



**ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР** МГТУ им. Н. Э. Баумана

**Выпускная квалификационная работа  
по курсу "Data science"**

**Московского государственного технического университета  
имени Н.Э. Баумана**

**Докладчик: Полунина Мария Михайловна**

**Москва, 2023 г.**

# Анализ набора данных рейтинга книг Goodreads-books

# Просмотр данных в целом - из чего состоит датасет:  
df.head(5)

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publication_date	publisher
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	0439785960	9780439785969	eng	652	2095690	27591	9/16/2006	Scholastic Inc.
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	0439358078	9780439358071	eng	870	2153167	29221	9/1/2004	Scholastic Inc.
2	4	Harry Potter and the Chamber of Secrets (Harry...	J.K. Rowling	4.42	0439554896	9780439554893	eng	352	6333	244	11/1/2003	Scholastic
3	5	Harry Potter and the Prisoner of Azkaban (Harr...	J.K. Rowling/Mary GrandPré	4.56	043965548X	9780439655484	eng	435	2339585	36325	5/1/2004	Scholastic Inc.



# Структура данных

```
# Тип данных в датасете:  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 11123 entries, 0 to 11122  
Data columns (total 12 columns):  
#   Column                Non-Null Count  Dtype  
---  -  
0   bookID                11123 non-null  int64  
1   title                 11123 non-null  object  
2   authors               11123 non-null  object  
3   average_rating        11123 non-null  float64  
4   isbn                 11123 non-null  object  
5   isbn13               11123 non-null  int64  
6   language_code        11123 non-null  object  
7   num_pages            11123 non-null  int64  
8   ratings_count         11123 non-null  int64  
9   text_reviews_count    11123 non-null  int64  
10  publication_date      11123 non-null  object  
11  publisher             11123 non-null  object  
dtypes: float64(1), int64(5), object(6)
```

'bookID' - Номер книги по порядку в данном датасете;

'title' - Название книги;

'authors' - Имя/Фамилия автора книги;

'average\_rating' - Средний рейтинг;

'isbn' - Международный стандартный книжный номер (10 цифр);

'isbn13' - Международный стандартный книжный номер (13 цифр);

'language\_code' - Язык издания;

'num\_pages' - Количество страниц;

'ratings\_count' - Количество учтенных рейтингов;

'text\_reviews\_count' - Количество текстовых отзывов;

'publication\_date' - Дата выхода книги в печать;

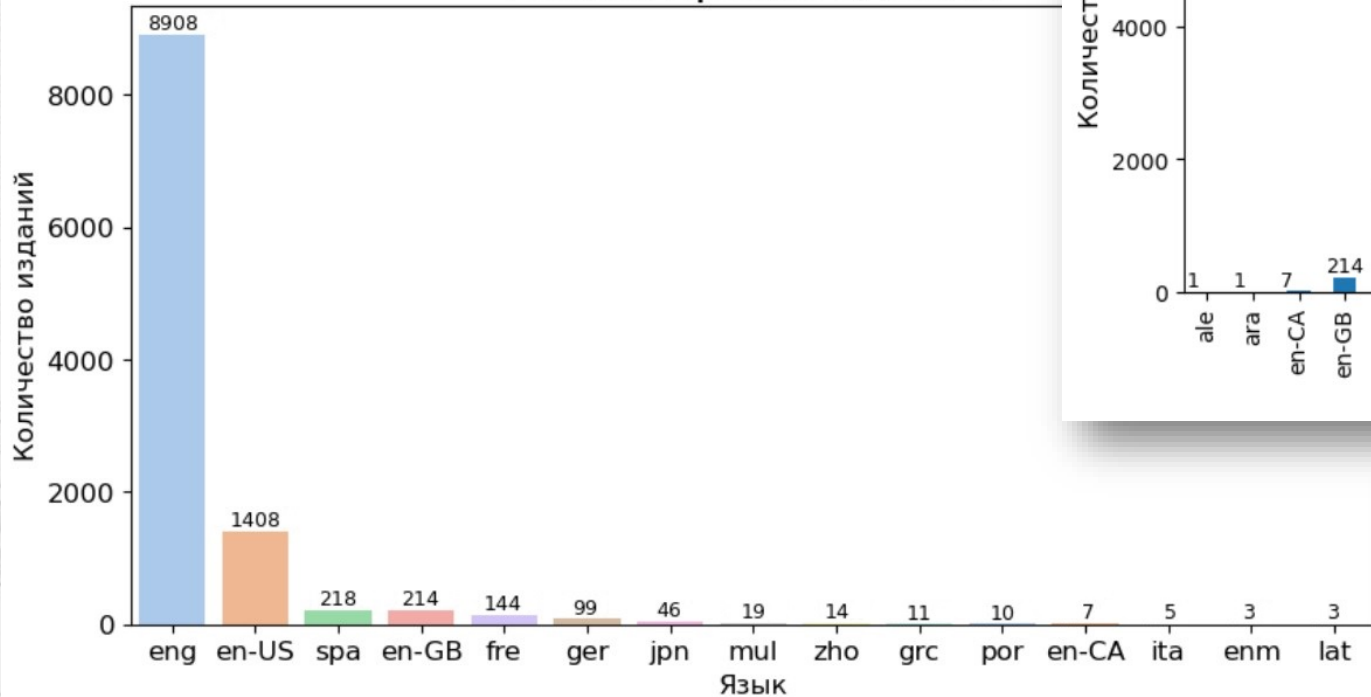
'publisher' - Издательство, выпустившее книгу.

