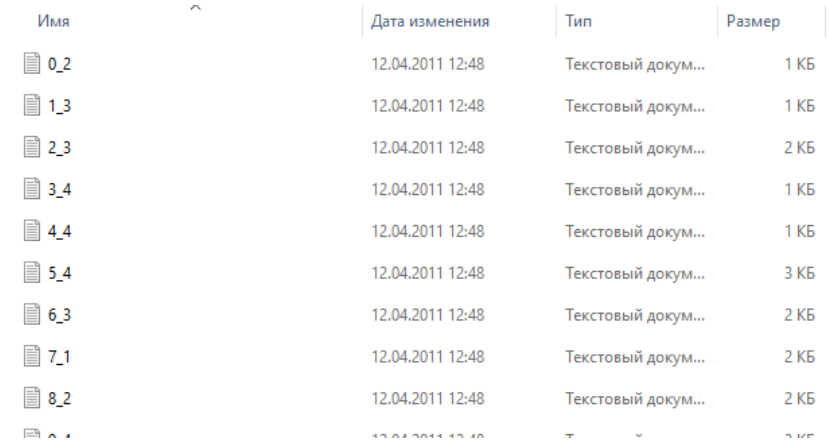
Описание задания:

1. Обучить модель на языке Python для классификации отзывов.
2. Разработать веб-сервис на базе фреймворка Django для ввода отзыва о фильме с автоматическим присвоением рейтинга (от 1 до 10) и статуса комментария (положительный или отрицательный).
3. Развернуть сервис в открытом доступе для оценки работоспособности прототипа.
4. Подготовить отчет о работе с оценкой точности полученного результата на тестовой выборке.
5. Отправить ответным письмом ссылку на прототип сервиса, ссылку на открытый репозиторий github с исходным кодом проекта, отчет о проделанной работе в формате pdf.

Моя работа состояла из следующих этапов:

1. Формирование датасетов:

Мне были предоставлены данные - 50 тыс. файлов типа .txt. Каждый файл содержал один отзыв, а его название было построено в виде “индекс_рейтинг”:



Имя	Дата изменения	Тип	Размер
0_2	12.04.2011 12:48	Текстовый докум...	1 КБ
1_3	12.04.2011 12:48	Текстовый докум...	1 КБ
2_3	12.04.2011 12:48	Текстовый докум...	2 КБ
3_4	12.04.2011 12:48	Текстовый докум...	1 КБ
4_4	12.04.2011 12:48	Текстовый докум...	1 КБ
5_4	12.04.2011 12:48	Текстовый докум...	3 КБ
6_3	12.04.2011 12:48	Текстовый докум...	2 КБ
7_1	12.04.2011 12:48	Текстовый докум...	2 КБ
8_2	12.04.2011 12:48	Текстовый докум...	2 КБ

Файлы поделены по папкам следующим образом:

./data/train/pos/ - папка с позитивными (рейтинг 6-10) комментариями для обучения,

./data/train/neg/ - папка с негативными (рейтинг 1-5) комментариями для обучения,

./data/test/pos/ - папка с позитивными (рейтинг 6-10) комментариями для обучения,

./data/test/neg/ - папка с негативными (рейтинг 1-5) комментариями для обучения.

На этом этапе были написаны две функции:

- Функция get_text_rating получает путь к файлу с комментарием, возвращает кортеж из текста комментария и рейтинга (он получен из имени файла).
- Функция get_df_comments_ratings принимает путь к папке, в которой хранятся текстовые файлы и возвращает датасет с комментариями и рейтингами.

С помощью этих функций я построила четыре таблицы (train pos, test pos, train neg, test neg), имеющие две колонки (comment – текст одного комментария, rating - рейтинг комментария).

Из таблиц для обучения случайно выбрала несколько негативных и позитивных комментариев, чтобы проверить качество разметки, разметка оказалась верна.

Объединила негативные и позитивные комментарии для обучения, то же сделала и для теста. Сохранила в .\data\train.csv и .\data\test.csv.

