

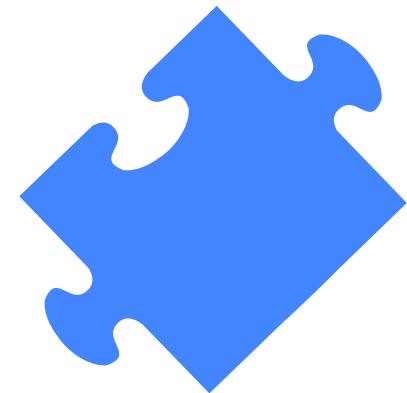
Кейс 2

Рекомендация типов для селлера

Команда 9:
Александр Тарабенко
Анастасия Золотухина
Виталина Локтева
Владислав Ежергин
Владислав Шуренков

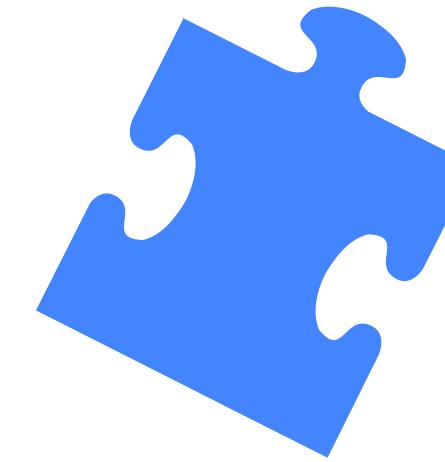


Команда:



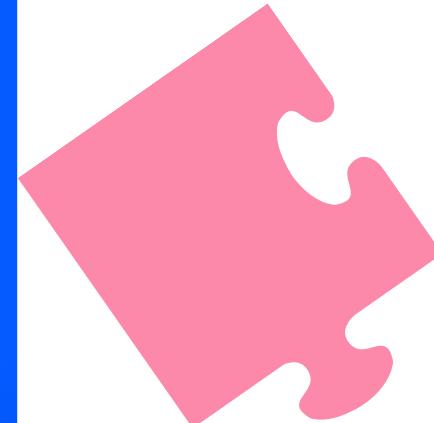
Team Leader:

Виталина Локтева



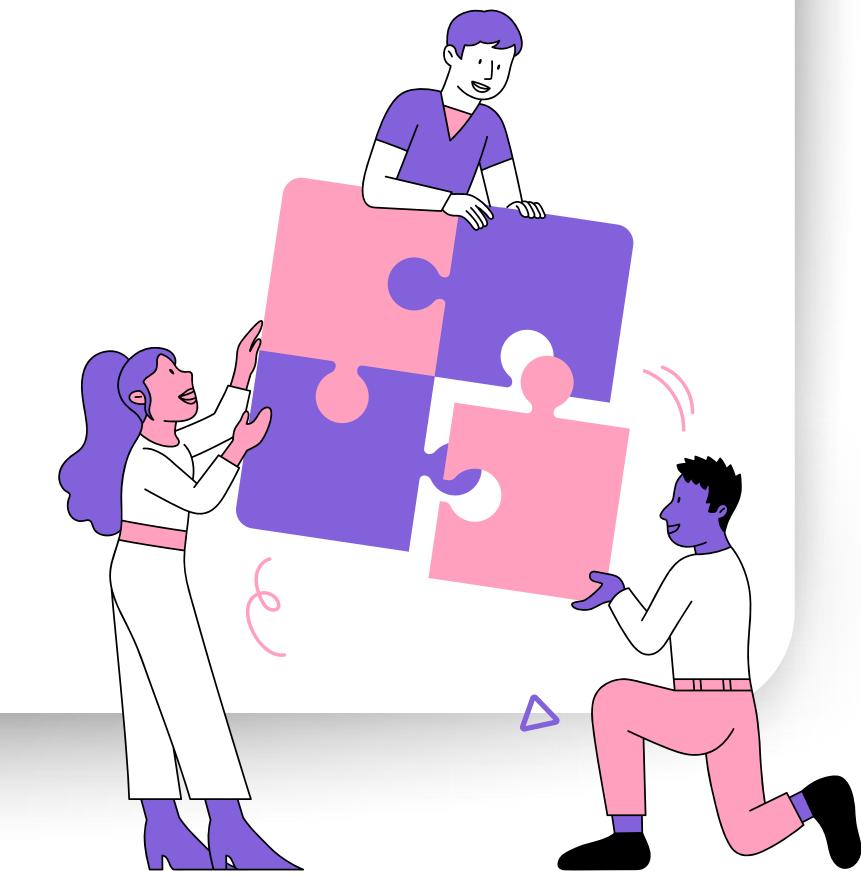
ML Engineer (Text):

