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

«Data Science»

на тему

«Выявление мошенников на торговой площадке Авито»

Слушатель Бунак Алексей Валерьевич

Москва, 2023

Оглавление

Введение ... 3
1. Аналитическая часть ... 5
1.1 Постановка задачи ... 5
1.2 Описание используемых методов ... 12
1.2.1 Линейная регрессия ... 12
1.2.2 Многослойный персептрон (MLP) ... 13
1.2.3 Метод опорных векторов для классификации ... 14
1.2.4 Метод k-ближайших соседей ... 16
1.2.5 Деревья решений ... 16
1.2.6 Случайный лес ... 18
1.2.7 Градиентный бустинг ... 19
1.2.8 Наивный байесовский алгоритм ... 21
1.2.9 Метод стохастического градиента ... 22
1.2.10 Нейронная сеть ... 23
1.3 Разведывательный анализ данных ... 24
1.3.1 Выбор признаков ... 25
1.3.2 Ход решения задачи ... 28
1.3.3 Препроцессинг ... 29
1.3.4 Перекрестная проверка ... 29
1.3.5 Метрики качества моделей ... 29
2. Практическая часть ... 32
2.1 Предобработка данных ... 32
2.2 Тестирование моделей ... 34
2.3 Разработка Веб-приложения ... 36
2.4 Создание репозитория на GitHub ... 37
Заключение ... 38
Планы на будущее ... 39
Библиографический список ... 40

Введение

Тема данной работы – выявление мошенников на торговой площадке Авито.

Авито – самый крупный сайт объявлений в русском сегменте сети Интернет. Там размещают объявления о товарах, услугах, вакансиях, автомобилях и вакансиях как юридические лица, так и физические лица.

Благодаря своей популярности не обходят площадку мимо и огромное число мошенников, пользуясь слабыми механизмами защиты Авито. Авито, как коммерческая контора не заинтересована в том, чтобы избавиться от мошенничества на их сайте. Они могут месяцами не блокировать аккаунты. Но вот если затронуть интересы самого Авито (не заплатить), то там наказание настигает быстро. Поэтому нужен независимый инструмент оценки продавцов на Авито.

Какие виды мошенничества бывают на Авито? Основные:

- Скам – это когда люди берут предоплату и просто пропадают, искать их сложно, почти бесполезно, так как карты на подставных лиц, а сами мошенники за пределами РФ зачастую
- Продажа подделок под видом оригинальной продукции
- Кража личных данных – когда размещают несуществующие вакансии и заставляют заполнять различные анкеты, высылать фото паспорта
- Сбор телефонных номеров – здесь объявление может быть абсолютно любым, конечная цель – ваш звонок, фиксация номера и дальнейшая работа по обману (все виды телефонных мошенников)
- Продажа несуществующих автомобилей и недвижимости – когда при визите вам говорят, что вот тот экземпляр только что продали, но у нас есть «похожий»
- Обман продавцов с фиктивной Авито.доставкой, когда жертва переходит на подставной сайт и вместо получения денег отсылает

мошенникам номер карты и все данные, а порой и производит оплату чего-либо

Есть и другие виды мошенников, но там они не несут прямой угрозы кошельку жертвы, поэтому я их не рассматриваю. Так как я не сотрудничаю с Авито, и могу взять только общедоступную информацию, то буду искать два первых типа мошенников.

Откуда у мошенников профили Авито?

- Могут регистрировать их самостоятельно
- Могут купить готовые профили в Интернете
- Могут обманом взять «в аренду» профили у доверчивых людей (обычно предлагают либо фиксированную оплату в день или процент с продаж)
- Могут обманом заставить выложить объявления (создается вакансия «менеджера по продажам в интернет магазин Авито» и просят разместить объявление, подключить чат-бота, а телефон указать мошенников)

1. Аналитическая часть

1.1 Постановка задачи

В данной работе я буду определять подозрительные профили пользователей Авито (продавцов) на основе общедоступных данных. А для обучения использовать данные, полученные двумя способами:

- Парсинг сайта Авито
- Данные чат-бота, работающего по открытому API Avito

Ранее я разработал приложение для интеграции «Авито – Телеграмм», оно позволяет общаться в Авито через бота в Телеграмм. Авито дает доступ к сообщениям, диалогам, объявлениям. В итоге, была накоплена база данных, которая продолжает обновляться, на данных момент прирост базы данных составляет более 10000 сообщений в сутки, более 1000 объявлений в сутки.

Из того что у меня есть на данный момент (Рисунок 1):

- Более 4.7 миллионов сообщений
- Более 600 тысяч диалогов
- Более 1.3 миллионов объявлений
- Более 688 тысяч пользователей Авито
- Более 18 тысяч отзывов

avito_blacklist	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	1 753	MyISAM	utf8mb4_general_ci	240.5 КиБ
avito_bots	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	3	InnoDB	utf8mb4_general_ci	48.0 КиБ
avito_bots_access	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	4	MyISAM	utf8mb4_general_ci	4.1 КиБ
avito_categories	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	50	MyISAM	utf8mb4_general_ci	4.8 КиБ
avito_chats	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	614 818	MyISAM	utf8mb4_general_ci	311.0 МБ
avito_chats_users	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2 012 696	MyISAM	utf8mb4_general_ci	354.4 МБ
avito_cities	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	2 602	MyISAM	utf8mb4_general_ci	209.2 КиБ
avito_emails	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	3 310	MyISAM	utf8mb4_general_ci	841.0 КиБ
avito_items	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	1 304 019	MyISAM	utf8mb4_general_ci	419.6 МБ
avito_messages	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	~4 774 927	InnoDB	utf8mb4_general_ci	3.2 ГиБ
avito_messages_content	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	275 810	MyISAM	utf8mb4_general_ci	145.4 МБ
avito_messages_errors	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	648	MyISAM	utf8mb4_general_ci	71.6 КиБ
avito_phones	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	88 737	MyISAM	utf8mb4_general_ci	17.1 МБ
avito_profiles	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	537 263	MyISAM	utf8mb4_general_ci	118.9 МБ
avito_profiles_changes	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	0	MyISAM	utf8mb4_general_ci	1.0 КиБ
avito_rating	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	1 989	MyISAM	utf8mb4_general_ci	159.0 КиБ
avito_reviews	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	18 739	MyISAM	utf8mb4_general_ci	9.9 МБ
avito_socials	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	1 645	MyISAM	utf8mb4_general_ci	376.2 КиБ
avito_stats	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	455 249	MyISAM	utf8mb4_general_ci	47.3 МБ
avito_users	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	688 359	MyISAM	utf8mb4_general_ci	127.6 МБ
crm_accounts	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	1 261	MyISAM	utf8mb4_general_ci	149.5 КиБ
crm_balance_history	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	0	MyISAM	utf8mb4_general_ci	1.0 КиБ
crm_chats	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	80	MyISAM	utf8mb4_general_ci	8.3 КиБ
crm_paid_functions	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	12	MyISAM	utf8mb4_general_ci	6.4 КиБ
crm_settings	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	1 363	MyISAM	utf8mb4_general_ci	153.1 КиБ
crm_users	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	5 670	MyISAM	utf8mb4_general_ci	2.4 МБ
tg_messages	Обзор	Структура	Поиск	Вставить	Очистить	Удалить	~116 314	InnoDB	utf8mb4_general_ci	33.3 МБ
28 таблиц	Всего						~10 907 321	InnoDB	utf8mb4_general_ci	4.7 ГиБ

Рисунок 1 – структура базы данных чат-бота (источник данных для обучения)

Так как я имею прямой доступ к данным для обучения, я проведу предварительную подготовку датасета на этапе выборки из таблиц базы данных. При парсинге возникали пробелы, например, не всегда можно получить город и категорию объявления по API, но у меня есть ссылка на объявления, а в ней всегда указан город и категория, поэтому в базе данных есть 2 вспомогательных таблицы, в которых указаны соответствия (Рисунок 2 и Рисунок 3):

category_id	name ▲ 1	url
9	Автомобили	avtomobili
92	Аквариум	akvarium
32	Аудио и видео	audio_i_video
33	Билеты и путешествия	bilety_i_puteshestviya

Рисунок 2 – пример из таблицы avito_categories

id	url	name
1	serpuhov	Серпухов
2	protvino	Протвино
3	rostov-na-donu	Ростов-на-Дону
4	moskva	Москва
5	sergiev_posad	Сергиев Посад
6	sverdlovskaya_oblast_artemovskiy	Артемовский
7	ramenskoe	Раменское

Рисунок 3 – пример из таблицы avito_cities

Для обучения я взял выборку из 52812 строк из таблицы avito_items (объявления), сделав JOIN таблиц с рейтингом, отзывами, а также с таблицей пользователей, рассмотрим структуру датасета (Рисунок 4):

