МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**по курсу**

«Инженер данных (Data engineer Pro)»

**Тема:** **«Классификация госконтрактов по объектам закупки»**

Слушатель Черненко Наталья Алексеевна(ФИО)

Москва, 2024

[Введение 4](#_Toc170432887)

[1. Аналитическая часть 6](#_Toc170432888)

[1.1. Постановка задачи 6](#_Toc170432889)

[1.2. Описание используемых методов 7](#_Toc170432890)

[1.3. Разведочный анализ данных. 8](#_Toc170432891)

[2. Практическая часть 12](#_Toc170432892)

[2.1. Предобработка данных 12](#_Toc170432893)

[2.2. Разработка и обучение модели 21](#_Toc170432894)

[2.3. Тестирование рекомендательной системы 22](#_Toc170432895)

[2.4. Разработка приложения 22](#_Toc170432896)

[2.5. Создание удаленного репозитория и загрузка результатов работы на него 22](#_Toc170432897)

[Заключение 23](#_Toc170432898)

Введение

В соответствии с Федеральным законом «О контрактной системе» (44-ФЗ) государственный контракт (госконтракт) - это соглашение между поставщиком и органами власти федерального, регионального или муниципального уровней. Карточки госконтрактов хранятся в специальном реестре. Карточка госконтракта содержит информацию о контракте: описание, сроки, исполнителей и т.д. В том числе хранится общероссийский классификатор продукции по видам экономической деятельности (ОКПД-2). Зачастую ОКПД-2 заявляется ошибочный. В этом заключается проблема – выделить ошибочно обозначенные госконтракты и переназначить их ОКПД-2.

Создание классификатора госконтрактов по ОКПД-2 в соответствии с объектом закупки, позволит эффективно перераспределять контракты по соответствующим им группам.

1. Аналитическая часть

1.1. Постановка задачи

На входе имеются данные карточек госконтрактов с ftp.zakupki.gov.ru. Необходимо на основе данных с ftp.zakupki.gov.ru научиться определять группу, к которой относится контракт с кодом ОКПД-2 41, 42, 43, 71.1.

Группы могут быть следующими:

1. Строительно-монтажные работы (СМР)
2. Проектно-изыскательские работы (ПИР)
3. Строительный надзор
4. Подключение коммуникаций
5. Прочее.

По ОКПД-2 контракты в общем случае должны разделяться так:

1. Строительно-монтажные работы (СМР) - 41, 42, 43(кроме нижеперечисленных)
2. Проектно-изыскательские работы (ПИР) - 41.1, 71.1
3. Подключение коммуникаций - 43.22
4. Строительный надзор – четкой группы нет.

Далеко не всегда контракты указываются с нужным кодом, поэтому есть проблема как такие контракты "отловить" и определить в нужную группу.

Поэтому задача предполагает классификацию контрактов на основе объекта закупки, который сформулирован естественным языком. Также предполагаем, что могут иметь значение цена контракта и его длительность.

На основе этого на входе данные о контрактах. На выходе необходимо получить группу для каждого контракта.

Иногда контракт может относиться одновременно в несколько групп.

В приложении к техническому заданию приведена таблица с указанием правильных групп по кодам, которые нам на не предоставили. В самом файле данные размечены только по кодам ОКПД2, которые могут быть неверными. Разметка данных в таком случае будет отдельной задачей.

Попробуем создать рекомендательную систему на основе объекта закупки с отнесением объекта к коду ОКПД2, которую потом можно будет использвать после разметки исходного файла по группам.

Для анализа нам предоставлен csv файл размером около 26Гб с количеством строк 29320202 по 27 значений в каждой, содержащий записи по всем кодам ОКПД2. Записи содержат много пропусков и дублей. Необходимо будет проанализировать взаимосвязь признаков и объектов, и принять решение об отборе признаков.

Предметом контрактов (объектом закупки) являются поставки товаров, выполнение работ, оказание услуг соответственно для федеральных нужд, нужд субъекта Российской Федерации, муниципальных нужд. Соответственно, в описаниях часто встречаются слова "поставка", "приобретение", "закупка", или стандартные словосочетания "выполнение работ", "оказание услуг", дополненные типом товара, указанием типа работ или услуг, также с использованием типичных для отраслей терминов. Также может быть указан объект работ в виде наименования организации или адреса. Сами описания представляют скорее определения, чем полноценные предложения.

1.2. Описание используемых методов

Для решения поставленной задачи будем использовать content-based подход для построения рекомендаций: измерение похожести между объектами на основе их содержания. Идея будет состоять в сравнении введенного текстового описания объекта закупки с имеющимися в базе, и предложении наиболее схожих с объектом закупок вариантов.

Выдача результатов в таких системах основана на ранжировании меры сходства объектов, в нашем случае описаний. Для ее определения можно использовать косинусную меру.

Косинусная мера - это мера сходства между двумя векторами в многомерном пространстве. Она вычисляет косинус угла между двумя векторами, показывая как близко они расположены относительно друг друга.

