Anti-Money Laundering Visualization

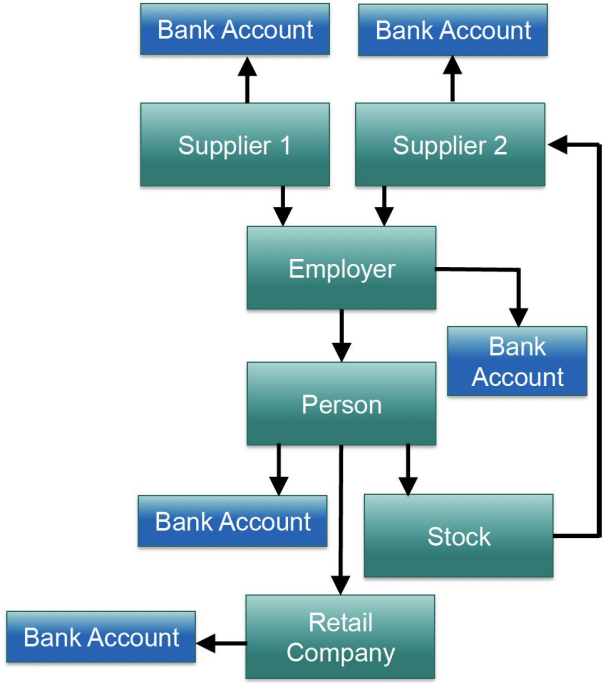
Group 4:
Debopriyo Ghosh (dg1114)
Nish Patel (nsp124)
Jahnavi Manchala (jm2658)

Mentor: Dr. James Abello
TA: Haoyang Zhang

Introduction

Introduction

- Money laundering detection is a critical issue in the financial sector
- Real financial transaction data is highly confidential and difficult to obtain for privacy reasons
- IBM has developed a simulation model to generate synthetic transactions
- The simulation includes simulated money laundering and these transactions are labeled in the dataset



Dataset

Dataset

- Posted by IBM on Kaggle
- ~17GB file size
- ~180 million records (~100 bytes per record)
- ~2.1M distinct bank accounts
- 15 distinct currencies
- 7 distinct payment formats
- Data was generated via IBM simulation from August 1, 2022 to November 5, 2022

Timestamp	From Bank	Account	To Bank	Account	Amount Receive	Receiving Currency	Amount Paid	Payment Currency	Payment Format	Is Laundering
9/1/22 0:22	800319940	8004ED620	808519790	872ABC810	120.92	US Dollar	120.92	US Dollar	Credit Card	0
9/1/22 0:05	8021ADE00	80238F220	9A7F59FA0	A23691240	33.97	US Dollar	33.97	US Dollar	Credit Card	1
9/1/22 0:14	801946100	8023F0980	83585F5A0	948893910	79.20	US Dollar	79.20	US Dollar	Credit Card	0
9/1/22 0:05	80010C840	800122AA0	80010C840	800122AA0	8,834.09	Euro	10351.64	US Dollar	ACH	0
9/1/22 0:05	80010C840	800122AA0	80010CF20	80012DA00	8,834.09	Euro	8834.09	Euro	ACH	0
9/1/22 0:08	80010CF20	80012DA00	80010CF20	80012DA00	9,682.16	US Dollar	8262.75	Euro	ACH	0
9/1/22 0:08	80010CF20	80012DA00	80010BD60	80011E460	9,682.16	US Dollar	9682.16	US Dollar	ACH	0
9/1/22 0:03	800319940	800466670	80029A010	8002F6F20	9,125.22	US Dollar	9125.22	US Dollar	ACH	0

Data Preprocessing

Data Preprocessing

- Performed data cleaning and preprocessing steps to prepare a csv to be used in the analysis
  - Check for nulls (no null values in any column)
  - Calculate conversion rates from each currency to USD and prepare a new column of payments in USD
    - ➢ This is useful to scale payment columns so transactions can be compared across currencies
  - Create a unique id by hashing bank + ‘_’ + account
    - ➢ Provides a way to have unique ids across banks/accounts
  - Convert timestamp columns into integer year, month, day, hour, minute columns to save space
- Output “clean” csv for use in visualizations and modeling

