IEEE-CIS Fraud Detection

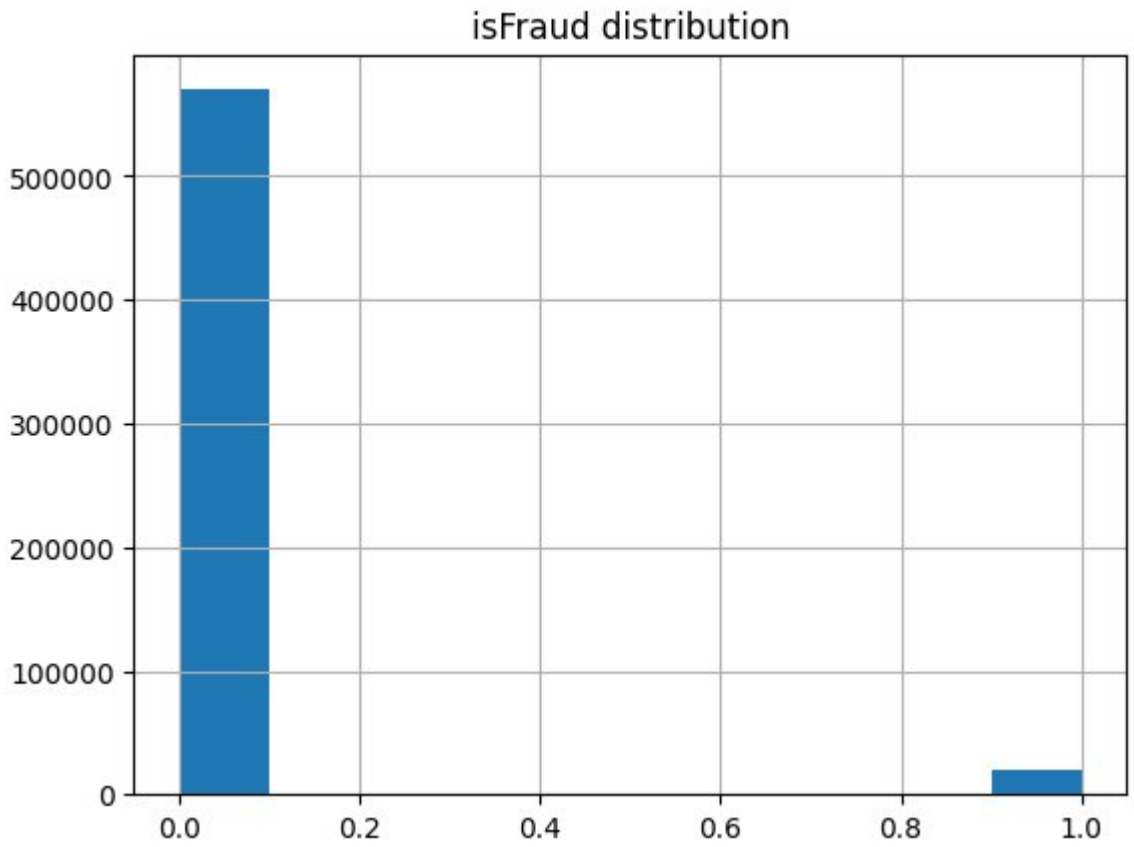
Ким Александр M05-413г

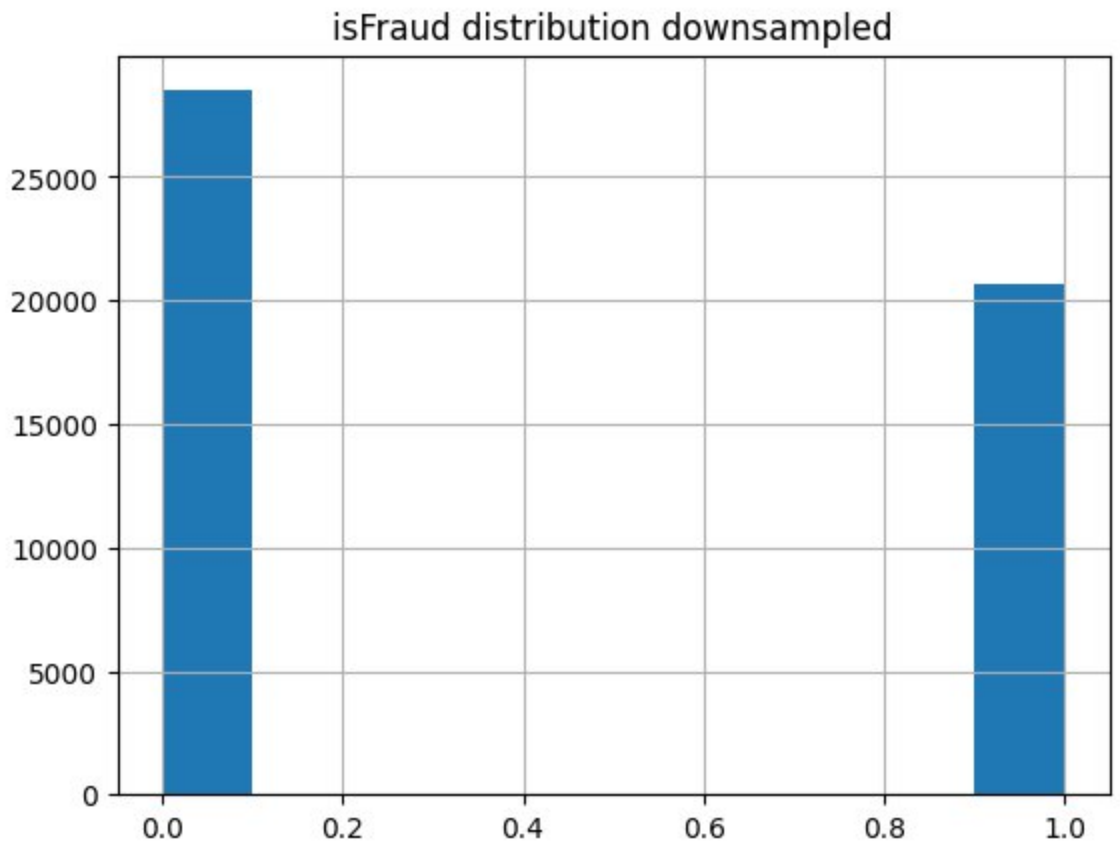
Kaggle: https://www.kaggle.com/bunsans

Описание

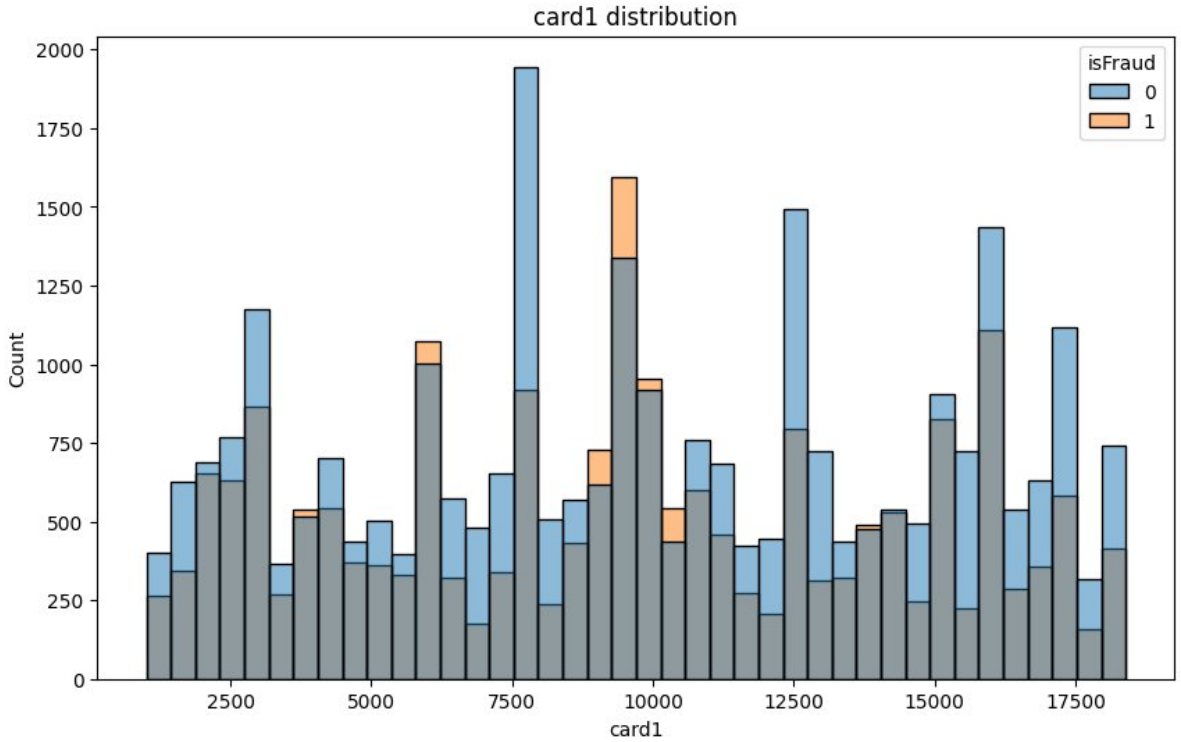
- Imagine standing at the check-out counter at the grocery store with a long line behind you and the cashier not-so-quietly announces that your card has been declined. In this moment, you probably aren’t thinking about the data science that determined your fate.
- Embarrassed, and certain you have the funds to cover everything needed for an epic nacho party for 50 of your closest friends, you try your card again. Same result. As you step aside and allow the cashier to tend to the next customer, you receive a text message from your bank. “Press 1 if you really tried to spend $500 on cheddar cheese.”
- While perhaps cumbersome (and often embarrassing) in the moment, this fraud prevention system is actually saving consumers millions of dollars per year. Researchers from the IEEE Computational Intelligence Society (IEEE-CIS) want to improve this figure, while also improving the customer experience. With higher accuracy fraud detection, you can get on with your chips without the hassle.
- IEEE-CIS works across a variety of AI and machine learning areas, including deep neural networks, fuzzy systems, evolutionary computation, and swarm intelligence. Today they’re partnering with the world’s leading payment service company, Vesta Corporation, seeking the best solutions for fraud prevention industry, and now you are invited to join the challenge.
- In this competition, you’ll benchmark machine learning models on a challenging large-scale dataset. The data comes from Vesta's real-world e-commerce transactions and contains a wide range of features from device type to product features. You also have the opportunity to create new features to improve your results.
- If successful, you’ll improve the efficacy of fraudulent transaction alerts for millions of people around the world, helping hundreds of thousands of businesses reduce their fraud loss and increase their revenue. And of course, you will save party people just like you the hassle of false positives.

