# Data Mining Homework 4

Fabio Fraschetti 1834942

January 2024

[Fabio Fraschetti]

## 1 Exercise

In this report I will describe the choices that I made in the python code DMHW4.py. First of all I started with data analysis of the dataframe loaded from kaggle. I saw that there aren't null values in each column and cancelled the columns that aren't useful for fraud-detection, that are oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest' and 'newbalanceDest, because it's written in the description of the dataset on kaggle. I checked the length of the dataframe, that is around 6 million rows that converted in a graph is huge, so I find what are the transaction types with more frauds and I saw that only Transfer and Cash Out have frauds. Because the most easy way to commit a fraud is to transfer the money into another account and make cash out to take cash without traceability because it's made from different accounts. So deleting all the other rows with other types of transactions, I reduced the dataframe size from 6 million to 2.7 million, of which only 8213 are fraud transactions, that is the 0.3% transactions. Anyway lets continue with the data analysis. I changed the types that were strings into 0 and 1 because now they are binary (Transfer and Cash Out) and mapped the ID origin and dest into incremental numbers too. I have done this to give those information into the model later. Now let's analyze the graphs that I made.

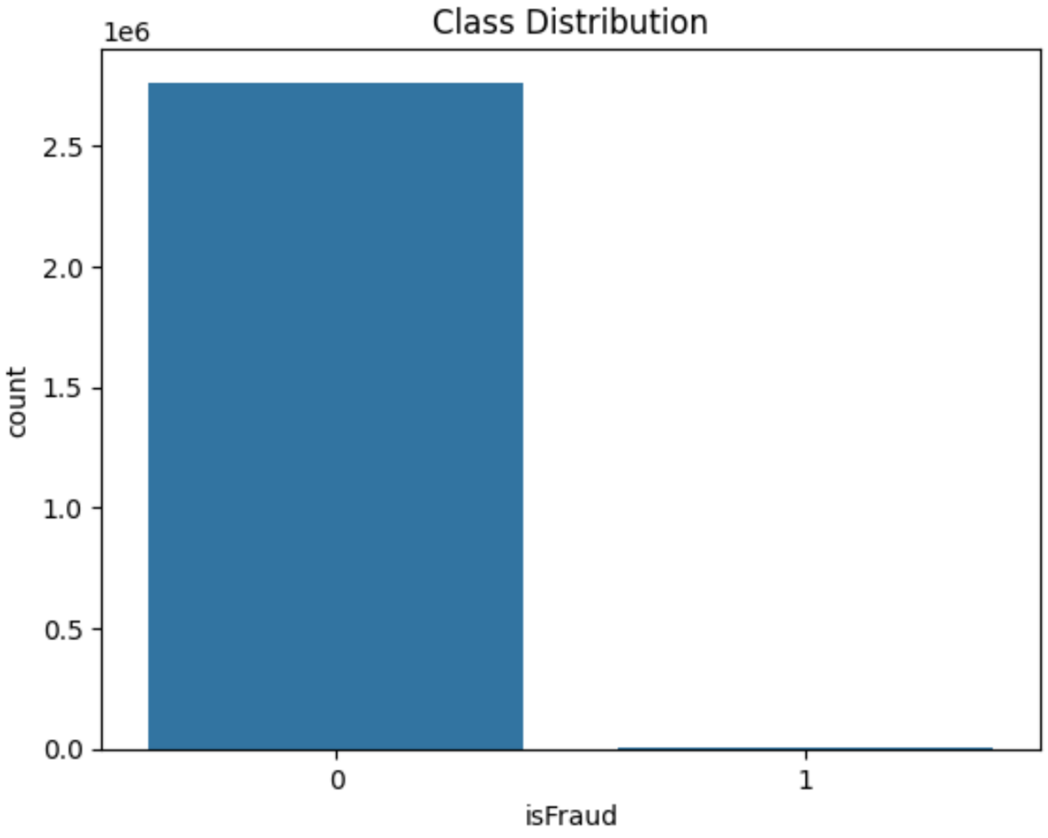

Figure 1

As I said before this is the graphic of the previous counts of frauds transactions respect to non-fraud transactions. I think this is one of the main problem of this system. I also to reduce the amount of transactions from 6 million to the half but it's not enough. I also tried to import some model to perform data augmentation but in the end I thought that if this is the real percentage of frauds I should work with this.

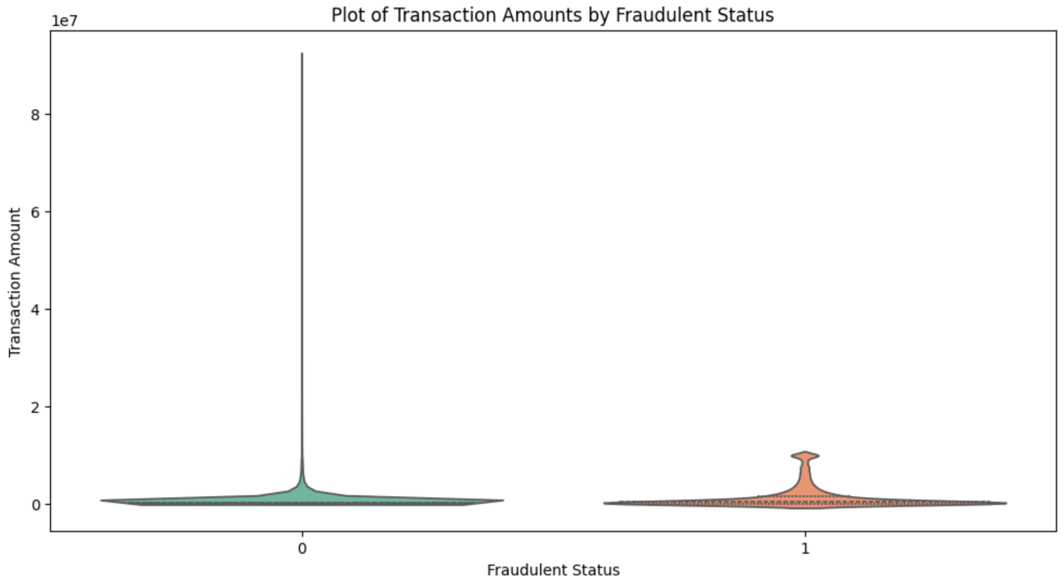

Figure 2

I think this is an important graphic to show because here we can see the correlation between the amount and the number of times that this amount appear in the dataframe represented by the width of the line, divided into fraud transactions and not fraud. For example here we can see that the non fraud transactions can have any amount, obviously greater is the amount less is the number of times that a transaction have this amount. While in the fraud transactions we have a great width for the small amounts but we have a bound to 10 million. This should be one of the things that the model should learn.

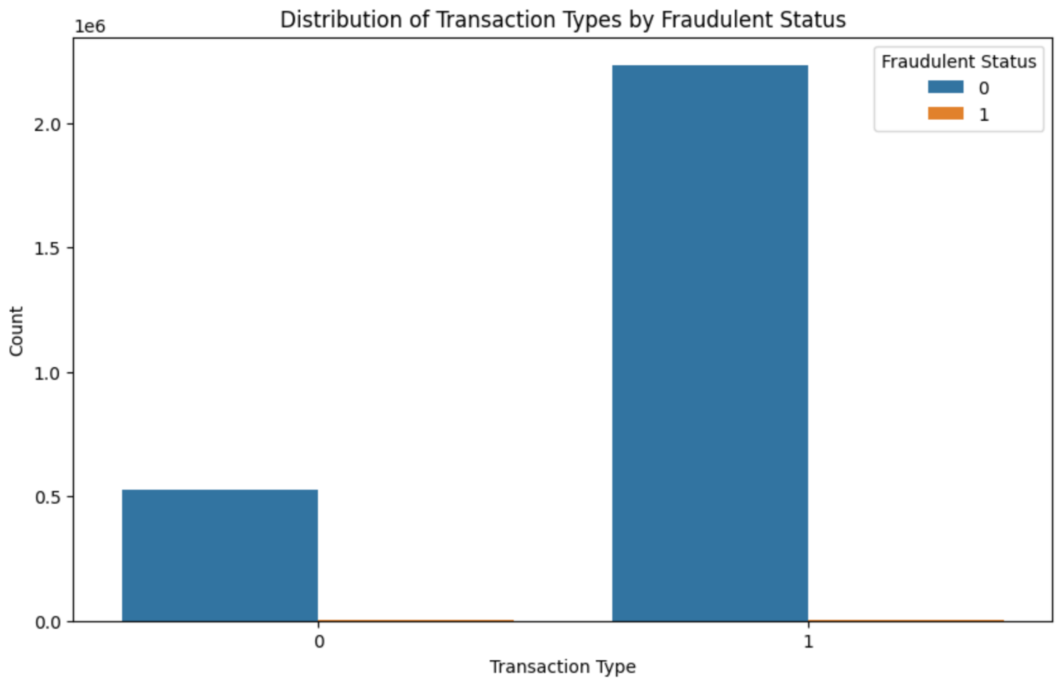

Figure 3

This are the real values of the graphic and as we can see there are many more Cash_out then Transfer, but almost the same fraud. Because a Transfer without a Cash_out isn't a completed fraud.

| type | isFraud | Count |
|---|---|---|
| TRANSFER | NOT Fraud | 528812 |
| TRANSFER | Fraud | 4097 |
| CASH_OUT | NOT Fraud | 2233384 |
| CASH_OUT | Fraud | 4116 |

Table 1: Transactions table

With this I completed the data analysis and the next step is to build a model that can say if the transaction is fraud or not. First of all I built a graph from the dataframe with node features the names of origin and dest and with edge features: steps, type of transaction and amount.

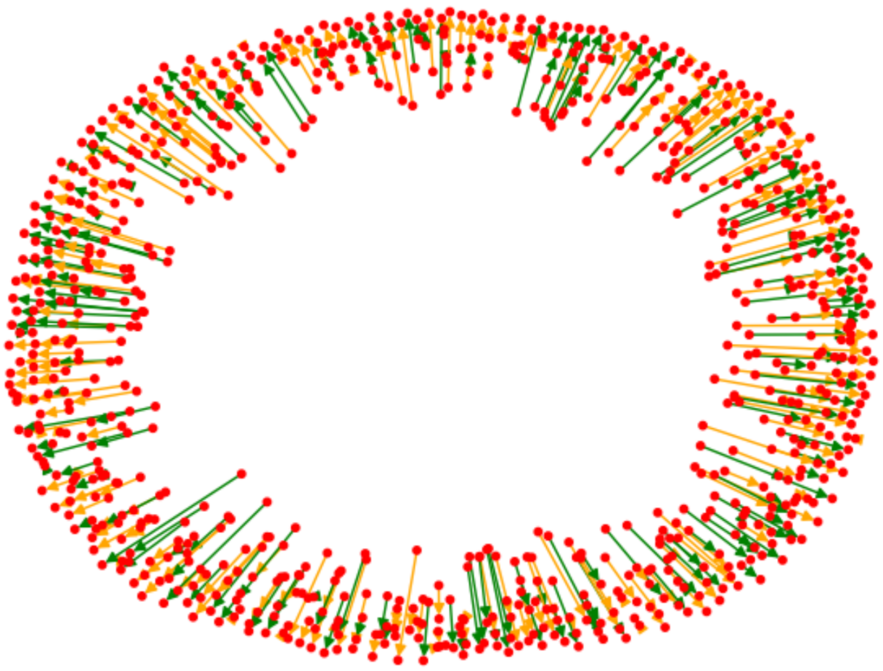

Figure 4

This is a simplified version of the graph composed by 400 nodes otherwise I couldn't show an image with some sense. All the nodes are in red and if the transaction type is Transfer the edge is green while if the transaction is cash out the edge is orange.
The next step is to create the right data structure to feed into the model. In the lesson we have seen the structure Data. So I started creating all the things that I can store in this structure. This is the shapes result Data(x=[3277509, 1], edge_index=[2, 2770409], edge_attr=[3, 2770409], y=[2770409], train_mask=[2770409], test_mask=[2770409]) where x contains the nodes with their features (name), edge_index contains the index of each edge and edge_attr contains the features (type,step and amount) for each edge, y contains the labels of fraud (0's and 1's of the dataframe column isFraud) and for the masks I divided the dataframe in 2 continuous parts with 0.8% train mask and 0.2% test mask. I made continuous parts and not random in order to have an high probability that the correlated fraud transactions are in the same set. The first 4 parameters are all tensors.
Now that I have the right structure to put into the model I developed 3 models to compare the results/difficulties. First of all I developed an MLP with 3 linear layers and 2 Relu functions

and trained it with Adam optimizer and CrossEntropyLoss.

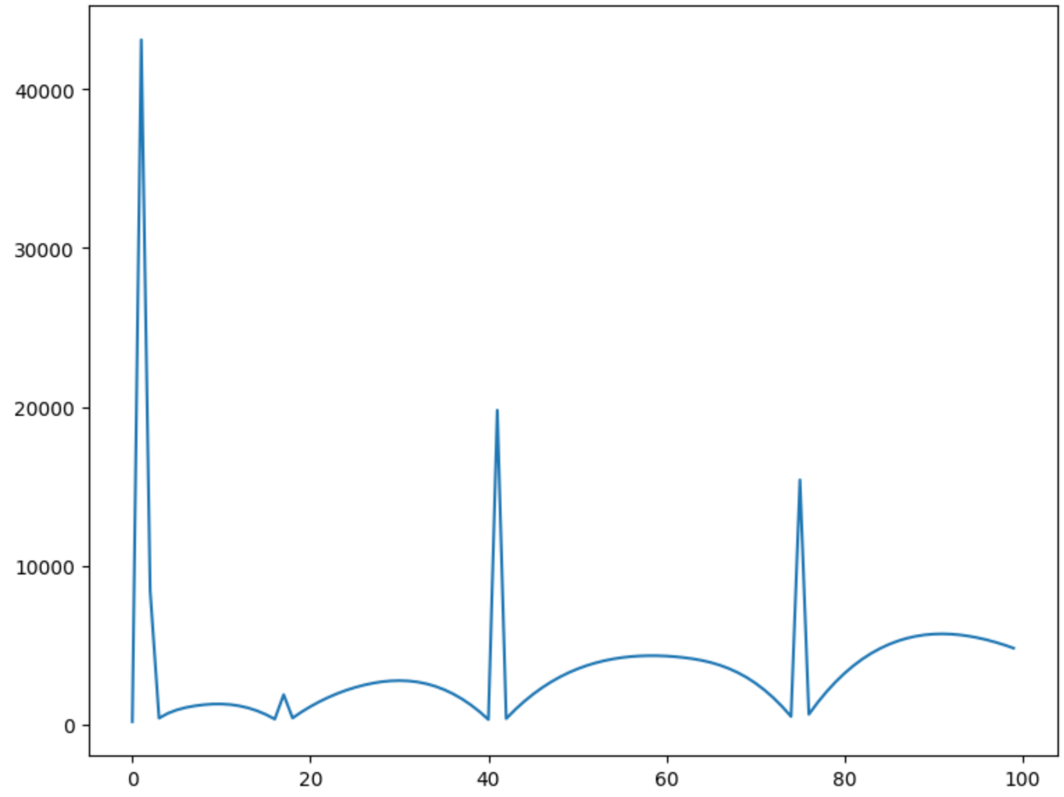

Figure 5

As we can see from this graphic I trained the model for 100 epochs and have more or less the same behaviour with same peaks of loss but with a descending trend. A good model shouldn't behave like that, but should be a parabolic curve with always a lower loss respect to the previous epoch. In this model I didn't take advantage of the graph structure but I simply feed the model with the data that I have and this is the loss results and the accuracy is 0.997 with f1 score always 0.

In the second model I used 2 layers of GCNConv and 2 linear layers, then I added encoder decoder because otherwise the model didn't run. In particular in decode I concatenated the edge_index and edge_attributes followed by a relu function this is done to merge the 2 properties and give them to the model. I made same parameter tuning with the hidden_dimension and I found that with 128 I have a quite good loss decrease as we can see in the figure below, where I reach a loss of 75 and an accuracy, as the model above, of 0.997 but in some cases the model reached an F1 score of 0.035/0.040, so the recall was very low but greater then the previous model. I used a weight_decay for decreasing the learning rate as the epochs increase. The shape of this loss is quite good compared to the other model but in the end the results on the accuracy are more or less the same.

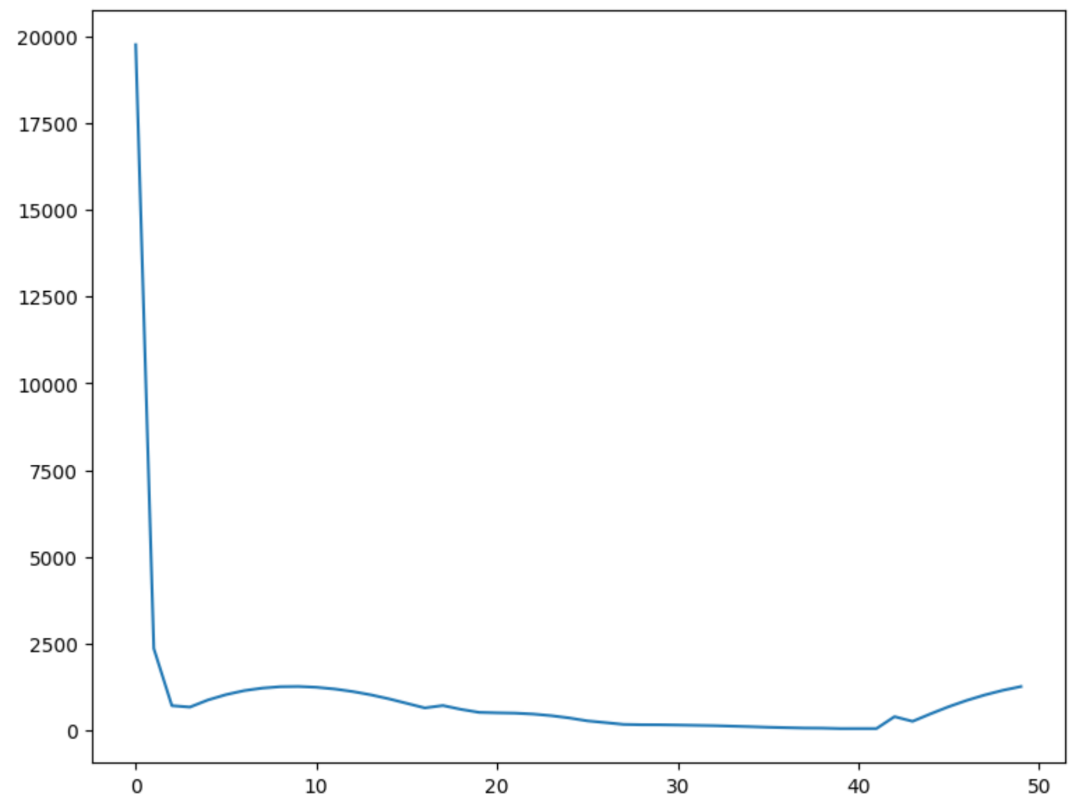

Figure 6

I tried also to build a model with EdgeModel but in this case the NN can't manage the edge attributes and so I couldn't be able to run this model.

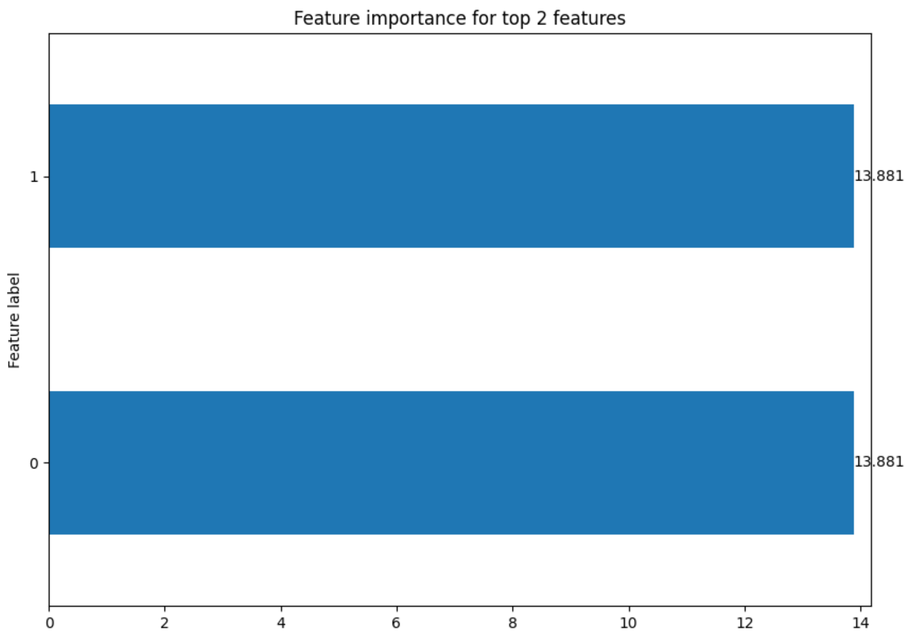

Figure 7

In the end I tried to show the explainability of this model by doing the GNNExplainer and I built the explainer but the image to show with visualize_feature_importance was so big to compute that my computer can't perform this task. Anyway I tried to modify the functions visualize_feature_importance and _visualize_score that are in the explainer to reduce

| **step** | **type** | **amount** | **nameOrig** | **nameDest** | **isFraud** |
|---|---|---|---|---|---|
| 1 | 1 | 181.00 | C1305486145 | C553264065 | 1 |
| 1 | 0 | 181.00 | C840083671 | C38997010 | 1 |

Table 2: Transactions table

the computational power needed and I tryed to show only the first 2 labels. I tried also with 50 labels but the results are meaningless because all the labels have the same importance as the image above, so I simply put the one with less features to analyze it better. As we can see in the image above, the model gives the same importance to the 2 different labels so it doesn't give the right importance to the labels.
In the end train a system like this isn't easy because it's difficult to detect a fraud given different origins and destinations, because it seems to be no meaning and no correlations also in the graph. For example in the 2 rows of the dataset represented in Table 2 we can see that name origin and name destination aren't correlated.

I showed that with GNN we can have a better loss respect to MLP, with the same accuracy. To appreciate the real importance of GNN maybe we should use another dataset.