Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут ім. Ігоря Сікорського”

Кафедра автоматизації проектування енергетичних процесів і систем

Звіт

з лабораторної роботи №3

«ОБРОБКА ТЕКСТОВОЇ ІНФОРМАЦІЇ ТА ПРОЕКТУВАННЯ ОЗНАК

ЗАСОБАМИ SPARK»

Виконав:

студент 5-го курсу

ТЕФ

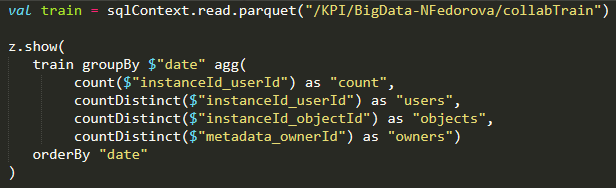
гр. ТМ-01мп  
Каліка Б.М.

Перевірив:

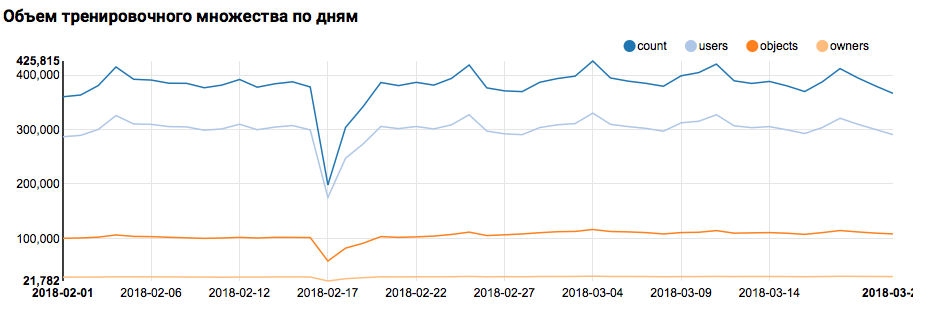
Проф. Федорова Н.В.

Київ – 2020

Із данних, що знаходяться в папці «KPI/BigData-NFedorova/collabTrain» обробляємо дані і групуємо їх по дням:

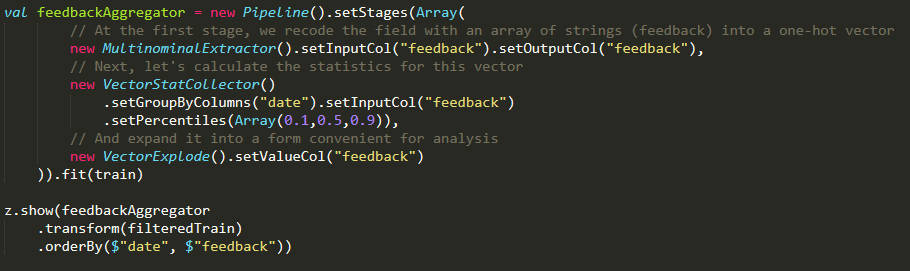


Із цих даних, за допомогою бібліотеки Zeppelin будуємо графік:



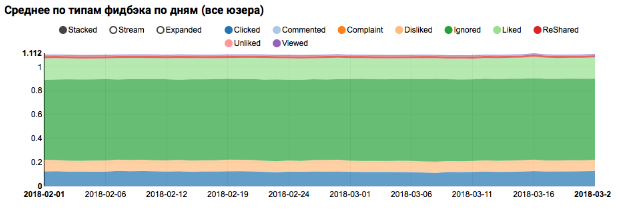
**Підрахунок основоної статистити по певним діям і ознаками за допомогою SparkML**

Для початку побудуємо простий конвеєр, за допомогою якого порахуємо статистику розподілу дій (поле feedback) користувачів в навчальній множині

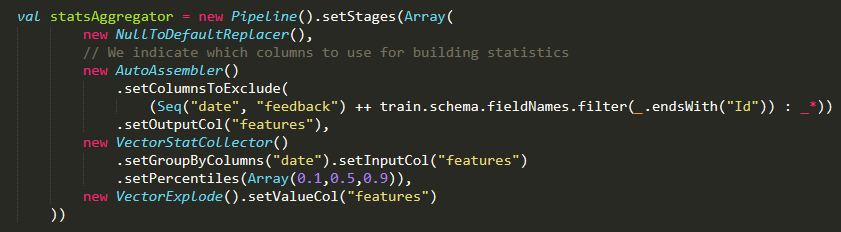


Результатом буде графік, який показує, що класи в датасета не збалансовані, однак дисбаланс для цільового класу Liked не є екстремальним

