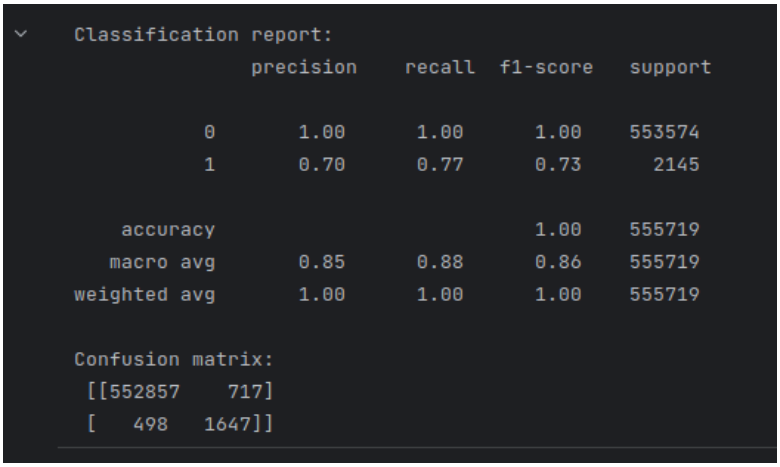
Модели:

1. Логистическая регрессия - это статистическая модель, которая используется для моделирования вероятности наступления определенного события. Логистическая регрессия применяется для бинарной классификации, где выходом является бинарное значение, например, 0 или 1.

2. Метод опорных векторов – это метод, который строит гиперплоскость или несколько гиперплоскостей в пространстве высокой размерности для разделения данных на классы. SVM также может быть использован для многоклассовой классификации.
Может работать с несбалансированными данными.

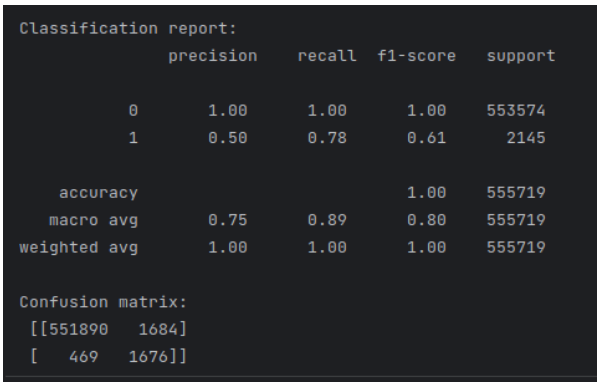
3. Решающие деревья - это модель, которая использует деревья для принятия решений на основе различных признаков. Решающие деревья также могут быть использованы для многоклассовой классификации.



Classification report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	553574
1	0.70	0.77	0.73	2145
accuracy			1.00	555719
macro avg	0.85	0.88	0.86	555719
weighted avg	1.00	1.00	1.00	555719

Confusion matrix:
[[552857 717]
[498 1647]]



Classification report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	553574
1	0.50	0.78	0.61	2145
accuracy			1.00	555719
macro avg	0.75	0.89	0.80	555719
weighted avg	1.00	1.00	1.00	555719

Confusion matrix:
[[551890 1684]
[469 1676]]

С уравновешиванием

без уравновешивания

4. Случайный лес – может работать с несбалансированными данными. Лучше сбалансировать. Случайный лес также может быть использован для задач бинарной классификации. Для этого, каждый дерево строится на случайном подмножестве данных и признаков, а затем все деревья комбинируются в один классификатор.
В задачах бинарной классификации, случайный лес может быть особенно полезен, так как он способен обрабатывать данные с большим количеством признаков, а также может работать с различными типами признаков. Кроме того, случайный лес может обработать

несбалансированные данные, когда количество примеров в одном классе значительно превышает количество примеров в другом.

```
Classification report:
                precision    recall  f1-score   support

             0       1.00      1.00      1.00    553574
             1       0.78      0.79      0.78      2145

      accuracy                           1.00    555719
     macro avg       0.89      0.89      0.89    555719
  weighted avg       1.00      1.00      1.00    555719

  Confusion matrix:
   [[553089    485]
   [   449   1696]]
  Share of Non-Fraud in Test Data: 0.9961
```

5. KNN - это алгоритм, который определяет класс объекта на основе его близости к другим объектам. Он используется для поиска K ближайших объектов и принятия решения на основе того, какие классы у этих объектов.

Основываясь на приведенном выше EDA, мы обнаружили, что такие функции, как сумма транзакции, возраст держателя кредитной карты, категория расходов, время транзакции и местоположение, имеют разную степень корреляции с мошенничеством с кредитными картами. Это помогает нам выбрать, какие функции мы хотим включить в наши модели данных. План состоит в том, чтобы обучить модели на наборе обучающих данных, который мы проанализировали выше, а затем использовать набор тестовых данных для оценки производительности модели.
- Плохо
6. Бустинг над решающими деревьями == Градиентный бустинг.

