Разработка алгоритма прогнозирования выполнения задачи

«Цифровой прорыв», г. Владивосток

8 сентября – 6 октября 2022 г.

Задача «Collector»

Постановка задачи

- Предсказать время выполнения задачи в секундах (целое число) на основе исторических данных
- Требуется максимизировать метрику R2
- Перед нами задача регрессии

Признаки из набора issues

Поле	Описание	Not Null	Признак(и)
id	Идентификатор задачи	+	Выбрасываем сразу
created	Дата создания задачи	+	Разбить на день, месяц, год
summary	Заголовок задачи в Jira	+	Представить текст, как вектор
project_id	Идентификатор проекта	+	номинальный признак
assignee_id	Идентификатор исполнителя	+	Выбрасываем, используем для связи
creator_id	Идентификатор создателя	+	Выбрасываем, используем для связи
overall_worklogs	Число секунд, потраченных на задачу	+	Количественный признак, целевой параметр, предсказываем

Признаки из набора employees

Поле	Описание	Not Null	Признак(и)
Id	Идентификатор сотрудника	+	Выбрасываем, используем только для связи
Active	Работает или уволен	+	Сразу выбрасываем (бинарный признак)
full_name	Имя и фамлиля сотрудника	+	Сразу выбрасываем
position	Должность	-	Номинальный признак
hiring_type	Схема найма	-	Номинальный признак
payment_type	Схема оплаты	-	Номинальный признак
salary_calculation_type	Схема рассчёта ЗП	-	Номинальный признак
english_level	Уровень английского	-	Порядковый признак
passport		+	Бинарный признак
is_nda_signed	Подписано NDA	+	Бинарный признак
is_labor_contract_signed		+	Бинарный признак
is_added_to_internal_chats		+	Бинарный признак
is_added_one_to_one		+	Бинарный признак

Признаки из набора comments

Поле	Описание	Not Null	Признак(и)
comment_id	Идентификатор комментария	+	Выбрасываем сразу
Text	Текст комментария	+	Представим, как вектор
issue_id	Идентификатор задачи	+	Выбрасываем, используем только для связи
author_id	Идентификатор автора задачи	+	Выбрасываем, используем только для связи