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52812 entries, 0 to 52811
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   item_id      52812 non-null  int64
 1   item_name    52812 non-null  object
 2   price        52812 non-null  int64
 3   category_id  52812 non-null  int64
 4   profile      52812 non-null  object
 5   location     52812 non-null  object
 6   score        52812 non-null  float64
 7   reviews      52812 non-null  int64
 8   counted      52812 non-null  int64
 9   status       52812 non-null  int64
dtypes: float64(1), int64(6), object(3)
memory usage: 4.0+ MB
```

Рисунок 4 – структура датасета для обучения

На этапе подготовки я заменил отсутствующие значения цены (price) и рейтинга (score), а также отзывов (reviews), учтенных отзывов (counted) на 0. В

итоге получил item_name (название объявления), category_id (категория), location (город), profile (продавец) и status (о нем подробнее).

username	balance	points	status
nongratttaa	0.00	0.00	fake
rabota0903	0.00	0.00	fake
its_polya	0.00	0.00	OK
blackhaaawk	0.00	0.00	scam
yes_kattya	0.00	0.00	scam
	0.00	0.00	OK

Рисунок 5 – пример таблицы crm_accounts

В чат-боте у моих пользователей есть 3 статуса (Рисунок 5): scam – когда людей кидают на деньги, fake – продавцы подделок под видом оригиналов и OK – все хорошо. Первые два статуса приравнены к 1, а OK приравнен к 0. Статус выставлялся вручную после анализа переписки (началось с того, что Авито заблокировало мое приложение, за скам и я месяц бился, добиваясь разблокировки), тогда и стало понятно, что мне придется выявлять мошенников внутри своего приложения. Собственно, тогда пришлось писать парсер, работающий на слова «мошенники», «верните деньги» и прочие слова, далее смотреть контекст диалога, ставить пометки владельцу профиля (так не бывает, что с одного профиля он занимается мошенничеством, а на других - нет). Логи аккаунтов мошенников ведутся отдельно и проводится анализ их деятельности для выявления новых схем мошенничества.

Я буду пытаться предсказать столбец status. Никаких выбросов и прочих аномальных данных у меня нет, так как собирались только реальные данные. На всякий случай я проверил, что никаких пропусков нет (Рисунок 6).

```
item_id        0
item_name      0
price          0
category_id    0
profile        0
location       0
score          0
reviews        0
counted        0
status         0
dtype: int64
```

Рисунок 6 – пропущенные значения в датасете

Анализ датасета на уникальные значения дал следующую картину (Рисунок 7):