2. Подготовка данных

```
--------------------------------
Обучающая выборка:
--------------------------------
Количество пустых значений:

comment    0
rating     0
dtype: int64
```

	rating
count	25000.000000
mean	5.477720
std	3.466477
min	1.000000
25%	2.000000
50%	5.500000
75%	9.000000
max	10.000000

	comment	rating
0	Story of a man who has unnatural feelings for ...	3
1	Airport '77 starts as a brand new luxury 747 p...	4
2	This film lacked something I couldn't put my f...	4
3	Sorry everyone,,, I know this is supposed to b...	1
4	When I was little my parents took me along to ...	1
...	...	...
24995	Seeing as the vote average was pretty low, and...	9
24996	The plot had some wretched, unbelievable twist...	8
24997	I am amazed at how this movie(and most others ...	10
24998	A Christmas Together actually came before my t...	8
24999	Working-class romantic drama from director Mar...	7

25000 rows × 2 columns

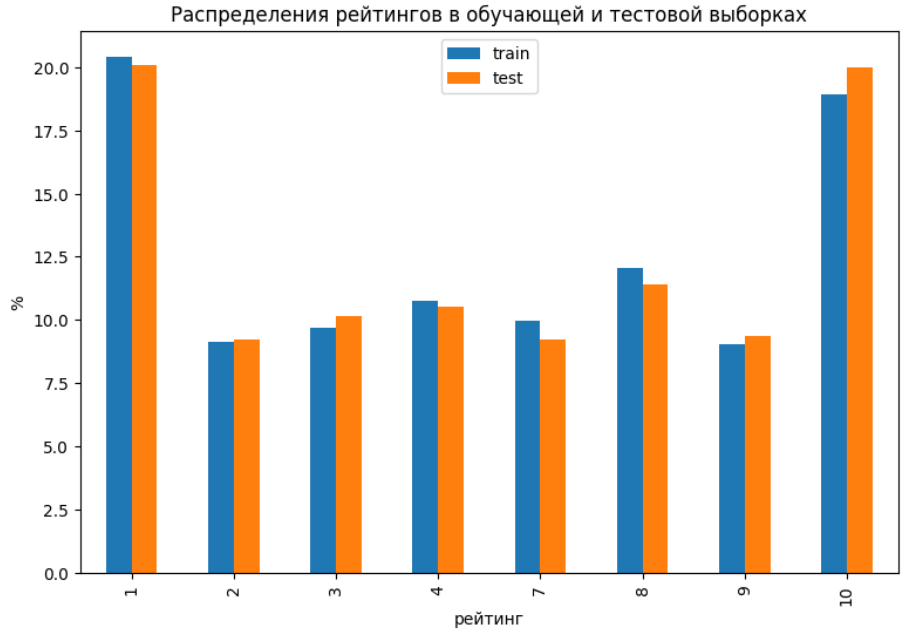
```
--------------------------------
Тестовая выборка:
--------------------------------
Количество пустых значений:

comment    0
rating     0
dtype: int64
```

	rating
count	25000.000000
mean	5.512960
std	3.490902
min	1.000000
25%	2.000000
50%	5.500000
75%	9.000000
max	10.000000

	comment	rating
0	Once again Mr. Costner has dragged out a movie...	2
1	This is an example of why the majority of acti...	4
2	First of all I hate those moronic rappers, who...	1
3	Not even the Beatles could write songs everyon...	3
4	Brass pictures (movies is not a fitting word f...	3
...	...	...
24995	I was extraordinarily impressed by this film. ...	8
24996	Although I'm not a golf fan, I attended a snea...	10
24997	From the start of "The Edge Of Love", the view...	8
24998	This movie, with all its complexity and subtle...	10
24999	I've seen this story before but my kids haven'...	7

25000 rows × 2 columns

Датасеты были сформированны корректно. 25000 комментариев в обучающей выборке (негативные/позитивные 50 на 50), 25000 комментариев в тестовой выборке (негативные/позитивные 50 на 50). Пустых ячеек нет. Негативными считаются комментарии от 1 до 4 включительно, позитивными - от 7 до 10 включительно. Нейтральных комментариев нет ни в обучении, ни в тесте (комментарии с рейтингами 5 или 6).



Распределения рейтингов для обучающей выборки и тестовой очень похожи. И там и там имеется дисбаланс, оценок 1 и 10 примерно в два раза больше, чем остальных. Это логично, люди обычно чаще ставят крайние оценки, чем средние.

Принято решение обработать текстовую колонку comment следующим образом:

- Привести текст к нижнему регистру.
- Удалить все символы, не являющиеся буквами английского алфавита, цифрами или пробелами.
- Лемматизировать слова.
- Удалить стоп-слова английского языка.

Для реализации этих пунктов я воспользовалась библиотекой nltk и мною были написаны следующие функции:

- get_cleared_comment - функция принимает текст, возвращает текст в нижнем регистре очищенный от лишних символов.
- get_wordnet_pos - функция принимает слово, возвращает словарь, где возвращается значение часть речи (pos_tag).
- get_lemmatized_comment - функция принимает комментарий, возвращает комментарий с лемматизировнным набором слов и с удаленными стоп-словами.