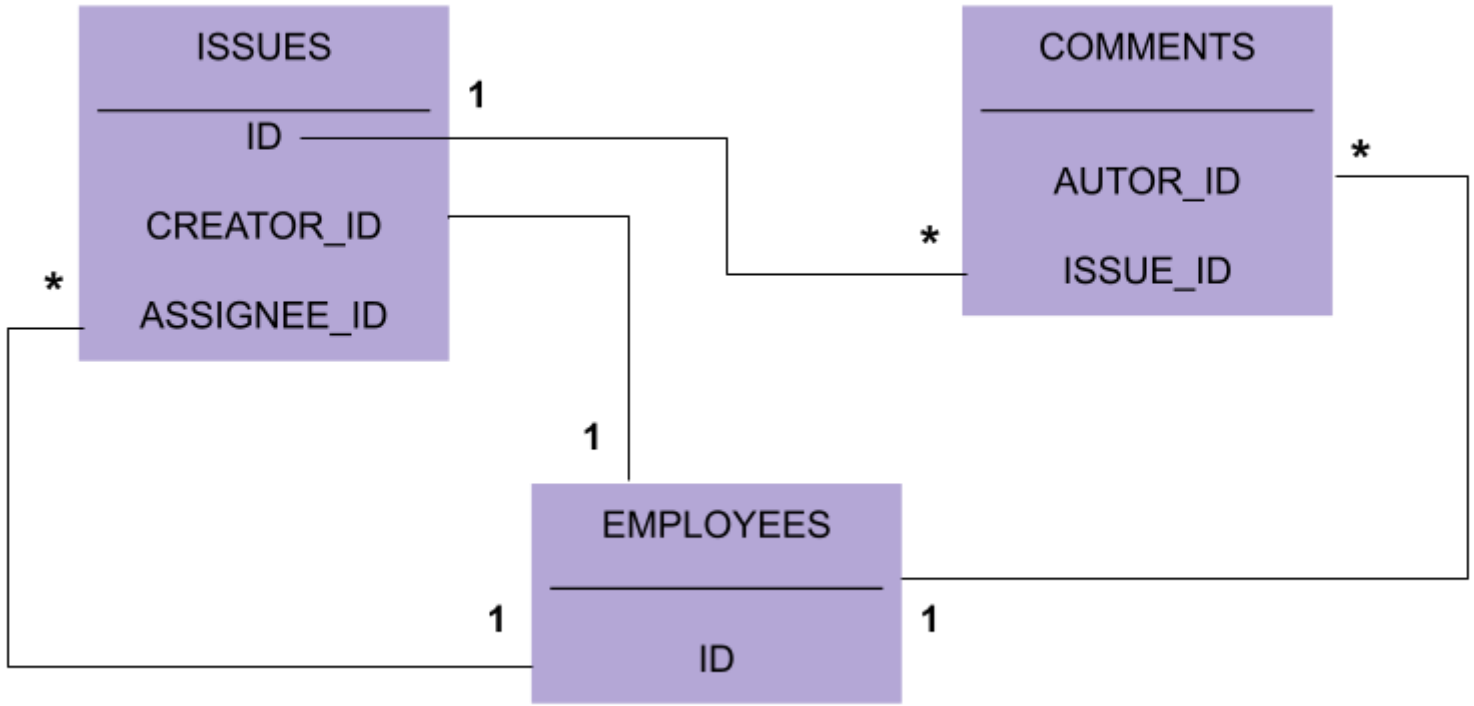
Преобразование текста в векторы doc2vec

- Используется алгоритм doc2vec: обобщение алгоритма word2vec
- Используется реализация алгоритма doc2vec из библиотеки genism
- Преобразуется текст из заголовка задачи issue.summary в вектор. Так как данные summary заполнены, каждой задаче сопоставляется вектор
- Преобразуется текст из текста комментария comment.text в вектор. Так как все данные text заполнены, каждому комментарию сопоставляется вектор
- Используется модель «distributed bag of words» (PV-DBOW). Порядок слов в тексте не учитываем. Пробовал учитывать, т.е. использовать «distributed memory» (PV-DM) - результат хуже

Кластеризация текстов K-Means

- Кластеризация – это разбиение заданной выборки объектов на непересекающиеся множества, чтобы кластер состоял из похожих объектов, а объекты разных кластеров существенно различались
- Тексты преобразованы в векторы, кластеризуются векторы
- Используется алгоритм «k-средних» из библиотеки h2o.ai
- K = 100 исходя из числа задач и комментариев. Попробовал разные K
- Каждому комментарию сопоставляется номер кластера комментариев
- Каждой задаче сопоставляется номер кластера задачи

Модель данных



Модель данных

Связи:

- Employee – Issue (по creator_id): один ко многим
- Employee – Issue (по assignee_id): один ко многим
- Employee – Comment: один ко многим
- Issue – Comment: один ко многим

Новые признаки из связей и по результатам кластеризации текстов

- Employee: Для каждого сотрудника считаем количество задач, в которых он assignee, и группируем по номеру кластера задачи (+N новых признаков)
- Employee: Для каждого сотрудника считаем количество задач, в которых он creator, и группируем по номеру кластера задачи (+N новых признаков)
- Employee: Для каждого сотрудника считаем количество комментариев, которые он написал, и группируем по номеру кластера комментариев (+M новых признаков)
- Issue: Для каждой задачи считаем количество комментариев и группируем их по кластеру комментария (+K новых признаков)

Новые признаки и проблемы

Проблема:

- По результатам экспериментов становится понятно, что результаты хуже, если просто добавить новые фичи в датасет
- Алгоритмы начинают считать их множеством признаков, таким образом сильно усиливается влияние одного признака (номера кластера)
- Нужно снова превратить группу признаков в один

Решение:

- Кластеризовать по количеству задач/комментариев в каждом кластере

Кластеризуем сущности по количеству в каждом кластере

Дано:

- Задачи, каждая из которых сопоставлена с номером кластера (по заголовку summary) и сотрудником исполнителем
- Сотрудники

Кластеризуем по количеству в каждом кластере

До

ISSUES

ID	CLUSTER	ASSIGNEE
1	15	11
2	16	11
3	16	11
3	16	12

Группировка задач по паре “кластер-исполнитель” и подсчёт таких пар

После

ASSIGNEE	CLUSTER	COUNT
11	16	2
11	15	1
12	16	1

Для каждого осрудника получаем количество задач из каждого кластера. Или ставим 0, если не находим

	C1	C2	..	**C15**	**C16**	...	Cn
A1	0	0	..	0	0	...	0
...	0	0	..	0	0	...	0
A11	0	0	..	**1**	**2**	...	0
A12	0	0	..	0	**1**	...	0
...	0	0	..	0	0	...	0
An	0	0	..	0	0	...	0

Кластеризуем!!!