# Показатели столбца «Язык произведения»

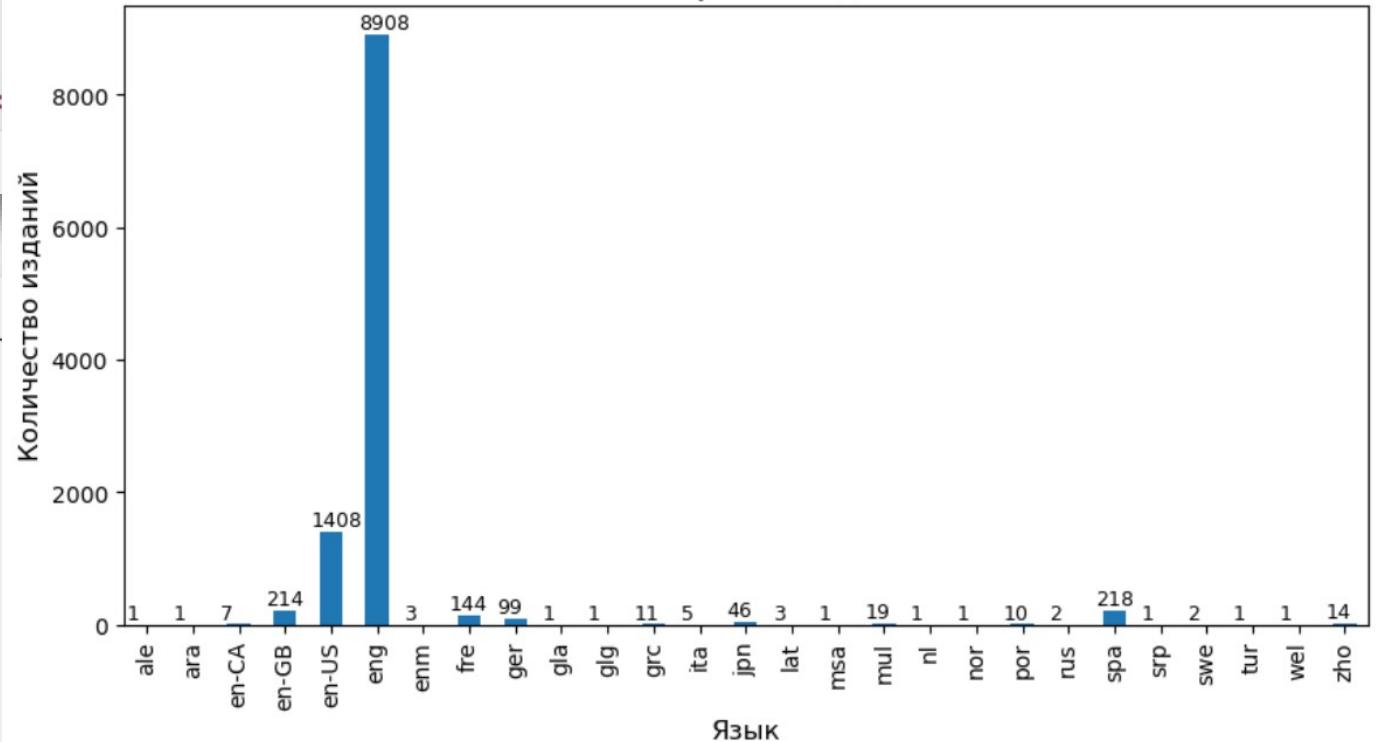
```
# Узнаем, с каким количеством авторов и языков будем иметь дело:  
num_author = len(pd.unique(df['authors']))  
num_lang = len(pd.unique(df['language_code']))  
print('Количество авторов: ', num_author, ', ', 'Языки произведений: ', num_lang)
```

Количество авторов: 6639, Языки произведений: 27.

## Языки произведений

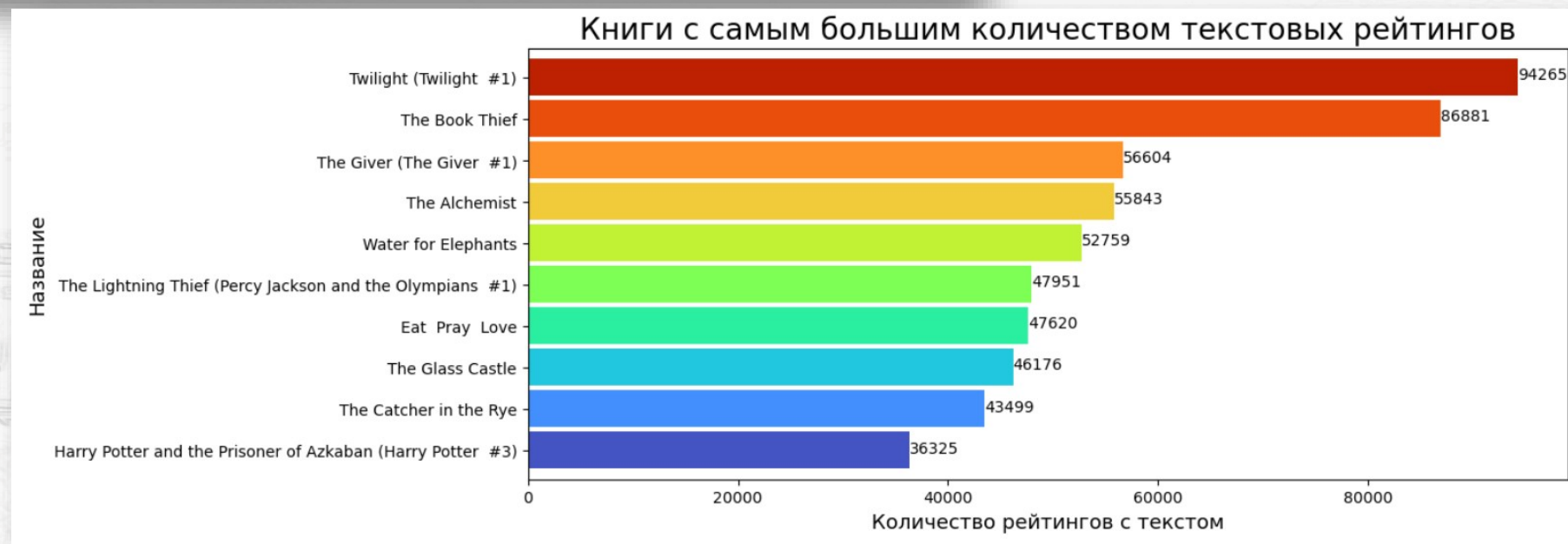
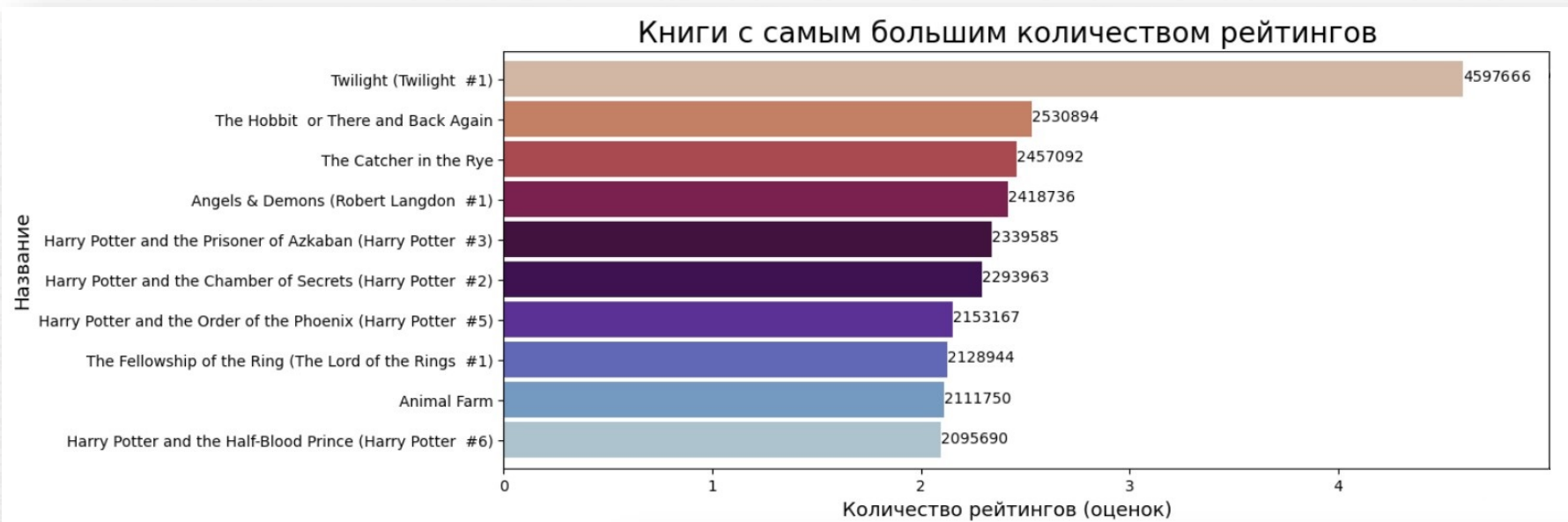