Вид датасета, после обработки:

	comment	rating
0	mr costner drag movie far longer necessary asi...	2
1	example majority action film generic boring re...	4
2	first hate moronic rapper could nt act gun pre...	1
3	even beatles could write song everyone like al...	3
4	brass picture movie fitting word really somewh...	3
...	...	...
24995	extraordinarily impressed film one best sport ...	8
24996	although golf fan attend sneak preview movie a...	10
24997	start edge love viewer transport strike world ...	8
24998	movie complexity subtlety make one thought pro...	10
24999	see story kid boy troubled past join military ...	7

25000 rows × 2 columns

Сохранила в .\data\train _lem.csv и .\data\test _lem.csv.

3. Обучение

Задача предсказания рейтинга представляет собой задачу порядковой регрессии. Следовало исследовать разную обработку текста и разные типы моделей.

Так как присутствовали ограничения по времени сдачи работы и ограничения, связанные разработкой прототипа веб-сервиса, я использовала обработку данных BOW, TF-IDF (только с униграммами и с униграммами и биграммами) и модели линейные, LightGBM, чтобы подбор параметров модели не занял большого количества времени, и модели не занимали много памяти на сервере.

Я воспользовалась библиотекой sklearn и lightgbm, а конкретно классами Pipeline, CountVectorizer, TfidfVectorizer, GridSearchCV, LinearRegression, Lasso, Ridge, LGBMRegressor.

Модели были обучены на обучающей выборке с 5-фолдовой кросс-валидацией. В качестве метрики, по которой отбиралась лучшая модель, была выбрана RMSE. Для линейных моделей я подбирала только гиперпараметр регуляризации, а для модели LightGBM только гиперпараметры количества деревьев и глубины дерева (глубину дерева подбирала небольшую, чтобы не было переобучения).

5.2.2 Ridge + TF-IDF + unigram + bigram

In [39]:
```
%%time
pipe_tfidf_ridge = Pipeline([('tfidf', TfidfVectorizer(ngram_range=(1, 2))), ('ridge', Ridge())])
param_grid = {'ridge__alpha': np.arange(0.1, 1.1, 0.1),
              'ridge__random_state': [RANDOM_STATE]}
gs_tfidf_ridge = GridSearchCV(estimator=pipe_tfidf_ridge,
                              param_grid=param_grid,
                              scoring='neg_root_mean_squared_error',
                              cv=5,
                              n_jobs=-1)
gs_tfidf_ridge.fit(X_train, y_train)
gs_tfidf_ridge.best_params_, gs_tfidf_ridge.best_score_
```

```
CPU times: total: 13.4 s
Wall time: 1min 35s
```

Out[39]:
```
({'ridge__alpha': 0.6, 'ridge__random_state': 42}, -2.081779817746546)
```

In [40]:
```
tfidf_ridge_results = pd.DataFrame(gs_tfidf_ridge.cv_results_)[['rank_test_score',
                                                                'mean_test_score',
                                                                'std_test_score',
                                                                'param_ridge__alpha']].sort_values(by='rank_test_score')
tfidf_ridge_results.columns = ['rank',
                               'mean_rmse',
                               'std_rmse',
                               'alpha']
tfidf_ridge_results['mean_rmse'] = tfidf_ridge_results['mean_rmse'].abs()
tfidf_ridge_results.head(10)
```

Out[40]:

	rank	mean_rmse	std_rmse	alpha
5	1	2.081780	0.031032	0.6
4	2	2.082161	0.031140	0.5
6	3	2.082211	0.030967	0.7
7	4	2.083255	0.030901	0.8
3	5	2.083606	0.031121	0.4
8	6	2.084904	0.030817	0.9
2	7	2.086495	0.031198	0.3
9	8	2.086910	0.030754	1.0
1	9	2.091298	0.031293	0.2
0	10	2.099036	0.031307	0.1

Лучшие результаты RMSE на кросс-валидации показала модель Ridge с гиперпараметром регуляризации равным 0.6 с кодировкой текста с помощью TF-IDF, были использованы униграммы и биграммы. Также Ridge обучается быстрее других моделей использованных в исследовании. Она и была в качестве модели для прототипа веб-приложения и сохранена в .\best_model.pkl.

(Подробнее этапы подбора модели можно посмотреть в ноутбуке проекта .\rating_of_comments.ipynb.)