```
item_id        52812
item_name      25865
price           2301
category_id       36
profile          611
location         339
score             23
reviews           47
counted           47
status             2
dtype: int64
```

Рисунок 7 – уникальные значения

Основное что меня здесь интересует – количество уникальных профилей, ведь их статус я и пытаюсь предсказать, а не объявлений.

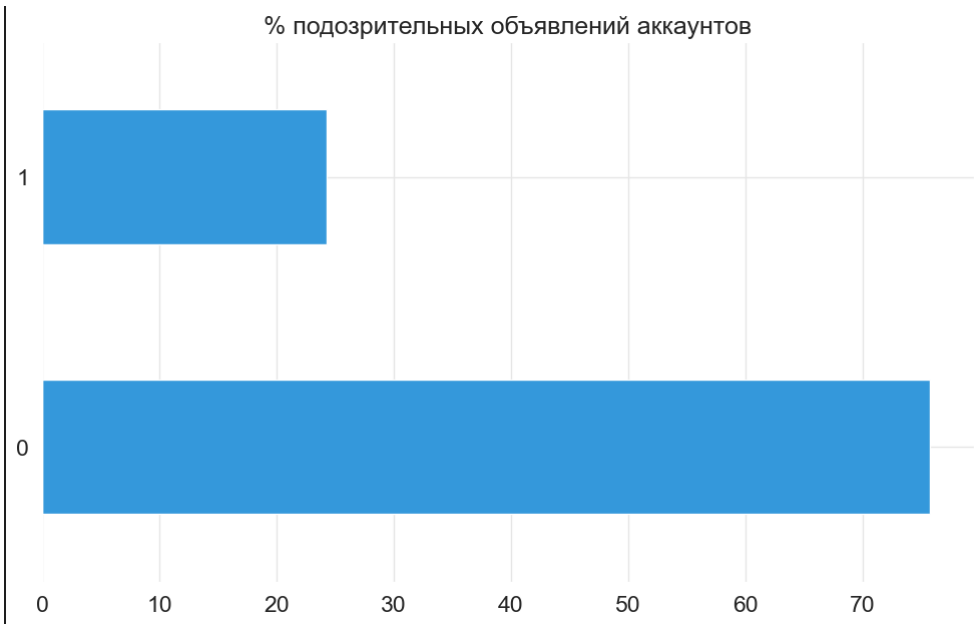

Рисунок 8 – подозрительные объявления

Если посмотреть на график, представленный на Рисунке 8, то можно увидеть, что примерно 24 процента объявлений от мошенников. Но если сгруппировать их по профилям, то как показывает график на Рисунке 9, все кардинально меняется и уже 70 процентов профилей являются мошенниками.

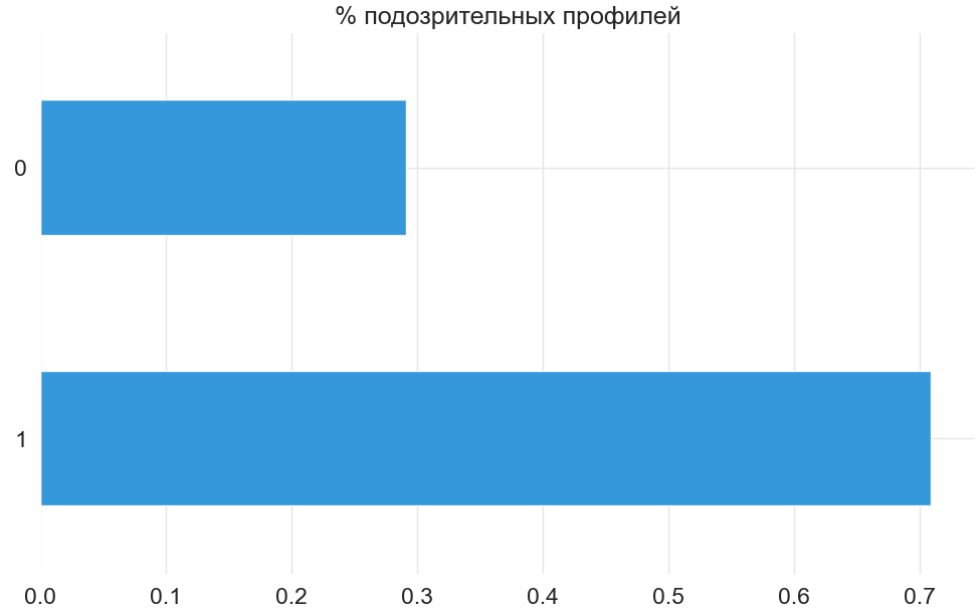

Рисунок 9 – подозрительные профили

Я решил посмотреть, что и где продают продавцы из сделанной мной выборки, поэтому взял самые часто встречающиеся слова из названий объявлений и города размещения. Результат на Рисунке 10 и Рисунке 11.

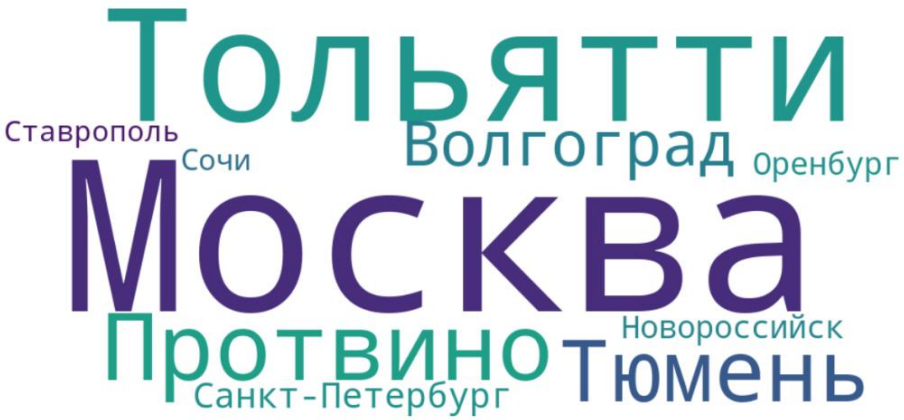

Рисунок 10 – города размещения объявлений

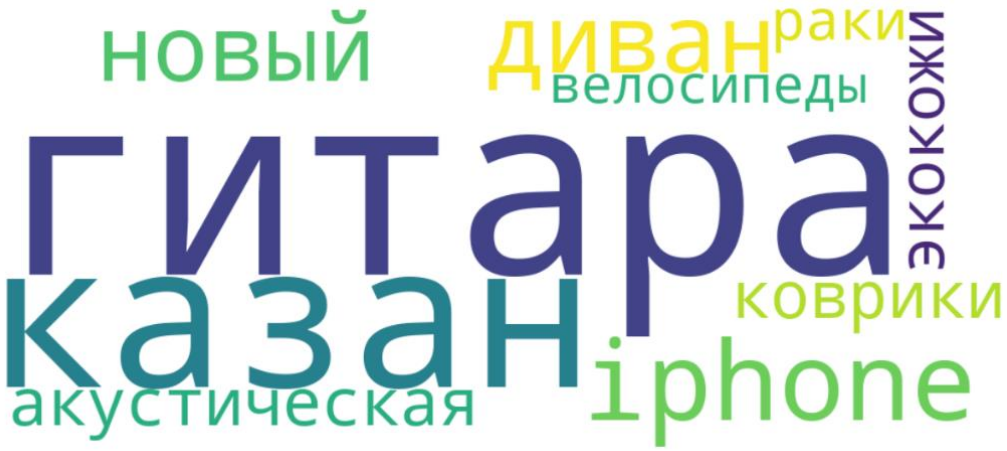

Рисунок 11 – популярные слова в названиях

Напоследок я решил провести попарное сравнение признаков, представленных в виде графиков на Рисунке 12

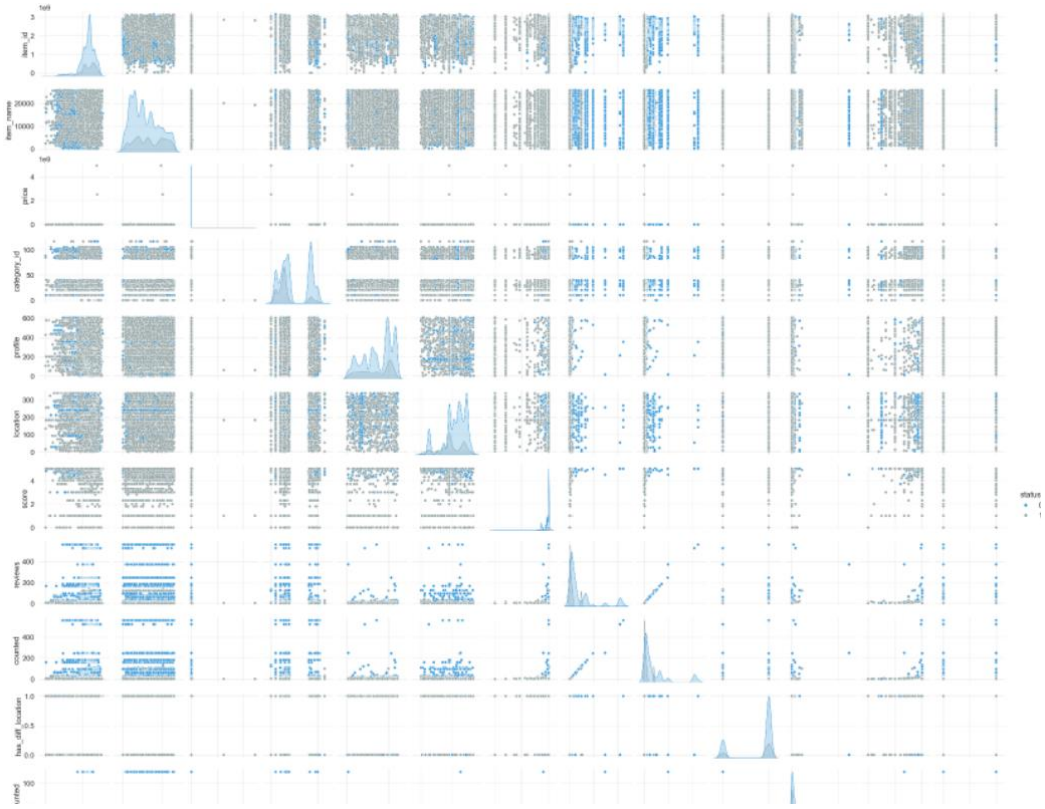

Рисунок 12 – попарное сравнение признаков

1.2 Описание используемых методов

Задача классификации — задача, в которой имеется множество объектов (ситуаций), разделённых, некоторым образом, на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать (см. ниже) произвольный объект из исходного множества.

В настоящее время разработано много методов классификации, постараюсь использовать самые популярные из них.

1.2.1 Линейная регрессия

Простая линейная регрессия имеет место, если рассматривается зависимость между одной входной и одной выходной переменными. Для этого определяется уравнение регрессии (1) и строится соответствующая прямая, известная как линия регрессии.

y=ax+b (1)

Коэффициенты a и b, называемые также параметрами модели, определяются таким образом, чтобы сумма квадратов отклонений точек, соответствующих реальным наблюдениям данных, от линии регрессии была бы минимальной. Коэффициенты обычно оцениваются методом наименьших квадратов.

Если ищется зависимость между несколькими входными и одной выходной переменными, то имеет место множественная линейная регрессия. Соответствующее уравнение имеет вид (2).

Y=b0+b1∗x1+b2∗ x2+⋯+bn∗xn, (2)

где n - число входных переменных.

Очевидно, что в данном случае модель будет описываться не прямой, а гиперплоскостью. Коэффициенты уравнения множественной линейной регрессии подбираются так, чтобы минимизировать сумму квадратов отклонения реальных точек данных от этой гиперплоскости.

Линейная регрессия — первый тщательно изученный метод регрессионного анализа. Его главное достоинство — простота. Такую модель можно построить и рассчитать даже без мощных вычислительных средств. Простота является и главным недостатком этого метода. Тем не менее, именно с линейной регрессии целесообразно начать подбор подходящей модели.

На языке python линейная регрессия реализована в sklearn.linear_model.LinearRegression.

1.2.2 Многослойный персептрон (MLP)

Многослойный персептрон (MLP) — это алгоритм обучения с учителем, который изучает функцию $f(\cdot): R^m \rightarrow R^o$ обучением на наборе данных, где m — количество измерений для ввода и o- количество размеров для вывода. Учитывая набор функций $X = x_1, x_2, \ldots, x_m$ и цель y, он может изучить аппроксиматор нелинейной функции для классификации или регрессии. Он отличается от логистической регрессии тем, что между входным и выходным слоями может быть

один или несколько нелинейных слоев, называемых скрытыми слоями. На рисунке 1 показан MLP с одним скрытым слоем со скалярным выходом.

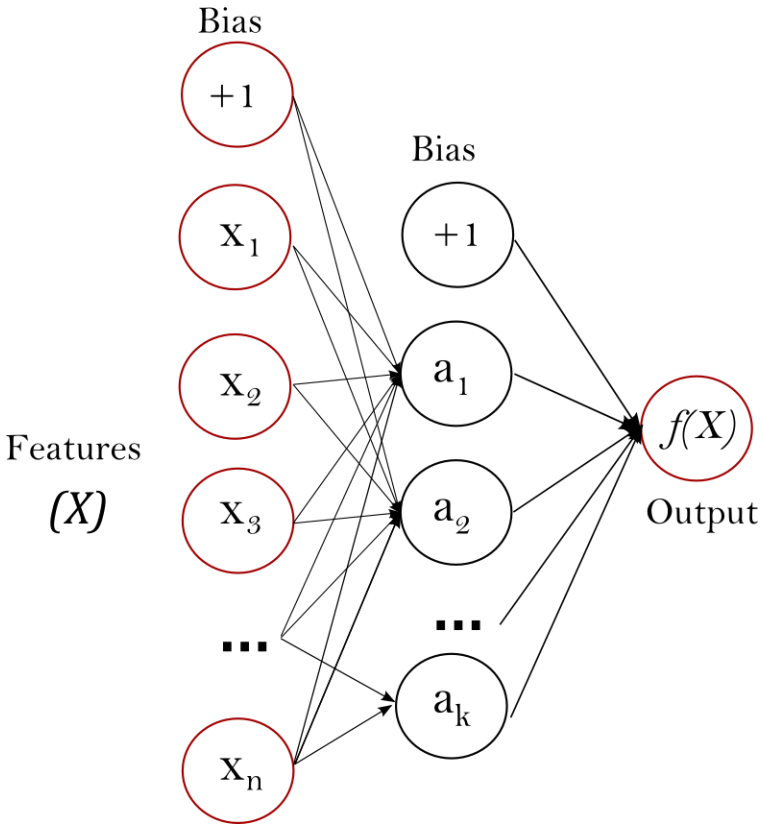

Рисунок 1: Один скрытый слой MLP.

Преимущества многослойного перцептрона:

- Возможность изучать нелинейные модели.
- Возможность изучения моделей в режиме реального времени (онлайн-обучение) с использованием partial_fit.

К недостаткам многослойного персептрона (MLP) можно отнести:

- MLP со скрытыми слоями имеют невыпуклую функцию потерь, когда существует более одного локального минимума. Поэтому разные инициализации случайных весов могут привести к разной точности проверки.
- MLP требует настройки ряда гиперпараметров, таких как количество скрытых нейронов, слоев и итераций.
- MLP чувствителен к масштабированию функций.

Класс MLPClassifier реализует алгоритм многослойного перцептрона (MLP), который обучается с использованием обратного распространения.

1.2.3 Метод опорных векторов для классификации

Метод опорных векторов (support vector machine, SVM) — один из наиболее популярных методов машинного обучения. Он создает гиперплоскость или

набор гиперплоскостей в многомерном пространстве, которые могут быть использованы для решения задач классификации и регрессии.

Чаще всего он применяется в постановке бинарной классификации. Основная идея заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Интуитивно, хорошее разделение достигается за счет гиперплоскости, которая имеет самое большое расстояние до ближайшей точки обучающей выборке любого класса. Максимально близкие объекты разных классов определяют опорные вектора. Если в исходном пространстве объекты линейно неразделимы, то выполняется переход в пространство большей размерности.

Решается задача оптимизации. Для вычислений используется ядерная функция, получающая на вход два вектора и возвращающая меру сходства между ними:

- линейная;
- полиномиальная;
- гауссовская (rbf).

Эффективность метода опорных векторов зависит от выбора ядра, параметров ядра и параметра C для регуляризации. Преимущество метода — его хорошая изученность.

Недостатки:

- чувствительность к выбросам;
- отсутствие интерпретируемости.

SVC (Support Vector Classification) — это вариант SVM, применяемый для задач классификации. Он использует метод опорных векторов для построения гиперплоскости, которая разделяет данные на классы. SVC ищет оптимальную гиперплоскость таким образом, чтобы максимизировать зазор между классами, то есть расстояние между гиперплоскостью и ближайшими точками каждого

класса (опорными векторами) было максимальным. В python реализацию SVC можно найти в sklearn.svm.SVC.

1.2.4 Метод k-ближайших соседей

Еще один метод классификации, который адаптирован для регрессии - метод k-ближайших соседей (k Nearest Neighbors). На интуитивном уровне суть метода проста: посмотри на соседей вокруг, какие из них преобладают, таковым ты и являешься.

В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны. Для реализации метода необходима метрика расстояния между объектами. Используется, например, эвклидово расстояние для количественных признаков или расстояние Хэмминга для категориальных.

Этот метод — пример непараметрической регрессии. Он реализован в sklearn.neighbors.KneighborsRegressor.

1.2.5 Деревья решений

Деревья решений (Decision Trees) - еще один непараметрический метод, применяемый и для классификации, и для регрессии. Деревья решений используются в самых разных областях человеческой деятельности и представляют собой иерархические древовидные структуры, состоящие из правил вида «Если ..., то ...».

Решающие правила автоматически генерируются в процессе обучения на обучающем множестве путем обобщения обучающих примеров. Поэтому их называют индуктивными правилами, а сам процесс обучения — индукцией деревьев решений. Дерево состоит из элементов двух типов: узлов (node) и листьев (leaf). В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу. В результате проверки множество примеров, попавших в узел, разбивается на два подмножества: удовлетворяющие

правилу и не удовлетворяющие ему. Затем к каждому подмножеству вновь применяется правило и процедура рекурсивно повторяется пока не будет достигнуто некоторое условие остановки алгоритма. В последнем узле проверка и разбиение не производится, и он объявляется листом.

В листе содержится не правило, а подмножество объектов, удовлетворяющих всем правилам ветви, которая заканчивается данным листом. Для классификации — это класс, ассоциируемый с узлом, а для регрессии — соответствующий листу интервал целевой переменной.

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут, по которому это будет сделано. Общее правило для классификации можно сформулировать так: выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, а количество объектов из других классов в каждом из этих множеств было как можно меньше. Для этого были выбраны различные критерии, например, теоретико-информационный и статистический.

Для регрессии критерием является дисперсия вокруг среднего. Минимизируя дисперсию вокруг среднего, мы ищем признаки, разбивающие выборку таким образом, что значения целевого признака в каждом листе примерно равны.

Огромное преимущество деревьев решений в том, что они легко интерпретируемы, понятны человеку. Они могут использоваться для извлечения правил на естественном языке. Еще преимущества — высокая точность работы, нетребовательность к подготовке данных.

Недостаток деревьев решений - склонность переобучаться. Переобучение в случае дерева решений — это точное распознавание примеров, участвующих в обучении и полная несостоятельность на новых данных. В худшем случае, дерево будет большой глубины и сложной структуры, а в каждом листе будет только один объект. Для решения этой проблемы используют разные критерии