Fundamental Questions

Fundamental Questions

- Currently, the industry relies heavily on manual analysis of transactions in order to flag and review fraudulent transactions with a high false positive rate
- As data volume has increased and new digital currencies have been introduced, the challenge of identifying potential money-laundering transactions with legacy rules-based has increased exponentially
- The three fundamental questions we are asking are:
    1. **What is the best data type for financial transaction data?**
    2. **How can we help analysts perform ad-hoc analysis of financial transaction data to identify possible accounts of interest?**
    3. **Once the accounts of interest are found, how can we help the analyst review financial transactions conducted by the accounts and their related accounts?**

Graphs

Graphs

- To answer fundamental question #1, **What is the best data structure for financial transaction data?,** a literature review was conducted to identify the current trends in the space
- Legacy rules-based analysis relied on the table data structure within relational databases and pre-defined thresholds for identifying potential fraudulent transactions ($10,000+ transactions must be reported to the IRS)
- The key component that was missing from these systems was the ability to easily view the context of the transactions and the accounts that money is being transferred from/to

Graphs

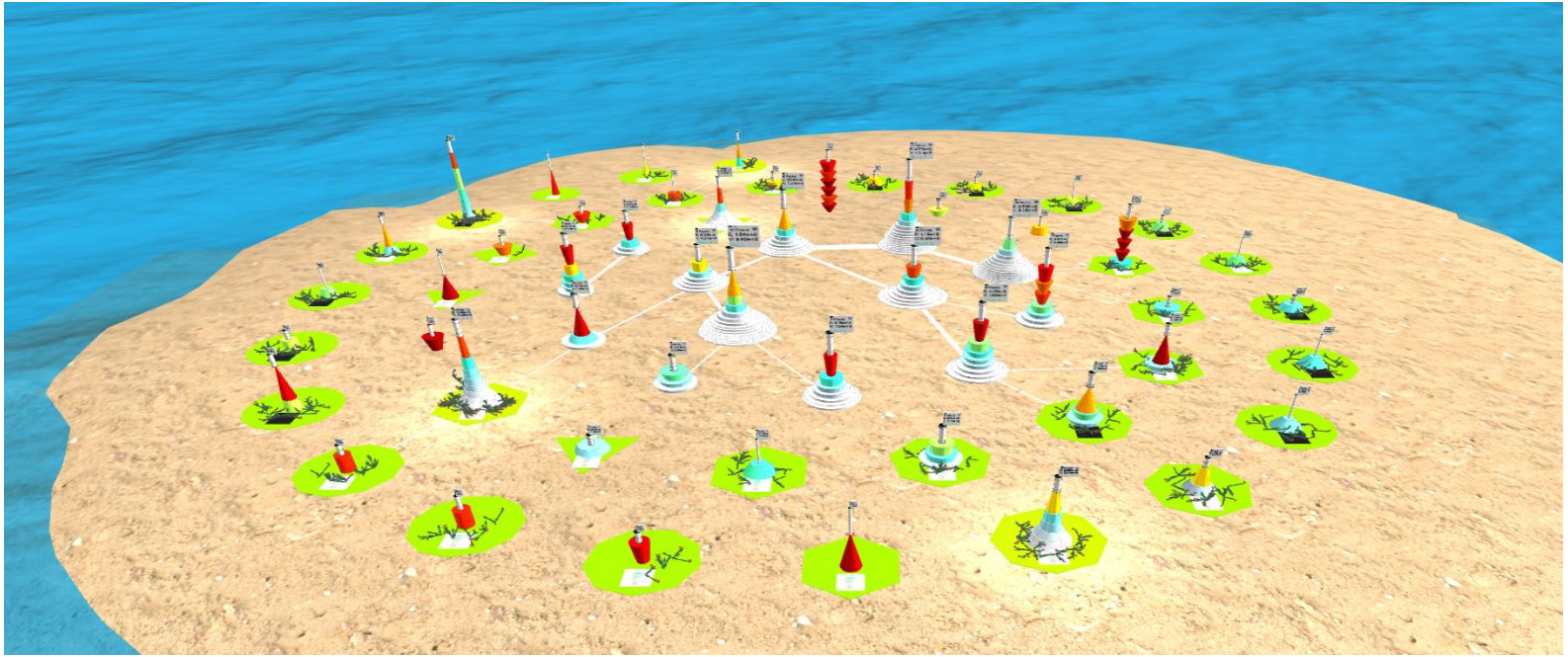
- As graph analytics have evolved due to the rise of social networks, the financial industry has also embraced graphs to model financial transactions as a series of relationships to provide the missing context
- Based on the literature review, we decided to utilize the graph data structure to model our data
- Given the need to store, query, and visualize graph data, the data was converted into nodes/edges via a python script and imported into a graph database (neo4j) that can natively handle the relationships and provide an efficient query mechanism