	C1	C2	..	**C15**	**C16**	...	Cn	**C**
A1	0	0	..	0	0	...	0	**1**
...	0	0	..	0	0	...	0	**1**
A11	0	0	..	**1**	**2**	...	0	**3**
A12	0	0	..	0	**1**	...	0	**2**
...	0	0	..	0	0	...	0	**1**
An	0	0	..	0	0	...	0	**1**

Результаты кластеризации по количеству в каждом кластере

- Employee: Для каждого сотрудника есть номер кластера задач, в которых он исполнитель (assignee_cluster), один признак
- Employee: Для каждого сотрудника есть номер кластера задач, в которых он создатель (creator_cluster), один признак
- Employee: Для каждого сотрудника есть номер кластера комментариев, которые он писал (comments_cluster), один признак
- Employee: Каждого сотрудника можно дополнительно кластеризовать по кластерам assignee, creator и comments (employee_cluster)
- Issue: Для каждой задачи есть номер кластера комментариев, которые к ней оставлены, один признак

Результаты экспериментов с объединёнными признаками

- Использование единого кластера employee_cluster даёт результаты хуже, чем использование assignee_cluster, creator_cluster и comments_cluster по отдельности

Восстановление пропусков в данных

- Значение корреляции Пирсона должности сотрудника (position) со временем выполнения задачи одно из самых больших среди всех известных данных
- При этом поле position у примерно половины сотрудников не заполнено, т.к. они уже не работают в компании
- Многие другие поля также не заполнены у (в основном, уволенных) сотрудников
- Восстанавливать будем, используя машинное обучение, сведя эту задачу к задаче классификации. Признаки: assignee_cluster, creator_cluster и comments_cluster. Целевая переменная: position. Помимо признаков-кластеров есть not null поля, которые тоже будем использовать

Классификация полей employee по кластерам

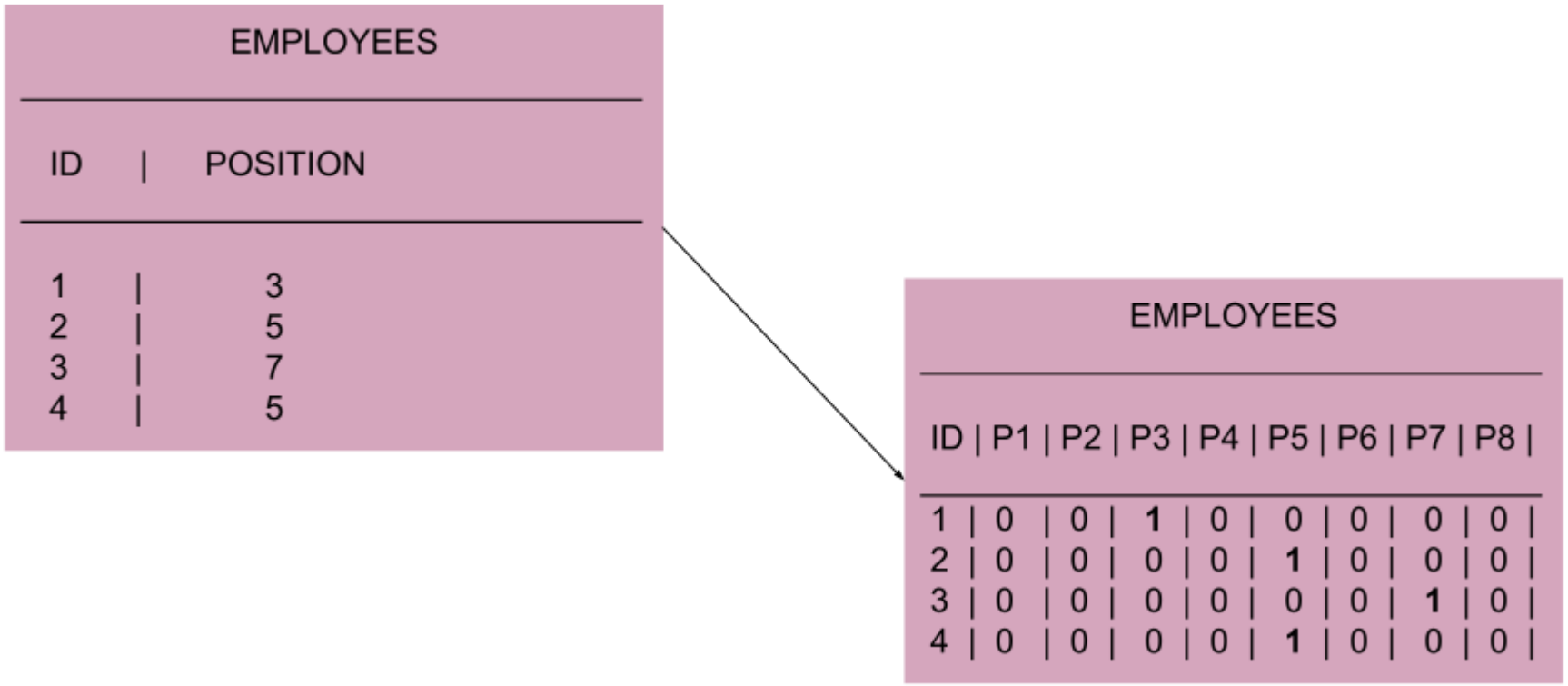
- Предсказываем пропуски position
- Предсказываем пропуски hiring_type
- Предсказываем пропуски payment_type
- Предсказываем пропуски salary_calculation_type
- Предсказываем english_level
- Все признаки заполнены! Пустых полей больше нет

