Brief Analysis

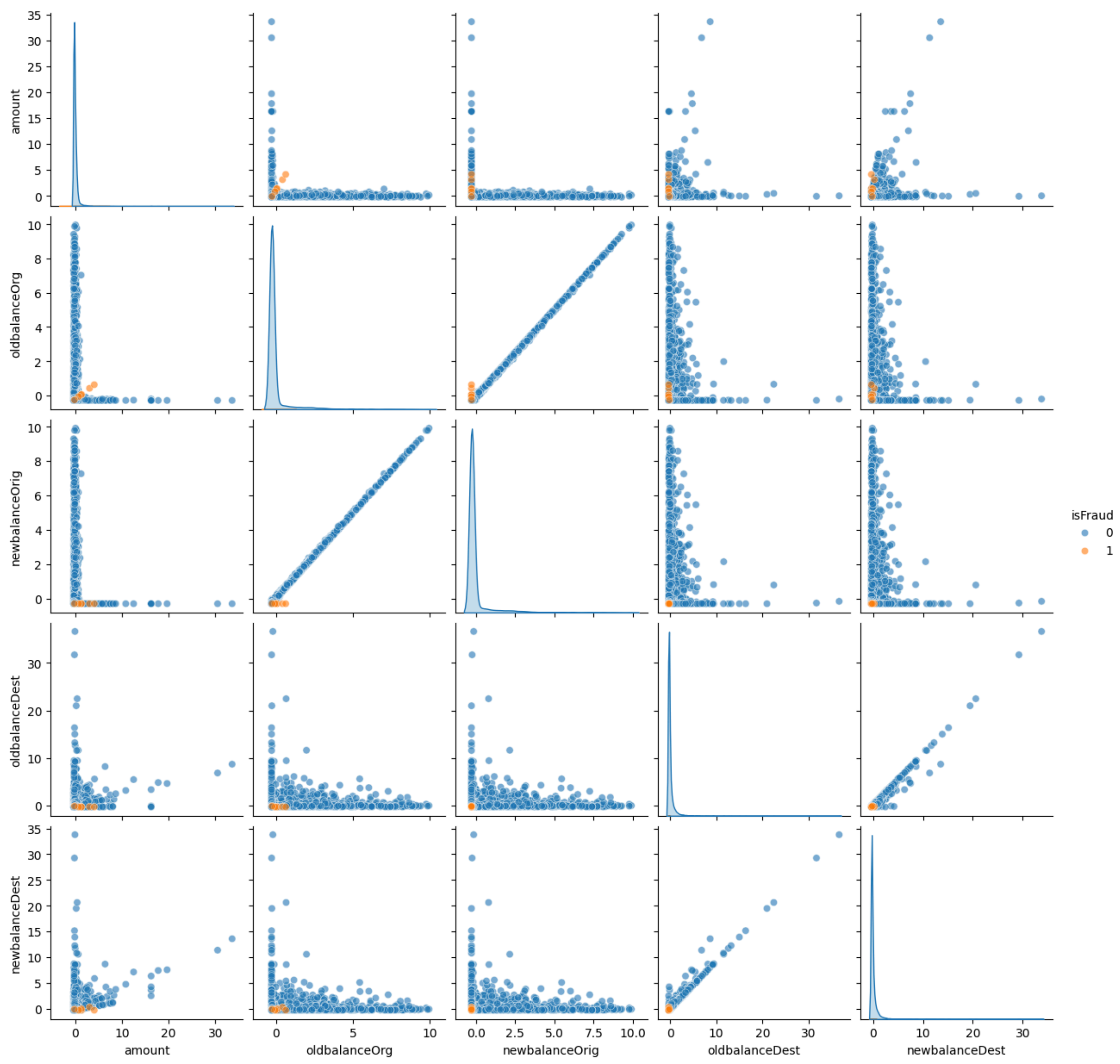
The dataset provided contains records of financial transactions for fraud detection. Included in the dataset were several features, namely:

- Transaction type
- Transaction amount
- Names (source and destination accounts)
- Old and new balances of the source and destination accounts
- Flagged as fraud
- Flagged as fraud by other models
- Transaction time

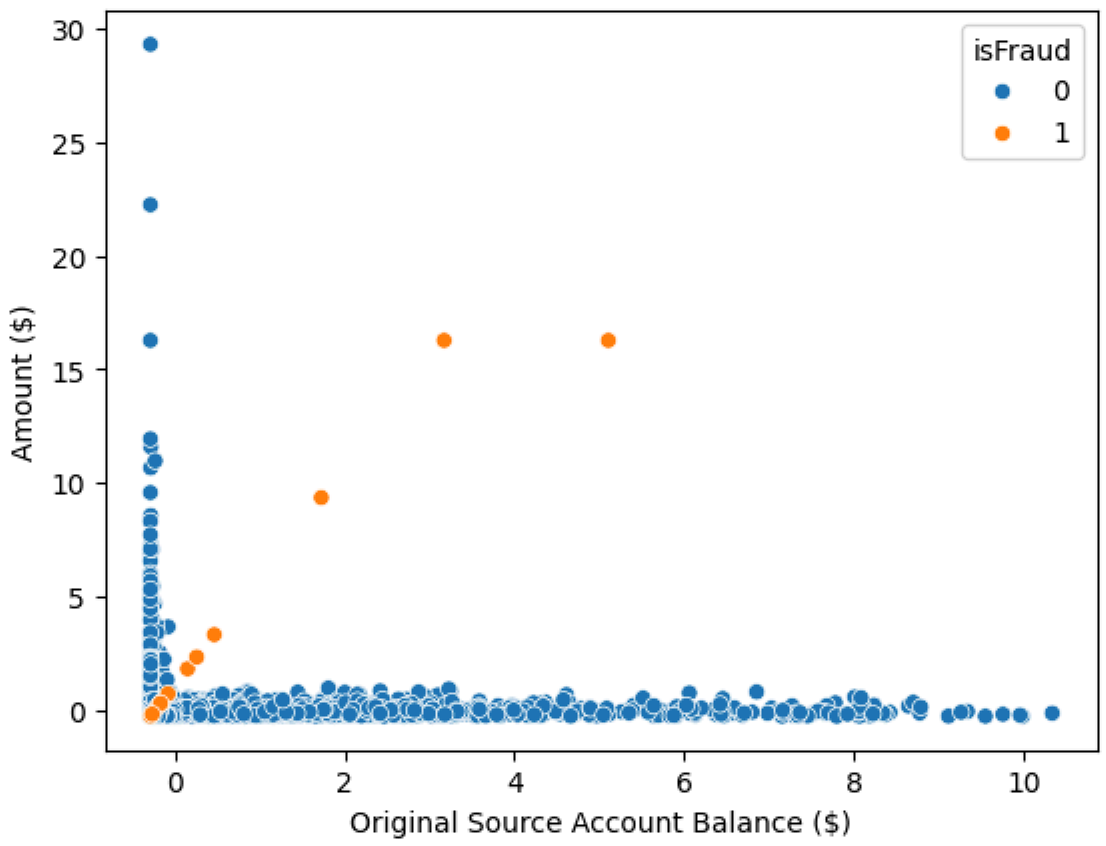
The dataset contained no missing values. Because the account balances were not along the same scale, the data had to be standardized before any meaningful modeling could occur. I also one-hot encoded the transaction type for simpler model training. Additionally, I chose to drop the following columns: step, source and destination names, and whether other models have flagged the transaction as fraud. Detecting fraud in this preliminary model should not need to take into account names or transaction times, but a more sophisticated model may want to take them into account.