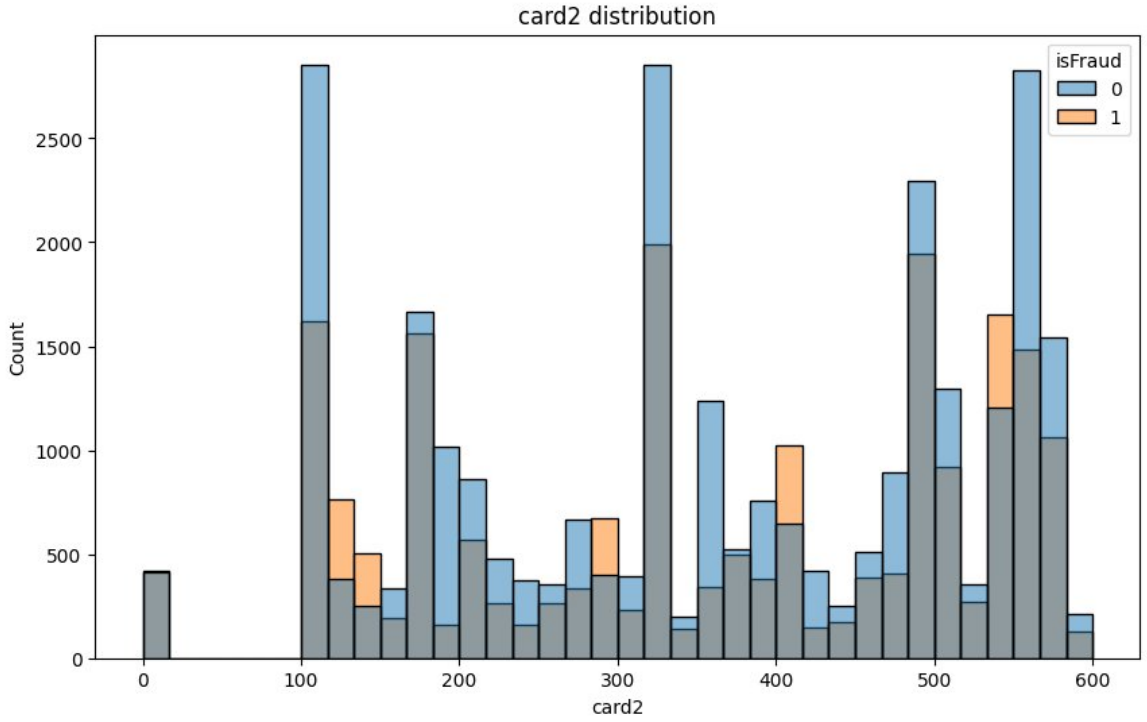
Распределение целевой переменной



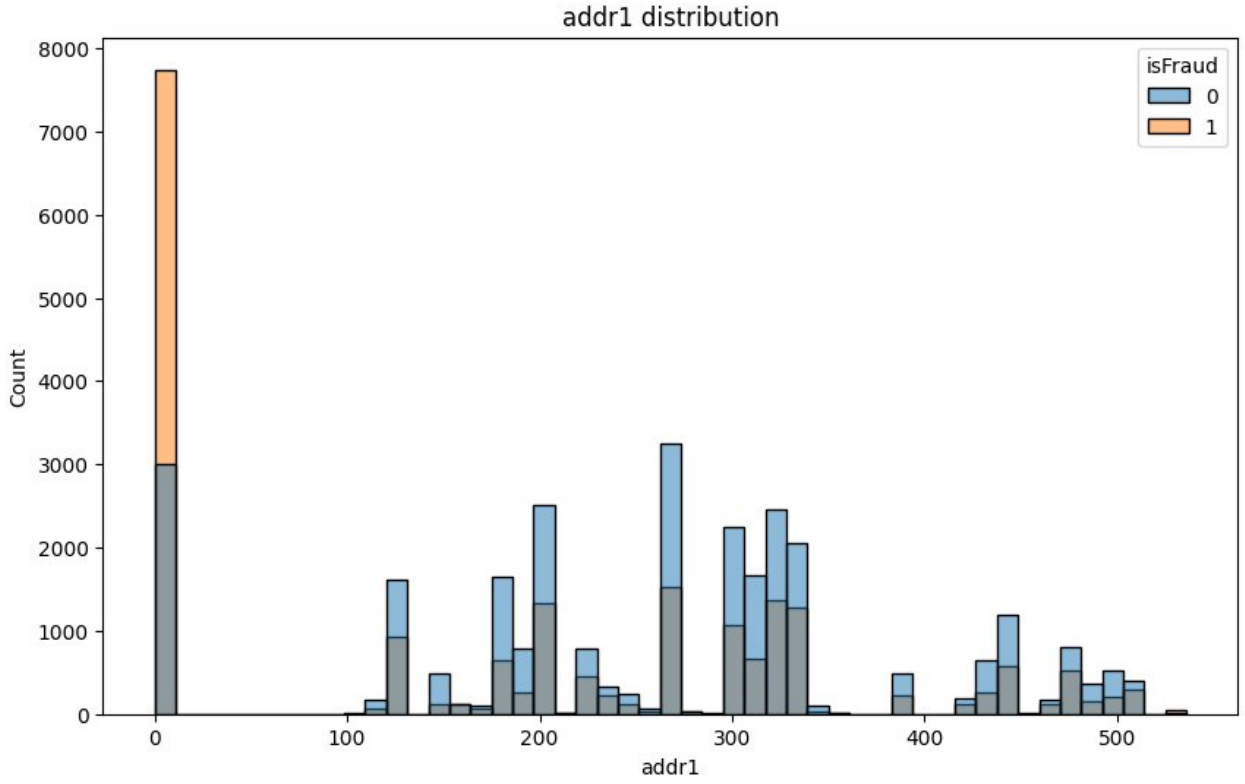


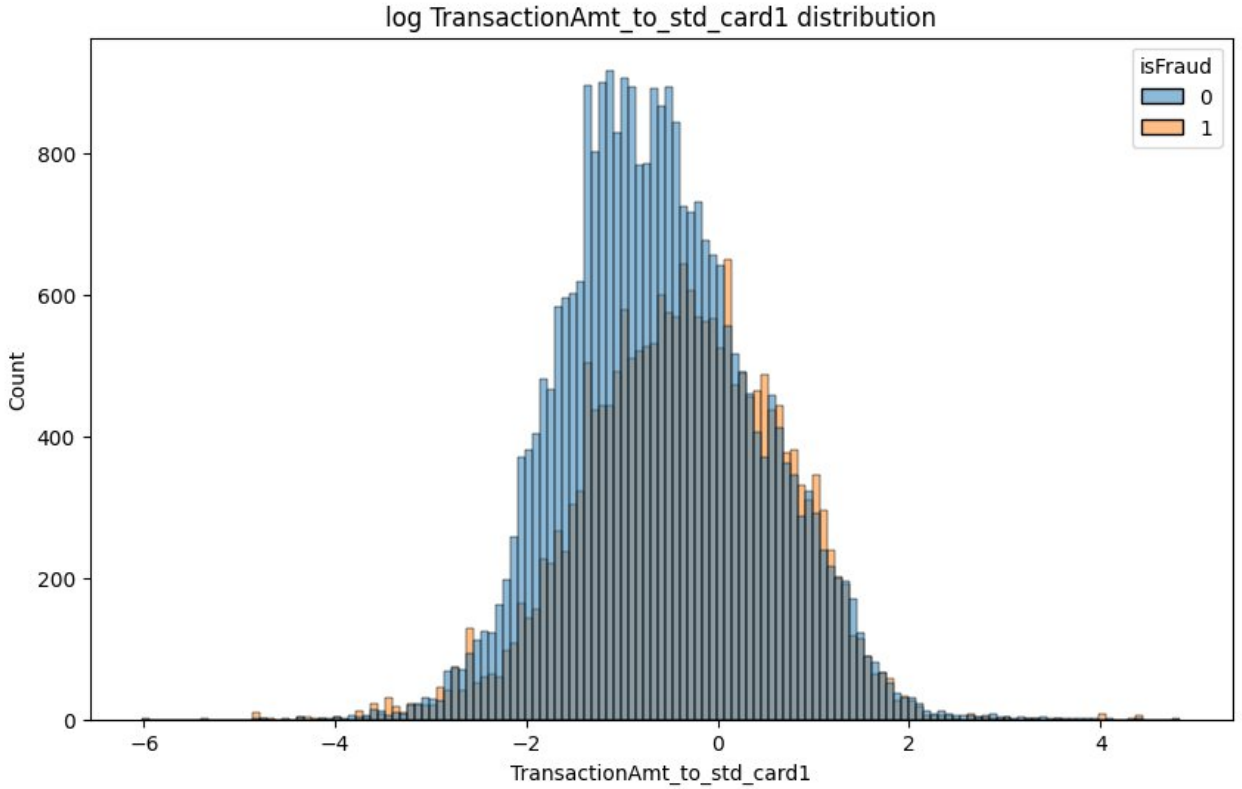
Распределение топ фичей



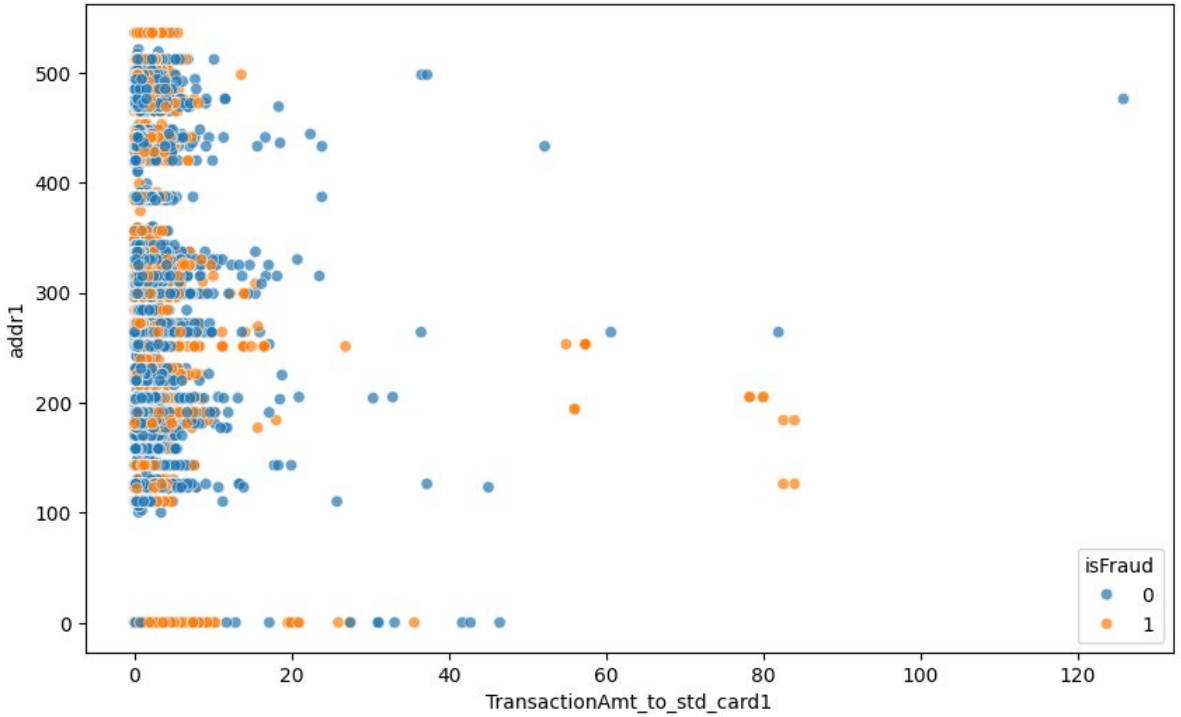


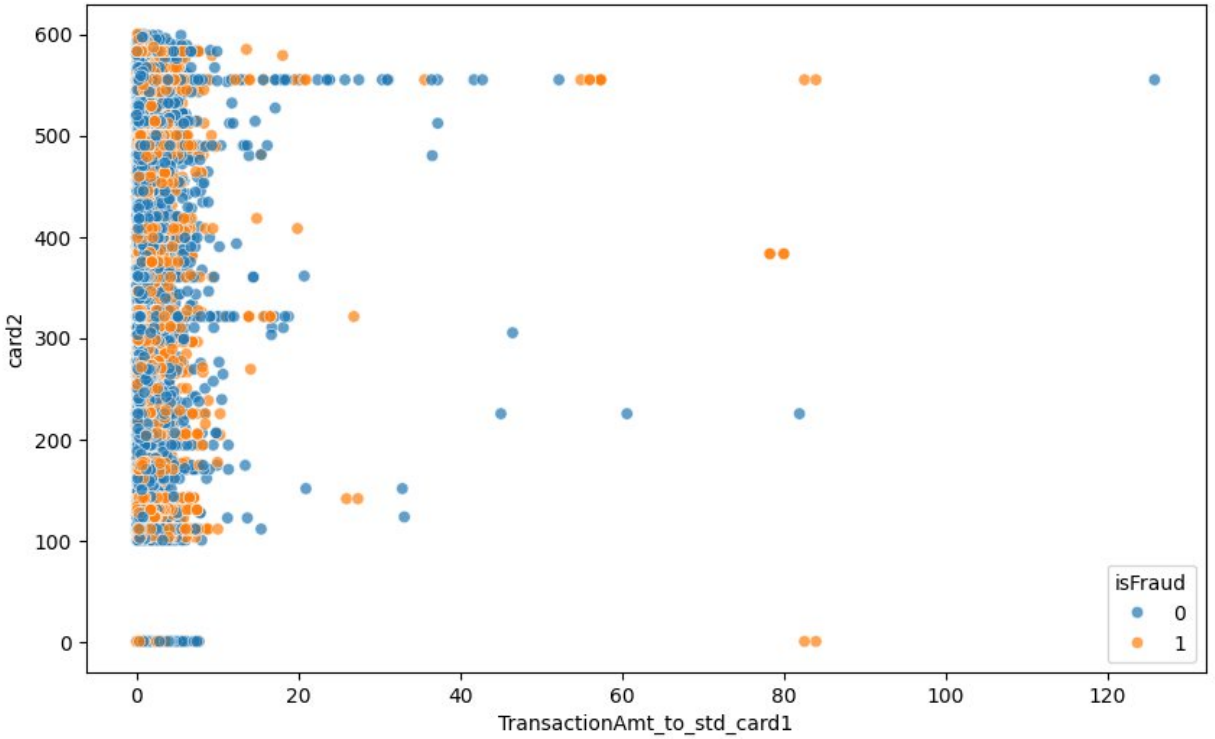
Распределение топ фичей



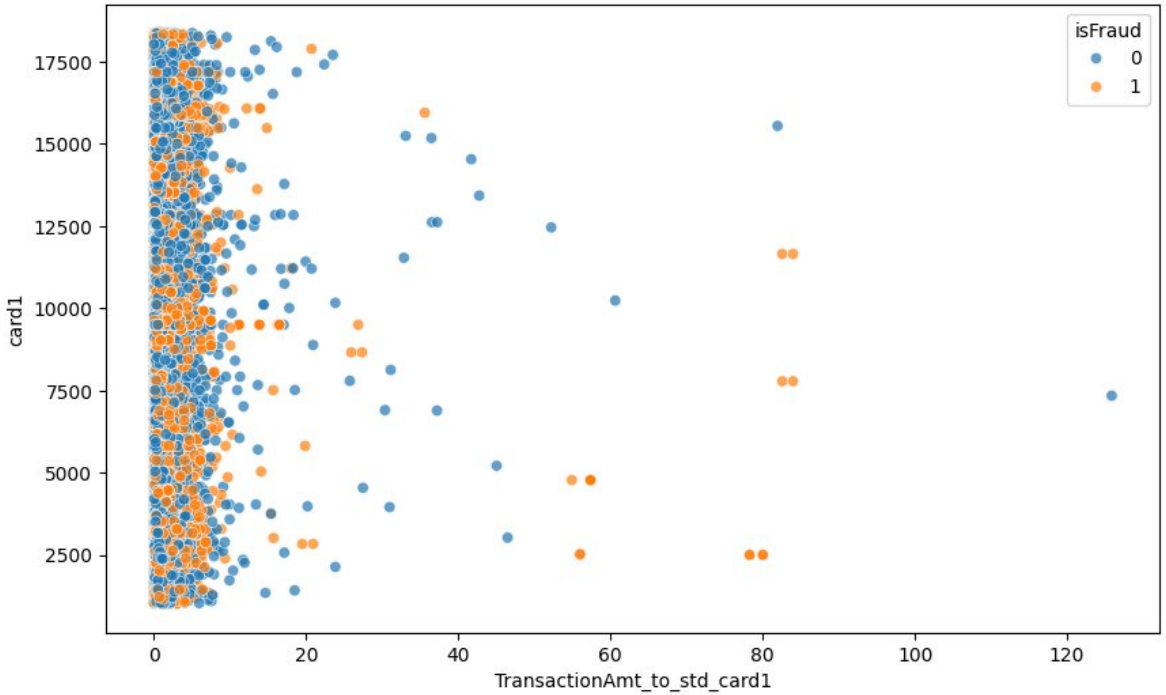


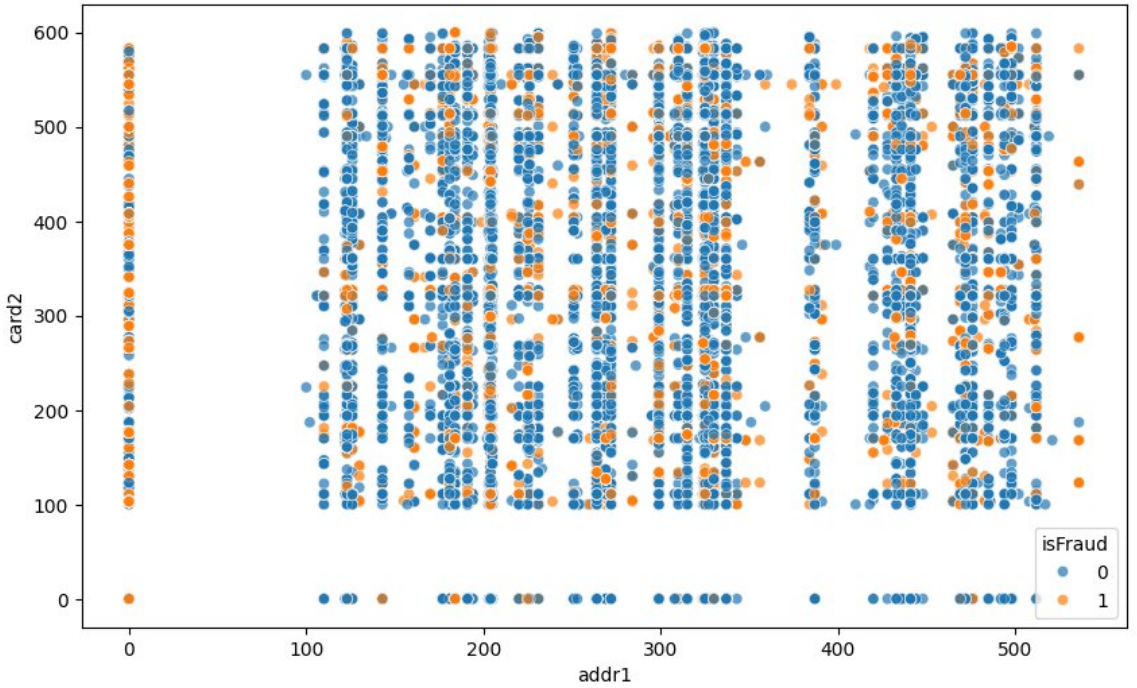
Попарное распределение топ фичей



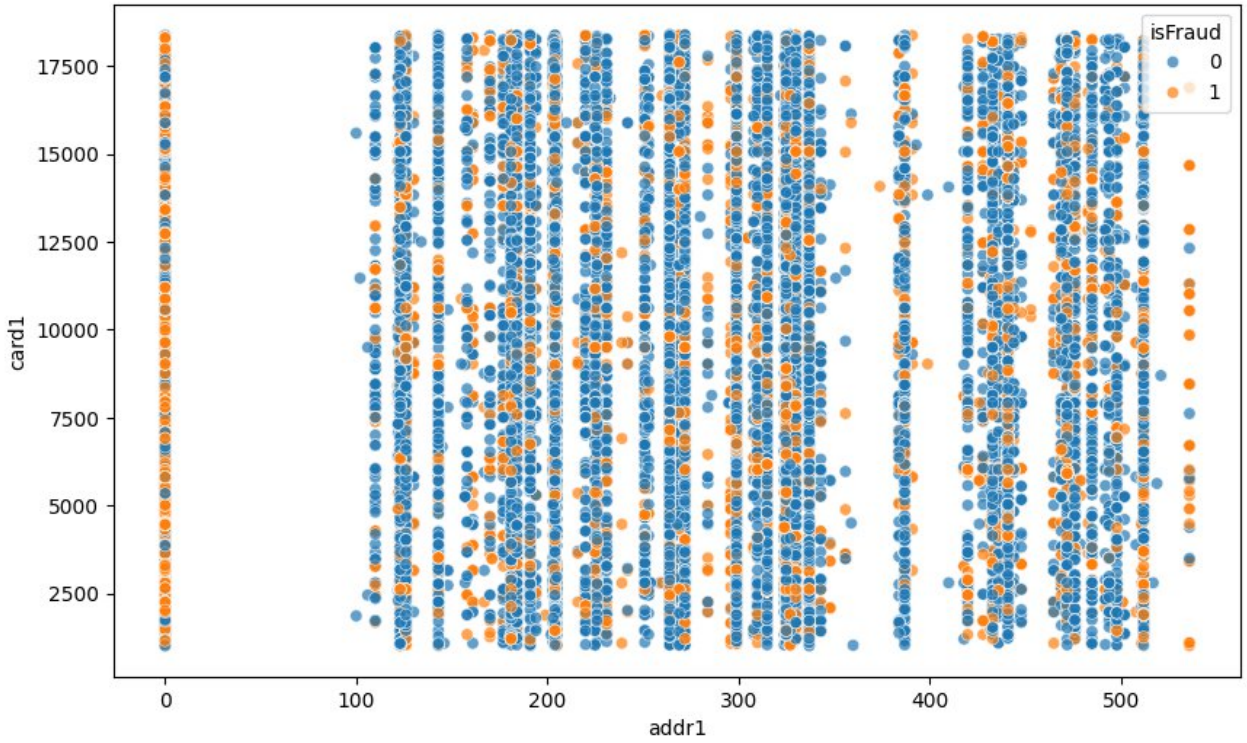


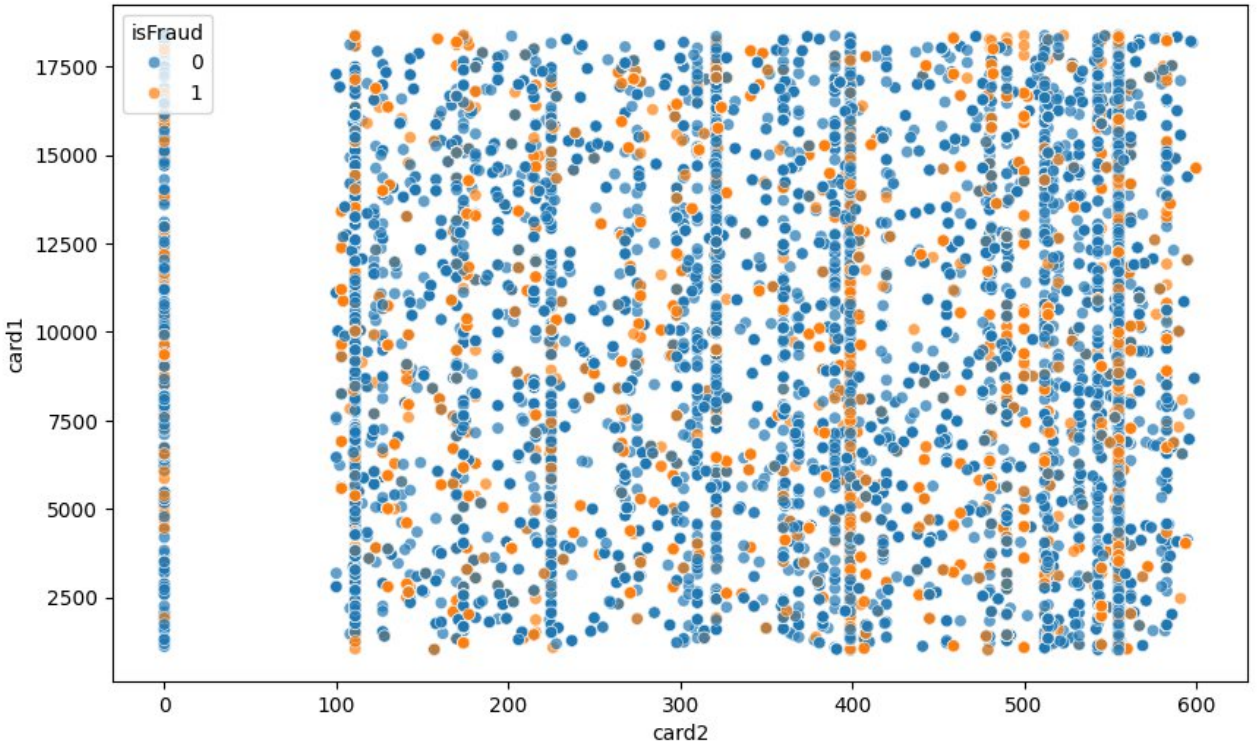
Попарное распределение топ фичей





Попарное распределение топ фичей





Временные и категориальные фичи