## Языки произведений



Две книги на русском:  
«Шинель» Н.В.Гоголя и  
«Мастер и Маргарита» М.Ю.Булгакова

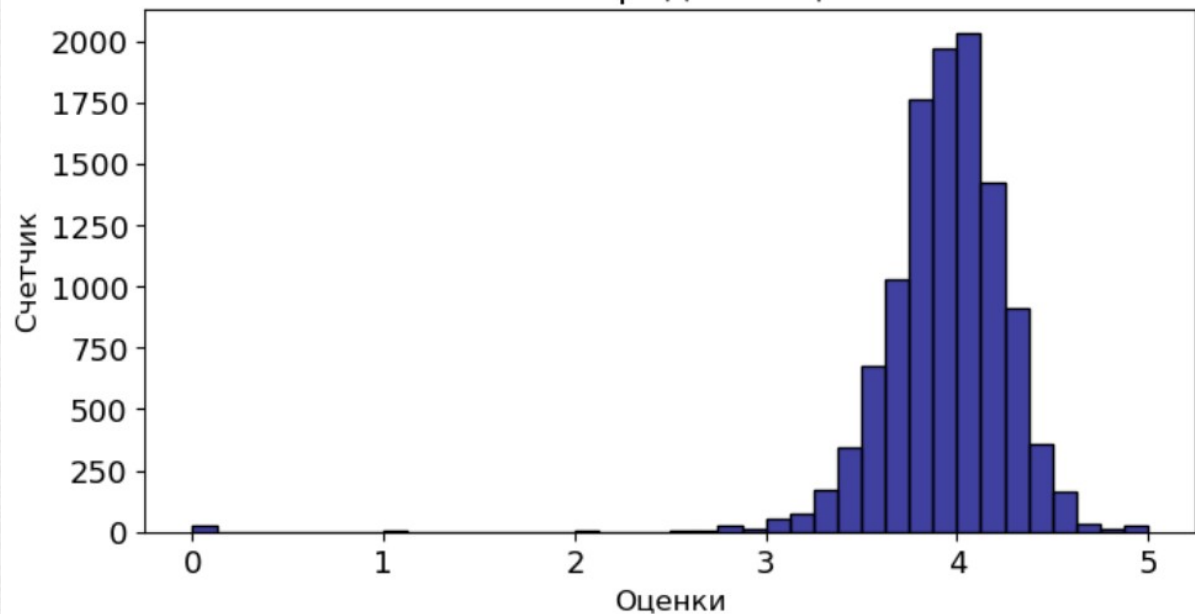
# Количество оценок и рейтинги с текстом





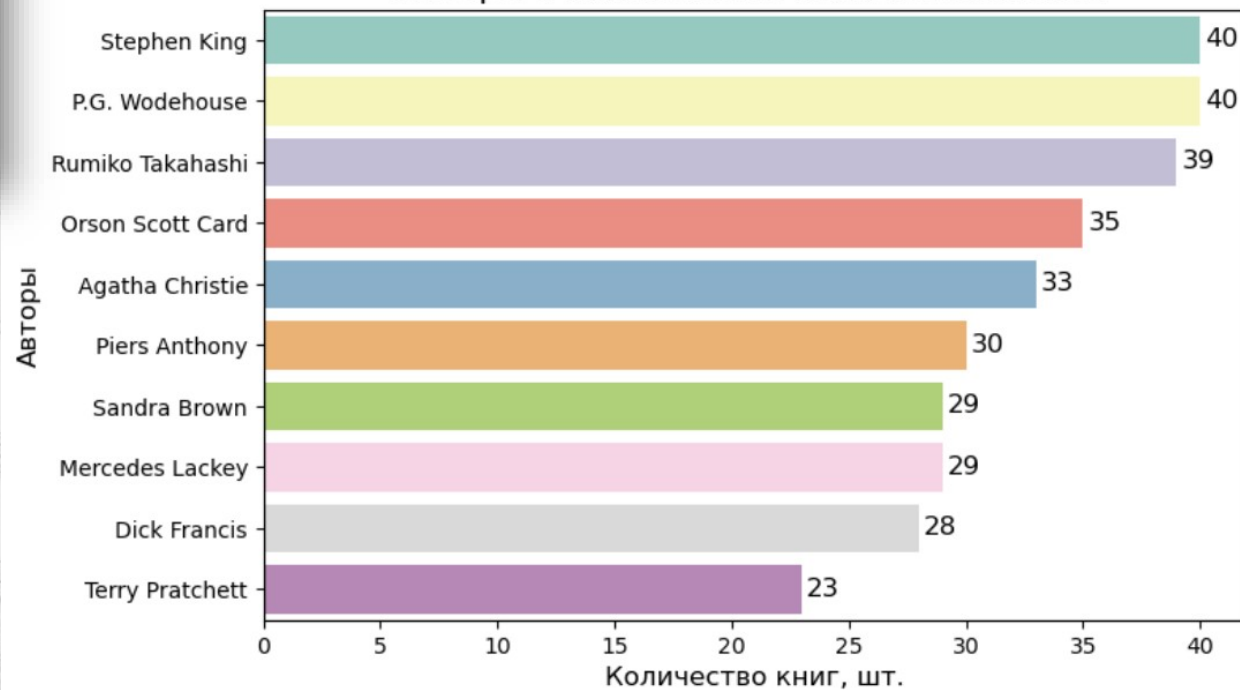
# Общий рейтинг и топ-10 авторов по количеству книг