EDA

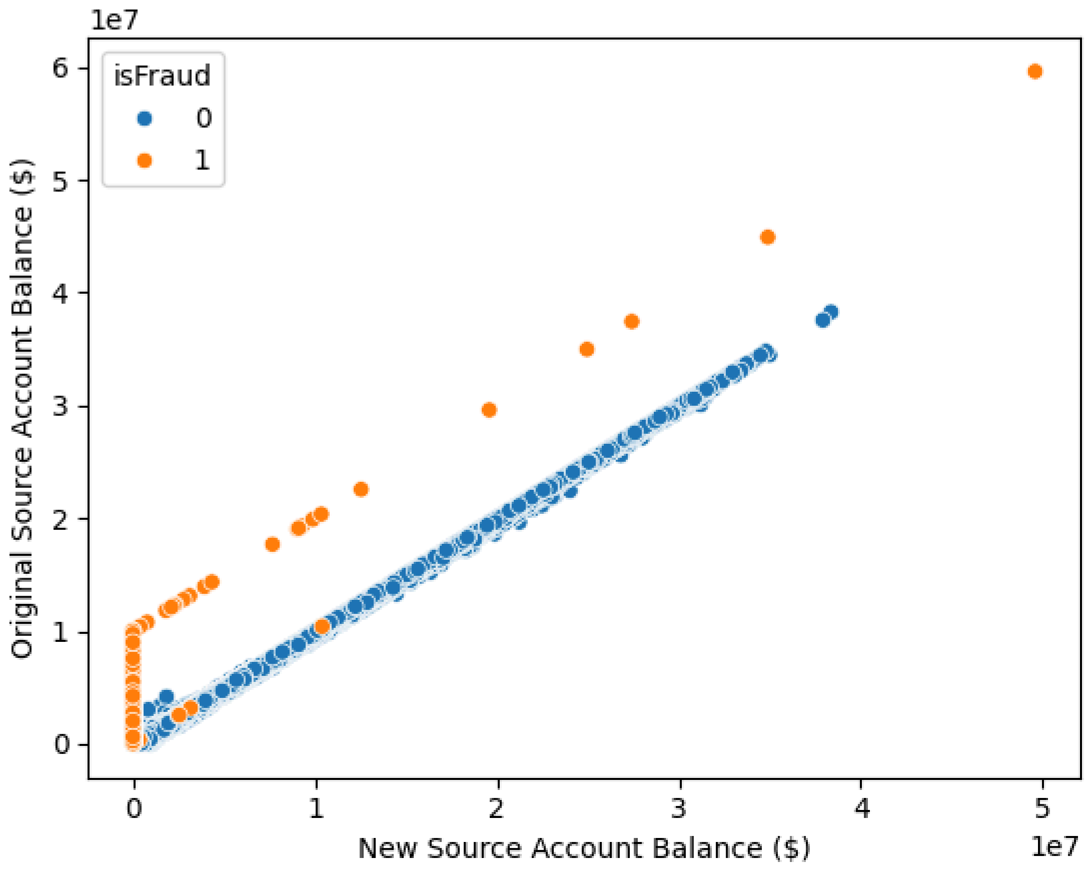
To explore trends in the data, I made a pairplot of the numeric variables.



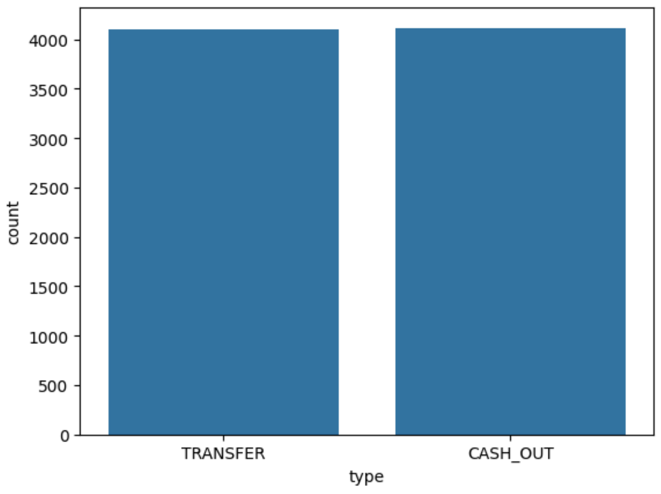
This showed an interesting correlation between the old account balance and the transaction amount. Additionally, there was a trend between the old and new source account balances.



isFraud
0
1
Amount ($)
Original Source Account Balance ($)



1e7
isFraud
0
1
Original Source Account Balance ($)
New Source Account Balance ($)
1e7



When exploring the most common type of transaction flagged as fraud, the dataset shows that there are only ‘transfer’ and ‘cash out’ transactions flagged as fraud.