Александр Тарасенко
Владислав Ежергин
Владислав Шуренков



Data Analyst:

Анастасия Золотухина



Цель:

Разработать модель машинного обучения для автоматической проверки корректности указанного продавцом типа товара, чтобы повысить качество классификации товаров и снизить количество ошибочно присвоенных категорий.

Задачи:

1. Анализ и подготовка данных
2. Формирование признаков
3. Обучение и оценка качества модели
4. Создание рекомендательной системы

EDA:

Train	Test
763013	ROWS
47730	DUPLICATES
673.5 MB	RAM
8	FEATURES
3	CATEGORICAL
0	NUMERICAL
5	TEXT

ASSOCIATIONS

ASSOCIATIONS

Количество признаков: 8

Основные поля:

main_photo – ссылка на изображение товара

name – название товара

type – тип товара, указанный селлером

category_I2, category_I4 – иерархические категории товара

is_markup – признак, указывает на возможный «манипулятивный» выбор категории

target – целевая переменная (1 – тип указан верно, 0 – неверно)

EDA:

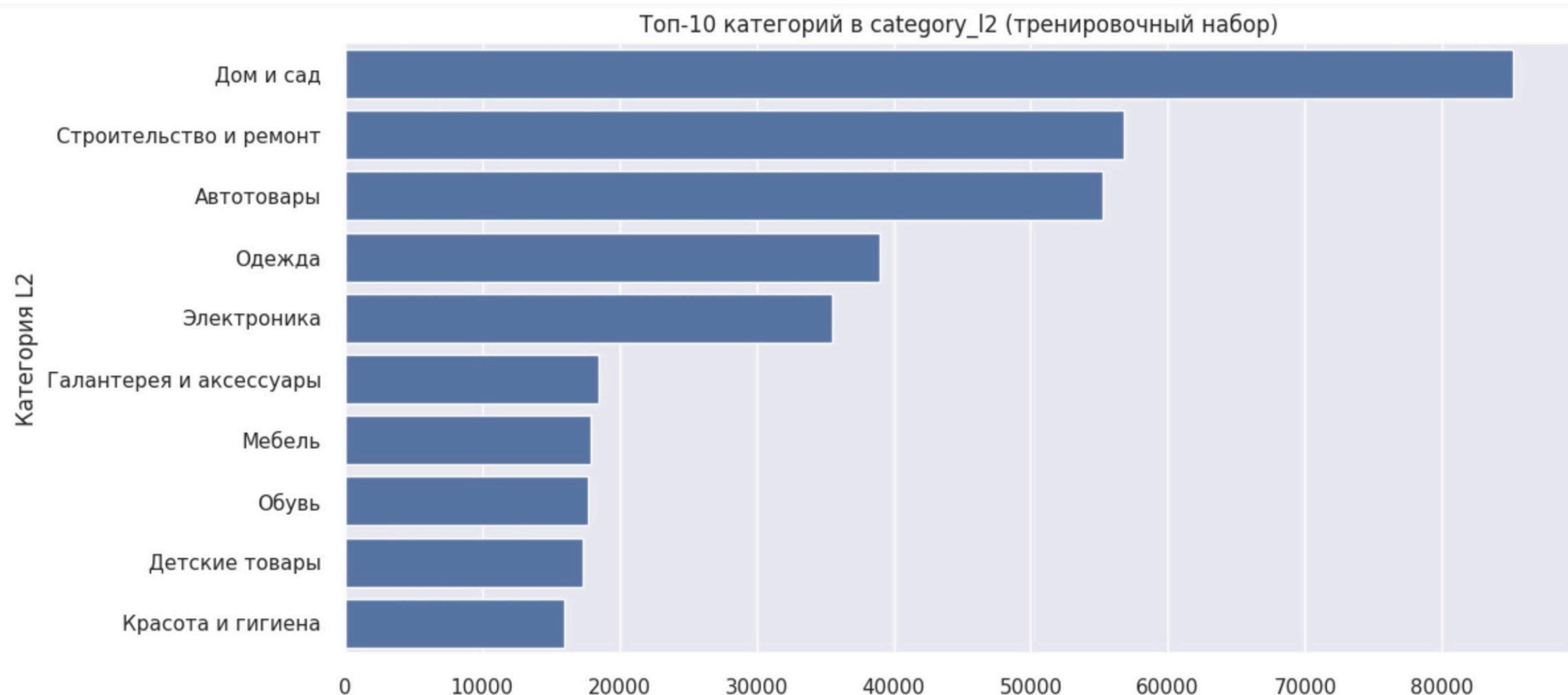
Анализ распределения целевой переменной



Распределение равномерное, классы сбалансированы.

EDA:

Топ-10 категорий для тренировочного набора



EDA:

Анализ столбцов паме и туре в тренировочном наборе и визуализация в виде облака слов

Облако слов для столбца "name" (тренировочный набор)



Столбец name

Облако слов для столбца "type" (тренировочный набор)



Столбец type

Результаты EDA:

Значительные пропуски в category_l2 и category_l4, встречаются дубликаты.