Рейтинг средних оценок

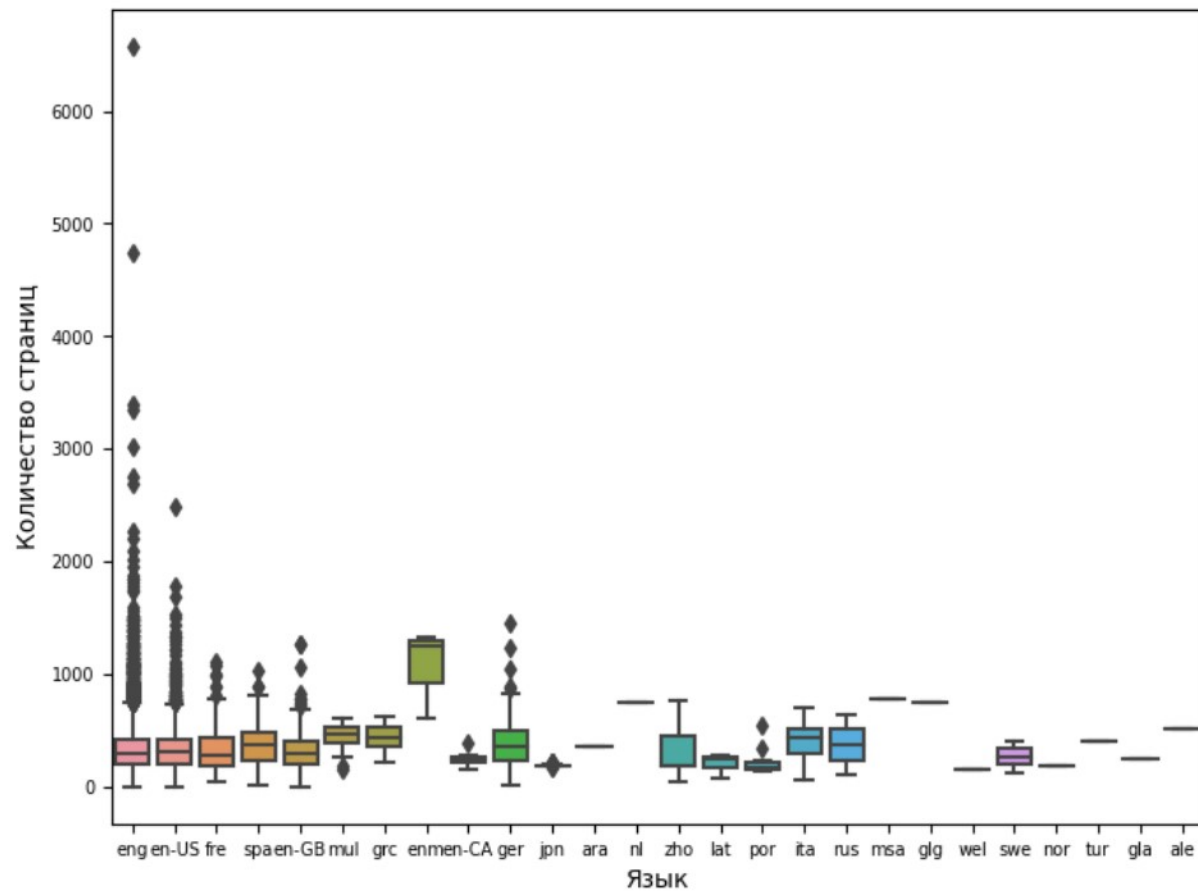
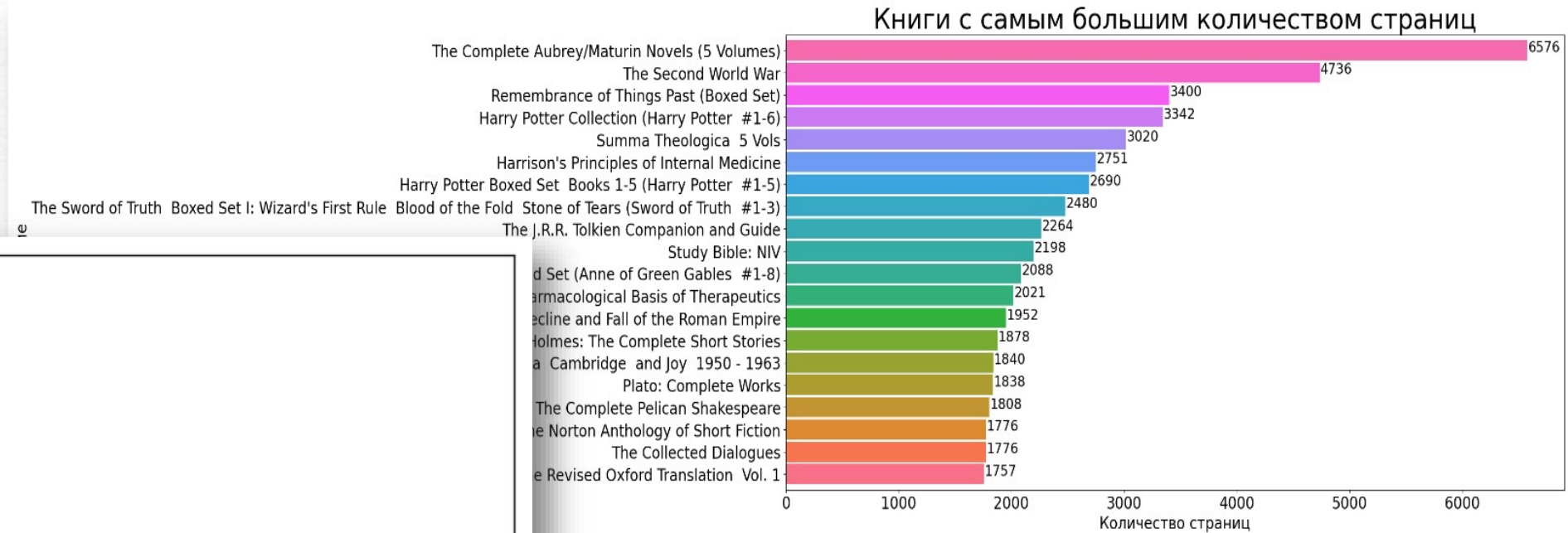