Graph City

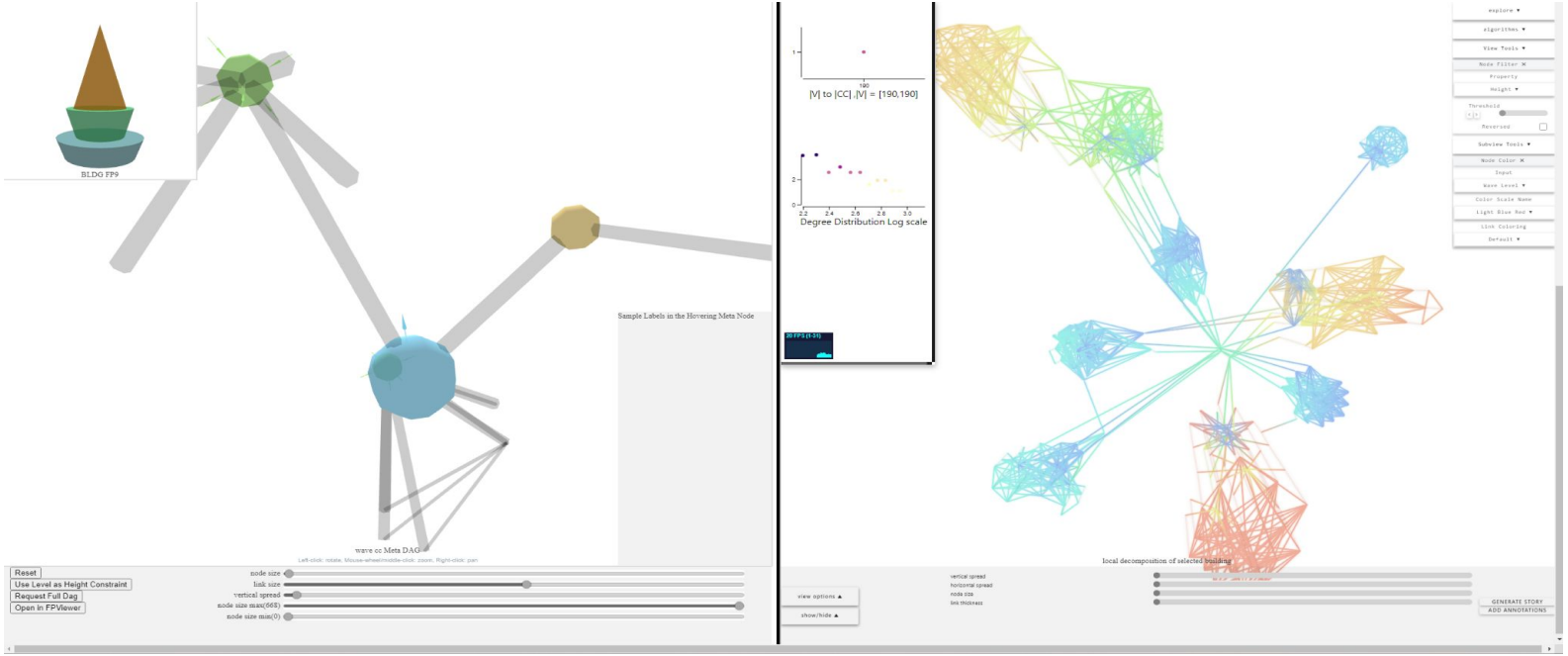
Graph City

- To answer fundamental question #2, **How can we help analysts perform ad-hoc analysis of financial transaction data to identify possible accounts of interest?,** the main issue to address was the screen bottleneck
- With massive graph networks, it is impossible for the analyst to visualize/analyze the graph using a traditional visualization
- Despite this, analysts will need to perform ad-hoc review of accounts to identify potential accounts of interest
- Visualizing the data with a graph city solves this problem by allowing analysts to see the entire graph and perform analysis one building at a time

Graph City



Graph City

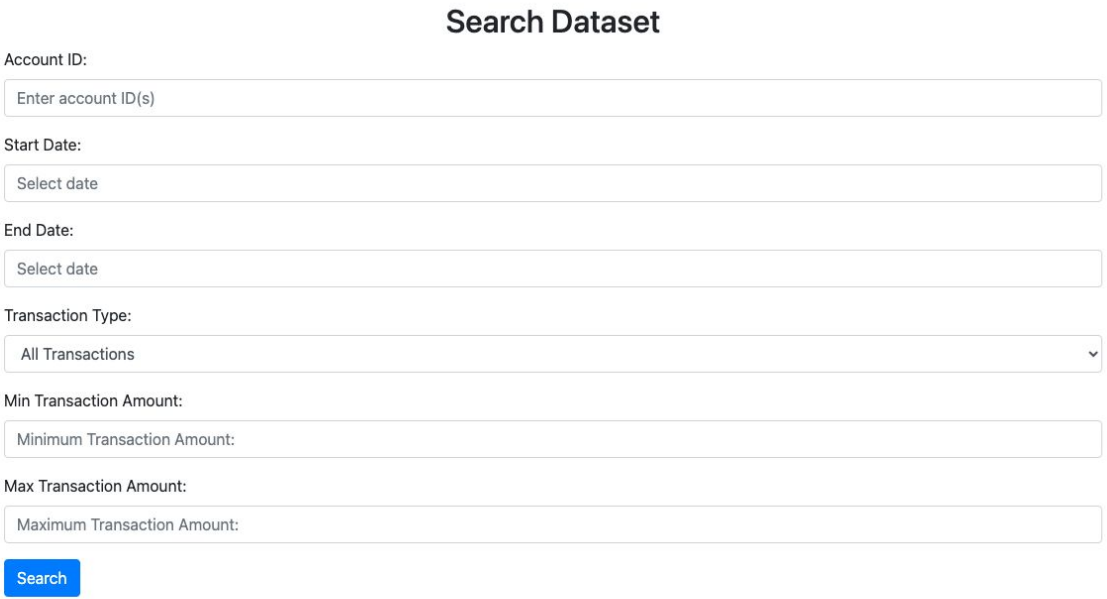

BLDG FP9
Sample Labels in the Hovering Meta Node
wave co Meta DAG
Reset
Use Level as Height Constraint
Request Full Dag
Open in FPViewer
node size
link size
vertical spread
node size max(668)
node size min(0)
|V| to |CC| .|V| = [190,190]
Degree Distribution Log scale
local decomposition of selected building
view options
show/hide
GENERATE STORY
ADD ANNOTATIONS

Search Application

Search Application

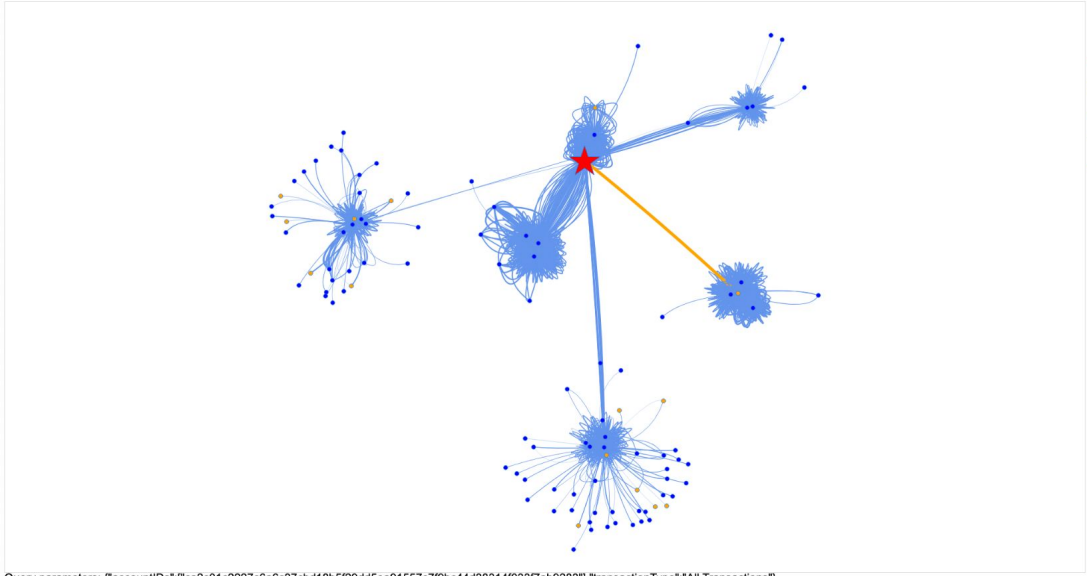
- To answer fundamental question #3, **Once the accounts of interest are found, how can we help the analyst review financial transactions conducted by the accounts and their related accounts?,** we needed to figure out how to leverage neo4j's advanced capabilities to allow analysts to view transactions for particular accounts and their related accounts.
- This was done by creating a search application that allows the user to filter on certain key attributes and visualize a graph of an account (or series of accounts)

Search Application



Search Application

- Once the query is submitted, a graph visualization is rendered that shows all transactions by that account and also any transaction by accounts that the account sent money to (depth 2)



Query parameters: {"accountIDs":["ca2c01c2227c6a6c37cbd18b5f29dd5ea91557c7f9bc44d38314f033f7ab9283"],"transactionType":"All Transactions"}
Query time taken: 7.30 seconds; Records processed per second: 23745880
# of Edges: 43036; # of Nodes: 106

- An interesting finding is that communities of interest are readily apparent when visualizing the accounts through the search application

Project Architecture

Project Architecture

aml.txt
aml_label.csv
(csv)

python

Raw Data
(csv)

pySpark

clean_data.csv
(csv)

python

accounts.csv
transactions.csv
(csv)