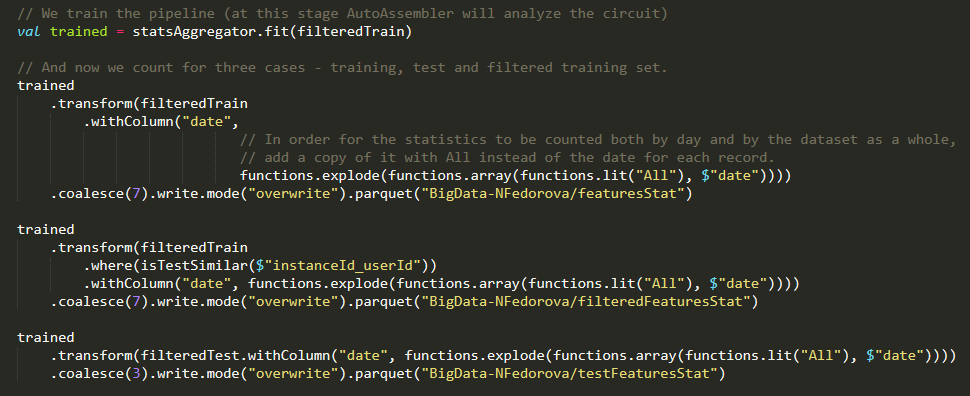
*Середнє по типам*



**Детальний статичний аналіз ознак**



Використовуючи отриманий конвеєр порахуємо статистику по трьом безлічі (навчальному, яка навчає з фільтром користувачів і тестового) і збережемо в окремі файли:



Отримавши три датасета зі статистикою ознак, проаналізуємо наступні речі:

- Чи є у нас ознаки, для яких спостерігаються великі викиди.

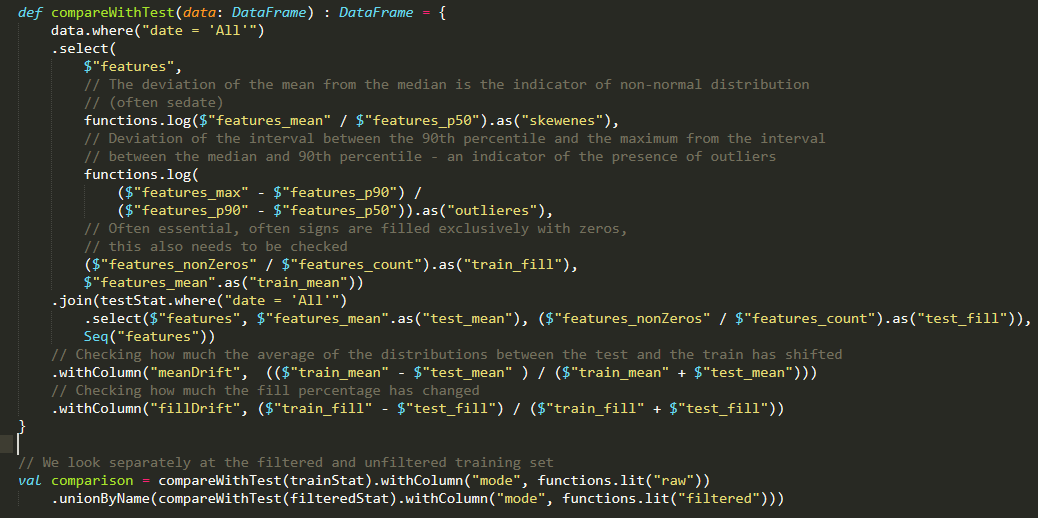
- Такі ознаки треба обмежити, або відфільтрувати записи-викиди.

- Чи є у нас ознаки з великим зміщенням середнього щодо медіани.

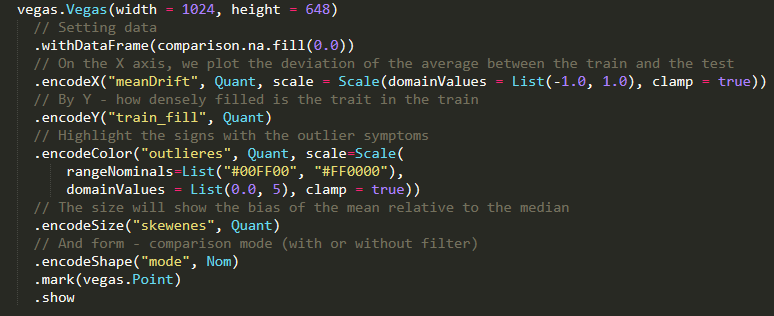
- Таке зміщення часто виникає при наявності статечного розподілу, ці ознаки має сенс логаріфмувати.

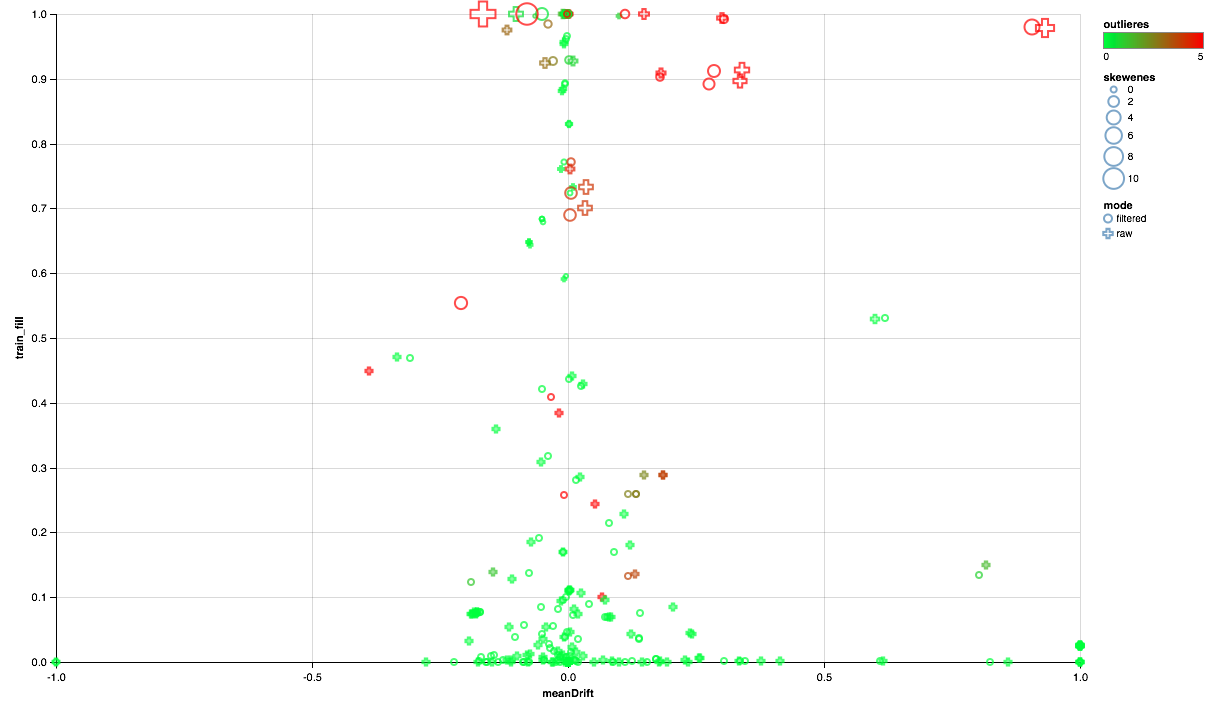
- Чи спостерігається зміщення середніх розподілів між навчальним і тестовим множествам.

Прояснити це нам допоможе такий запит:



Для побудови графіка скористаємося бібліотекою Vegas. Для цього налаштуємо певні конфігурації:





Пояснення до графіка:

По осі Х відкладено зміщення центрів розподілів між тестовим і навчальним множествам (чим ближче до 0, тим стабільніше ознака).

По осі У відкладений відсоток ненульових елементів (чим вище, тим для більшого числа точок є дані за ознакою).