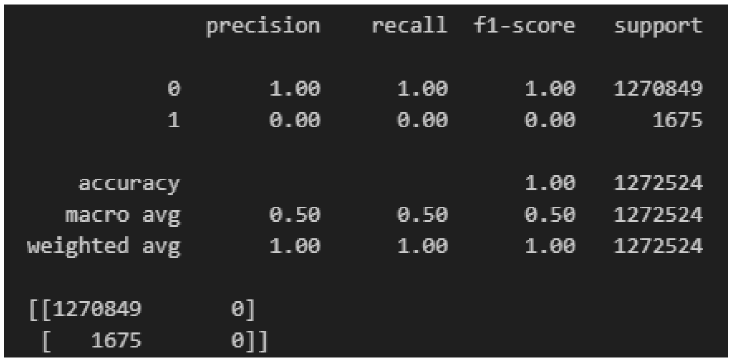
Model Selection

This is a regression problem. As such, I wanted to try a binary K Neighbors Classifier to see how it performed. I believed that fraudulent transactions might be in their own bubble in 5-dimensional space.

As a followup model, because there was such a large amount of data with so few features, a random forest classifier model would be appropriate for this problem.

Evaluation

Unexpectedly, the kNN classifier did horribly. It predicted non-fraudulent for every transaction, which is due to there being so few fraudulent transactions in the dataset. Additionally, the kNN classifier took 40 minutes to train without parallelization, and 10 minutes with parallelization for what was essentially worse than a coin flip.



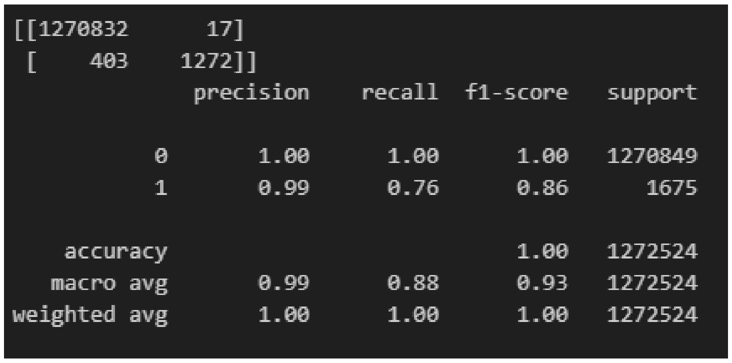
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270849
           1       0.00      0.00      0.00      1675

    accuracy                           1.00   1272524
   macro avg       0.50      0.50      0.50   1272524
weighted avg       1.00      1.00      1.00   1272524

[[1270849       0]
 [   1675       0]]
```

I tested three different Random Forest Classifiers for the second model. One with 200 trees and a max depth of 20, one with 50 trees and a max depth of 20, and one with 50 trees and a max depth of 10. The forests with a max depth of 20 performed the same, although the one with 200 trees took over 10 minutes to train while the one with 50 trees took 3 minutes to train. The one with the max depth of 10 performed worse than the other two, so I stuck with the classifier with 50 trees and a max depth of 20.



```
[[1270832      17]
 [    403    1272]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1270849
           1       0.99      0.76      0.86      1675

    accuracy                           1.00   1272524
   macro avg       0.99      0.88      0.93   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Future Considerations

This dataset had such few fraudulent anomalies that the models trained on the data tend to favor classifying transactions as non-fraudulent over fraudulent. Because of this, we may need to weigh positive classifications higher, include more fraudulent data in the dataset, or tune the models more.