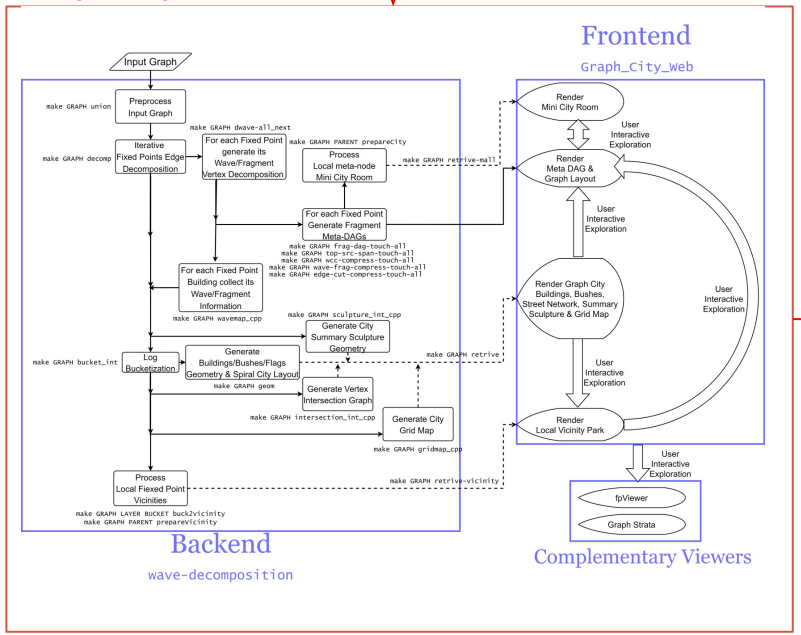
neo4j import function

neo4j

Node.js server

Search application
(javascript, neovis.js)

Graph City

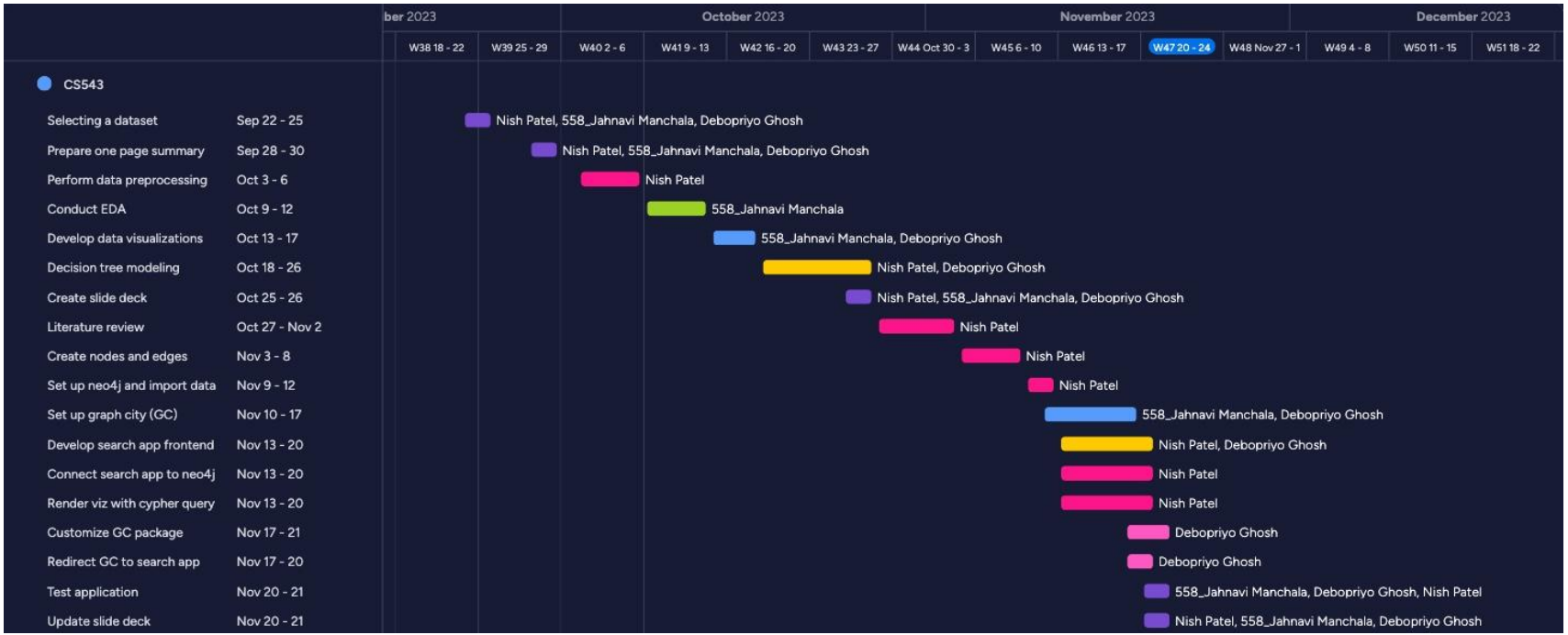


Project Architecture

- Node attributes:
  - accountID
  - is_laundering
  - bank
- Transaction attributes:
  - amount_usd
  - bank_from
  - bank_to
  - currency_from
  - currency_to
  - is_laundering
  - payment_format
  - year, month, day
  - hour, minute