- Временная фича: TransactionDT использовалась для сортировки данных и были идеи использовать TimeSeriesSplit, но было использовано только для обучения начальной модели для определения топовых фичей(lightgbm)
- Категориальные фичи изначально были закодированы LabelEncoder для определения топ фичей, после фичи с небольшым количеством различных значений преобразовывались OHE

Numeric фичи и выбросы

- Были преобразованы standart scaler
- Выбросы были обработаны только у нескольких фичей, так как в случае с задачей определения фрода, слишком большие значения в транзакциях могут как раз указывать на аномалии и более подозрительные операции

Новые признаки

Основной упор был сделан на признаки отношения:

Размер транзакции к среднему значению(стандартному отклонению) сгруппированных фич card1, card2 и тд

Так же добавлены фичи отношения дельты по времени с n-ой поледней операции к среднему значению сгруппированных фич card1, card2 и тд

Функции для просчета

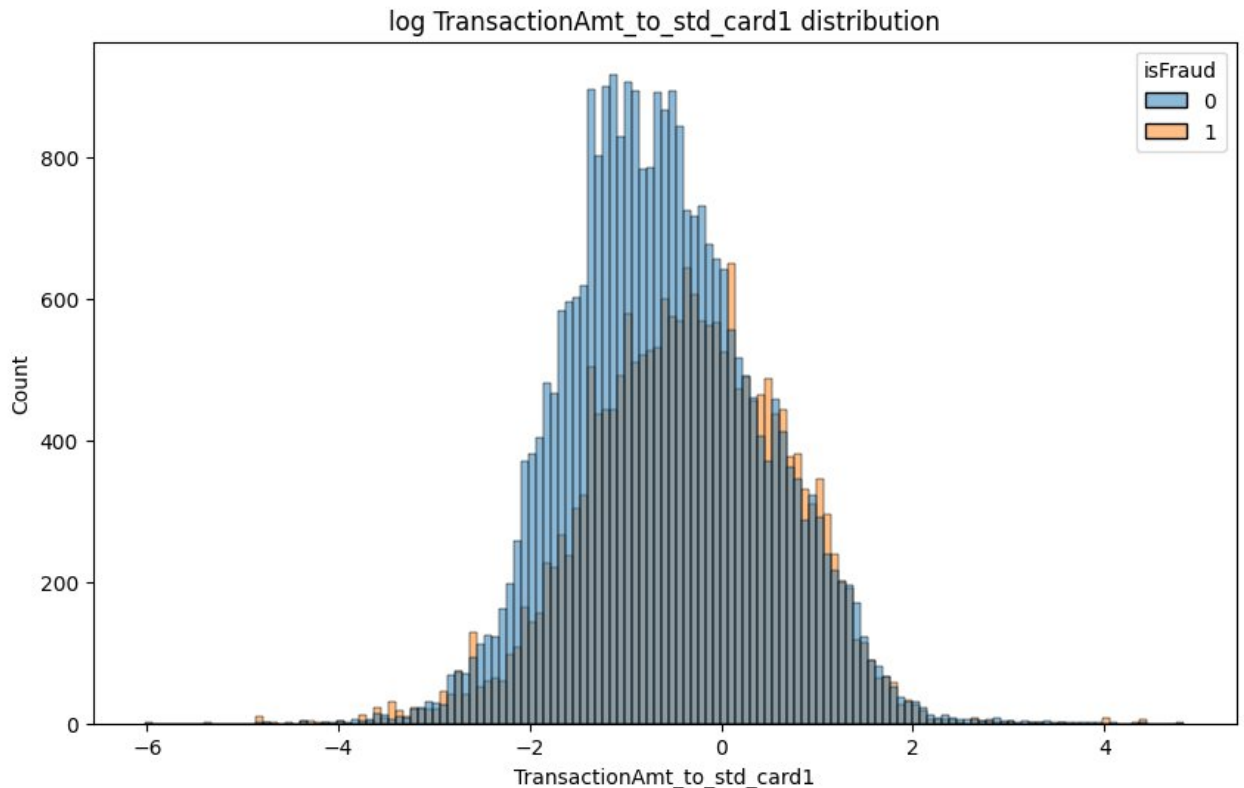
```python
def make_ratio_features_tr_amt(data, card_num):
    data[f"TransactionAmt_to_std_card{card_num}"] = data[
        "TransactionAmt"
    ] / data.groupby([f"card{card_num}"])["TransactionAmt"].transform("std")
    data[f"TransactionAmt_to_mean_card{card_num}"] = data[
        "TransactionAmt"
    ] / data.groupby([f"card{card_num}"])["TransactionAmt"].transform("mean")
    return data


def make_ratio_features_time_Delta_from_last(data, D_num, card_num):
    data[f"D{D_num}_to_mean_card1"] = data[f"D{D_num}"] / train.groupby(
        [f"card{card_num}"]
    )[f"D{D_num}"].transform("mean")
    return data
```

Новые топ фичи

- Как результат большая часть фичей с отношением размера транзакции зашли в топ, но нельзя отметать тот факт, что размер транзакции сама по себе сильная фича и возможно что ее эффект размазался на остальные вариации.