На входе имеем датафреймы:
train_features и test_features, имеющие следующую структуру:

```
train_features.info()
Executed in 409ms, 10 Apr at 12:14:35

Data columns (total 9 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   age          1296675 non-null  float64
 1   amount(usd)  1296675 non-null  float64
 2   hour_of_day  1296675 non-null  int64
 3   category     1296675 non-null  object
 4   merchant     1296675 non-null  object
 5   state        1296675 non-null  object
 6   city_pop     1296675 non-null  int64
 7   job          1296675 non-null  object
 8   is_fraud     1296675 non-null  int64
dtypes: float64(2), int64(3), object(4)
memory usage: 89.0+ MB
```

```
test_features.info()
Executed in 184ms, 10 Apr at 12:15:14

Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   age          555719 non-null  float64
 1   amount(usd)  555719 non-null  float64
 2   hour_of_day  555719 non-null  int64
 3   category     555719 non-null  object
 4   merchant     555719 non-null  object
 5   state        555719 non-null  object
 6   city_pop     555719 non-null  int64
 7   job          555719 non-null  object
 8   is_fraud     555719 non-null  int64
dtypes: float64(2), int64(3), object(4)
memory usage: 38.2+ MB
```

Признаки category, merchant, state, job – категориальные, то есть просто строки, слова. Для модели это не подходит и нужно их преобразвать.

Способы преобразования:

+1. OrdinalEncoder - это метод кодирования категориальных признаков в числовые значения, где каждому уникальному значению категориального признака присваивается уникальный целочисленный код. Этот метод используется, когда значения категориальных признаков могут быть упорядочены или имеют важность в порядке следования.

Другими словами: закодировать каждое уникальное значение категориального признака уникальным целочисленным кодом, основанным на их порядке или значимости.

+2. Label Encoding - В этом методе каждое уникальное значение признака заменяется соответствующим числовым значением. Например, если признак "цвет" принимает значения "красный", "зеленый" и "синий", то "красный" может быть заменен на 0, "зеленый" на 1 и "синий" на 2. Этот подход может работать хорошо для моделей, которые могут обрабатывать числовые данные, но он может приводить к неправильной интерпретации порядка между значениями, если порядок не является существенным.

Друими словами: закодировать каждое уникальное значение категориального признака уникальным целочисленным кодом. Этот метод применяется, когда значения категориального признака не имеют порядка и не важны их значения.

!-3. One-Hot Encoding - Это самый распространенный подход к представлению категориальных признаков. В этом методе каждый уникальный значок признака представляется в виде бинарного вектора длиной, равной количеству уникальных значений признака. Вектор состоит из нулей и одной единицы, которая соответствует индексу соответствующего значения признака. Этот подход может приводить к большому количеству признаков в случае большого количества уникальных значений.

Другими словами: создать бинарные столбцы для каждого уникального значения категориального признака. Этот метод применяется, когда значения категориального признака не имеют порядка, и все значения равноправны.

+4. Count Encoding - заменить каждое значение категориального признака количеством раз, которое оно появляется в исходном наборе данных.

-5.Target Encoding - заменить каждое значение категориального признака средним значением целевой переменной для этого значения.

Если буду использовать, то все категориальные признаки будут иметь одинаковые значения. Одинаковые столбцы.

Будем применять на практике:
1. Ordinal Encoding:

```
print(classification_report(y_test, predict))
Executed in 390ms, 10 Apr at 12:43:06
```

	precision	recall	f1-score	support
0	1.00	0.94	0.97	553574
1	0.05	0.75	0.09	2145
accuracy			0.94	555719
macro avg	0.52	0.85	0.53	555719
weighted avg	1.00	0.94	0.96	555719

```
[[519631  33943]
 [   533   1612]]
```

2. Label Encoding:

```
print(classification_report(y_test, predict))
Executed in 350ms, 10 Apr at 13:07:59
```

	precision	recall	f1-score	support
0	1.00	0.94	0.97	553574
1	0.05	0.75	0.09	2145
accuracy			0.94	555719
macro avg	0.52	0.85	0.53	555719
weighted avg	1.00	0.94	0.97	555719

```
[[521693  31881]
 [   538   1607]]
```

3. Count Encoding:

```
print(classification_report(y_test, predict))
Executed in 473ms, 10 Apr at 13:23:19
```

	precision	recall	f1-score	support
0	1.00	0.73	0.84	553574
1	0.01	0.77	0.02	2145
accuracy			0.73	555719
macro avg	0.50	0.75	0.43	555719
weighted avg	0.99	0.73	0.84	555719

```
[[401555 152019]
 [   484   1661]]
```

4. One-Hot Encoding:
Очень долго работает

Пока лучше использовать Label Encoding.

Борьба с несбалансированностью:
1. SMOTE. - на пару процентов лучше
2. AUC-ROC.
3. F1-F2(F1beta).
4. ADASYN:

```
print(classification_report(y_test, predict))
```
Executed in 379ms, 10 Apr at 14:46:54

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96    553574
           1       0.03      0.76      0.06      2145

    accuracy                           0.92    555719
   macro avg       0.52      0.84      0.51    555719
weighted avg       1.00      0.92      0.95    555719
```

```
[[506884  46690]
 [   524   1621]]
```

Модель логистической регресии: