**Final Report: Pattern and Anomaly=Based Detection of Potential Money Laundering Transactions and Entities**

**Group members: Nathan Fleming**

**Introduction**

As anti money laundering (AML) laws become more complex, so do the solutions to comply with the laws. The government relies on companies to be able to find and prevent fraudulent transactions that promote elicit enterprises. As banking has become more complex, the old simple solutions cannot adequately address the billions of transactions that are processed every day. To address this problem, DataBricks created a solution that leverages the lakehouse data structure. The lakehouse data structure allows us to store large amounts of data cheaply, and allows us to interact with the data simply, like a data warehouse. This solution uses graph theory, data correlation, and network analysis to find and track fraudulent transactions. For my project I will use databricks and Amazon Web Services (AWS) to implement a solution to find fraudulent transactions and entities. Once this is complete, I will tune the solutions to try to find fraudulent schemes faster.

**Related Work**

A literature review of AML techniques for identifying money laundering yielded several paper that increased my understanding of the issue.

1. Chen, Z., Van Khoa, L.D., Teoh, E.N. et al., “Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection”

This paper helped with data preparation, data transformation, and data analytics techniques. This paper helped gain an understanding of standard techniques used in the industry.

2. K. Plaksiy, A. Nikiforov and N. Miloslavskaya, "Applying Big Data Technologies to Detect Cases of Money Laundering and Counter Financing of Terrorism”

This paper showed how graph theory can help tie entities together in ways that traditional SQL cannot.

3. R. I. T. Jensen and A. Iosifidis, "Fighting Money Laundering With Statistics and Machine Learning”

This paper gave a great summary of typical patterns used to detect ML.

**Dataset**

This Project consists of transactions and entities. Transactions is a database consisting of 14,532 rows and twelve columns. The first column is txn\_dt, it consists of a date in the format of yyyy-mm-dd. The second column is txn\_id a identifier that is an integer. The third column is originator\_id, an integer that identifies where a transaction came from. Fourth is a beneficiary\_id, an integer that signifies where the transaction is going. Fifth is txn\_amount, the amount of the transaction. Fifth is originator type, a string that identifies the type of entity that sent the transaction. Sixth is beneficiary\_type, a string that represents where the transaction is going. Seventh is rptd\_originator\_address, a string that represents the address of the entity sending the transaction. Eighth is rptd\_beneficiary\_address, a string that represents the address of the entity receiving the transaction. Ninth is rptd\_originator\_name, a string that represents the name of the entity sending the transaction. Tenth is is rptd\_beneficiary\_name, a string that represents the name of the entity receiving the transaction. Eleventh is originator\_bank\_country, a string representing the country the transaction came from. Twelve is beneficiary\_bank\_country, The country that the transaction went to.

The second database named entities consists of 10108 rows each with seven columns. The first column is address, a string for the address of the entity. The second column is email\_addr, a string for an email associated with the entity. The third colum is entity\_id, an int that gives an id number to the entity. The fourth column is name, a string for the name of the entity. The fifth column is phone\_number a string for the phone number. The sixth column is risk score, an int where 0 is low risk and 10 is high risk. The last column is entity\_type, a string to describe the type of the entity.

**Methodology**

There are many different ways to try and find suspicious activity, but the first solution is resolving synthetic identities. To start, a simplistic SQL query looking for shared emails is used, but this is a very surface level check that is very simply thwarted. To get a better understanding of the relationships in the data, the GraphFrame library is used to create a graph of all the columns in the entity database. From there an edge is added from that node to the nodes adjacent to the column in the row. This lets us visualize the interconnectedness of the data. From there, all the edges that are leaves are dropped because they give us no insight into how the data connects. In the original solution, only leaves are dropped, however, on subsequent revisions if a node did not have a combined degree of four it was dropped to see if the solution could be accelerated. From this point, an SQL query is run to identify suspicious nodes and it is given a score based on how connected the entities are. The number of entities flagged is then compared using the seaburn library.

A second method for money laundering is smurfing or structuring. This occurs when a network of bad actors all pass small sums of money within a network of banks till they all end at the same place. Bad actors use this method to avoid being flagged as suspicious because of the low individually transferred sums. Graph theory can again be used to create alerts for this suspicious behavior. To start, we again create a graph of all the entities. Within a graph, a motif is created to represent the behavior we are looking for. To start, a motif of four individuals sending money to a final person is used but later a motif checking for 3 people sending money to a bad actor is compared to see what transactions can be caught.

A third method for money laundering is round tripping. To make money seem legitimate, bad actors send money out and in a circle the money comes back so the actor can make the income seem legitimate. To detect this pattern, we can use the same graph we created for smurphing and simply create a different motif and find it in the graph.

A fourth way to prevent money laundering is risk score propagation. Risk score propagation is the process by which an entity that has a low risk score itself but is connected to entities with high risk scores receives a high risk score itself because of those associations. This prevents money launders from using long transaction chains to separate them from suspicious behavior. To demonstrate this prevent method, we again go back to our graph, but we use the coalesce and sum function from pyspark to propagate risk scores from the leaves to the node creating a sum of each node’s own risk and the risk of all nodes it is attached to.

