Operation raccoon

Problem Definition

- FX markets are subject to volatility
- Some clients trade in different currencies
  - Spot-trades tied to transactions/GetRate
  - Some make efforts to manage risk with various levels of complexity
    - Forwards
    - Options / Structured Options
    - FX Derivatives
- Some clients do transactions based on FX movements:
  - Those who do: Predict transactions to arm salespeople with meaningful insights
  - Those who don’t: Find clients who may benefit from products based on volume

Also due to 2X multiplier on the board, all time-series based solutions will not be entertained

Hypothesis

If a client regularly makes a transaction based on a specific FX movement, *then* the next time this movement is observed, that client will make a transaction.

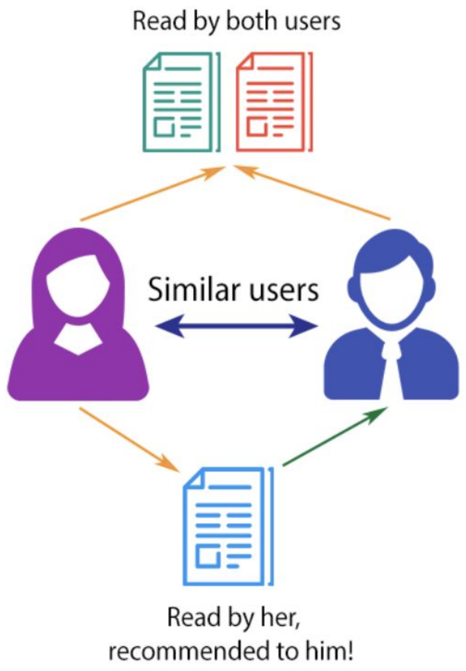
Data

- MI360 contains nearly 3 years of transactions by over 500,000 clients
  - `$_VA` indicates value of the transaction to HSBC, no rows NAN
  - `Client ID_` is an alpha-numeric identifier for a client, no rows NAN
  - `Transaction Date` is the date of a transaction, no rows NAN
  - Other columns that may be interesting are industry, geographic region, etc.
- KDB holds FX rates
  - Very rich information, highs and lows for every day for many currencies
  - Split up over many volumes and joins are spooky
- Currency Data also holds FX rates
  - `Date` col acts as a tag for the rest of the row
  - Every other entry is a ratio of currency value relative to USD at close

Goal

- Rank all clients by their propensity to make an FX related transaction
- Create a simplified output for salesperson, perhaps “Top 5” list
- For each client, describe what movement was interesting:
  - Arm salesperson with what movements raised flag for particular client
  - “Hello <client>, we’ve noticed that when <movement> happens, you usually <transaction>, just now <movement> has happened. May I help you with <transaction> today?”

Traditional Approach?

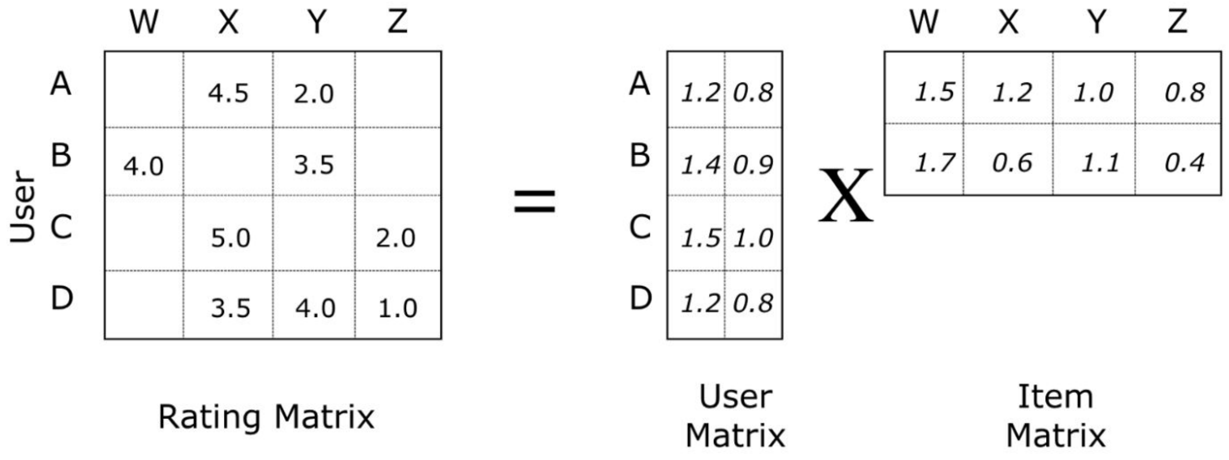


- K-Nearest Neighbours, the use of collaborative filtering

- Negatives:
    - Cold start problem
    - Popularity bias
    - Scalability issues

Inspiration

- Netflix had a similar problem with recommendations for movies for its users
- Matrix Factorization was used to solve this problem - works as below:



$$\begin{array}{c|cccc} & W & X & Y & Z \\ \hline A & & 4.5 & 2.0 & \\ B & 4.0 & & 3.5 & \\ C & & 5.0 & & 2.0 \\ D & & 3.5 & 4.0 & 1.0 \end{array} = \begin{array}{c|cc} A & 1.2 & 0.8 \\ B & 1.4 & 0.9 \\ C & 1.5 & 1.0 \\ D & 1.2 & 0.8 \end{array} \times \begin{array}{cccc} W & X & Y & Z \\ \hline 1.5 & 1.2 & 1.0 & 0.8 \\ 1.7 & 0.6 & 1.1 & 0.4 \end{array}$$

Matrix Factorization of Movie Ratings Data

Netflix Problem cont.

- “A **user** has these **ratings** for these **movies**, next time we see **user**, what **movie** should we suggest to maximize **rating**?”

Intuitively user and client are both IDs of *somebody*, ratings and FX Moves are ordinal, and movies, like transactions, are discrete events. By making the obvious substitutions we arrive at the problem being written:

- “A **client** has these **FX moves** for these **transaction volumes**, next time we see **client**, what **transaction volume** should we suggest to maximize **FX moves**?”

But these substitutions do not answer our goal (we are recommending clients, not volume)

Netflix Problem cont. cont.

- “A **user** has these **rating**s for these **movie**s, next time we see **user**, what **movie** should we suggest to maximize **rating**?”

Instead shoehorn the goal “we see FX move, what client should we suggest to maximize transaction volume” by substituting user for FX move, movie for client, and rating for transaction volume

- “An **FX move** has these **transaction volume**s for these **client**s, next time we see **FX move**, what **client** should we suggest to maximize **transaction volume**?”

Methodology

- Preprocess data by obtaining the total value added for each FX Move. E.g.

	USD+1	USD+2	USD-1	USD-2	GBP+1	GBP+2
Company A	1000	NaN	NaN	NaN	3000	5000
Company B	NaN	3000	NaN	2000	NaN	NaN

- Note: This is aggregated value added and it is time series independent
- Apply a matrix decomposition algorithm to predict the value of the NaNs
- Recommend the top 5 companies based on highest value added

Planned Roadmap

1. Extract, Transform, Load
   a. Both datasets into PySpark dataframe
   b. Throw out all unused cols, throw out rows beyond common date range
   c. Split data into training and verification sets still based on date
   d. Load into new dataframe of format needed for SVD
2. Train Model
   a. Create simplified models for both Movements and Clients
   b. Verify that trained models can predict matrix saved for verification (if not, panic)
3. Create Interface
   a. Enter new FX information
   b. Generate recommendations based on all moves (different than single user recommendation)
   c. Receive client suggestions and reasons they were chosen