Сравнение наборов: распределения признаков train и test в целом схожи.

Рекомендации:

1. Обработать пропуски (заполнение, отдельная категория).
2. Удалить дубликаты.
3. Создать новые признаки (из названий и URL изображений).

Варианты решений:

Решение	Методология	Результат
Решение 1 Ансамбль моделей	<ul style="list-style-type: none">- Объединение name + type- Лемматизация- TF-IDF-векторизация- PyCaret для подбора моделей- Ансамбль: LightGBM + CatBoost + XGBoost	ROC-AUC: 0.8222
Решение 2 2-уровневая классификация и рекомендации	<ul style="list-style-type: none">- Модель 1: бинарная классификация (верный тип или нет)- Модель 2: рекомендация наиболее вероятных type (многокласс)- TF-IDF + PCA	ROC-AUC: 0.8322
Решение 3 Логическая проверка через словарь	<ul style="list-style-type: none">- Модель 1: $P(\text{target}=1)$ по name + type- Модель 2: category_I2 по name- Словарь {category_I2: [types]}	ROC-AUC для модели 1: 0.88

Подготовка данных:

Удаление дубликатов:

Количество дублирующихся строк в обучающих данных: 47734

Количество дублирующихся строк в тестовых данных: 5224

Дублирующиеся строки были удалены.

Количество дублирующихся строк в обучающих данных после удаления: 0

Количество дублирующихся строк в тестовых данных после удаления: 0

Удаление столбцов:

1. Удалили 'stratify_column' (комбинация category_l4, is_markup, target).
2. Удалили 'main_photo' (обработка признаков из изображений требует слишком много аппаратных и временных ресурсов).
3. Удалили 'category_l2', 'category_l4', 'is_markup'.

