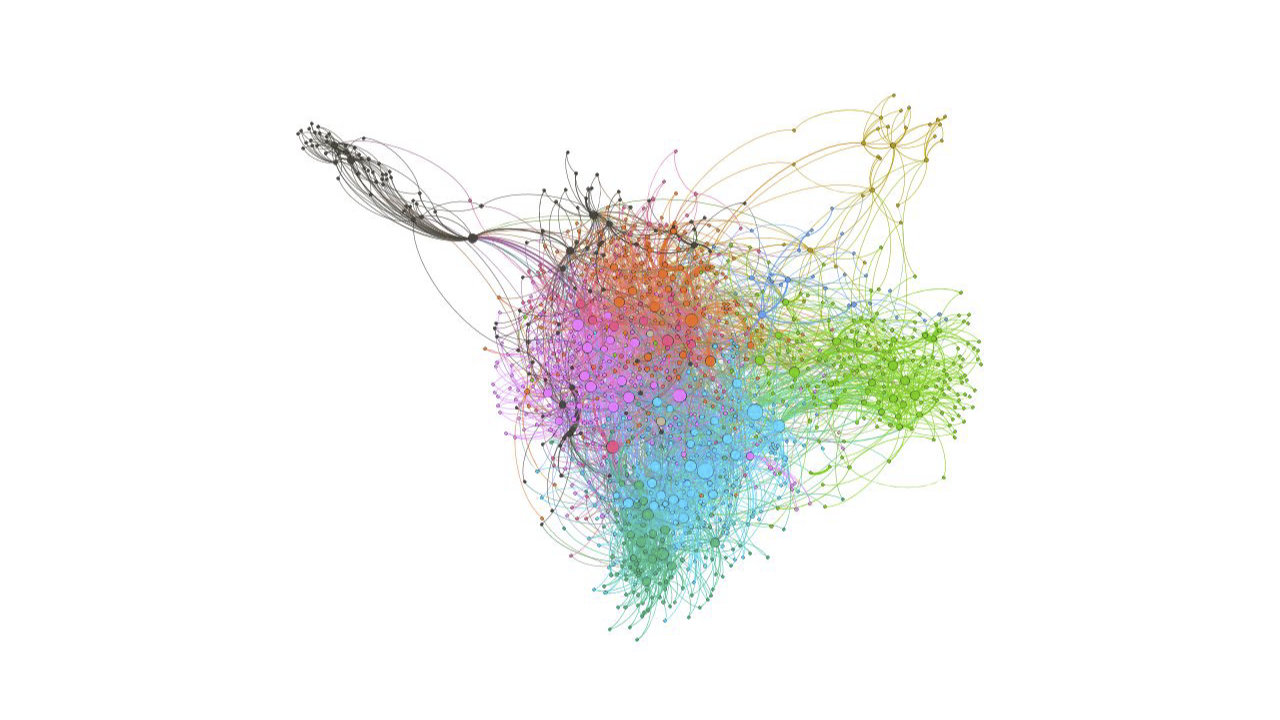
Graph Minig



Выявление неявных связей при анализе графов или как увидеть незримое.

Неявные связи в графах. Что это такое и с чем его едят? Разберем на примерах.

Начнем с небольшой теории. Все знают, что граф — это множество узлов объединенных множеством ребер. С множеством узлов все понятно взяли, к примеру города Росси, клиентов банка, компьютеры в сети и получили множество объектов, которые и будут являться узлами для графа. А что же с ребрами? На первый взгляд все просто: города соединены дорогами, клиенты совершают переводы денежных средств, а компьютеры постоянно обмениваются информацией, все же очень просто. Да, но нет. Все что мы перечислили относится к явным типам связей, то есть связь между объектами либо есть, либо ее нет. Так что же тогда неявные связи? Неявные связи немного сложнее, они могут зависть от явных связей, или же быть самостоятельными, например, возьмем к примеру 2-х людей которые работают на абсолютно разных работах, живут в разных концах города, на первый взгляд не имеют ничего общего, но при этом они оба по выходным приходят на матч любимой команды - это и есть неявная связь.

В графах все строится на сущностях и связах между ними. Так вот и поговорим о связях. Связи могут быть явными (например: транзакции, дороги между городами и т.п), иными словами, если присутствует прямой факт существования взаимосвязи между объектами. Неявные связи – это выделение связи между сущностями, которые имеют общие признаки (например: работники одного предприятия, фанаты баскетбольной команды и т.п).

Разберемся на примере гипотетической задачи. Есть 3 файла с данными:

1. Данные о звонках мошенников клиентам;
2. Жалобы клиентов на мошенников (например: у клиента украли определенную сумму денежных средств);
3. Транзакции клиентов.

P.S. Все данные синтетические и не имеют ничего общего с реальными.

Для демонстрации нам понадобятся следующие python-библиотеки:

* Pandas – для работы с файлами;
* NetworkX – для создания графа связей и визуализации;
* Matplotlib и Numpy – нужны для более красивой визуализации графа;

И так взглянем на данные.

Таблица 1 "Словарь данных"

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Описание** | **Тип данных** |
| Звонки | CustomerID | Уникальный идентификатор клиента | Int |
| Date | Дата звонка | DateTime |
| Number | Номер телефона | String |
| CustomerId | Уникальный идентификатор клиента | Int |
| Жалобы | Complaint ID | Уникальный идентификатор жалобы | Int |
| Date received | Дата жалобы | Дата |
| TRANSACTION\_ID | Уникальный идентификатор операции | Int |
| Транзакции | TX\_DATETIME | Дата и время операции | DateTime |
| CUSTOMER\_ID | Уникальный идентификатор клиента (отправитель) | Int |
| RECEIVER\_CUSTOMER\_ID | Уникальный идентификатор клиента (получатель) | Int |
| TX\_AMOUNT | Сумма транзакции | Float |
|  |  |  |

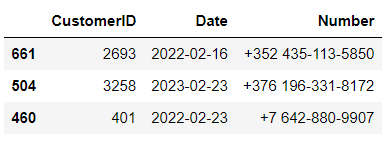


Рисунок 1 "Данные звонков мошенников клиентам"

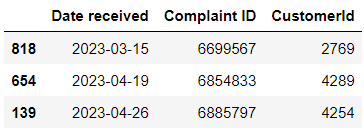


Рисунок 2 "Данные жалоб клиентов"

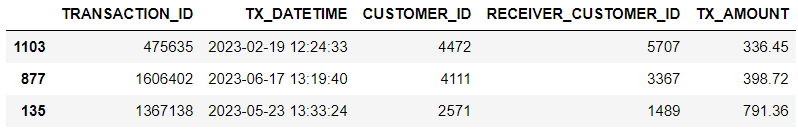
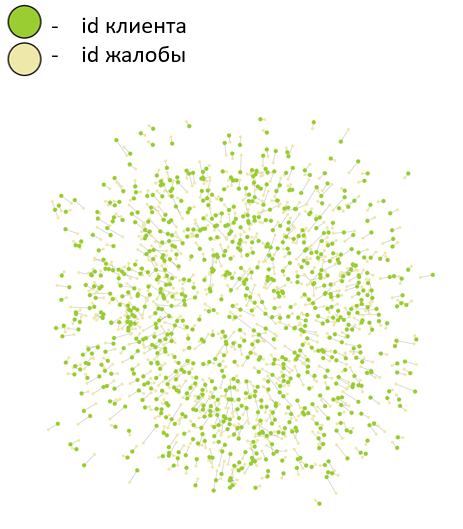
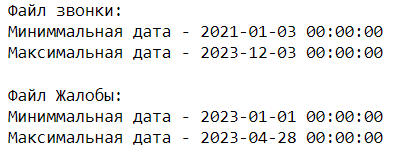


Рисунок 3 "Данные о транзакциях клиентов"

Для начала создаем граф, добавляем в него данные о клиентах и жалобах, дополнительно добавим атрибут времени жалобы.



Прежде чем добавлять данные по звонкам, необходимо сопоставить даты.



Исходя из полученной информации добавим данные только по звонках мошенников за 2023 год. Жертвы, которым звонили пометим оранжевым цветом.