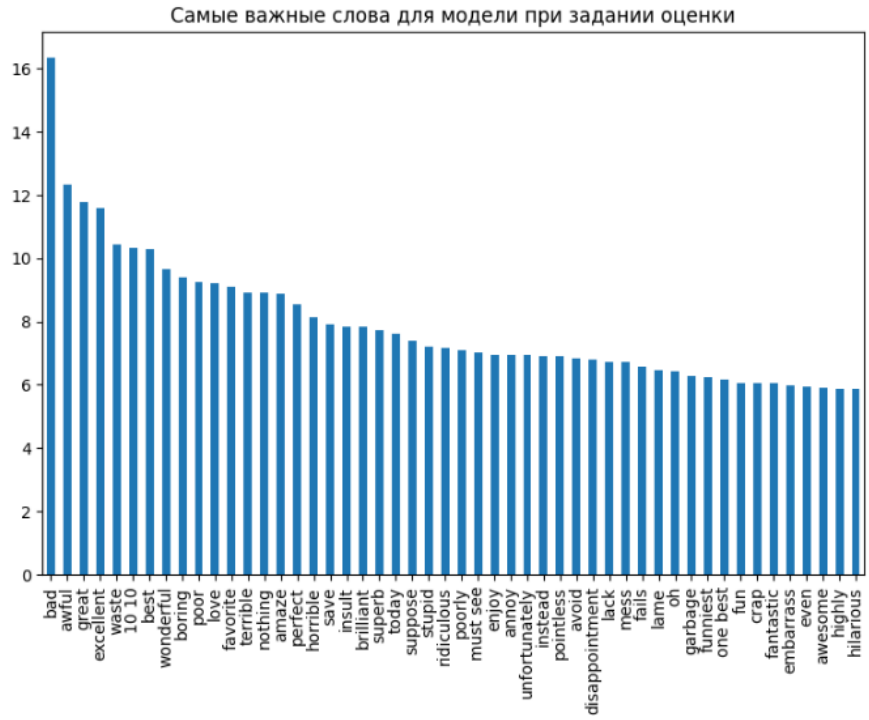
4. Оценка лучшей модели

Я протестировала лучшую модель на тестовой выборке с помощью метрики RMSE и сравнила результат с результатом лучшей модели на обучающей выборке и результатом на тестовой выборке “наивного” классификатора, обученного на обучающей выборке. Получены следующие результаты:

- RMSE лучшей модели на обучающей выборке: 0.86
- RMSE лучшей модели на тестовой выборке: 2.13
- RMSE "наивного" регрессора на тестовой выборке: 3.49

RMSE на тесте для лучшей модели меньше, чем для "наивной", обучение сработало корректно.

“Решающими” словами для модели стали:



Решающее значение для модели в определении категории комментария представляет наличие либо очень положительных слов, либо резко отрицательных. Примечательно, что в 50 важных для модели слов попали биграммы "10 10" и "must see".

Полученная модель предсказывает вещественное число, но стояла задача предсказать оценку комментария - целое число от 1 до 10 включительно. Для преобразования предсказаний модели в предсказание рейтинга была написана функция.

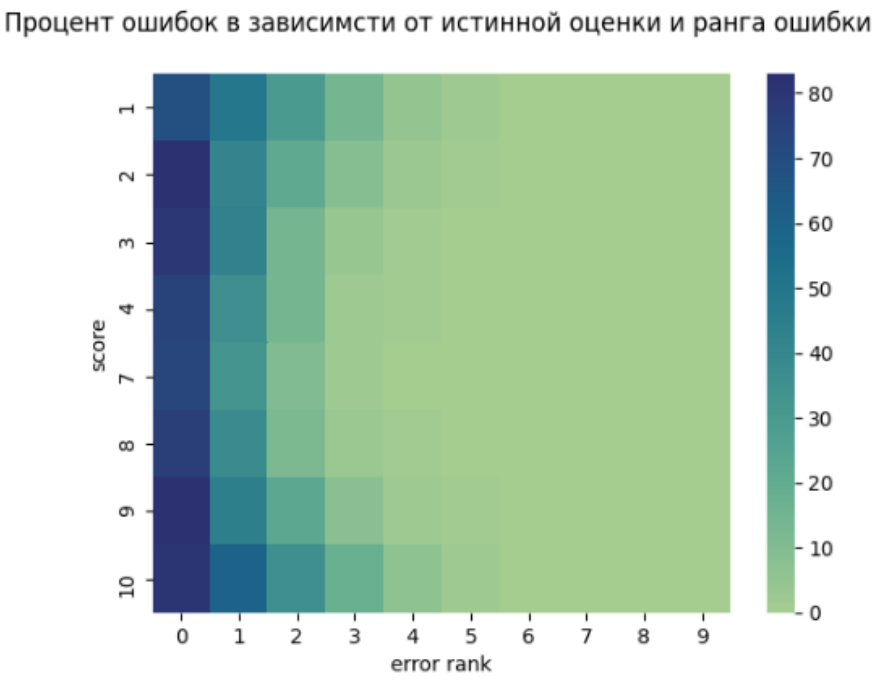
get_score_pred - функция, принимает вектор предсказаний модели и преобразует его в вектор оценок следующим образом:

1) Значения вектора округляются с помощью round.
2) Если полученное значение вектора меньше 1, то оно заменяется на 1.
3) Если полученное значение вектора больше 10, то оно заменяется на 10.

Я преобразовала с помощью get_score_pred предсказания (лучшей и “наивной” моделей) и сравнила результаты метрики MAE (эта метрика представляет собой абсолютное отклонение от истинного результата, легко интерпретируется, в данном случае измеряется в баллах). Получены следующие результаты:

- MAE определения класса лучшей модели на обучающей выборке: 0.65
- MAE определения класса лучшей модели на тестовой выборке: 1.59
- MAE определения класса "наивного" регрессора на тестовой выборке: 3.29

На тестовой выборке модель абсолютно ошибается в среднем на 1.6 балла, это хороший результат, так как у "наивной" модели средняя абсолютная ошибка в два раза больше.



По результатам лучшей модели на тестовой выборке был построен график процента ошибок в зависимости от рейтинга (score) и порядка ошибки (error rank). Ошибкой считается только абсолютная разница между истинным и предсказанным значением большая error rank баллов. Из графика видно, что модель достаточно плохо точно предсказывает оценку комментария, но даже процент расхождений больших 2 баллов очень мал, это значит, что модель довольно четко распознает тональность комментария. Так же можно заметить, что процент ошибок выше при крайних значениях оценки и ниже при серединных.

Определение статуса комментария — это задача классификации, для ее решения требуются модели-классификаторы. Но я приняла решение использовать лучшую модель (регрессионную), полученную при решении задачи оценивания комментария. Это обусловлено следующими пунктами:

1) Модель должна быть встроена в веб-сервис, она должна предсказывать достаточно быстро. Время предсказания одной моделью оценки и статуса будет меньше, чем, если будут предсказывать две модели.
2) Одна модель занимает меньше памяти на сервере, чем две.
3) Дополнительные исследования качества лучшей модели показали, что она довольно хорошо определяет тональность комментария.
4) Экономия времени подбора гиперпараметров и типа модели.

Я создала функцию get_status, которая принимает вектор оценок и выдает вектор статусов. Для оценок <= 5.5 - статус будет отрицательным (значение 0), для оценок > 5.5 - статус будет положительным (значение 1).

Так как дизбаланса классов нет, соотношение положительных и отрицательных статусов в выборках 1:1, то в качестве метрики можно выбрать Accuracy - отношение правильно предсказанных записей ко всем записям.

Я преобразовала с помощью get_status предсказания (лучшей и “наивной” моделей) и сравнила результаты метрики Accuracy:

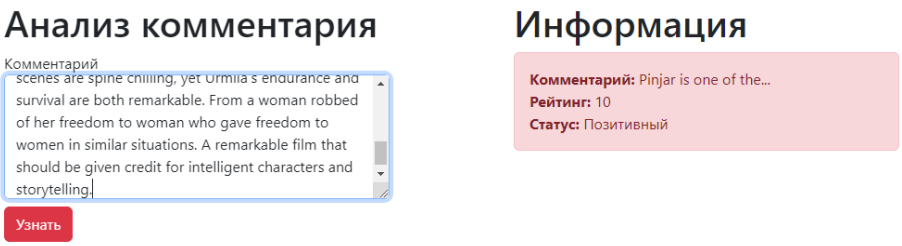
Accuracy лучшей модели на обучающей выборке: 0.99

Accuracy лучшей модели на тестовой выборке: 0.88

Accuracy "наивной" модели на тестовой выборке: 0.5

Лучшая модель дала хороший результат accuracy на тесте, он превышает результат "наивной" модели на 38%.

1. Прототип сервиса



Финальным этапом моей работы была разработка веб-сервиса на базе фреймворка Django для ввода отзыва о фильме с автоматическим присвоением рейтинга (от 1 до 10) и статуса комментария (положительный или отрицательный).

Веб-сервис был реализован и размещен на хостинге pythonanywhere, им можно воспользоваться по ссылке https://anastasianehodova.pythonanywhere.com/

Посмотреть все файлы использованные в работе можно в репозитории https://github.com/AnastasiaNehodova/movie