Подготовка данных:

Лемматизация

процесс приведения слова к его нормальной (словарной) форме (лемме)

После лемматизации:

Train data

Уникальных слов в 'name': 322398 (было 503371)

Уникальных слов в 'type': 9054 (было 13488)

Test data

Уникальных слов в 'name': 143000 (было 227023)

Уникальных слов в 'type': 8493 (было 12554)

Подготовка данных:

Добавление новых признаков:

Объединение столбцов name и type в единый текстовый признак:

текстовый векторизатор, такой как TF-IDF, может анализировать объединенный текст для выявления слов и фраз из name, которые часто встречаются вместе со словами и фразами в правильном type.

Добавлены признаки: len_name - длина названия, len_type - длина типа.

Пример:

Для продукта с названием "Apple iPhone 15 Pro" и типом "Смартфон", объединенный текст будет "Apple iPhone 15 Pro Смартфон". Модель узнает, что совместное появление "iPhone" и "Смартфон" является сильным индикатором правильной классификации (**target=1**).

Если бы типом был "Тостер", текст "Apple iPhone 15 Pro Тостер" был бы очень необычным по сравнению с обучающими данными, что сигнализировало бы о неверной классификации (**target=0**).

Параметры TF-IDF:

Векторизация текстов (name, type).

Размерность: 504 признака.

max_features (макс. количество признаков):

10 000 – этого достаточно, чтобы уловить ключевые слова без перегрузки памяти.

ngram_range (диапазон n-грамм)

Лучший вариант: (1, 2) (униграммы + биграммы).

Униграммы ("iphone", "15") помогают уловить отдельные слова.

Биграммы ("iphone 15") сохраняют смысл словосочетаний.

Дополнительные настройки

min_df = 2 – игнорировать слова, встречающиеся реже 2 раз (уменьшает шум).

stop_words = 'english' – убрать стоп-слова (если текст на английском).

lowercase = True (по умолчанию) – привести все слова к нижнему регистру.

Настройка среды PyCaret:

PyCaret — это библиотека для автоматизированного машинного обучения (AutoML) в Python.

PyCaret упрощает процесс машинного обучения, автоматизируя:

1. Предобработку данных (нормализация, кодирование категориальных признаков, обработка пропусков).
2. Обучение моделей (поддержка классификации, регрессии, кластеризации и др.).
3. Подбор гиперпараметров (с помощью `tune_model`).
4. Сравнение моделей (`compare_models()` показывает лучшие алгоритмы).
5. Развёртывание (сохранение и загрузка моделей).

Классификаторы:

Топ моделей по ROC-AUC:

1. LightGBM
2. CatBoost
3. XGBoost

	Model		Accuracy	AUC	Recall	Prec.	F1	Карра	MCC	TT (Sec)
	lightgbm	Light Gradient Boosting Machine	0.7335	0.8221	0.4985	0.8870	0.6382	0.4524	0.4996	38.3720
	catboost	CatBoost Classifier	0.7270	0.8125	0.4778	0.8943	0.6227	0.4382	0.4913	49.8660
	xgboost	Extreme Gradient Boosting	0.7279	0.8104	0.4778	0.8973	0.6235	0.4400	0.4938	71.9320
	lr	Logistic Regression	0.7359	0.8057	0.6133	0.7799	0.6866	0.4640	0.4741	32.8460
	dt	Decision Tree Classifier	0.7404	0.7420	0.7112	0.7311	0.7210	0.4784	0.4786	51.2600
	knn	K Neighbors Classifier	0.5433	0.6805	0.8873	0.5091	0.6470	0.1187	0.1612	43.3680
	dummy	Dummy Classifier	0.5283	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	22.5090

Создание ансамбля: из трех лучших моделей с использованием голосования.