Основная масса оценок находится  
в диапазоне от 3.5 до 4.3 баллов

Авторы с наибольшим количеством книг

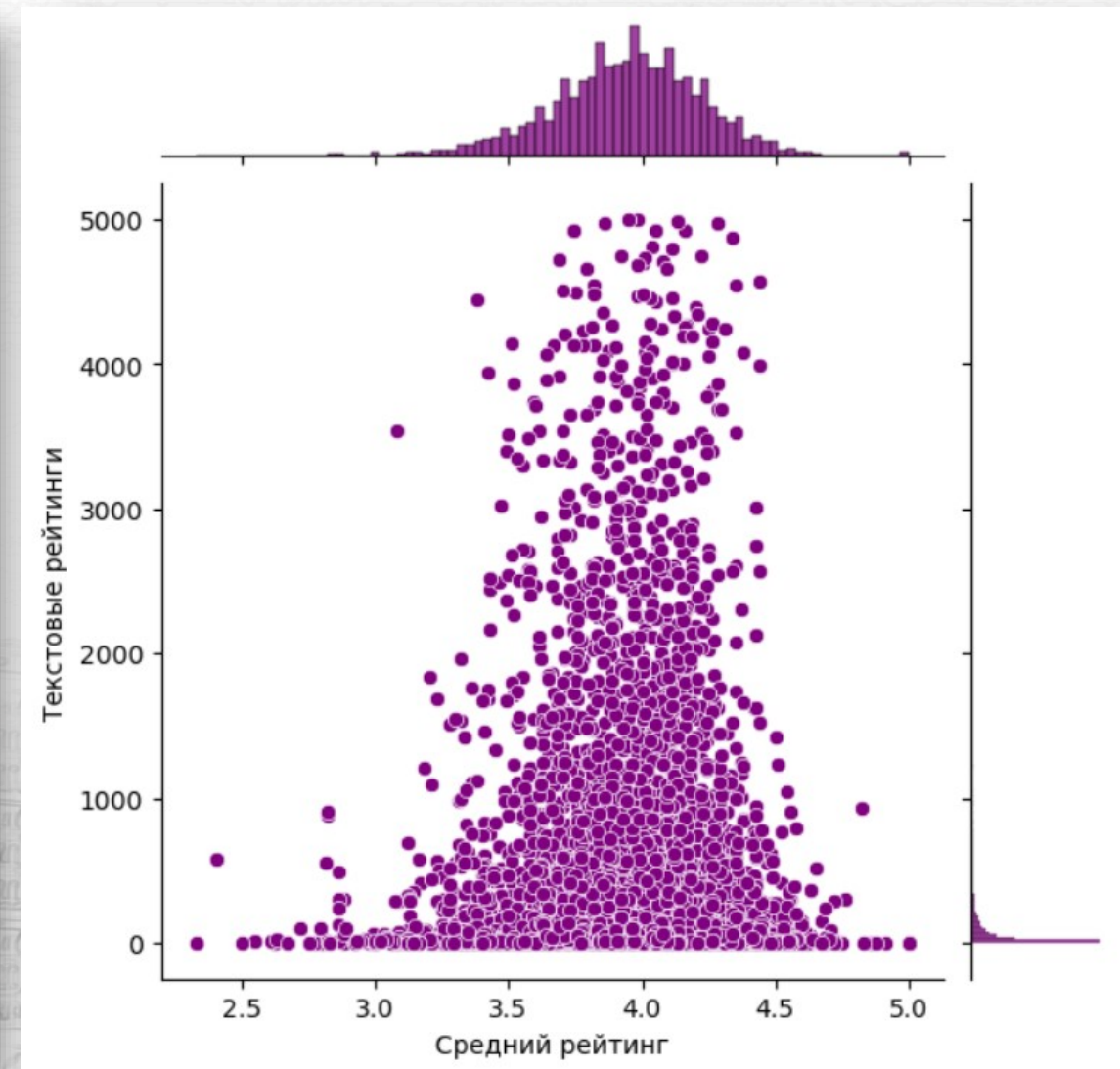
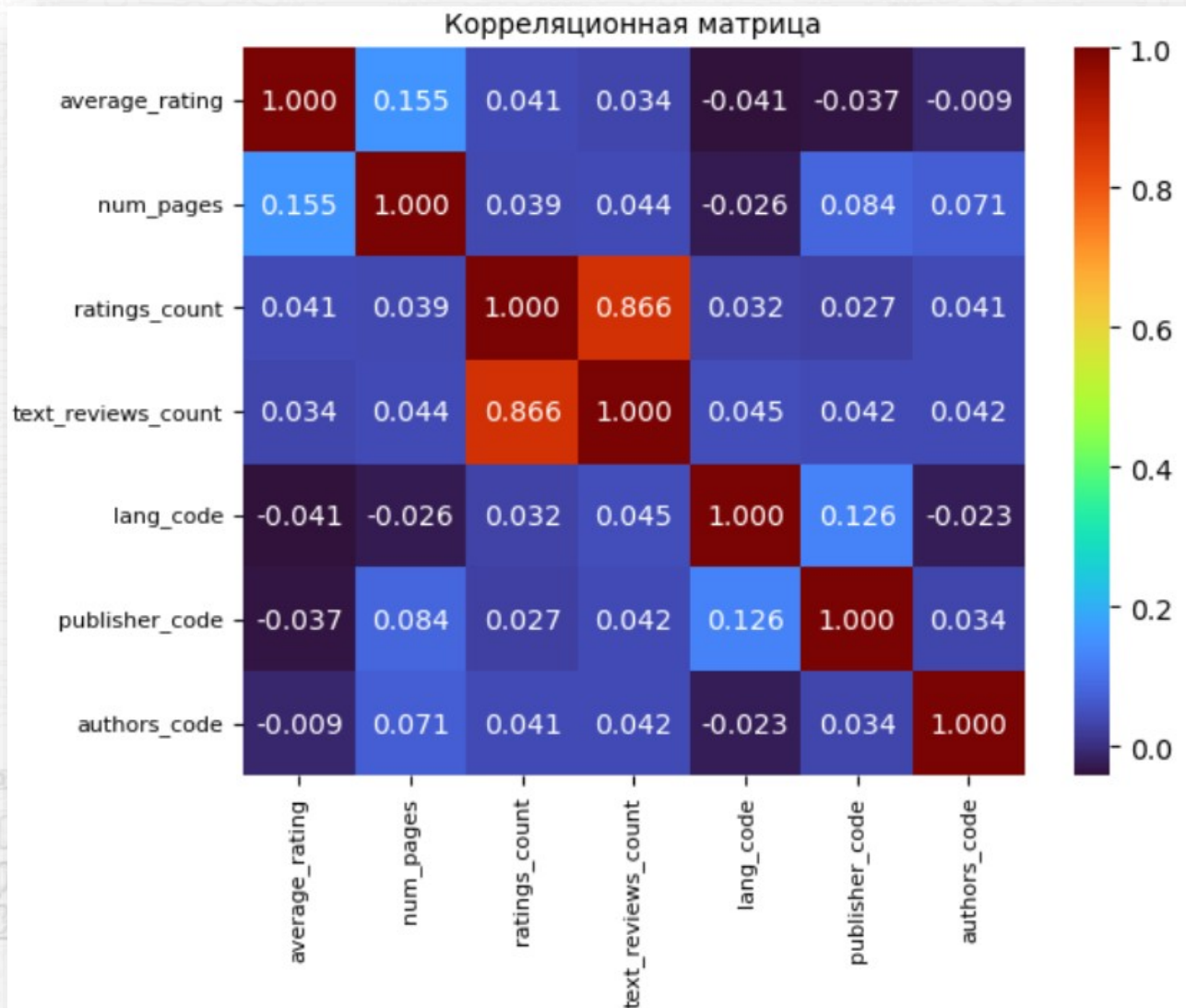


# Количество страниц: выбросы и общие данные



Для анализа выбросов по количеству страниц использовался «Ящик с усами» для всех языков