Кодирование признаков: порядковые

- Уровень английского (english_level) – порядковый признак. Чтобы алгоритм машинного обучения понимал, какой уровень английского выше, а какой ниже, преобразуем данные столбца по правилу:
  - A1 → 0
  - A2 → 1
  - B1 → 2
  - B2 → 3
  - C1 → 4

Кодирование признаков: номинальные

- У номинальных признаков нет отношения порядка, поэтому преобразуем их в набор столбцов через one hot encoding



Результаты экспериментов с после кодирования признаков employee

- Однозначно отбрасываем все признаки employee, кроме position, так как алгоритм машинного обучения работает намного хуже
- Финальные признаки employee: position, assignee_cluster, creator_cluster, comments_cluster
- Добавление employee.position (one hot encoding) в набор признаков тоже ухудшает результат
- Отказ от one hot encoding и переход на label encoding для position
- Идея: вероятно, можно попробовать кодировать position через frequency encoding (не реализовано)

Поиск решения с помощью LightAutoML

- По итоговому набору признаков ищем решение задачи регрессии с помощью LightAutoML
- LightAutoML строит ансамбль из моделей ансамбль моделей LightGBM и CatBoost
- Точность решения (R2): **0.024783761079684496**

```
INFO:lightautoml.automl.base:Layer 1 training completed.

INFO:lightautoml.automl.blend:Blending: optimization starts with equal weights and score 0.024783761079684496
INFO:lightautoml.automl.blend:Blending: iteration 0: score = 0.028225012245828185, weights = [0.         0.         0.4035965  0.4430443  0.15335923]
INFO:lightautoml.automl.blend:Blending: iteration 1: score = 0.028226348864145878, weights = [0.         0.         0.39829376 0.4535524  0.14815387]
INFO:lightautoml.automl.blend:Blending: iteration 2: score = 0.028226378090335125, weights = [0.        0.        0.3975065 0.4550901 0.1474034]
INFO:lightautoml.automl.blend:Blending: iteration 3: score = 0.028226377929846502, weights = [0.         0.         0.3973499  0.4553048  0.14734533]
INFO:lightautoml.automl.blend:Blending: iteration 4: score = 0.028226377877376696, weights = [0.         0.         0.39736453 0.45529374 0.14734176]
INFO:lightautoml.automl.presets.base:Automl preset training completed in 243.71 seconds

INFO:lightautoml.automl.presets.base:Model description:
Final prediction for new objects (level 0) =
        0.39736 * (5 averaged models Lvl_0_Pipe_1_Mod_1_Tuned_LightGBM) +
        0.45529 * (5 averaged models Lvl_0_Pipe_1_Mod_2_CatBoost) +
        0.14734 * (5 averaged models Lvl_0_Pipe_1_Mod_3_Tuned_CatBoost)
```

Что нового я узнал, пока решал задачу

- Алгоритмы машинного обучения очень чувствительны к качеству данных. Предварительная подготовка данных для анализа – самый важный этап работы
- Алгоритмы word2vec и doc2vec требуют большого количества тестовых данных. Чем меньше данных, тем более случайным получается результат
- Алгоритмы word2vec и doc2vec требуют тщательной настройки, которую трудно автоматизировать. Подбор гиперпараметров – почти искусство. Без понимания внутренней работы алгоритма лучше использовать значения гиперпараметров по-умолчанию

Контактная информация

- Почта: valeriy@manenkov.com
- Telegram: @vmanenkov
- Все контакты: http://v.manenkov.com