Gantt Chart Of Project Timeline

Project Timeline



Project Demo

References

References

- https://ibm.ent.box.com/v/AML-Anti-Money-Laundering-Data/file/780515045707
- https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml
- https://datawalk.com/whitepaper-graph-analytics-the-new-game-changer-for-aml/
- https://neo4j.com/developer-blog/graph-visualization-with-neo4j-using-neovis-js/
- https://github.com/neo4j-contrib/neovis.js

References

- https://github.com/endlesstory0428/Graph-Cities
  - @article{Abello2022GigaGC, title={Giga Graph Cities: Their Buckets, Buildings, Waves, and Fragments}, author={James Abello and H. Zhang and Daniel Nakhimovich and Chengguizi Han and Mridul Aanjaneya}, journal={IEEE Computer Graphics and Applications}, year={2022}, volume={42}, pages={53-64} }
  - @inproceedings{Abello2021GraphCT, title={Graph Cities: Their Buildings, Waves, and Fragments}, author={James Abello and Daniel Nakhimovich and Chengguizi Han and Mridul Aanjaneya}, booktitle={EDBT/ICDT Workshops}, year={2021} }
  - @article{Abello2020GraphW, title={Graph Waves}, author={James Abello and Daniel Nakhimovich}, journal={Big Data Res.}, year={2020}, volume={29}, pages={100327} }
  - @article{Abello2013FixedPO, title={Fixed points of graph peeling}, author={James Abello and François Queyroi}, journal={2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)}, year={2013}, pages={256-263} }

Project 2

Project 2 - Nish

- For project 2, I will be continuing on with this dataset and will attempt to utilize graph convolutional neural network to predict if a transaction is a money laundering transaction or not
- A key benefit of using graphs as the data type (as mentioned in question 1) is to be able to utilize the relationships between accounts to identify potential fraud
- Given 90%+ false positive rates using traditional methods, graph convolutional neural networks should be able to utilize the context from the relationships to produce lower false positive rates
- I will be working individually for Project 2

Project 2- Debopriyo

- I have decided to use the DIV2K dataset (https://data.vision.ee.ethz.ch/cvl/DIV2K/) to implement Image Super-Resolution Using TensorFlow.
- My goal is to develop a deep learning model for image super-resolution, enhancing the quality and resolution of low-resolution images.
- Implement a convolutional neural network (CNN) architecture suitable for super-resolution
- Develop a user interface where users can upload low-resolution images and receive enhanced versions
- I will be working individually for Project 2

DIV2K Dataset

The DIV2K dataset is divided into:

1000 2K resolution images divided into: 800 images for training, 100 images for validation, 100 images for testing

- train data: starting from 800 high definition high resolution images obtain corresponding low resolution images and provide both high and low resolution images for 2, 3, and 4 downscaling factors
- validation data: 100 high-definition high-resolution images are used for generating low resolution corresponding images
- test data: 100 diverse images are used to generate low resolution corresponding images

Project 2 - Jahnavi

- For project 2, i will be working on Plant Village dataset from Kaggle to detect diseases in potatoes.
- I will be using CNN to build a model.
- Then i will build a mobile app using which a farmer can take a picture and app will tell you if the plant has a disease or not.
- This dataset consists of around 25000 images of potato leaves divided as healthy, Light-blight and heavy-blight.
- I will be dividing into 64 batches , each containing of 32 samples and i will be using 80% of data to train, 10% data for validation and the rest 10% to testing.
- I will be working individually for project 2.