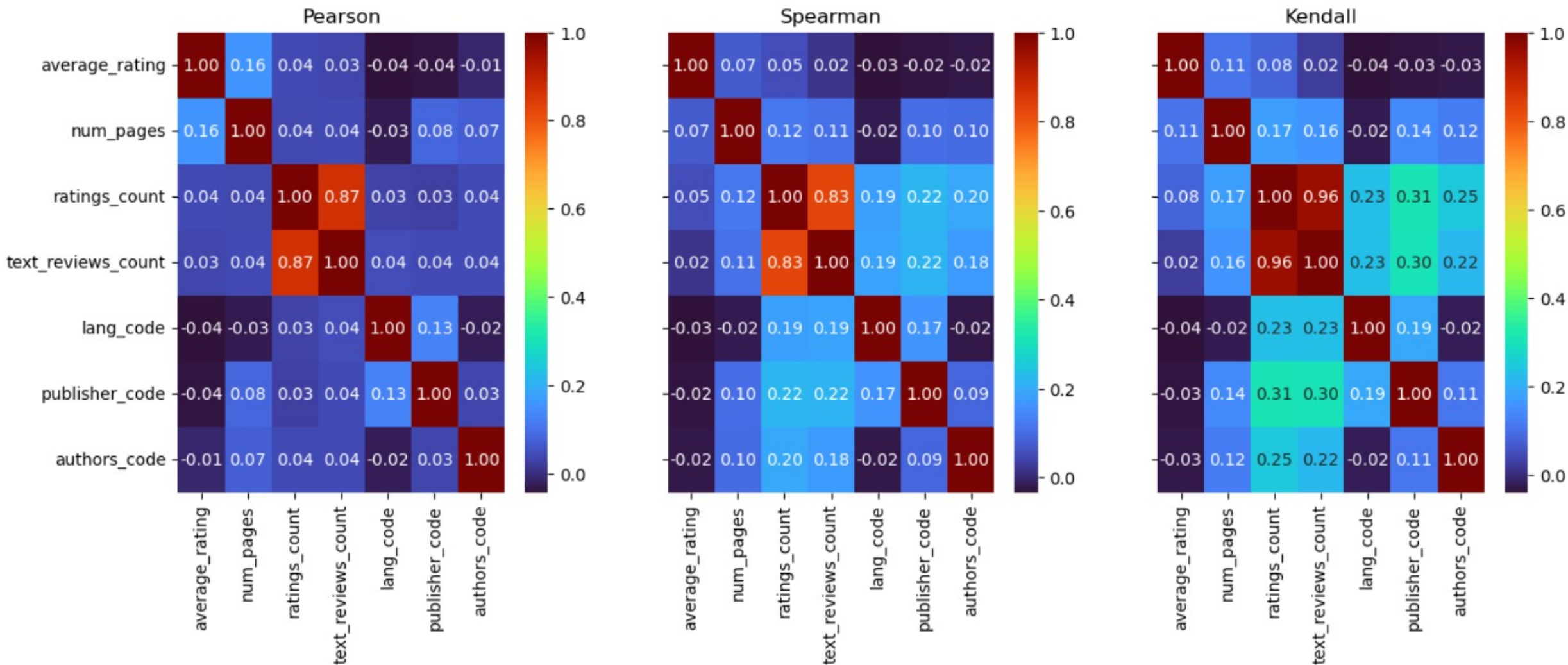
# Корреляция показателей





# Корреляция показателей

Корреляционные матрицы, построенные различными методами



# Линейная регрессия (LinearRegression)

```
# Линейная регрессия
model = LinearRegression()
parameters = {'fit_intercept': [True, False], 'normalize': [True, False]}

grad_Linear = GridSearchCV(model, parameters, refit=True)
grad_Linear.fit(X_train, y_train)

print('Лучший результат: ', grad_Linear.best_score_*100, '\nлучшие параметры: ', grad_Linear.best_params_)
```

Лучший результат: 3.062007672686584  
Лучшие параметры: {'fit\_intercept': True, 'normalize': False}

```
# Предсказание на модели линейной регрессии
pred_lr = grad_Linear.predict(X_test)

# Результаты
print('MAE: ' + str(np.sqrt(mean_absolute_error(y_test, pred_lr))))
print('R2: ', r2_score(y_test, pred_lr))
```

MAE: 0.4709750932696985  
R2: 0.02139284140943143

```
# Сравнение актуальных оценок и предсказанных моделью
pred = pd.DataFrame({'Обучающие данные': y_test.tolist(),
                    'Предсказание': pred_lr.tolist()}).head(5)
pred.head()
```

	Обучающие данные	Предсказание
0	4.30	3.893007
1	3.80	3.970500
2	3.49	3.906495
3	3.78	3.908899
4	4.04	3.933579



# AdaBoostRegressor (DecisionTreeRegressor)

```
# AdaBoost регреcсор
model = AdaBoostRegressor(DecisionTreeRegressor(max_depth=4))
parameters = {'learning_rate': [0.001, 0.01, 0.02, 0.1, 0.2, 1.0], 'n_estimators': [10, 50, 100]}

grad_Ada = GridSearchCV(model, parameters, refit=True)
grad_Ada.fit(X_train, y_train)

print('Лучший результат: ', grad_Ada.best_score_*100, '\nлучшие параметры: ', grad_Ada.best_params_)

Лучший результат: 5.983036547551877
Лучшие параметры: {'learning_rate': 0.2, 'n_estimators': 10}
```

```
# Предсказание на модели AdaBoost
pred_adaboost = grad_Ada.predict(X_test)

# Результаты
print('MAE: ' +str(np.sqrt(mean_absolute_error(y_test, pred_adaboost))))
print('R2: ', r2_score(y_test, pred_adaboost))

MAE: 0.467228435449686
R2: 0.0575481447639038
```

```
# Сравнение актуальных оценок и предсказанных моделью
pred = pd.DataFrame({'Обучающие данные': y_test.tolist(),
                     'Предсказание': pred_adaboost.tolist()}).head(5)
pred.head()
```

	Обучающие данные	Предсказание
0	4.30	3.895287
1	3.80	3.895306
2	3.49	3.883739
3	3.78	3.900847
4	4.04	3.926560



# Случайный лес (Random Forest Regression)

```
# Случайный лес
model = RandomForestRegressor()
parameters = {'n_estimators': [50, 100, 150], 'max_depth': [3, 5, 7, 10, 12],
              'min_samples_split': [5, 10, 15], 'min_samples_leaf': [5, 10, 15]}

grad_rf = GridSearchCV(model, parameters, refit=True, cv=10)
grad_rf.fit(X_train, y_train)

print('Лучший результат: ', grad_rf.best_score_*100, '\nлучшие параметры: ', grad_rf.best_params_)

Лучший результат: 10.625476474667291
Лучшие параметры: {'max_depth': 12, 'min_samples_leaf': 15, 'min_samples_split': 5, 'n_estimators': 100}
```

```
# Предсказание на модели Случайного леса
pred_rf = grad_rf.predict(X_test)

# Результаты
print('MAE: ' + str(np.sqrt(mean_absolute_error(y_test, pred_rf))))
print('R2: ', r2_score(y_test, pred_rf))

MAE: 0.459771189368407
R2: 0.11181124070859261
```