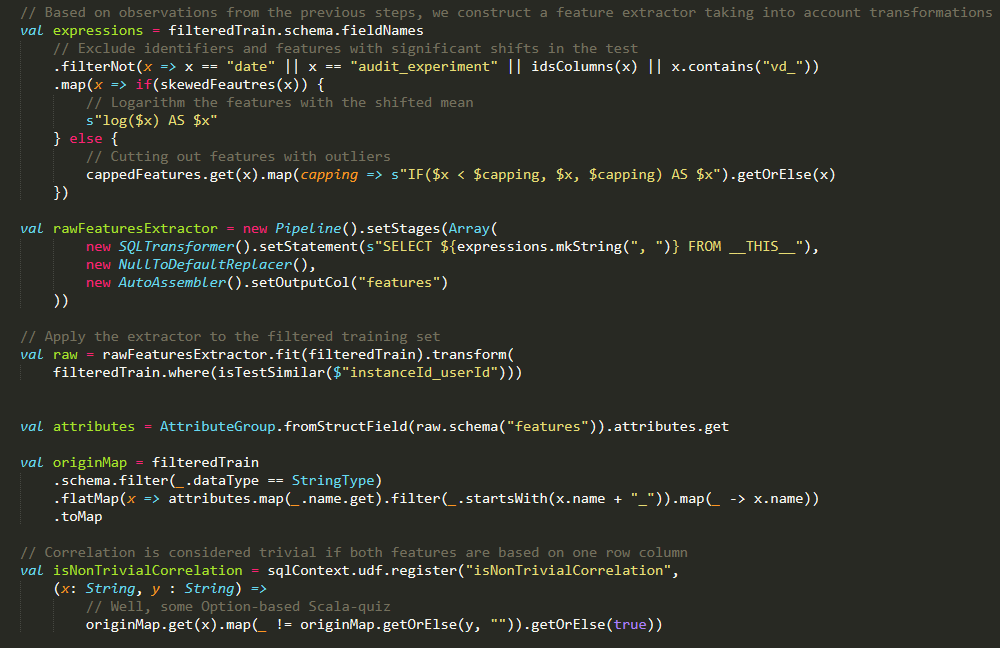
Розмір показує зміщення середнього щодо медіани (ніж точка більше, тим більш ймовірний статечної закон розподілу для неї).

Колір показує наявність викидів (чим червоніше, тим викидів більше).

Ну і форма відрізняє режим порівняння: з фільтром користувачів в навчальній множині або без фільтру.

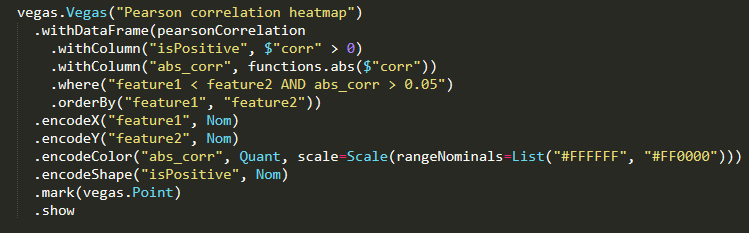
**Кореляційний аналіз**

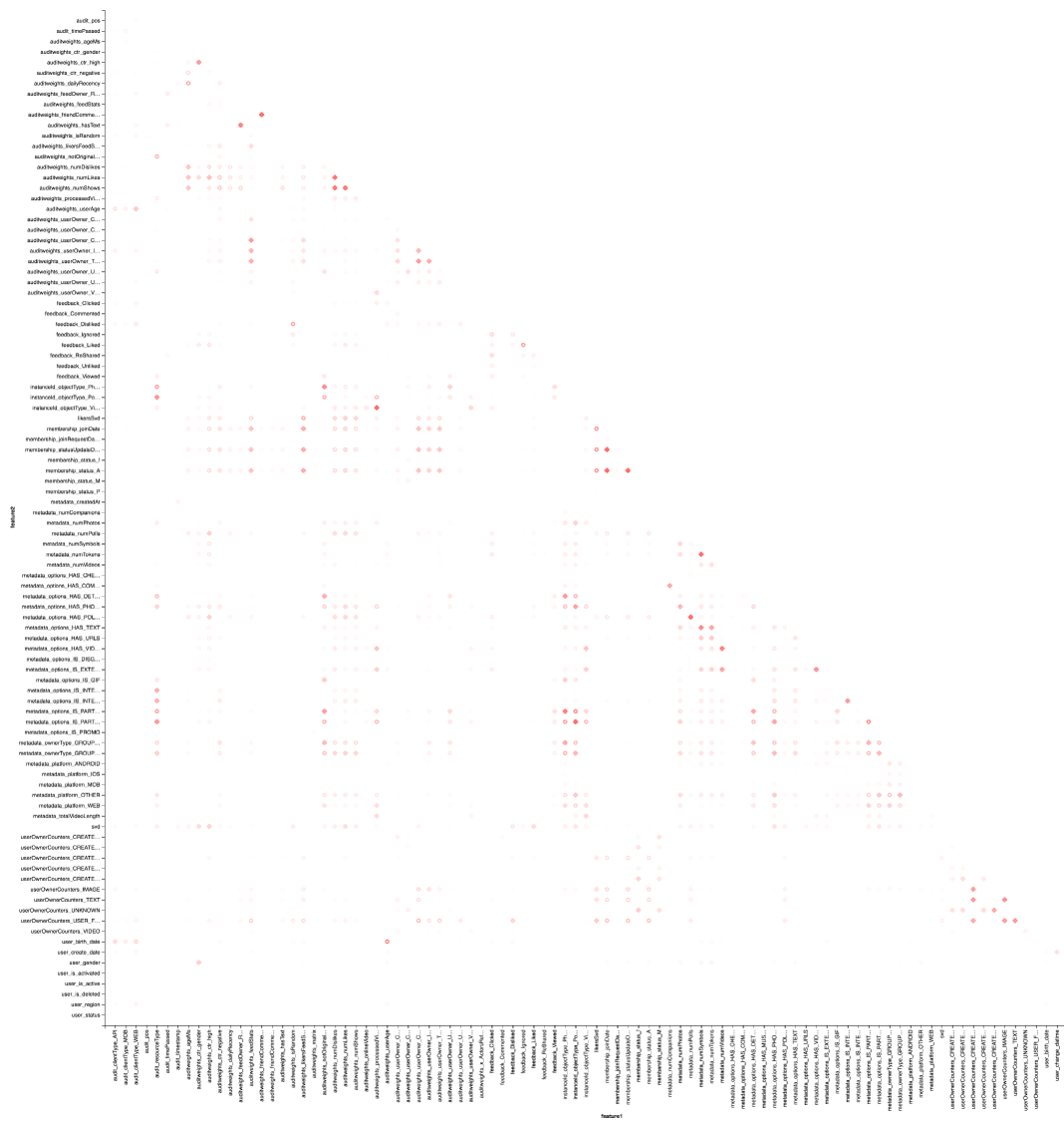
Після отримання уявлення про те, як розподілені ознаки і як вони співвідносяться між навчальним і тестовим множеством, спробуємо проаналізувати кореляції. Для цього налаштуємо екстрактор ознак на базі попередніх спостережень:



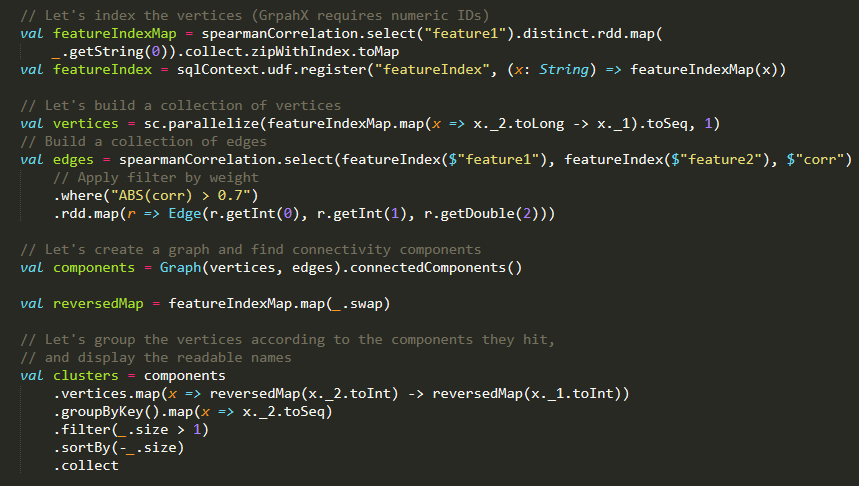
Маючі датасета з колонкою-вектором розрахувати крос-кореляції засобами Spark досить просто, але результатом буде матриця, для розгортання якої в наборі пар доведеться трохи підправити:

Візуалізовуємо за допомогою Vegas:

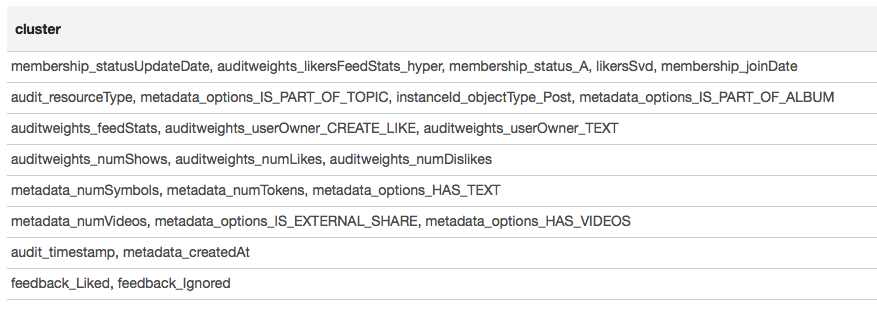




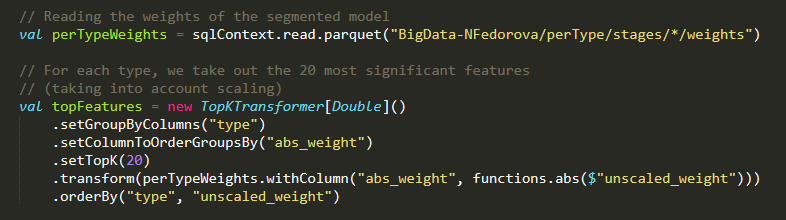
За теплової карті видно, що деякі кореляції явно є. Спробуємо виділити блоки найбільш сильно скоррелірованних ознак, для цього використовуємо бібліотеку GraphX: матрицю кореляції перетворимо в граф, відфільтруємо ребра по вазі, після чого знайдемо компоненти пов'язаності і залишимо тільки невироджені (з більш ніж одного елемента). Така процедура за своєю суттю аналогічна застосування алгоритму DBSCAN і виглядає наступним чином:



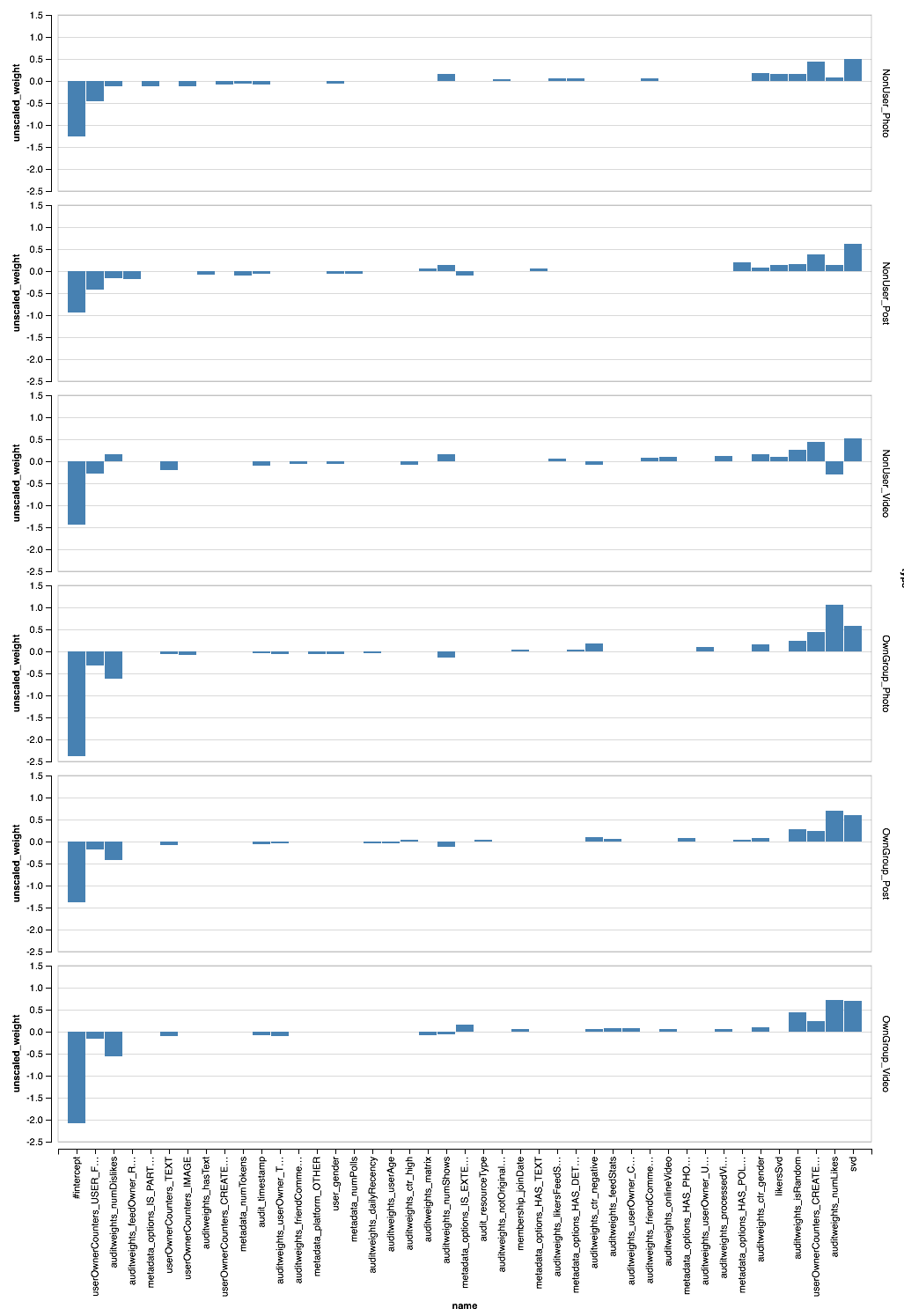
Результат представимо у вигляді таблиці:



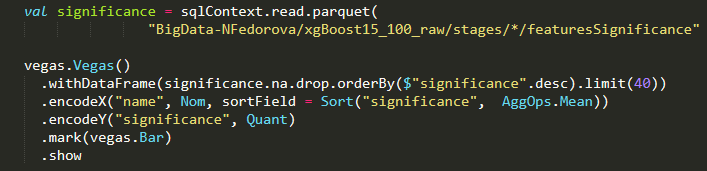
**Аналіз результатів**

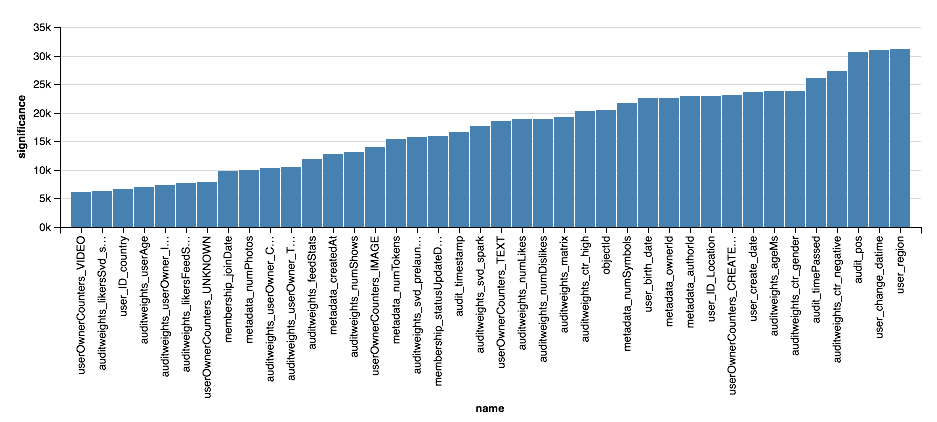


Отримані ваги визуалізуємо за допомогою Vegas



Видно, що сегментація дозволила по-різному розважити різні ознаки. Також можна заглянути всередину XGBoost





Можна помітити, що значимість ознак, на думку XGBoost, це просто кількість вузлів зі сплітом з цього атрибуту, що не завжди добре. Але той факт, що XGBoost часто вибирає для розгалужень ознаки, пов'язані з демографією користувача, наводить на думки.