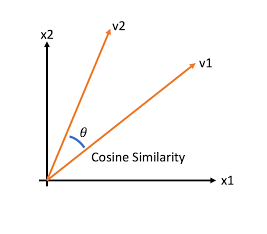


Рисунок 1 – Косинусная мера

Значения косинусной меры сходства находятся в диапазоне от -1 до 1, где:

* 1 указывает на то, что направление векторов полностью совпадает
* 0 указывает на то, что векторы перпендикулярны друг другу
* -1 указывает на то, что направления векторов противоположны

То есть чем ближе к единице значение меры, тем более схожи тексты.

Эта мера сходства особенно полезна в работе с векторами, так как она фокусируется на соотношении направления векторов, а не их длине.

1.3. Разведочный анализ данных.

В целом процесс обработки любых данных начинается с чтения файла. Так как предложенный для анализа файл имеет большой объем, на этом этапе мы с помощью небольших скриптов, написанных на языке Python, определим диалект, найдем размерность данных, и сразу отберем из него строки с нужными для анализа кодами ОКПД2.

Работа скриптов предполагает построчное считывание и запись файлов, что занимает большое количество времени. Использование базы данных может ускорить этот процесс.

Работать с дублями и пропусками в данных, отбирать признаки для анализа будем уже в Jupyter NoteBook с использованием библиотеки Pandas.

Для визуализации данных используем Библиотеки Matplotlib и Seaborn.

Предобработка самих текстовых данных предполагает следующие шаги:

1. приведение к одному регистру
2. замена “нетекстовой” информации
3. удаление пунктуации
4. токенизация
5. удаление стоп-слов, редких и частых слов
6. стемминг
7. лемматизация
8. векторизация

Первые три пункта будут выполнены с использованием регулярных выражений и методов из модуля re.

Также в текстах могут присутствовать лишние слова, такие как предлоги, местоимения, модальные глаголы, вводные слова. Так как наш текст не является художественным или разговорным, в нем присутствуют в основном глаголы, зато много сокращений, например, в адресах. На этом этапе просто удалим все слова, длиной больше трех символов.

Токенизация обобщенно представляет собой разделение входного текста на элементы (предложения, слова, разделители). Так как описания объектов закупки у нас уже представляют отдельные предложения, токенизировать будем сразу по словам. После первых трех шагов токенизацию можно осуществить методом split c пробелом в качестве разделителя. Или можно воспользоваться методом wordtokenizer из библиотеки nltk.

Стемминг – это приближенный процесс, в ходе которого от слов отбрасываются окончания, приставки, суффиксы в расчете на то, что в большинстве случаев это себя оправдает.

Лемматизация - это более точный процесс с использованием лексикона и морфологического анализа слов, в результате которого возвращается словарная форма слова(лемма)

Так как стемминг быстрее обрабатывают данные, и у нас достаточно простой текст, будем использовать стеммер SnowballStemmer из библиотеки nltk.

Обучения модели предполагает, что все данные должны быть представлены в числовом виде. Для решения этой проблемы существуют различные техники перевода текстов в векторы. TF-IDF является основополагающим методом в задачах классификации документов, направленных на распределение документов по разным группам.

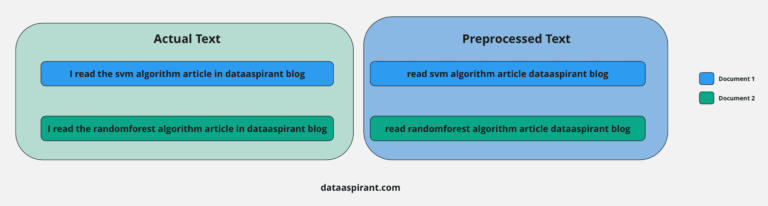


Рисунок 3 – Пример предобработки текста

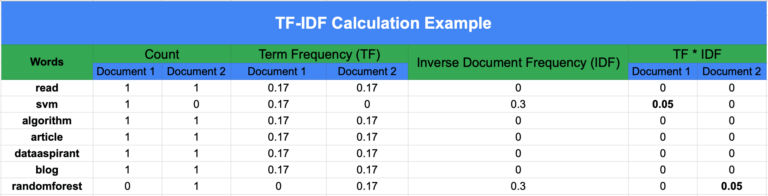


Рисунок 4 – Пример вычисления вектора

TF – Term Frequency: измеряет как часто слово встречается в документе

IDF – Inverse Document Frequency: понижает вес слов, которые часто встречаются в корпусе и повышает вес слов, которые встречаются редко.

Другими словами, TF-IDF фокусируется на важных словах. “Важными” считаются слова, которые часто встречаются в одном документе, но редко в других.

Преимущества метода:

1. Выделяет важные слова. Может найти редкие и полезные для анализа документа слова.
2. Сбалансированные веса слов.
3. Достаточно прост для понимания и использования.
4. Широко используется, поддерживается многими инструментами и библиотеками.