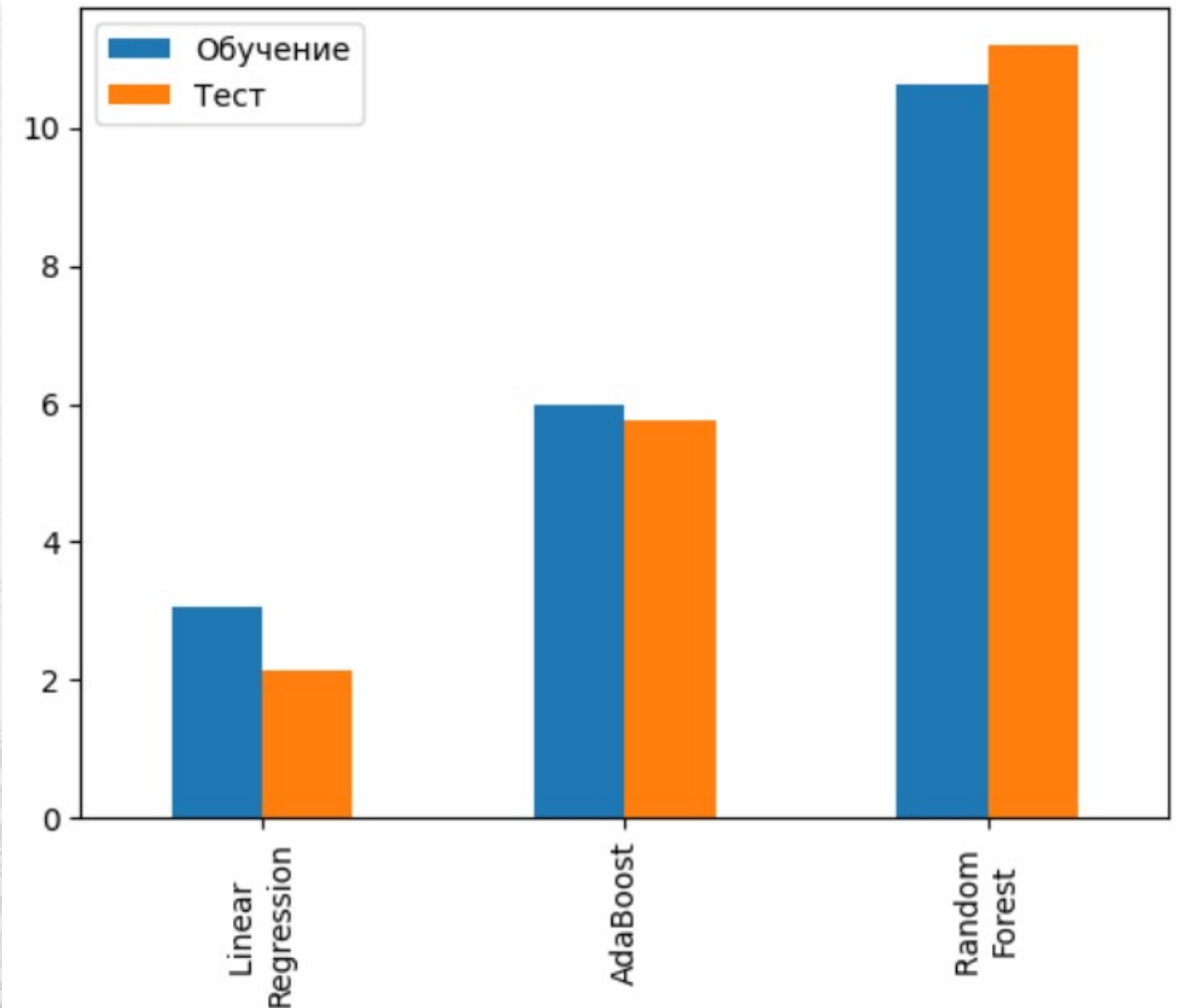
```
# Сравнение актуальных оценок и предсказанных моделью
pred = pd.DataFrame({'Обучающие данные': y_test.tolist(),
                    'Предсказание': pred_rf.tolist()}).head(5)
pred.head()
```

	Обучающие данные	Предсказание
0	4.30	3.843004
1	3.80	3.969051
2	3.49	3.774735
3	3.78	3.920451
4	4.04	3.909089

# Сравнение работы моделей

	Модель	Обучение	Тест
<b>Linear Regression</b>	Linear Regression	3.062008	2.139284
<b>AdaBoost</b>	AdaBoost	5.983037	5.754814
<b>Random Forest</b>	Random Forest	10.625476	11.181124

По итогам работы всех моделей  
алгоритм Случайного Леса  
(RandomForestRegressor) показал  
наилучшие результаты





# Полносвязная нейронная сеть

```
# Модель полносвязной нейронной сети
model = Sequential()

model.add(Dense(128, input_dim = x_train.shape[1], activation='relu'))
model.add(Dense(256, activation='softmax'))
model.add(Dense(256, activation='softmax'))
model.add(Dense(256, activation='softmax'))
model.add(Dense(10, activation='linear'))

model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mean_absolute_error'])
model.summary()
```

```
model.summary()
```

Model: "sequential\_1"

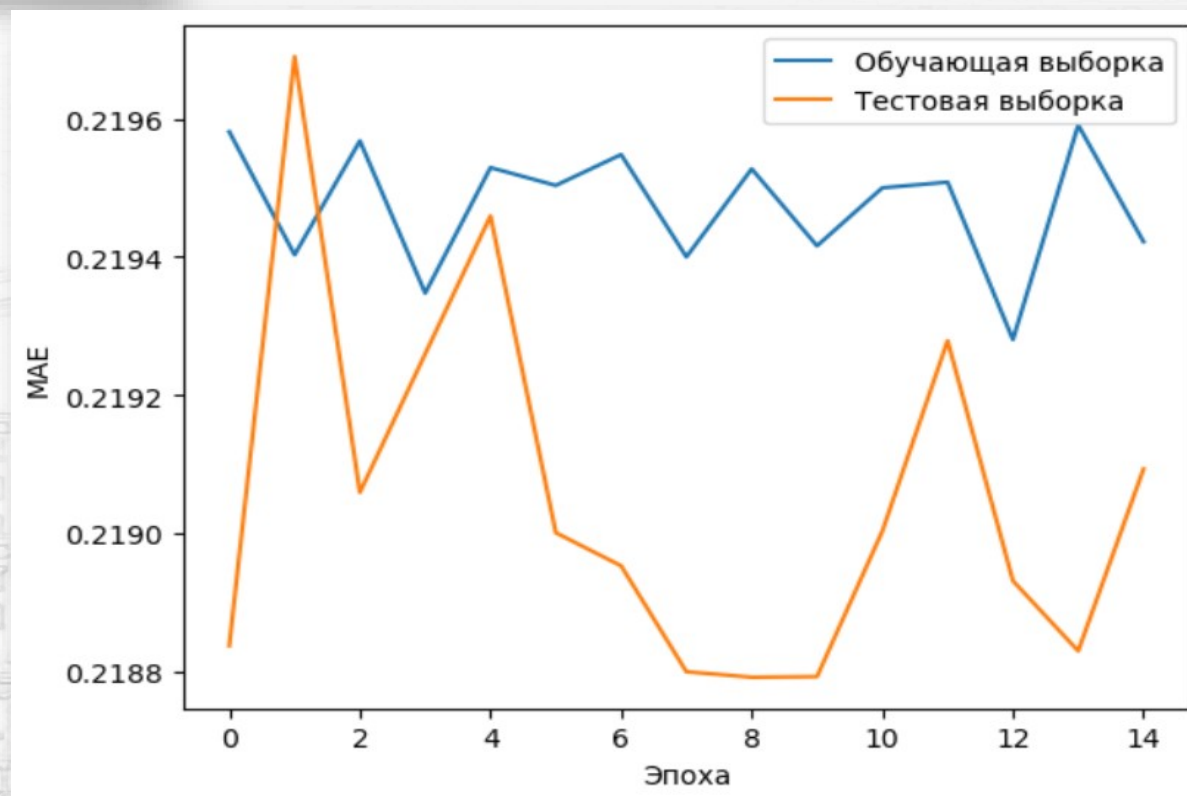
Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 128)	896
dense_6 (Dense)	(None, 256)	33024
dense_7 (Dense)	(None, 256)	65792
dense_8 (Dense)	(None, 256)	65792
dense_9 (Dense)	(None, 10)	2570
Total params: 168,074		
Trainable params: 168,074		
Non-trainable params: 0		