▶ print(comparison_df)

Model	Accuracy	AUC	Recall	Prec.	F1	Карра	MCC
LightGBM	0.7321	0.8218	0.5022	0.8769	0.6387	0.4498	0.4940
CatBoost	0.7267	0.8080	0.4791	0.8906	0.6230	0.4375	0.4895
XGBoost	0.7289	0.8125	0.4824	0.8936	0.6265	0.4422	0.4941
Blender	0.7292	0.8222	0.4820	0.8954	0.6266	0.4428	0.4953

Результаты:

Лучшей моделью оказался ансамбль, который показал ROC AUC 0.8222 (хороший результат, модель справляется с задачей классификации)

Система рекомендаций:

Примеры:

Тип верный

```
[ ] user_input_1 = {  
    'name': 'Подвесная люстра Vitaluce V1339/5',  
    'type': 'Люстра потолочная',  
    'category_12': 'Строительство и ремонт',  
    'category_14': 'Светильник для дома',  
    'is_markup': 0  
}  
  
classify_and_recommend(user_input_1)
```

```
→ {'result': 'Тип указан правильно',  
    'probability': np.float64(1.0),  
    'recommendations': 'оставляем Люстра потолочная'}
```

Тип неверный

```
[ ] user_input_2 = {  
    'name': 'Телефон Samsung',  
    'type': 'Авто',  
    'category_12': 'Одежда',  
    'category_14': 'Брюки',  
    'is_markup': 1  
}  
  
classify_and_recommend(user_input_2)
```

```
→ {'result': 'Тип, скорее всего, неверный',  
    'probability': np.float64(0.12),  
    'recommendations': [{  
        'type': 'other', 'score': np.float64(0.336)},  
        {'type': 'Смартфон', 'score': np.float64(0.123)},  
        {'type': 'Расходник для печати', 'score': np.float64(0.018)},  
        {'type': 'Пальто пуховое', 'score': np.float64(0.008)},  
        {'type': 'Куртка', 'score': np.float64(0.008)},  
        {'type': 'Шкаф распашной', 'score': np.float64(0.007)},  
        {'type': 'Пальто утепленное', 'score': np.float64(0.007)},  
        {'type': 'Комплект одежды', 'score': np.float64(0.005)},  
        {'type': 'Костюм спортивный', 'score': np.float64(0.004)},  
        {'type': 'Кабель питания', 'score': np.float64(0.004)}]}
```

Результаты:

- 1. Загрузка и предобработка данных.** Данные были загружены из файлов Parquet, очищены от пропусков и дубликатов, а также были созданы новые текстовые признаки путем объединения столбцов 'name' и 'type'.
- 2. Лемматизация.** Была применена лемматизация к текстовым данным для уменьшения количества уникальных слов и улучшения качества признаков. До лемматизации классификатор Catboost выдавал результат с ROC-AUC 78%, после - 80%.
- 3. Векторизация текста.** Был использован TF-IDF для преобразования текстовых данных в числовые признаки, которые могут быть использованы моделями машинного обучения.
- 4. Сравнение моделей.** Был использован PyCaret для сравнения различных моделей машинного обучения и выбора лучших из них.
- 5. Создание ансамбля.** Был создан ансамбль из трех лучших моделей (LightGBM, CatBoost и XGBoost) с использованием голосования.

Результаты:

Лучшей моделью оказался ансамбль, который показал ROC AUC 0.8222.

Что можно улучшить?

- **Подбор гиперпараметров.** Можно попробовать улучшить производительность моделей, подобрав их гиперпараметры с помощью таких техник, как Grid Search или Randomized Search.
- **Инженерия признаков.** Можно попробовать создать новые признаки из существующих данных, чтобы предоставить моделям больше информации. Например, можно извлечь длину названия продукта или количество слов в описании.
- **Использование других моделей.** Можно попробовать использовать другие модели, такие как нейронные сети , чтобы увидеть, могут ли они дать лучшие результаты.
- **Обработка изображений.** Можно попробовать извлечь признаки из изображений продуктов с помощью сверточных нейронных сетей (CNN) типа ResNet, EfficientNet и добавить их к существующим признакам.

**СПАСИБО ЗА
ВНИМАНИЕ!**