Ограничения метода:

1. Не понимает семантику документа, так как основан только на частоте появления слов.
2. Проблемы с редкими словами, которые могут получить слишком высокую оценку.
3. Зависимость от длины документа. В длинном документе определенное слово может встретиться гораздо чаще, чем в коротком
4. Не понимает семантику документа, так как считает частоту отдельных слов.
5. Использует модель Bag-of-Words. Игнорирует порядок слов, из-за которого может меняться значение предложения.

Таблица 1 - Сравнение с моделью BoW:

|  |  |  |
| --- | --- | --- |
| Признак | TF-IDF | BoW |
| Определение важности слова | Да | Нет |
| Возможность работы с распространенными словами(стоп-слова) | Да | Нет |
| Учитывает порядок слов | Нет | Нет |
| Простота понимания и использования | Да | Да |
| Возможность работы с большими наборами данных | Да | Да |

2. Практическая часть

2.1. Предобработка данных

Для начала импортируем необходимые библиотеки:

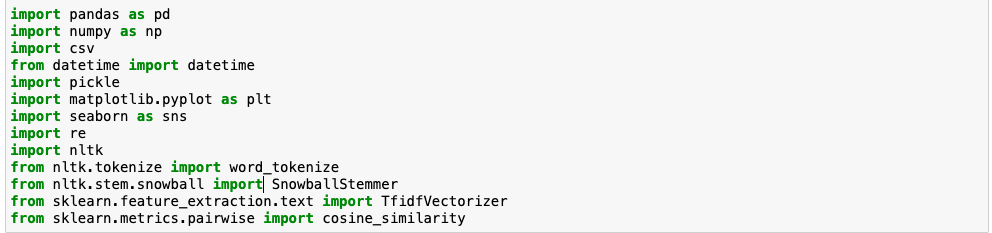


Рисунок 5 – Пример кода

Никакого стандарта для файлов такого формата не существует. Но сначала хотя бы посмотрим на данные, предварительно считав диалект скриптом detect\_csv\_dialect.py:

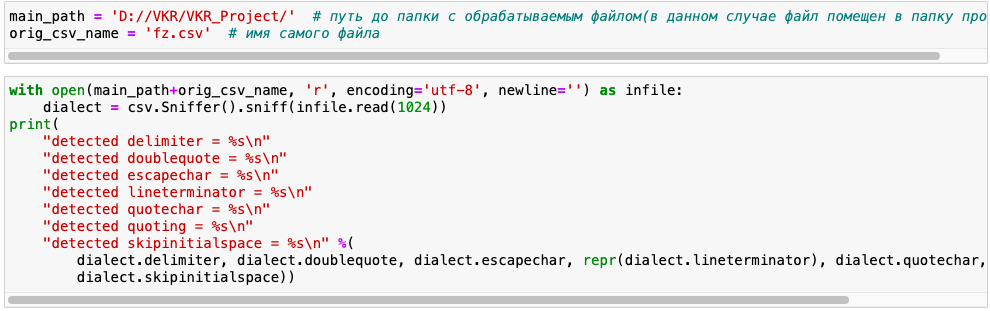


Рисунок 6 – Пример кода

Похоже на диалект excel, который используется по умолчанию:

1. delimiter ','
2. doublequote True
3. escapechar None
4. lineterminator \r\n
5. quotechar "
6. quoting csv.QUOTE\_MINIMAL
7. skipinitialspace False
8. strict False

Исходим из того, что файл все-таки соответствует некоторым общепринятым практикам, указанным в RFC 4180 Definition of the CSV Format. Попытка считать файл методом read\_csv из библиотеки pandas привела к ошибке:

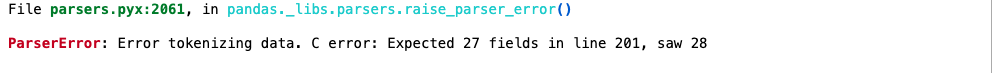


Рисунок 7– Пример кода

После считывания первых 10 строк обнаруживаем, что в файле отсутствует заголовок, и первая строка принимается за него. Уточнили, что большинство записей содержит 27 столбцов. Пока при загрузке в датафрейм будем использовать параметр header = None.

Одним из решений проблемы токенизации является пропуск строк с ошибками. При таком количестве объектов для проведения анализа это возможно. Но оставим этот вариант на крайний случай.