Эксперимент с нейронной сетью:

MAE =  $\sim 0.2192$

Лучший результат в регрессионных ML-моделях:

MAE =  $\sim 0.4597$





# Рекомендательная система: подготовка

*# Разбивка книг на 5 рейтинговых классов*

```
df2.loc[ (df2['average_rating'] >= 0) & (df2['average_rating'] <= 1), 'rating_between'] = 'between 0 and 1'
df2.loc[ (df2['average_rating'] > 1) & (df2['average_rating'] <= 2), 'rating_between'] = 'between 1 and 2'
df2.loc[ (df2['average_rating'] > 2) & (df2['average_rating'] <= 3), 'rating_between'] = 'between 2 and 3'
df2.loc[ (df2['average_rating'] > 3) & (df2['average_rating'] <= 4), 'rating_between'] = 'between 3 and 4'
df2.loc[ (df2['average_rating'] > 4) & (df2['average_rating'] <= 5), 'rating_between'] = 'between 4 and 5'
```

1. Разделение книг на пять классов рейтингов;

*# Новые классы рейтинга в отдельной таблице*

```
rating_df = pd.get_dummies(df2['rating_between'])
rating_df.head()
```

	between 0 and 1	between 1 and 2	between 2 and 3	between 3 and 4	between 4 and 5
0	0	0	0	0	1
1	0	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	1

2. Кодирование новых классов рейтингов при помощи get\_dummies;

*# Кодирование колонки с языком производства*

```
language_df = pd.get_dummies(df2['language_code'])
language_df.head()
```

	ale	ara	en-CA	en-GB	en-US	eng	enm	fre	ger	gla	...	nl	nor	por	rus	spa	srp	swe	tur	wel	zho
0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 27 columns

3. Кодирование языковых классов при помощи get\_dummies;

# Рекомендательная система: подготовка


```
# Объединяем в одну таблицу признаки для рекомендательной системы
features = pd.concat([rating_df, language_df, df2['average_rating'], df2['ratings_count']], axis=1)
features.head()
```

	between 0 and 1	between 1 and 2	between 2 and 3	between 3 and 4	between 4 and 5	ale	ara	en- CA	en- GB	en- US	...	por	rus	spa	srp	swe	tur	wel	zho	average_rating	ratings_count
0	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	4.57	2095690
1	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	4.49	2153167
2	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	4.42	6333
3	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	4.56	2339585
4	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	4.78	41428

5 rows × 34 columns

```
# MinMax-масштабирование
min_max_scaler = MinMaxScaler()
features = min_max_scaler.fit_transform(features)
print(features)
```

```
[[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
 9.14000000e-01 4.55816060e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
 8.98000000e-01 4.68317403e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
 8.84000000e-01 1.37743803e-03]
 ...
```



4. Объединение всех закодированных значений, а также среднего рейтинга и количества учтенных рейтингов ;



5. Масштабирование данных при помощи MinMaxScaler.



# Рекомендательная система: моделирование и тестирование

```
# Создание модели KNN - ближайших соседей
model = neighbors.NearestNeighbors(n_neighbors=7,
                                   algorithm='kd_tree')

model.fit(features)
dist, idlist = model.kneighbors(features)
```

```
# Функция работы рекомендателя:
def BookRecommender(book_name):
    book_list_name = []
    book_id = df2[df2['title'] == book_name].index
    book_id = book_id[0]
    for newid in idlist[book_id]:
        book_list_name.append(df2.loc[newid].title)
    return book_list_name
```

Модель рекомендательной системы создана при помощи алгоритма KNN – метод ближайших соседей.

Поиск книги производится при помощи функции, используя название произведения.

```
# Проверяем работу рекомендательной системы
BookNames = BookRecommender('The Green Mile')
BookNames
```

```
['The Green Mile',
 'Death Note Vol. 1: Boredom (Death Note #1)',
 'Voyager (Outlander #3)',
 'The Complete Stories and Poems',
 'Lover Awakened (Black Dagger Brotherhood #3)',
 'The Ultimate Hitchhiker's Guide to the Galaxy (Hitchhiker's Guide to the Galaxy #1-5)',
 'Maus I: A Survivor's Tale: My Father Bleeds History (Maus #1)']
```



# Рекомендательная система: мини-программа

```
# Мини-программа для рекомендации
ask = df.title[int(input('Введите порядковый номер книги от 0 до 11123: '))]
print(ask)
rec_books = BookRecommender(input('Копируйте название книги: '))
print('Похожие книги: ', rec_books)
```

Введите порядковый номер книги от 0 до 11123:

```
# Мини-программа для рекомендации
ask = df.title[int(input('Введите порядковый номер книги от 0 до 11123: '))]
print(ask)
rec_books = BookRecommender(input('Копируйте название книги: '))
print('Похожие книги: ', rec_books)
```



Введите порядковый номер книги от 0 до 11123: 6543

Oh The Places You'll Go!

Копируйте название книги: Oh The Places You'll Go!

Похожие книги: ['Oh The Places You'll Go!', 'The Acme Novelty Library #17', 'Let Justice Roll Down', "The Ballet Companion: A Dancer's Guide to the Technique Traditions and Joys of Ballet", 'Autobiographies: Narrative of the Life of Frederick Douglass / My Bondage and My Freedom / Life and Times of Frederick Douglass', "Playing Shakespeare: An Actor's Guide", 'Collected Stories']



**ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР** МГТУ им. Н. Э. Баумана

# Спасибо за внимание!

