Research project report

Khasianov Rasul

11 сентября 2018 г.

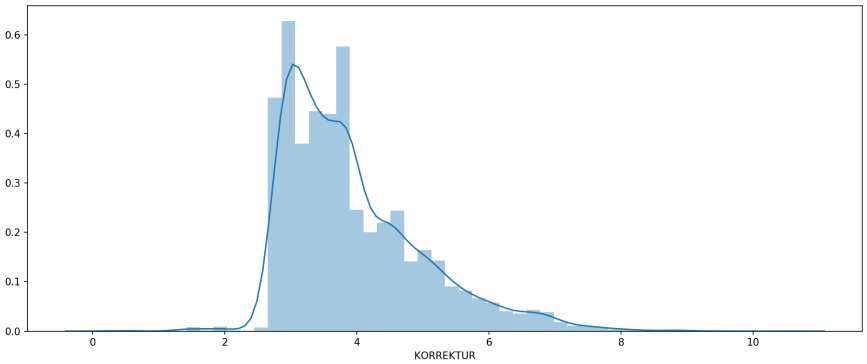
1 Описание датасета

Поставлена задача классификации с несбалансированными данными. Целью задачи предсказать возникновение fraud. Всего имеется 4 файла с данными. Для начала будем работать только с одним из них.

1.1 Описание признаков

1. ID: unique id per receipt: 301453 unique id, about 30% of data and 5280 fraud id, about 1.75% of the number of unique values.
2. KORREKTUR (= target), 1:

Рис. 1: Logarithm of korrektur



3. RECHNUNGSBETRAG: INVOICE AMOUNT: Invoice amount per document, 2.
4. ALTER: AGE: age of the customer, 3.
5. GESCHLECHT: SEX: gender of the customer (0/1-coded): about 48% and 52% respectively.

Рис. 2: Logarithm of Rechnungsbetrag

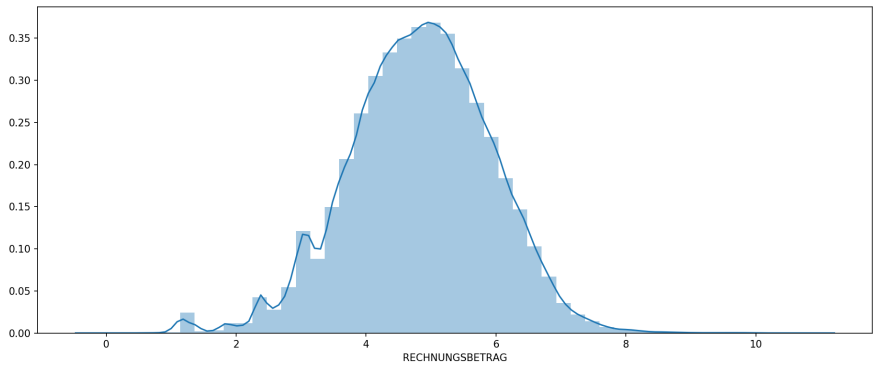
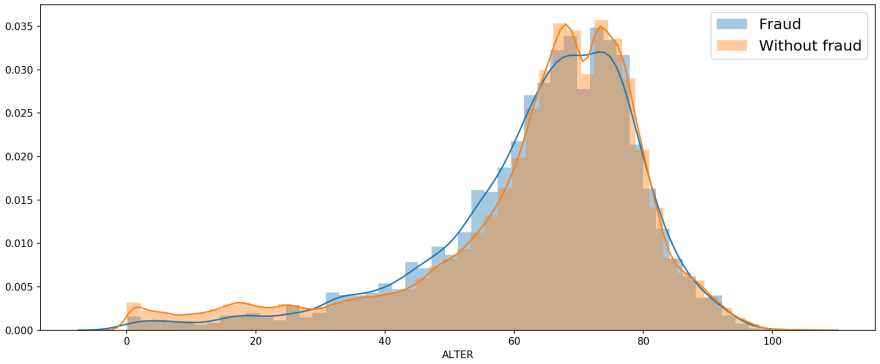


Рис. 3: Age of clients



6. VERSICHERUNG: INSURANCE: type of insurance of the customer - full or supplementary insurance (0/1-coded): about 1% and 99% respectively.

7. FACHRICHTUNG: FOCUS: Doctor's specialty: general practitioner, dermatologist, ophthalmologist, etc. (mapped to anonymous values) per document: **here problems with data. The same with VERSICHERUNG**.

---

8. NUMMER: NUMBER: Fee number describing the treatment (mapped to anonymous values) per line: 1940 different treatments

9. NUMMER KAT: NUMBER KAT: Upper category of the fee number (mapped to anonymous values) per line: 17 different categories.