Попробуем в явном виде считать 28 колонок, чтобы посмотреть что не так с 201 строкой. Для этого определяем список из индексов и используем его в параметре use\_cols, чтобы в явном виде считать ошибочную строку, используем параметр skiprows и nrows для чтения ограниченного количества строк:

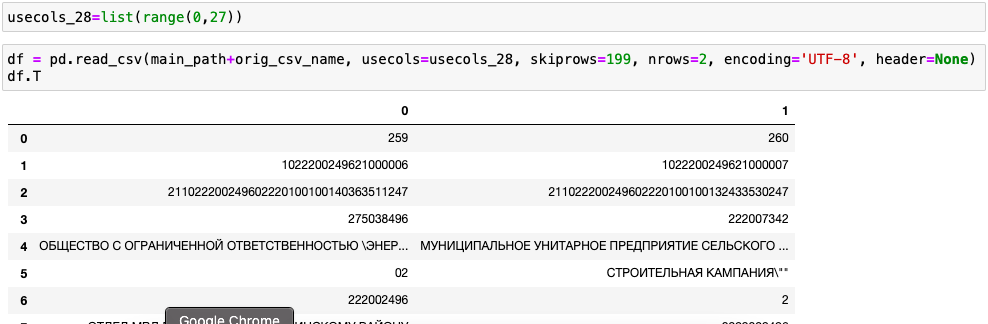


Рисунок 7 – Пример кода

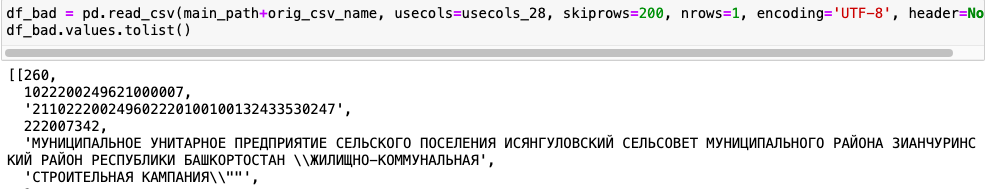


Рисунок 8 – Пример кода

Проверяем запись по номеру контракта в реестре контрактов на сайте [https://zakupki.gov.ru](https://zakupki.gov.ru/). В карточке контракта указан поставщик “МУНИЦИПАЛЬНОЕ УНИТАРНОЕ ПРЕДПРИЯТИЕ СЕЛЬСКОГО ПОСЕЛЕНИЯ ИСЯНГУЛОВСКИЙ СЕЛЬСОВЕТ МУНИЦИПАЛЬНОГО РАЙОНА ЗИАНЧУРИНСКИЙ РАЙОН РЕСПУБЛИКИ БАШКОРТОСТАН "ЖИЛИЩНО-КОММУНАЛЬНАЯ,СТРОИТЕЛЬНАЯ КАМПАНИЯ"”. То есть в строке имеется наименование в двойных кавычках, которое содержит в себе ','. Внутри кавычек также видим символ ''. Подозреваем проблемы с экранированием символов, которые приводят к неправильному парсингу. Добавим в функцию чтения параметр escapechar = '\'.

Чтение проходит без ошибок.

После проверки на следующем куске данных снова получаем ошибку токенизации и снова в ячейке с называнием:

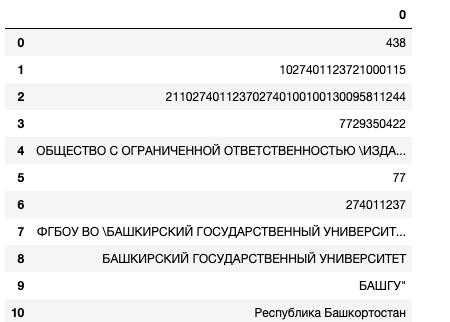


Рисунок 9 – Пример кода

С помощью скрипта raw\_file\_count\_rows.py считаем весь файл целиком и посчитаем кол-во строк без escapechar.

1. TOTAL number of rows in the CSV file: 29320202
2. Number of GOOD rows in the CSV file: 28538110
3. Number of BAD rows in the CSV file: 782092
4. Percentage of BAD rows in the CSV file: 2.66741682066174

Сохранили "плохие" строки в отдельный файл bad\_rows.csv. При просмотре файла кажется, что ошибка при чтении всего одна и только с наименованиями.

После подсчета строк с параметром escapechar = '\' получили:

1. TOTAL number of rows in the CSV file: 29320202
2. Number of BAD rows in the CSV file: 0
3. Number of GOOD rows in the CSV file: 29320202
4. Percentage of BAD rows in the CSV file: 0.0

Cчитаем проблему чтения решенной.

Проверив несколько контрактов в реестре контрактов на сайте zakupki.gov.ru, удалось установить названия практически всех столбцов.

Список признаков:

1. идентификатор записи, число
2. реестровый номер контракта, число
3. идентификационный код закупки, число
4. ИНН поставщика, число
5. поставщик(наименование юридического лица или ФИО физического лица), строка
6. неизвестно, число
7. ИНН заказчика, число
8. наименование заказчика, строка
9. регион заказчика(поставки), строка
10. код бюджетной классификации, число
11. статус контракта, строка
12. предмет контракта(объект закупки), строка
13. источник финансирования, строка
14. дата заключения контракта, дата
15. дата начала исполнения, дата
16. дата окончания исполнения контракта(срок исполнения), дата
17. дата
18. дата
19. дата
20. основание расторжения контракта, строка //предположительно связано со статусом контракта "исполнение прекращено"
21. начальная(максимальная) цена контракта, число
22. цена контракта в рублях, число
23. изменения в цене контракта, число //предположительно связанно с изменениями условий контракта
24. обоснование внесений изменений в контракт, строка
25. обозначение раздела в ОКПД2, символ из спиcка заглавных латинских букв [A-U]. Тут же появляется новое значение для пропуска '<НЕ РАСПРЕДЕЛЕНО>'
26. объект закупки по коду ОКПД2, строка
27. код ОКПД2, строка

Большинство признаков - категориальные данные, которые будут считываться в скриптах как строки. Для визуализации подойдут столбчатые диаграммы. Для использования в анализе их необходимо будет закодировать.

Количественные данные есть только в колонках, связанных со стоимостью контракта.

Реестровый номер закупки идентифицирует ее, большинство категориальных данных однозначно с ним связаны или с кодом ОКПД2. Эти два признака и будут использоваться в качестве основных в дальнейших методах предварительной обработки данных.

Для отбора строк по кодам ОКПД2 используем скрипт select\_by\_OKPD2.py. Отобранные строки записаны в файл selected\_rows.csv с разделителем '|'. Снизили размер файла до 2 Гб. Продолжим работать с дублями и пропусками.

Так как данных много, отобразим количество пропусков с помощью графика.

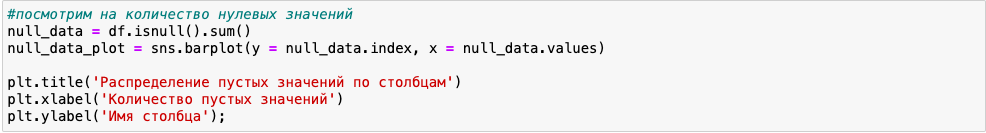


Рисунок 10 – Пример кода



Рисунок 11 – Распределение пустых значений по столбцам

В идеале конечно нужно заполнить все строки по имеющейся на сайте госзакупок информации. Но пока попробуем просто отобрать признаки. Номер закупки оставляем для поиска дублей. Основной признак для анализа по ТЗ – объект закупки, дополнительные признаки по ТЗ – стоимость(фактическая цена) и длительность в днях(разница между датами начала и окончания контракта). Но в нашей рекомендательной системе они использоваться не будут. Оставим их для поиска дублей. Максимум пропусков как раз находится в признаках, которые не планируется использовать. Эти колонки мы отбросим. В ходе преобразования дат и вычисления длительности контракта, получили 3 строки с пустыми значениями, которые также удалили.

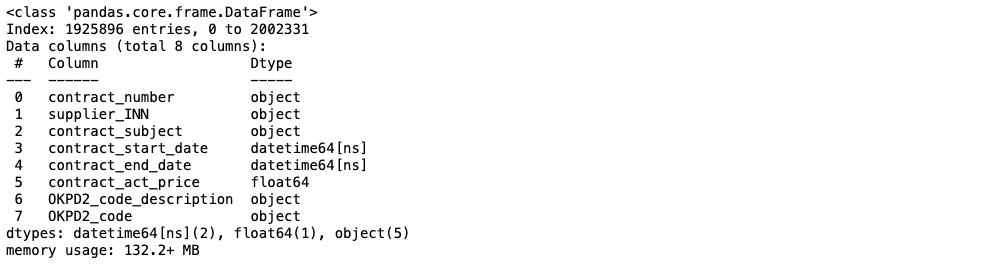


Рисунок 12 – Результат обработки пропусков

Получили небольшое снижение размерности и полностью заполненные значения признаков. Переходим к работе с дублями.

В самом начале уже есть дублирование номеров контрактов и большое количество пропущенных значений. Можно заранее уточнить логику формирования файла, и сформулировать правила определения задублированных строк. Делать это имеет смысл после заполнения пропусков и перед отбором признаков. Но даже при заполненных пропусках у нас нет возможности уточнить какие строки считать дублями. Поэтому полностью совпадающие строки будут удалены.

Также в техническом задании указано, что один контракт может относиться сразу к нескольким группам ОКПД2. Например, контракт 1020200039521000238. Такие контракты имеют несколько кодов ОКПД2 и внесены в файл отдельными строками с единым номером контракта. При просмотре нескольких таких контрактов замечено, что у них общее описание, общая цена контракта и даты. Можно попробовать добавить в такие объекты закупки описания кодов ОКПД2, загрузить дополнительно цену по категориям. При обработке сохраним их в отдельный файл.

Еще один тип дублей – контракты с одним объектом закупки и разными КБК. Их также можно отнести к полным дублям.

Другой проблемой может являться обобщенное описание предмета закупки. Можно записать такие объекты и отдельно проверить как на них работает модель.

В итоге к значимым для анализа дублей признакам можно отнести реестровый номер закупки, цену контракта в рублях, дату начала и окончания исполнения и код ОКПД2:

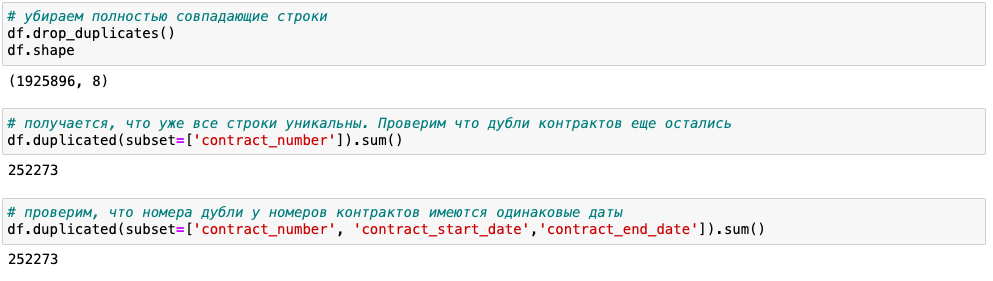


Рисунок 13 – Пример кода

То есть дальше уже фильтруем только по номерам контрактов. Перед этим сначала вычислим длительность контрактов в днях. После подсчитываем записи с по номеру контракта. Их будем считать уникальными:

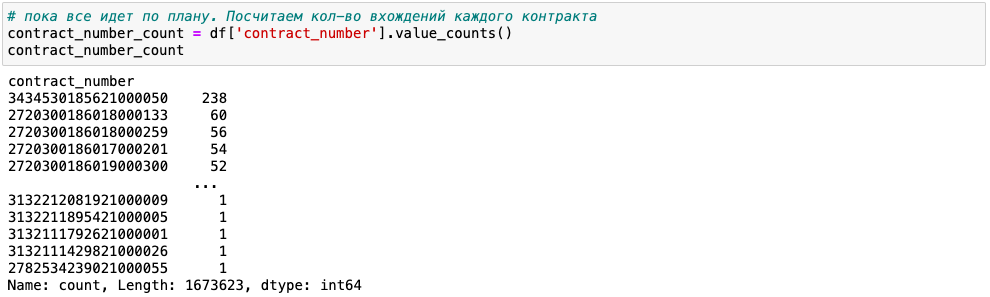


Рисунок 14 – Пример кода

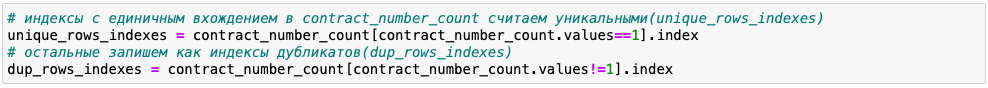


Рисунок 15 – Пример кода

Считаем, что с основной очисткой справились датафрейма справились. У нас таблица размером (1474956, 7).

Теперь посмотрим на сами данные.

1. contract\_number - уже в принципе не очень нужен, но можно оставить как идентификатор
2. supplier\_INN - для задач классификации нужно будет закодировать
3. contract\_subject - основной для анализа признак, оставляем строкой
4. contract\_duration\_days - уже число
5. contract\_act\_price - уже число

OKPD2\_description - будет использоваться в выводе рекомендательной системы, оставляем строкой

OKPD2\_code - необходимо закодировать

Колонку с кодами OKPD2\_code, которых всего 10, сразу переведем в категориальные данные.

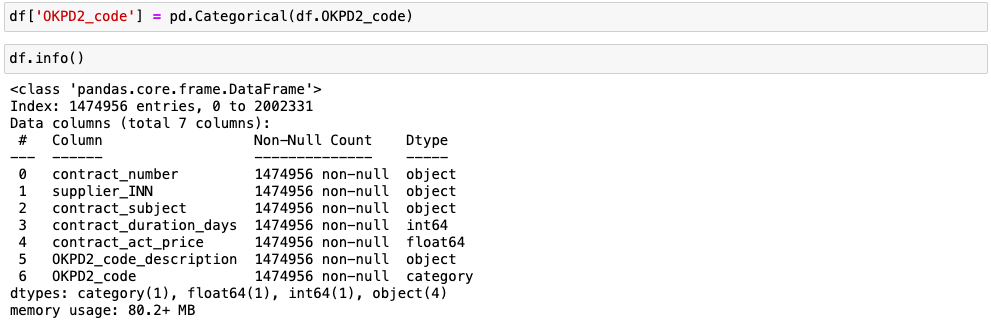


Рисунок 16 – Пример кода

Посмотрим насколько изменилось распределение закупок по категориям:

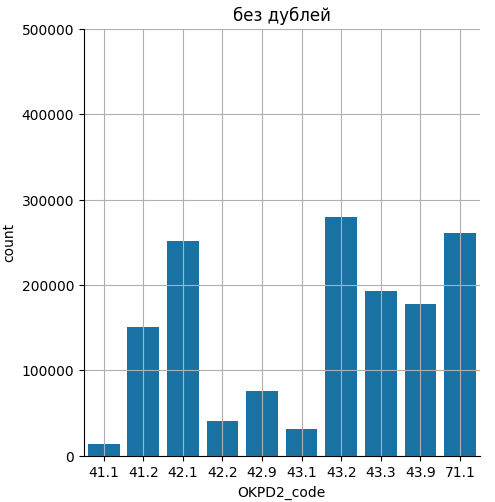
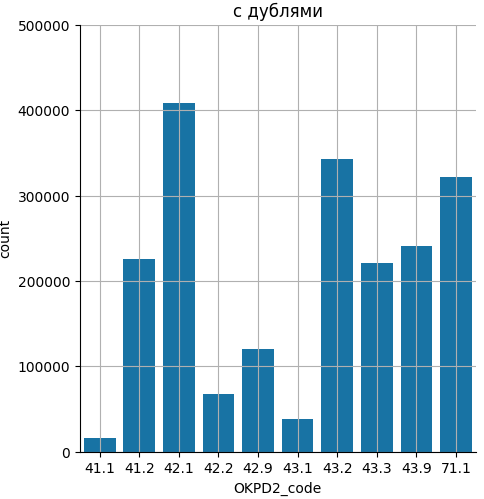


Рисунок 16 – Сравнение дублей

Графики похожи, выбивается только одна категория, в остальном удаление дублей не повлияло критично количество предметов закупки по категориям. Отмечаем изначальную несбалансированность классов. Одним из вариантов решения проблемы является выборка снижение количества примеров в мажоритарных классах.

Далее переходим к предобработке текстовых данных. Начинаем с удаления пунктуации и стоп-слов (слова более трех символов)

Результат работы:

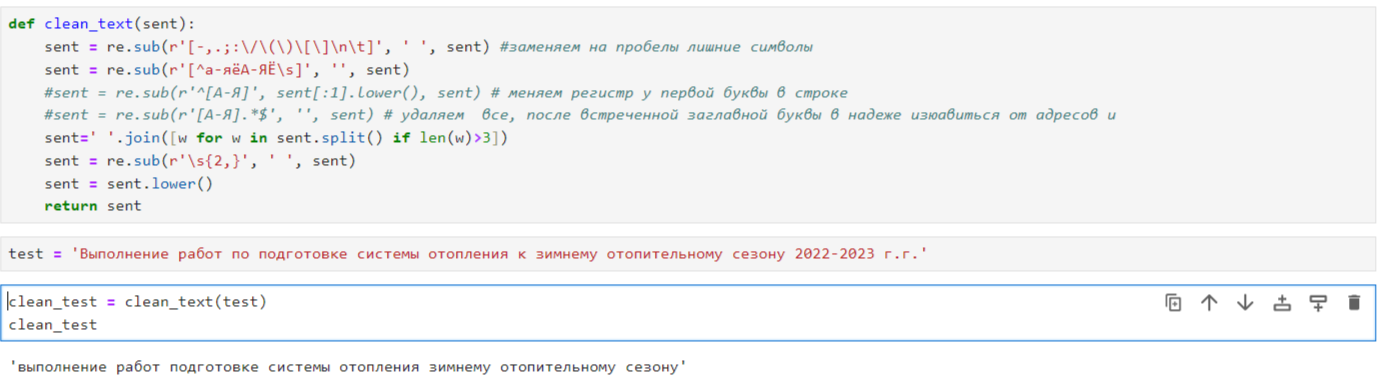


Рисунок 17 – Результаты очистки текста

Текст уже выглядит неплохо. Полностью доступен в файле check.csv. Подход с удалением трехсимвольных слов сработал. Причем данные пригодны для грубого разбиения методом сплит вместо использования токенизатора из библиотека. Но в работе использовался метод word\_tokenize из библиотеки nltk. Полученные результаты добавляем отдельными колонками в датафрейм.

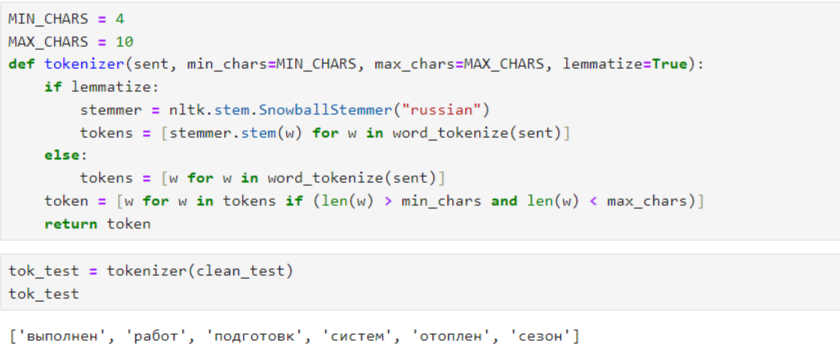


Рисунок 18 – Результат токенизации

Создаем новые столбцы с очищенными данными

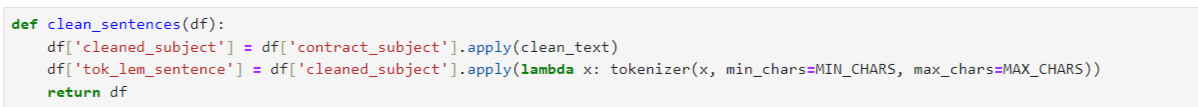


Рисунок 19 – Пример кода

2.2. Разработка и обучение модели

Функция extract\_best\_indices принимает на вход полученные в следующем методе косинусные меры, упорядочивает их в порядке убывания, и выдает значения индексов объектов с самым высоким значением меры:

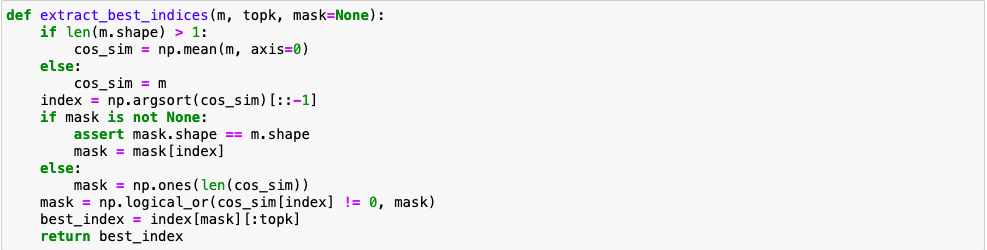


Рисунок 20 – Пример кода

На этапе предобработки мы уже подготовили данные для оценки меры подобия.

Функция get\_recomendations использует косинусную меру подобия, взятую из библиотеки sklearn. Объект для поиска преобразуется в частотный вектор, после просматриваются метрики сходства с векторами из матрицы tfidf\_mat для всех доступных нам описаний объектов закупки. Возвращаются индексы наиболее похожих по описанию объектов:

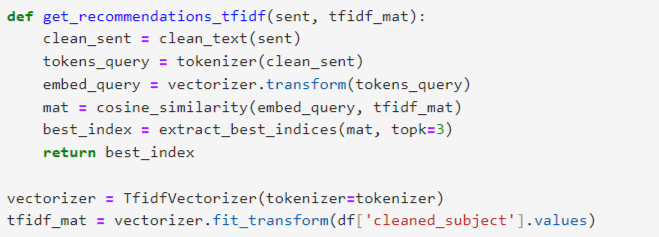


Рисунок 21 – Пример кода

2.3. Тестирование рекомендательной системы

Получен не очень хороший результат:

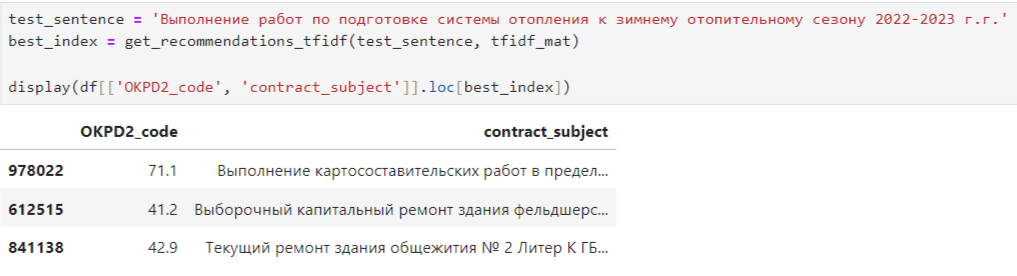


Рисунок 22 – Результат работы системы

Скорее всего самым важным оказалось слово выполнение.

2.4. Разработка приложения

Функционал приложения предполагает ввод объекта описания в виде строки и получение списка рекомендаций на основании объекта закупки с указанием кода ОКПД2. Рекомендации выдаются в порядке убывания.

2.5. Создание удаленного репозитория и загрузка результатов работы на него

<https://github.com/n4ernenko/VKR_Bauman>

Заключение

В ходе работы был проведен первичный анализ, предобработка данных и создана рекомендательная система с использованием методов обработки естественного языка. Так как в решении были использованы максимально простые подходы и методы предобработки и подсчета метрики сходства, результат получился неточным и плохо интерпретируемым. В качестве меры улучшения качества можно попробовать использовать классический BoW, изменить подходы к предобработке. Например, создать собственный набор стоп-слов на основе частотного словаря, удалив часто и редко встречающиеся слова. Была предпринята попытка удаления наименований объектов и адресов в описании закупки. Для начала был понижен регистр первой буквы в предложении и удалении всех букв, после первой встреченной заглавной буквы. В результате был получен новый объем пустых значений. Это связано с тем, что некоторые строки были полностью написаны заглавными буквами, некоторые начинались с цифр или аббревиатур. От этого способа было решено отказаться на данном стадии исследования. Полученную систему после дополнительной настройки можно использовать для распределения объектов закупки по классам для последующего обучения моделей классификации.