остановки алгоритма. Деревья решений реализованы в sklearn.tree.DecisionTreeRegressor.

1.2.6 Случайный лес

Случайный лес (RandomForest) — представитель ансамблевых методов. Если точность дерева решений оказалось недостаточной, мы можем множество моделей собрать в коллектив. Формула итогового решателя (3) — это усреднение предсказаний отдельных деревьев.

$$a(x) = \frac{1}{N} \sum_{i=1}^{N} b_i(x) \quad (3),$$

где

- N – количество деревьев;
- i – счетчик для деревьев;
- b – решающее дерево;
- x – сгенерированная нами на основе данных выборка.

Для определения входных данных каждому дереву используется метод случайных подпространств. Базовые алгоритмы обучаются на различных подмножествах признаков, которые выделяются случайным образом.

Преимущества случайного леса:

- высокая точность предсказания;
- редко переобучается;
- практически не чувствителен к выбросам в данных;
- одинаково хорошо обрабатывает как непрерывные, так и дискретные
- признаки, данные с большим числом признаков;
- высокая параллелизуемость и масштабируемость.

Из недостатков можно отметить, что его построение занимает больше времени. Так же теряется интерпретируемость.

Метод реализован в sklearn.ensemble.RandomForestRegressor.

1.2.7 Градиентный бустинг

XGBoost — это контролируемый метод машинного обучения для классификации и регрессии. XGBoost — это сокращение от "экстремального повышения градиента". Этот метод основывается на дереве решений и работает лучше, чем другие методы, такие как произвольное дерево и градиентное повышение. Он лучше работает со сложными большими наборами данных, используя различные методы оптимизации.

Для соответствия набору обучающих данных с использованием XGBoost, делается первичный прогноз. Невязки вычисляются на основе прогнозированного и наблюдаемого значений. Дерево решений создается с невязками на основе оценки подобия для невязок. Вычисляется подобие данных в листе, а также усиление в подобии при последующем разбиении. Усиления сравниваются, чтобы определить объект и порог для узла. Выходное значение для каждого листа также рассчитывается с использованием невязок. Для классификации значения обычно рассчитываются с использованием перечня шансов и вероятностей. Результат дерева становится новой невязкой для набора данных, который используется для построения другого дерева. Этот процесс повторяется до тех пор, пока невязки не перестанут уменьшаться или определенное количество раз. Каждое последующее дерево учится у предыдущих деревьев, и ему не присваивается равный вес, в отличие от того, как работает Случайный лес.

Чтобы использовать эту модель для прогнозирования, выходные данные каждого дерева, умноженные на скорость обучения, добавляются к начальному прогнозу, чтобы получить окончательное значение или классификацию.

XGBoost использует следующие параметры и методы для оптимизации алгоритма и обеспечения лучших результатов и производительности:

Регуляризация - параметр регуляризации (лямбда) используется при расчете показателей подобия, чтобы снизить чувствительность к отдельным данным и избежать переобучения.

Сокращение — параметр сложности дерева (гамма) выбирается для сравнения выигрышей. Ветвь, где коэффициент усиления меньше значения гаммы, удаляется. Это предотвращает избыточную подгонку за счет обрезки ненужных ветвей и уменьшения глубины деревьев.

Скетч взвешенных квантилей — вместо проверки каждого возможного значения в качестве порога для разделения данных используются только взвешенные квантили. Выбор квантилей осуществляется с помощью алгоритма скетча, который оценивает распределение по нескольким системам в сети.

Параллельное обучение — этот метод делит данные на блоки, которые можно использовать параллельно для создания деревьев или других вычислений.

Поиск разделения с учетом разреженности — XGBoost обрабатывает разреженность данных, пробуя оба направления в разбиении и находя направление по умолчанию, вычисляя усиление.

Доступ с учетом кэша — этот метод использует кэш-память системы для расчета показателей сходства и выходных значений. Кэш-память является более быстрой памятью для доступа по сравнению с основной памятью и повышает общую производительность модели.

Блоки для вычислений вне ядра - этот метод работает с большими наборами данных, которые не помещаются в кэш или основную память и должны храниться на жестких дисках. Этот набор данных разбивается на блоки и сжимается. Несжатые данные в основной памяти работают быстрее, чем при

чтении с жесткого диска. Другой метод, называемый сегментированием, используется, когда данные должны храниться на нескольких жестких дисках.

1.2.8 Наивный байесовский алгоритм

Наивный байесовский классификатор (Naive Bayes classifier) – это очень популярный в машинном обучении алгоритм, который в основном используется для получения базовой точности набора данных.

Теорема Байеса позволяет рассчитать апостериорную вероятность P(A | B) на основе P(A), P(B) и P(B | A).

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Где:

- P(A | B) – апостериорная вероятность (что A из B истинно)
- P(A) – априорная вероятность (независимая вероятность A)
- P(B | A) – вероятность данного значения признака при данном классе. (что B из A истинно)
- P(B) – априорная вероятность при значении нашего признака. (независимая вероятность B)

Плюсы

- Алгоритм легко и быстро предсказывает класс тестового набора данных. Он также хорошо справляется с многоклассовым прогнозированием.
- Производительность наивного байесовского классификатора лучше, чем у других простых алгоритмов, таких как логистическая регрессия. Более того, вам требуется меньше обучающих данных.
- Он хорошо работает с категориальными признаками (по сравнению с числовыми). Для числовых признаков предполагается нормальное распределение, что может быть серьезным допущением в точности нашего алгоритма.

Минусы

- Если переменная имеет категорию (в тестовом наборе данных), которая не наблюдалась в обучающем наборе данных, то модель присвоит 0 (нулевую) вероятность и не сможет сделать предсказание.

Это часто называют нулевой частотой. Чтобы решить эту проблему, мы можем использовать технику сглаживания. Один из самых простых методов сглаживания называется оценкой Лапласа.

- Значения спрогнозированных вероятностей, возвращенные методом *predict_proba*, не всегда являются достаточно точными.
- Ограничением данного алгоритма является предположение о независимости признаков. Однако в реальных задачах полностью независимые признаки встречаются крайне редко.

1.2.9 Метод стохастического градиента

Градиентные методы — это широкий класс оптимизационных алгоритмов, используемых не только в машинном обучении. Здесь градиентный подход будет рассмотрен в качестве способа подбора вектора синаптических весов w в линейном классификаторе. Пусть $y^*: X \to Y$ - целевая зависимость, известная только на объектах обучающей выборки: $X^l = (x_i, y_i)_{i=1}^l$, $y_i = y^*(x_i)$.

Возможно 2 основных подхода к реализации градиентного спуска:

Пакетный (batch), когда на каждой итерации обучающая выборка просматривается целиком, и только после этого изменяется w. Это требует больших вычислительных затрат.

Стохастический (stochastic/online), когда на каждой итерации алгоритма из обучающей выборки каким-то (случайным) образом выбирается только один объект. Таким образом вектор w настраивается на каждый вновь выбираемый объект.

Преимущества SG

- Метод приспособлен для динамического (online) обучения, когда обучающие объекты поступают потоком, и надо быстро обновлять вектор w.
- Алгоритм способен обучаться на избыточно больших выборках за счёт того, что случайной подвыборки может хватить для обучения.

- Возможны различные стратегии обучения. Если выборка избыточно большая, или обучение происходит динамически, то допустимо не сохранять обучающие объекты. Если выборка маленькая, то можно повторно предъявлять для обучения одни и те же объекты.

Недостатки SG

- Алгоритм может не сходиться или сходиться слишком медленно
- Как правило, функционал Q многоэкстремален и процесс градиентного спуска может "застрять" в одном из локальных минимумов. Для борьбы с этим используют технику встряхивания коэффициентов (jog of weights).
- При большой размерности пространства признаков n и/или малой длине выборки l возможно переобучение, то есть классификация становится неустойчивой, и вероятность ошибки увеличивается. При этом сильно возрастает норма вектора весов.
- Если функция активации имеет горизонтальные асимптоты, то процесс может попасть в состояние "паралича". При больших значениях скалярного произведения $<w,x_i>$ значение φ' становится близким к нулю и вектор w перестаёт существенно изменяться.

1.2.10 Нейронная сеть

Нейронная сеть — это последовательность нейронов, соединенных между собой связями. Структура нейронной сети пришла в мир программирования из биологии. Вычислительная единица нейронной сети — нейрон или персептрон.

У каждого нейрона есть определённое количество входов, куда поступают сигналы, которые суммируются с учётом значимости (веса) каждого входа. Смещение – это дополнительный вход для нейрона, который всегда равен 1 и, следовательно, имеет собственный вес соединения. Так же у нейрона есть функция активации, которая определяет выходное значение нейрона. Она используется для того, чтобы ввести нелинейность в нейронную сеть. Примеры активационных функций: relu, сигмоида. У полносвязной нейросети выход

каждого нейрона подается на вход всем нейронам следующего слоя. У нейросети имеется:

- входной слой — его размер соответствует входным параметрам;
- скрытые слои — их количество и размерность определяем специалист;
- выходной слой — его размер соответствует выходным параметрам.

Прямое распространение – это процесс передачи входных значений в нейронную сеть и получения выходных данных, которые называются прогнозируемым значением.

Прогнозируемое значение сравниваем с фактическим с помощью функции потери. В методе обратного распространения ошибки градиенты (производные значений ошибок) вычисляются по значениям весов в направлении, обратном прямому распространению сигналов. Значение градиента вычитают из значения веса, чтобы уменьшить значение ошибки. Таким образом происходит процесс обучения. Обновляются веса каждого соединения, чтобы функция потерь минимизировалась.

Для обновления весов в модели используются различные оптимизаторы. Количество эпох показывает, сколько раз выполнялся проход для всех примеров обучения. Нейронные сети применяются для решения задач регрессии, классификации, распознавания образов и речи, компьютерного зрения и других. На настоящий момент это самый мощный, гибкий и широко применяемый инструмент в машинном обучении.

1.3 Разведывательный анализ данных

Для анализа составим матрицу корреляции признаков (Рисунок 13)

Особой связи не видно, количество отзывов и учтенных отзывов близки и большинства пользователей. Это не новость. Буду создавать свои признаки, основанные на личном опыте.

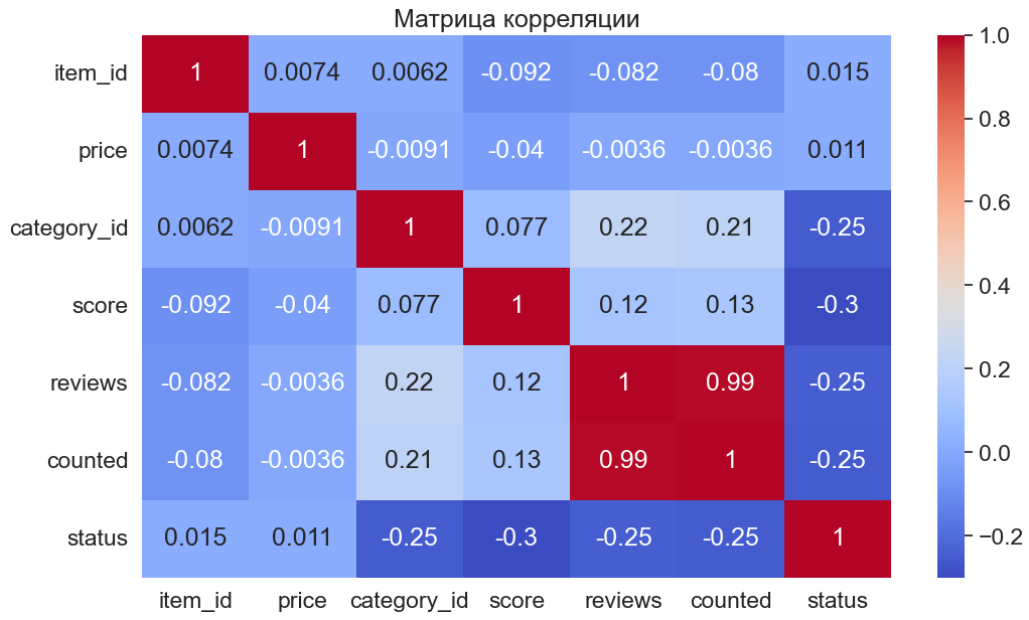

Рисунок 13 – матрица корреляции признаков исходного датасета

1.3.1 Выбор признаков

Как показала практика, часто получается так, что в профиле выкладывались объявления о продаже каких-то вещей в Саратове, к примеру, а мошенническое объявление выставлено в Москве для большего охвата. Поэтому я ввел параметр has_diff_location, который принимает значение 1, если в профиле объявления выставлены в разных городах (Рисунок 14):

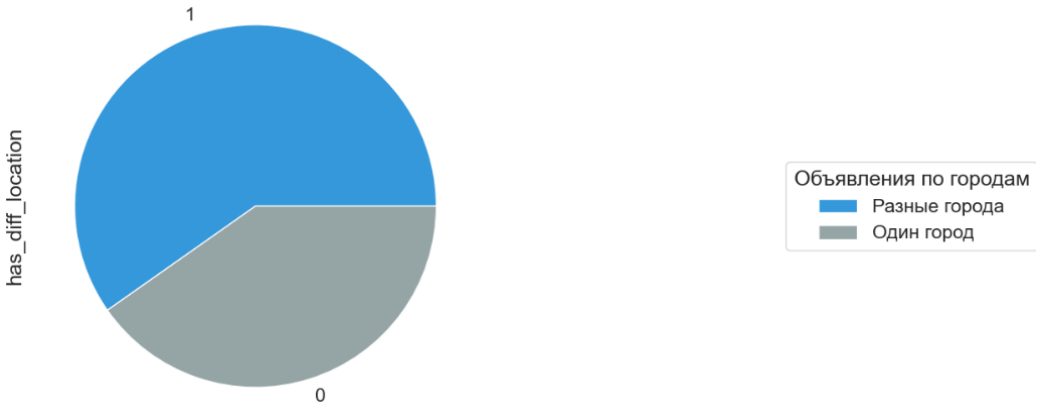

Рисунок 14 – объявления по городам

Далее я проверю свою теорию, что у профилей мошенников чаще встречаются объявления в разных городах, что подтверждает Рисунок 15:

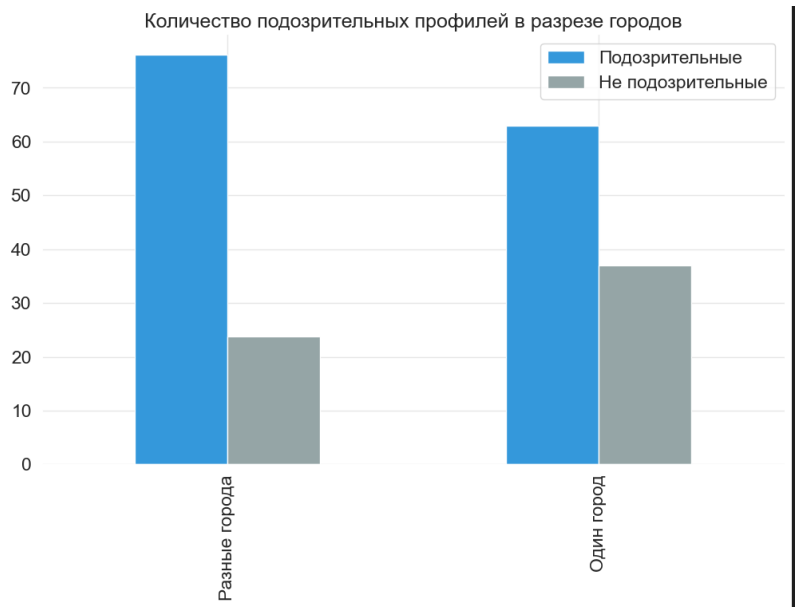

Рисунок 15 – распределение профилей в разрезе городов

Я предположил, что рейтинг напрямую может показывать подозрительный профиль. Поэтому игнорировать данный параметр нельзя. Но и полагаться на него полностью также не выйдет. Мошенники легко и непринужденно через Техническую Поддержку Авито удаляют негативные отзывы, а также покупают фальшивые отзывы на свои профили:

Рисунок 16 – зависимость статуса от рейтинга

Как видно из графика на Рисунке 16, мошеннические профили зачастую имеют рейтинг ниже 4 звезд. Еще один немаловажный фактор – неучтенные в

рейтинге отзывы, так называемые «без оценки», часть отзывов переносится туда, и там крайне редко оказываются положительные отзывы. Так что я смело ставлю фактор наличия таких отзывов в негативные факторы. Но просто наличие таких отзывов ничего не дает, я буду рассматривать процентное соотношение неучтенных отзывов к общему количеству (Рисунок 17):

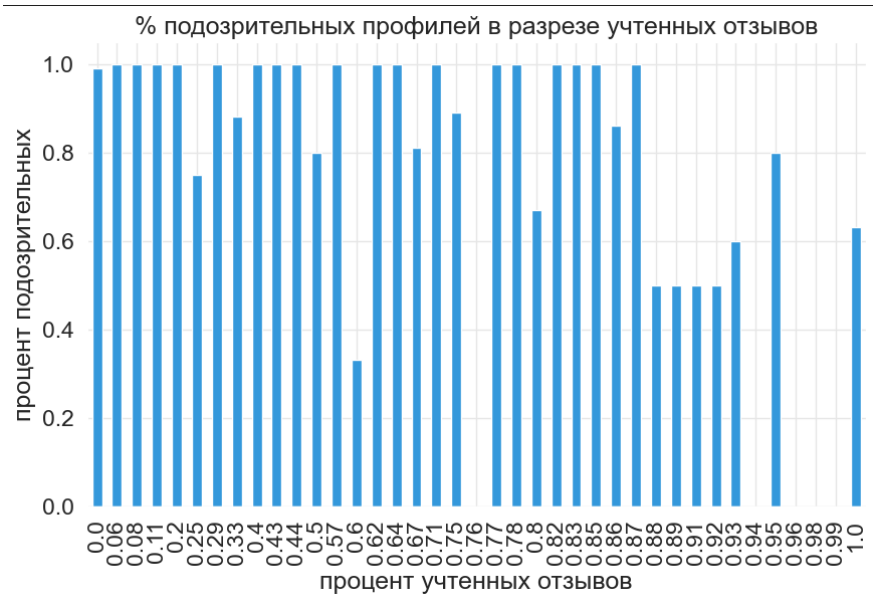

Рисунок 17 – зависимость статуса от учтенных отзывов

Как показывает график, если в профиле менее 90 процентов отзывов учтено, то это крайне негативный фактор.

Также, на основе опыта могу сказать, что, если продают «детскую кухню IKEA», да еще и по цене ниже 5000 руб – это мошенники. Разные модели AirPods с припиской «оригинал», по цене от 3000 до 7000 руб – продажа подделки (мошенники). Как было сказано ранее, я анализирую действия мошенников, которые пользуются моим приложением и выявляю новые объявления, новые схемы. Поэтому был введен параметр bad_item, в котором модератор, в моем лице, вручную вносит связки СТОП-СЛОВ и цены. Так называемое «плохое объявление», как показала практика, это почти идеальный триггер для определения подозрительных аккаунтов, но все же не самодостаточный:

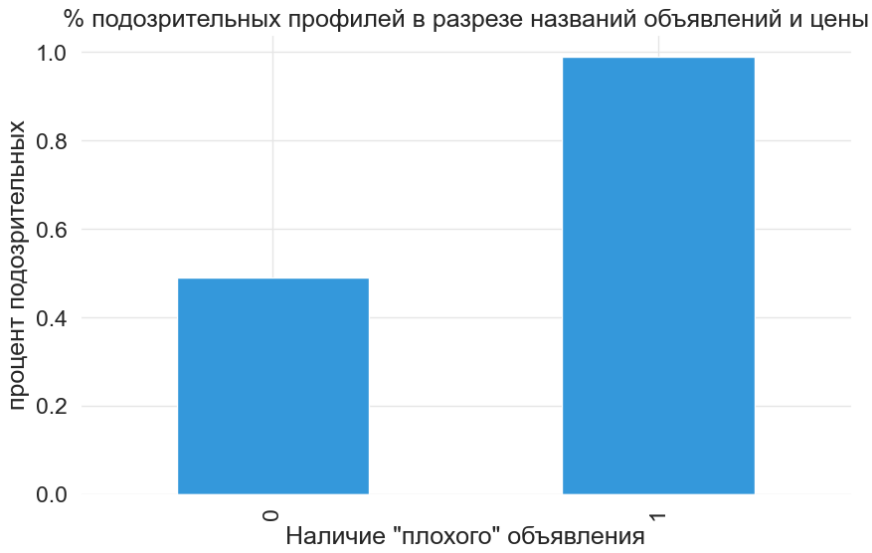

Рисунок 18 – зависимость статуса от наличия «плохого» объявления

На графике, представленном на Рисунке 18 видно, что наличие «плохого» объявления дает почти полную уверенность в том, что перед нами мошенник. Этот параметр сильно зависит от человеческого фактора, поэтому подозрительные профили присутствуют и в выборке, где нет «плохих» объявлений. Я просто не внес все возможные комбинации. Но по объективным причинам все внести и не получится. Профиль может быть на «прогреве» и у него пока нет таких объявлений, а первое время на нем крутятся вполне безобидные товары. Тогда следует выждать время.

1.3.2 Ход решения задачи

Ход решения каждой из задач и построения оптимальной модели будет следующим:

- разделить данные на тренировочную и тестовую выборки. На тестирование оставлю 20% данных;
- выполнить препроцессинг, то есть подготовку исходных данных;
- попробовать разные модели обучения
- сравнить разные модели обучения
- сохранить модели обучения и в дальнейшем использовать для переобучения

1.3.3 Препроцессинг

Цель препроцессинга, или предварительной обработки данных — обеспечить корректную работу моделей. Его необходимо выполнять после разделения на тренировочную и тестовую выборку, как будто мы не знаем параметров тестовой выборки.

Препроцессинг для категориальных и количественных признаков выполняется по-разному. Категориальный признак один – Статус, его мы и предсказываем. Но я ввел три новых признака, и они уже нормированы, bad_item и has_diff_location, a count_procent принимает значения от 0 до 1. Profile я закодирую с помощью LabelEncoder.

1.3.4 Перекрестная проверка

Для обеспечения статистической устойчивости метрик модели используем перекрестную проверку или кросс-валидацию. Чтобы ее реализовать, выборка разбивается необходимое количество раз на тестовую и валидационную. Модель обучается на тестовой выборке, затем выполняется расчет метрик качества на валидационной. В качестве результата мы получаем средние метрики качества для всех валидационных выборок. Перекрестную проверку реализует функция cross_validate из sklearn.

1.3.5 Метрики качества моделей

Существует множество различных метрик качества, применимых для классификации. В этой работе я использую: Точность (Accuracy)

- Точность (Accuracy): Это доля правильно классифицированных примеров относительно общего числа примеров. Она вычисляется как отношение числа верно предсказанных примеров к общему числу примеров в тестовом наборе. Точность может быть полезной метрикой, когда классы в данных сбалансированы, то есть количество примеров в каждом классе

примерно одинаково. Однако, она может давать неправильную оценку качества модели, когда классы несбалансированы.

- Полнота (Recall или True Positive Rate): Это доля верно предсказанных положительных примеров относительно общего числа положительных примеров. Она вычисляется как отношение числа верно предсказанных положительных примеров к общему числу положительных примеров в тестовом наборе. Полнота может быть важной метрикой, когда важно минимизировать ложноотрицательные результаты, то есть случаи, когда модель предсказывает неправильно отрицательные примеры как положительные.
- Точность (Precision): Это доля верно предсказанных положительных примеров относительно общего числа положительных примеров и ложно положительных примеров. Она вычисляется как отношение числа верно предсказанных положительных примеров к сумме числа верно предсказанных положительных примеров и ложно положительных примеров. Точность может быть важной метрикой, когда важно минимизировать ложноположительные результаты, то есть случаи, когда модель предсказывает неправильно положительные примеры.
- F1-мера (F1-score): Это гармоническое среднее между точностью и полнотой. Она вычисляется как отношение удвоенного произведения точности и полноты к их сумме. F1-мера является более сбалансированной метрикой, которая учитывает как точность, так и полноту.
- Матрица ошибок (Confusion Matrix): Это таблица, которая показывает количество верно и неверно классифицированных примеров для каждого класса. Она может быть полезной для детального анализа ошибок модели и оценки ее производительности в разных классах. Матрица ошибок включает в себя четыре значения: true positives (TP) - число верно предсказанных положительных примеров, true negatives (TN) - число верно предсказанных отрицательных примеров, false positives (FP) - число неверно предсказанных положительных примеров и false negatives (FN) - число неверно предсказанных отрицательных примеров.

- Кривая ROC (Receiver Operating Characteristic): Это графическое представление производительности классификационной модели, которое показывает соотношение между чувствительностью (True Positive Rate) и специфичностью (1 - False Positive Rate) на различных уровнях порога классификации. Кривая ROC может быть полезной метрикой, особенно когда классы несбалансированы.
- Площадь под кривой ROC (AUC-ROC): Это числовая метрика, которая представляет собой площадь под кривой ROC. Она является одним из показателей производительности модели, где значение ближе к 1 указывает на лучшую производительность, а значение ближе к 0.5 указывает на случайное предсказание.
- Матрица классификации (Classification Report): Это отчет, который предоставляет информацию о точности, полноте, F1-мере и других метриках для каждого класса, а также их средневзвешенные значения. Он предоставляет детальную информацию о производительности модели для каждого класса и может быть полезным инструментом для анализа результатов классификации.

2. Практическая часть

2.1 Предобработка данных

После добавления новых признаков я получил следующий датасет (Рисунок 19)

```
RangeIndex: 52812 entries, 0 to 52811
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   item_id            52812 non-null  int64
 1   item_name          52812 non-null  object
 2   price              52812 non-null  int64
 3   category_id        52812 non-null  int64
 4   profile            52812 non-null  object
 5   location           52812 non-null  object
 6   score              52812 non-null  float64
 7   reviews            52812 non-null  int64
 8   counted            52812 non-null  int64
 9   status             52812 non-null  int64
 10  has_diff_location  52812 non-null  int64
 11  uncounted          52812 non-null  int64
 12  count_procent      52812 non-null  float64
 13  bad_item           52812 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 5.6+ MB
```

Рисунок 19 – обработанный датасет

Я получил новые категориальные признаки (Рисунок 20).

```
item_id             52812
item_name           25865
price                2301
category_id            36
profile               611
location              339
score                  23
reviews                47
counted                47
status                  2
has_diff_location       2
uncounted              14
count_procent          39
bad_item                2
dtype: int64
```

Рисунок 20 – уникальные значения

Для обучения надо избавиться от нечисловых признаков, что я и делаю, закодировав значения (Рисунок 21):

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52812 entries, 0 to 52811
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   item_id            52812 non-null  int64
 1   item_name          52812 non-null  int32
 2   price              52812 non-null  int64
 3   category_id        52812 non-null  int64
 4   profile            52812 non-null  int32
 5   location           52812 non-null  int32
 6   score              52812 non-null  float64
 7   reviews            52812 non-null  int64
 8   counted            52812 non-null  int64
 9   status             52812 non-null  int64
 10  has_diff_location  52812 non-null  int64
 11  uncounted          52812 non-null  int64
 12  count_procent      52812 non-null  float64
 13  bad_item           52812 non-null  int64
dtypes: float64(2), int32(3), int64(9)
memory usage: 5.0 MB
```

Рисунок 21 – готовим датасет к обучению

Но если я запущу обучение на таком датасете, то я получу совершенно неверную картину. Так как здесь объявления, а мы ищем не плохое объявление, а плохого продавца в целом. Поэтому сгруппирую данные по профилю, заодно избавлюсь от лишних данных, получаю следующую картину (Рисунок 22):

```
RangeIndex: 611 entries, 0 to 610
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   score              611 non-null    float64
 1   category_id        611 non-null    int64
 2   status             611 non-null    int64
 3   has_diff_location  611 non-null    int64
 4   uncounted          611 non-null    int64
 5   count_procent      611 non-null    float64
 6   bad_item           611 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 33.5 KB
```

Рисунок 22 – датасет готов для обучения

Я решил проверить, как в полученном датасете с корреляцией признаков и составил новую матрицу (Рисунок 23):

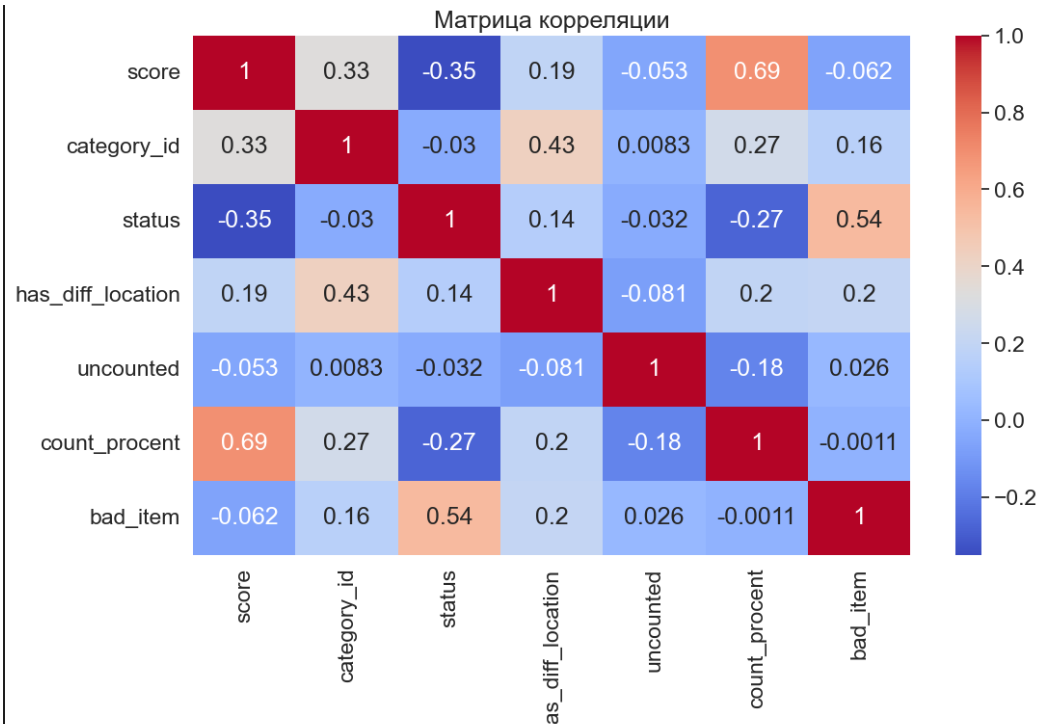

Рисунок 23 – матрица корреляции

Теперь зависимости прослеживаются гораздо лучше, а значит алгоритмам будет проще ориентироваться.

2.2 Тестирование моделей

Я прогоню датасет через все алгоритмы, которые указывал в разделе 1.2 и составлю таблицу точности (Таблица 1)

Метод	Точность	После кросс-вали-дации	Скорость
Линейная регрес-сия	89.43%	85.68%	1.4сек
Многослойный персептрон (MLP)	87.8%	85.06%	24.5сек
Метод опорных векторов	87.8%	85.06%	1м 30сек

Метод k-ближайших соседей	84.55%	86.08%	1.6сек
Деревья решений	88.62%	85.86%	1.8сек
Случайный лес	**91.06%**	**87.72%**	1м 3сек
Градиентный бустинг	87.8%	85.07%	3.1сек
Наивный байесовский алгоритм	89.43%	83.24%	1.2сек
Метод стохастического градиента	76.42%	77.92%	**1.1сек**
Нейронная сеть	**89.43%**		27.2 сек

Таблица 1 – Сравнение алгоритмов по точности и скорости

Все алгоритмы дают схожую точность. Для этого есть несколько причин:

- Очень малая выборка (53 тысячи строк из 1.3 миллионов)
- Человеческий фактор (bad_item настраивает человек)
- Малое количество признаков (я не имею доступа ко всем данным сайта Авито)
- Упрощение модели (для демонстрации работы мне пришлось убрать часть информации, которая будет оцениваться в реальном проекте)

Нейронная сеть строилась на базе библиотеки tensorflow.Keras, имеет входной слой с 7 признаками и выходной слой с 1 признаком (активация - Сигмоида). Я добавил 2 скрытых слоя по 512 нейронов. Запускал на 100 эпохах. Оптимизатор - 'adam', loss='binary_crossentropy', metrics=["accuracy"].

Изменение количества эпох или размерности слоев не дает значимого результата.

2.3 Разработка Веб-приложения

Хоть данные модели и являются частью бэкенд сервиса, по условиям задачи я должен разработать веб-приложение.

Оно будет представлять из себя 3 файла, не считая моделей:

- HTML – форма для ввода данных (также задействую язык JavaScript для отправки формы и вывода результата)
- PHP – скрипт для обработки полученных данных и запуска ру-скрипта
- Python – для запуска модели ML

К сожалению, нейронную сеть запустить не получится, так как приложение будет размещено на сервере под управлением Debian (без GUI), а на поиск способов запуска на такой конфигурации нет времени, но ML-модели дают похожий результат, так что считаю это допустимым.

В веб-версии можно будет выбрать любой из 9 методов машинного обучения, которые я использовал ранее (скриншот на Рисунке 24).

Посмотреть можно по адресу (позже будет удалено):
https://yxml.ru/bot/ml/avito.html

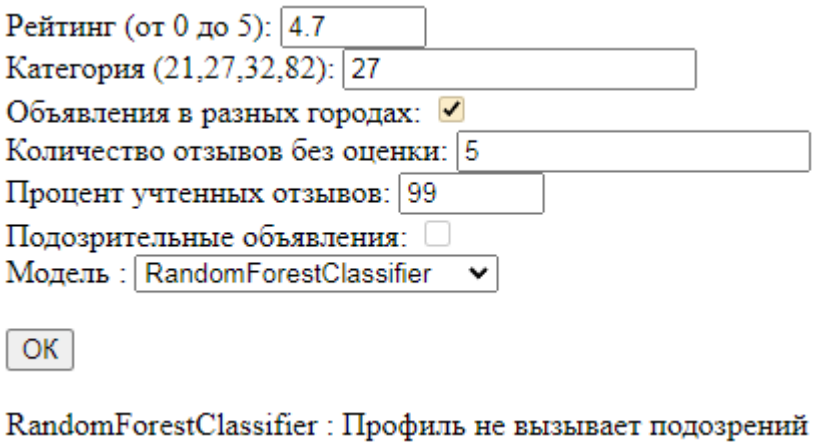

Рисунок 24 – скриншот веб-приложения

2.4 Создание репозитория на GitHub

Для выполнения данной работы был создан репозиторий, который находится по адресу: https://github.com/1nSaneRu/avito-scam-checker в нем расположен ноутбук с тестами, 2 датасета, файлы веб-приложения, 10 сохраненных моделей, данная пояснительная записка и презентация. Позже репозиторий будет скрыт из общего доступа.

Заключение

В ходе выполнения данной работы я прошел практически весь Dataflow pipeline, рассмотрел большую часть операций и задач, которые приходится выполнять специалисту по работе с данными.

Этот поток операций и задач включает:

- изучение теоретических методов анализа данных и машинного обучения;
- изучение основ предметной области, в которой решается задача;
- извлечение и трансформацию данных. Я использовал свою базу данных, накопленную за год работы другого проекта;
- проведение разведочного анализа данных статистическими методами;
- DataMining — извлечение признаков из датасета и их анализ;
- разделение имеющихся данных на обучающую, валидационную, тестовую выборки;
- выполнение предобработки (препроцессинга) данных для обеспечения корректной работы моделей;
- построение аналитического решения. Это включает выбор алгоритма решения и модели, сравнение различных моделей;
- визуализация модели и оценка качества аналитического решения;
- сохранение моделей;
- разработка и тестирование приложения для поддержки принятия решений специалистом предметной области, которое использовало бы найденную модель;
- внедрение решения и приложения в эксплуатацию;

Для выполнения работы я использовал имеющиеся и полученные знания MySQL, PHP, JavaScript, NodeJS, HTML, Python, GitHub, VSCode, Debian, Apache2, Nginx. Большая часть из них касалась развертывания удаленного сервера и организация работы приложения на нем.

Планы на будущее

Это не самостоятельный проект, а лишь его часть, а весь проект – это создание расширения для браузеров (Chrome, Firefox), которое будет парсить данные автоматически при заходе на объявление Авито, передавать на сервер для анализа и выдавать оценку, насколько подозрительным является продавец, а также сообщать негативные факторы, которые были обнаружены. Для выявления новых способов мошенничества у пользователей будет возможность сообщить о своих подозрениях, либо же напрямую сообщить о факте мошенничества. Полученную информацию я буду анализировать и вносить правки в датасет, изменять признаки и переобучать модель. Возможно для упрощения процесса разработки будет использоваться AutoML, например TPOT.

Библиографический список

1. Композиционные материалы: учебное пособие для вузов / Д. А. Иванов, А. И. Ситников, С. Д. Шляпин ; под редакцией А. А. Ильина. — Москва: Издательство Юрайт, 2019 — 253 с. — (Высшее образование). — Текст: непосредственный.
2. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.
3. ГрасД. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.
4. Документация по языку программирования python: – https://docs.python.org/3.8/index.html.
5. Документация по библиотеке numpy: – https://numpy.org/doc/1.22/user/index.html#user.
6. Документация по библиотеке pandas: – https://pandas.pydata.org/docs/user_guide/index.html#user-guide.
7. Документация по библиотеке matplotlib: – https://matplotlib.org/stable/users/index.html.
8. Документация по библиотеке seaborn: – https://seaborn.pydata.org/tutorial.html.
9. Документация по библиотеке sklearn: – https://scikitlearn.org/stable/user_guide.html.
10. Документация по библиотеке keras: – https://keras.io/ api/.
11. Loginom Вики. Алгоритмы: – https://wiki.loginom.ru/ algorithms.html.
12. Alex Maszański. Метод k-ближайших соседей (k-nearest neighbour): – https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearestneighbour-2021-07-19.
13. Yury Kashnitsky. Открытый курс машинного обучения. Тема 3. Классификация, деревья решений и метод ближайших соседей: – https://habr.com/ru/company/ods/blog/322534/.

14. Yury Kashnitsky. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес: – https://habr.com/ru/company/ods/blog/324402/.
15. Alex Maszański. Машинное обучение для начинающих: алгоритм случайного леса (Random Forest): – https://proglib.io/p/mashinnoeobuchenie-dlya-nachinayushchih-algoritm-sluchaynogo-lesa-random-forest-2021-08- 12.
16. Alex Maszański. Решаем задачи машинного обучения с помощью алгоритма градиентного бустинга: – https://proglib.io/p/reshaemzadachi-mashinnogo-obucheniya-s-pomoshchyu-algoritma-gradientnogo-bustinga2021-11-25.