85	TransactionAmt_to_mean_card1	363.08
86	TransactionAmt_to_std_card5	370.20
87	P_emaildomain	386.04
88	card5	386.08
89	D10	389.00
90	D2	418.60
91	D4	423.80
92	C13	456.36
93	TransactionAmt_to_std_card2	484.20
94	dist1	499.96
95	D15	585.08
96	TransactionAmt_to_std_card1	653.48
97	addr1	951.48
98	card2	1016.40
99	card1	1195.76


log TransactionAmt_to_std_card1 distribution
isFraud
0
1
800
600
400
200
0
Count
−6
−4
−2
0
2
4
TransactionAmt_to_std_card1

KNeighborsClassifier

```
n_neighbors = trial.suggest_int("n_neighbors", 1, 10)
p = trial.suggest_int("p", 1, 4)
```

- Лучшие параметры:

{'n_neighbors': 7,
 'p': 1}

- Лучшее значение **roc auc: 0.820**

LinearSVC

```
loss = trial.suggest_categorical("loss", ["hinge", "squared_hinge"])
C = trial.suggest_float("C", 0.01, 5)
```

- Лучшие параметры:

{'loss': 'squared_hinge',
 'C': 3.46}

- Лучшее значение **roc auc: 0.826**

DecisionTreeClassifier

```
criterion = trial.suggest_categorical("criterion", ["gini", "entropy", "log_loss"])
max_depth = trial.suggest_int("max_depth", 5, 12)
```

- Лучшие параметры:

{'criterion': 'entropy',
 'max_depth': 7}

- Лучшее значение **roc auc: 0.836**

BaggingClassifier on DecisionTree

```
criterion = trial.suggest_categorical("criterion", ["gini", "entropy", "log_loss"])
min_samples_leaf = trial.suggest_int("min_samples_leaf", 1, 7)
```

- Лучшие параметры:

{'criterion': 'entropy',
 'min_samples_leaf': 6
 'estimators': 20}

- Лучшее значение **roc auc: 0.894**

RandomForestClassifier

```
criterion = trial.suggest_categorical("criterion", ["gini", "entropy"])
min_samples_leaf = trial.suggest_int("min_samples_leaf", 1, 10)
max_features = trial.suggest_categorical("max_features", ["sqrt", "log2", 0.3])
```

- Лучшие параметры:

{'criterion': 'gini',
 'min_samples_leaf': 1,
 'max_features': 0.3
 'estimators': 100}

- Лучшее значение **roc auc: 0.901**

GradientBoostingClassifier

```
learning_rate = trial.suggest_float("learning_rate", 0.001, 1)
subsample = trial.suggest_float("subsample", 0.85, 1)
max_depth = trial.suggest_int("max_depth", 2, 12)
max_features = trial.suggest_categorical("max_features", ["sqrt", "log2", 0.3])
```

- Лучшие параметры:

{'learning_rate': 0.049,
 'subsample': 0.96,
 'max_depth': 12,
 'max_features': 0.3,
 'estimators': 300 }

- Лучшее значение **roc auc: 0.919**

Stacking

- В качестве базовых моделей стекинга были взяты KNN, RandomForest, GradientBoosting
- Финальной моделью был GradientBoosting

Использовался метод cross_val_score на 4 фолдах

- **roc auc: 0.9198**

Результаты

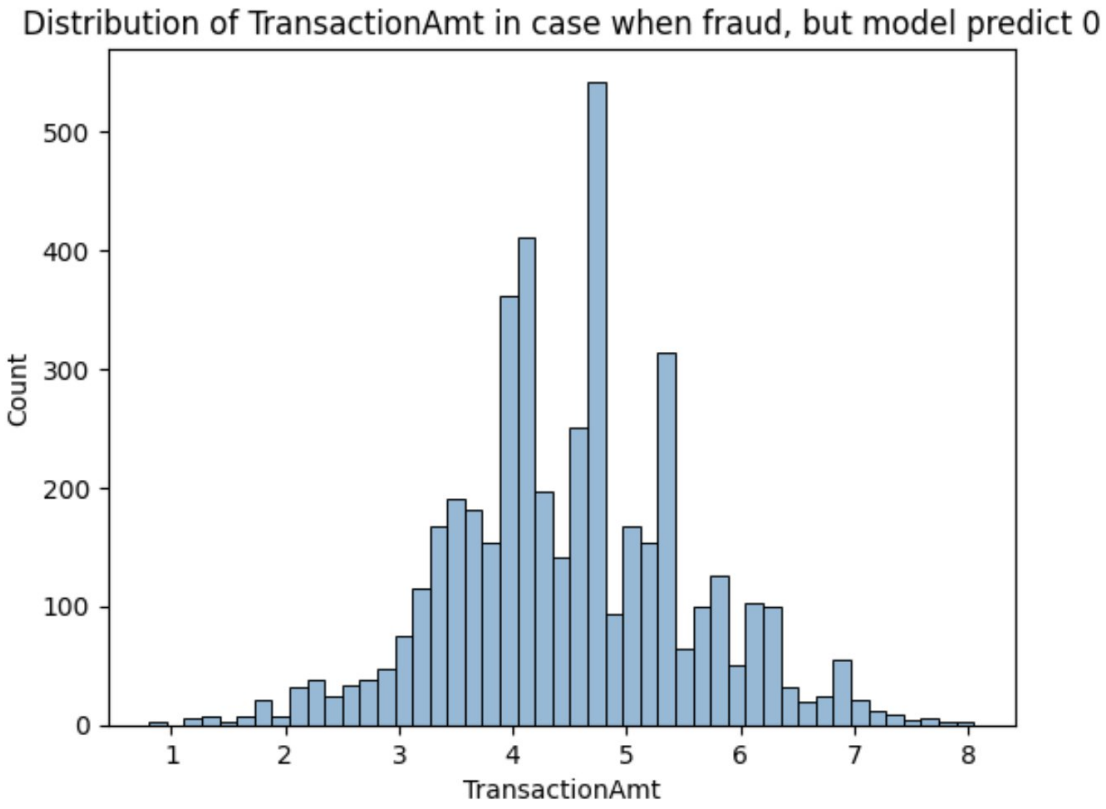
- Лучше всего себя показал GradientBoosting* с 300 estimators. Возможно при увелечении их количества можно получить качество лучше.
- Итоговое качество на Kaggle: 0.8919

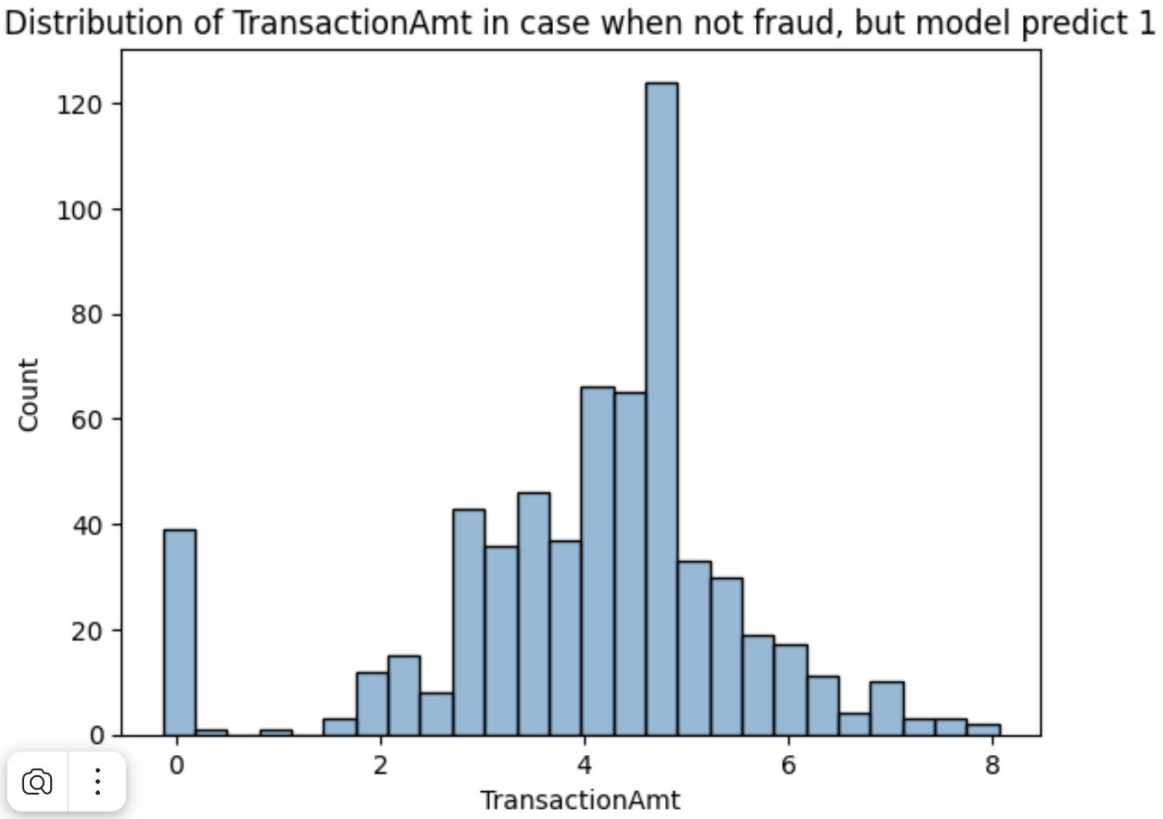
Этого не достаточно для попадания в лидерборд.

Интуитивно кажется, что наиболее сложней всего детектировать примеры фродовых транзакций с низким TransactionAmount и наоборот при «белой» транзакцией с большим TransactionAmount модель часто ошибается.

*После посылки был так же обучен стэкинг на котором качество вышло лучше

- Однако при рассмотрении распределении этой фичи(в log масштабе) на ошибках модели, можно заметить, что модель часто ошибочно предполагает о фродовости при малых суммах перевода





Варианты решения

- Как вариант решения это увеличение мощностей для обучения и перебора параметров, так как подбор параметро производился на downsample выборке. Так же возможны эксперименты с catboost и lightgbm.
- И создание новых фичей делающих меньше упор на размер транзакции.

IEEE-CIS Fraud Detection

Can you detect fraud from customer transactions?

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

Leaderboard

Raw Data Refresh

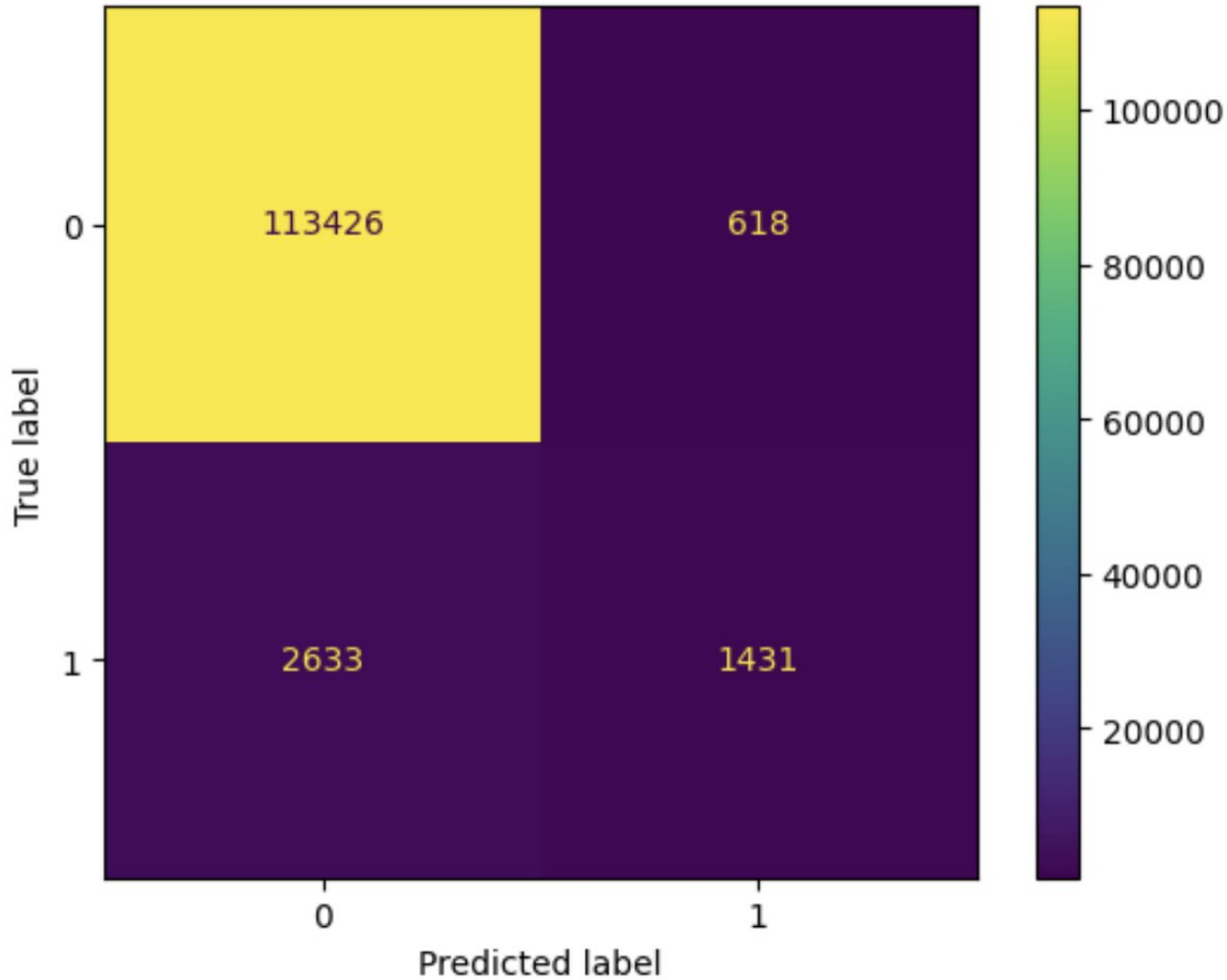
YOUR RECENT SUBMISSION

submission_grad_boost (1).csv
Submitted by Alexandr Kim · Submitted 28 minutes ago

Score: 0.915760
Private score: 0.891945

Jump to your leaderboard position

	precision	recall	f1-score	support
0	0.98	0.99	0.99	114044
1	0.70	0.35	0.47	4064
accuracy			0.97	118108
macro avg	0.84	0.67	0.73	118108
weighted avg	0.97	0.97	0.97	118108


113426
618
2633
1431
0
1
True label
0
1
Predicted label
100000
80000
60000
40000
20000