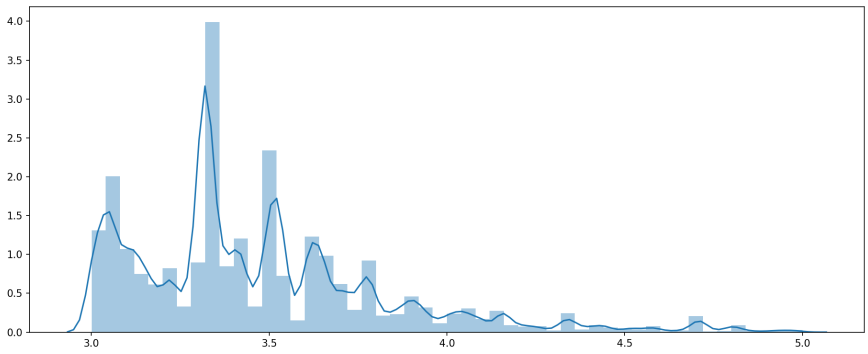
10. ANZAHL: NUMBER: number of treatments per line

11. FAKTOR: FACTOR: increase factor of treatment per line

12. BETRAG: AMOUNT: Cost of treatment per line, 4

13. ART: TYPE: Material cost type (mapped to anonymous values) per line: 11 different types

Рис. 4: Logarithm of betrag / anzahl



14. TYP: TYPE: Special type of billing ("analogue calculation 0/1-coded) per line: 0 – 90%, 1 – 1%, nan – 9%

15. LEISTUNG: PERFORMANCE: Type of benefits (grouping of treatments into collectively agreed types of benefits, mapped to anonymous values) per line: 24 different types

2 Метрика

Для оценки качества алгоритма необходимо определиться метриками, которые будут эффективно отражать обучаемость алгоритма на данных, имеющих несбалансированную природу. Ниже в таблице 1 приведен confusion matrix. Очевидно, что в задаче fraud detection очень важно находить все actual positive события. Поэтому чем ниже будет FN по отношению к TP, тем лучше мы будем выявлять случаи мошенничества. При условии, что мы можно дополнительно потребовать небольшого значения FP для того (иначе на каждый predicted positive будет приходиться большое количество FP, а это потребует излишние усилия для проверки события)

	Actual positive	Actual negative
Predicted positive	TP	FP
Predicted negative	FN	TN

Таблица 1: Confusion matrix

Приведем несколько очевидных метрик в контексте нашей задачи:

1. Recall = True Positive Rate $\frac{TP}{TP+FN}$. В идеальном случае эта метрика должна равняться 1 и это будет значить, что мы выявили все случаи мошенничества.

2. Precision = $\frac{TP}{TP+FP}$. Вероятно, мы не сможем потребовать высокого значения данной метрики. Скорее она нам скажет, какой процент от предсказанных значений является Actual positive.

3. False Positive Rate = $\frac{FP}{FP+TN}$. Дает характеристику событиям, не являющимися мошенническими. Минимален, когда все не fraud события классифицируются верным образом. Максимален, когда все обычные события предсказываются как fraud.

Рассмотрим две кривые:

1. Roc-curve. X: FPR, Y: TPR
2. PR-curve. X: Recall, Y: Precision

Классическая Roc-curve показывает, как хорошо оба класса классифицируются. Качество одного класса оценивается высоким значением TPR, а качество второго класса оценивается низким значением FPR. Поэтому хорошим предсказанием является «баланс» между этими значениями.

PR-curve отличается тем, что она характеризует, как хорошо классифицируется один класс (fraud). Мы хотим максимизировать значение TP по отношению к predicted positive и к actual positive.

3 Преобразование признаков

В данных имеются, 4 постоянных признака для каждого ID (оставил их почти, как есть), и 8 признаков, которые имеют разную длину для каждого ID.

1. BETRAG (+ ANZAHL: удельное значение — Betrag / Anzahl): переменная типа Real, поэтому преобразовал данный признак 2 способами: с помощью простых статистик (mean, std, median, min, max) и с помощью гистограммы для логарифма от Betrag (задал 100 признаков на интервал [2, 6]). Получается для каждого ID было известно количество попаданий в логарифмический интервал (аналог bag of words).
2. FAKTOR, TYP: аналогично задал с помощью обычных статистик.
3. NUMMER, NUMMER KAT, LEISTUNG, ART — имеют категориальную природу, поэтому закодировал с помощью bag of words.

4 Результаты и сравнение

Дипломная работа «Autoencoder for Fraud Detection: An Empirical Application» посвящена именно этой же задачи и этим же данным. Они тремя способами подготавливают данные и обучают на них автоэнкодер: практически на сырых данных: только NUMMER, NUMMER KAT, LEISTUNG были закодированы с помощью one-hot-encoding (1), на немного преобразованных данных: NUMMER, NUMMER KAT, TYPE, NUMBER, FACTOR, AMOUNT и PERFORMANCE – хотя по коду, это не так, видимо (2), на данных, в которых использовался bag of words подход преобразования категориальных данных: NUMMER, NUMMER KAT, LEISTUNG – но и здесь, код достаточно странный (3).

После этого они обучили автоэнкодер и посчитали для каждого из них AUC Score. Для модели 1 – 0.69, 2 – 0.76, 3 – 0.79.

Я использовал в основном два алгоритма: Xgboost и Lightgbm. Обучался пока что на 30% данных (так как немного не хватало ресурсов), и получил следующие результаты:

Description	AUC ROC	Average PR
1. LightGBM	0.8345	0.0994
2. LightGBM with over-sampling (ADASYN)	0.8455	0.0985
3. LightGBM with under-sampling (RepeatedEditedNearestNeighbours)	0.8439	**0.1019**
4. LightGBM with under-sampling (InstanceHardnessThreshold)	**0.8515**	0.1011
1. Xgboost	0.843	0.1077
2. Xgboost with under-sampling (EditedNearestNeighbours)	0.8424	**0.1115**
3. Xgboost with under-sampling (InstanceHardnessThreshold)	**0.8437**	0.1071
4. Xgboost with with Ensemble of samplers	0.8434	0.1017

Таблица 2: Результаты

Для балансировки данных использовал библиотеку imblearn, с помощью которого получил небольшой прирост качества.