The last way the notebook attempted to find money laundering is with entity resolution. In this section, the splink library was used to catch money launders changing values of entities slightly to avoid the data being correlated. For example if two entities have an email nathanemail@gmail,com and [nathanema1l@gmail.com](mailto:nathanema1l@gmail.com) with addresses “123 street street” and “124 street street” these intities are probably fraudulent because someone is trying to slightly very data of another entity to get more entities available to launder money. The original notebook only used a database of 4 entities to display how someone might catch the scheme. I used splink to try and flag suspicious transactions and got results. However, without any baseline technique to compare to, this experiment is impossible to evaluate.

**Evaluation**

The evaluation of this model is very difficult because the data is missing any baseline for accuracy. Because of that, I chose to change the parameters the model uses and compare the number of entities flagged and the amount of time it takes to run the computation over multiple trials across the methods.

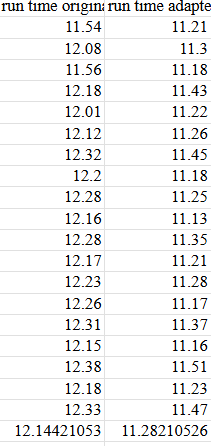
**Results**

The first change that we wanted to calculate was pruning less connected pieces from the graph. Given that money launders typically use multiple entities, I had a hypothesis that removing more than just the leaves of the graph would speed up calculations and while still being able to find suspicious transaction.

A computer screen shot of text

Description automatically generated

Below Are the runtimes compared.



After making the changes and doing the rest of the calculations, the new method caught 100% of the suspicious entities with a synth score higher than two. Below is a bar graph created by seabrin.

A graph of a bar graph

Description automatically generated

As you can see the most likely perpetrators are still caught with the smaller graph size. Without a dataset to tell us if the low synth scores are money launders or false positives, it is impossible to tell if we missed important information.

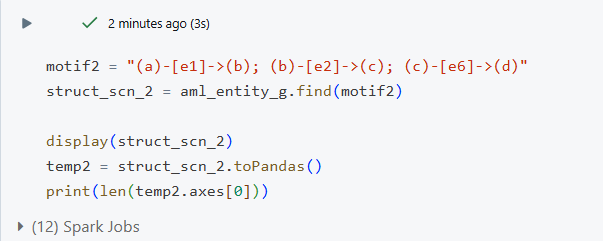
To test the computation time, I performed 20 trials on the same dataset to see if the change sped up computation. I got the following results

A screenshot of a computer screen

Description automatically generated

As you can see, a dependent samples t-test was conducted to examine the difference in run time of the larger entity graph (mean run time =12.14 Standard Deviation .22) and the smaller entity graph (mean run time= 11.28 Standard Deviation .11) for the same sample of 20 runs. The analysis revealed a significant decrease in run time, t(19) = 19.73, p < .001. The Cohens d effect size is large d=4.41 indicating a substantial difference in run time between the two methods on the same date sets. This shows that being more selective in the connectedness of the graph we create can help decrease the run time and increase efficiency. This may not seem significant, but because of the large volume of transactions and entities, a significant speed increase within this small data set shows that with a larger data set a substantial amount of time could be saved.

The second hypothesis I had for speeding up AML calculations was to create simpler motifs to determine structuring or smurfing.



Once this simpler motif was created, I joined it to the original motif to see if a simpler motif would catch all instances of a more complex motif.

A close-up of text

Description automatically generated

Upon running the comparison, the smaller motif caught 100% of the 3844 instances identified with the larger model and found 902 more instances. After seeing that the smaller motif identified more transactions as potentially problematic, I wanted to compare the run time for finding the motifs. I got the following results.

A screenshot of a table

Description automatically generated

A screenshot of a computer screen

Description automatically generated

As you can see a dependent samples t-test was conducted to examine the difference in run time of the larger motif (mean run time= .2153 Standard Deviation .004) and the smaller motif (mean run time = .04 Standard Deviation 0). for the same sample of 20 runs. The analysis revealed a significant decrease in run time. t(19) = 156.98, p < .001. The Cohens d effect size is 35.101 which is extremely large and indicates a substantial difference in run time. The increased number of suspicious transactions found and the time decrease of the computation gives credence to the hypothesis that a smaller search motif may be more efficient. One disadvantage to a smaller search motif is that the number of entities found with the technique decreases. This gives investigators less information to go use, but these entities can be quickly found once we have identified suspicious transactions.

**Conclusion**

In conclusion, I started this project with the goal of being able to use Databricks to identify fraudulent transactions. Once that was completed, I wanted to find ways to reduce computation time and still find accurate results. To this end, I used the preapproved data bricks project to find fraudulent transactions. From there I tested different ways of improving computation time and found that reducing the graph motif size increased the number of flagged transactions and reduced running time. I also determined that pruning the entity graph more aggressively before running computation increased performance and did just as well at catching highly suspicious transactions but not as well at catching transaction with low suspicion scores.

**References**

1. Chen, Z., Van Khoa, L.D., Teoh, E.N. *et al.* Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection
2. K. Plaksiy, A. Nikiforov and N. Miloslavskaya, "Applying Big Data Technologies to Detect Cases of Money Laundering and Counter Financing of Terrorism
3. R. I. T. Jensen and A. Iosifidis, "Fighting Money Laundering With Statistics and Machine Learning
4. [PySpark Overview — PySpark 3.5.3 documentation](https://spark.apache.org/docs/latest/api/python/index.html)
5. [splink · PyPI](https://pypi.org/project/splink/)