Fraud Detection in Banking

PROJECT

TABLE OF CONTENT

- DATASET & SOFTWARE Page 3
- BUSINESS REASON Page 4
- DM PROCESS Page 5
- MODELING & RESULTS Page 7
- 3 W's Page 9

DATASET & SOFTWARE

The dataset represents a list of transactions from various customers made at different time and amount at BankSim (an agent-based simulator of payments based on a sample of aggregate transactional data provided by a bank in Europe). Fundamentally, its main purpose is to create synthetic data that may be used for fraud detection research. The dataset contains:

- 4412 Customers. (Male & Female)
- 600000 Transactions made in the past 6 months.
- 50 Merchants.
- 15 Product categories.

For more information on the set, the reader can check this page Synthetic data from a financial payment system | Kaggle.

As far as the software application is concerned, I chose KNIME analytics to lead this project. For novices, Knime Analytics Platform is open-source software that allows users to access, blend, analyze, and visualize data, without any coding. Furthermore, the website explains "its low-code, the no-code interface offers an easy introduction for beginners, and an advanced data science set of tools for experienced users.

BUSINESS REASON

Fraudulent behavior may be seen across many different fields such as e-commerce, healthcare, and banking systems. The PwC global economic crime explained Fraud is a billion-dollar business that is increasing every day and surveyed 7000 companies which found that 50% of them experienced fraudulent activity of some kind.

Although fraud seems to be daunting for businesses and people, I think it may be possible to detect it using intelligent systems such as artificial intelligence. For example, AI in fraud detection would help limit the number of transactions promptly (velocity rule) or deny the transactions which come from previously known fraudulent IPs (internet protocol) or domains.

However, even though the intuition looks appealing somehow, the trained machine learning model might fire a lot of false positives or false negatives because of predefined threshold values. For instance, let's think about a simple rule that denies a transaction with an amount bigger than $10000 for a user. If that user is an experienced fraudster, he/she might know the system would have a threshold, and so he/she might make a transaction just below the threshold ($10000 – x) to bypass it.

Despite all that, machine learning could help reduce the risk of fraud which consequently will save money for businesses and people.

From a personal perspective, I chose this case because fraud affected my life several years ago. Back in the summer of 2020, I experience a sim swap (a type of identity theft). It is an account takeover fraud that generally targets a weakness in two-factor authentication in which the second step is a text or call placed to a mobile phone. The fraud exploits a mobile phone service provider's ability to seamlessly port a phone number to a device containing a different SIM. It all starts with a fraudster gathering personal details about the victim by buying them from criminals, or by socially engineering the victim. Then, the fraudster contacts the mobile telephone provider and with sophisticated techniques convinces the company to port the victim's phone number to the fraudster's SIM. Once this happens, the victim's phone will lose connection to the network, giving the ability to the fraudsters to intercept any one-time passwords sent via text or phone to the victim. Since so many services still permit password resets with only access to a recovery phone number, the fraud let criminals gain access to almost any account tied to the hijacked number.

Such an experience cost me $40000 and is still pending litigation (3 years now!) against the bank for letting third parties hijack my financial account. In 2021 I won the first trial, but the bank refused to pay and take responsibility by blaming my negligence.

DATA MINING PROCESS

Knime analytics has a RapidMiner-Like user interface. So, the first step I've taken is to read the dataset in the console. In the node repository section, there's an operator called CSV reader. I dragged and dropped into the workflow window. From the looks of it, the dataset contains ten attributes.

- STEP: It contains 180 days from when the simulation began. Virtually 6 months.
- CUSTOMER
- ZIPCODEORIGIN
- MERCHANT
- ZIPMERCHANT
- AGE:
  0: <= 18,
  1: 19-25,
  2: 26-35,
  3: 36-45,
  4: 46-55,
  5: 56-65,
  6: > 65
  U: Unknown
- GENDER:
  E: Enterprise,
  F: Female,
  M: Male,
  U: Unknown
- CATEGORY: purchase's category.
- AMOUNT
- FRAUD: Target variable which shows if the transaction is fraudulent (1) or not (0).

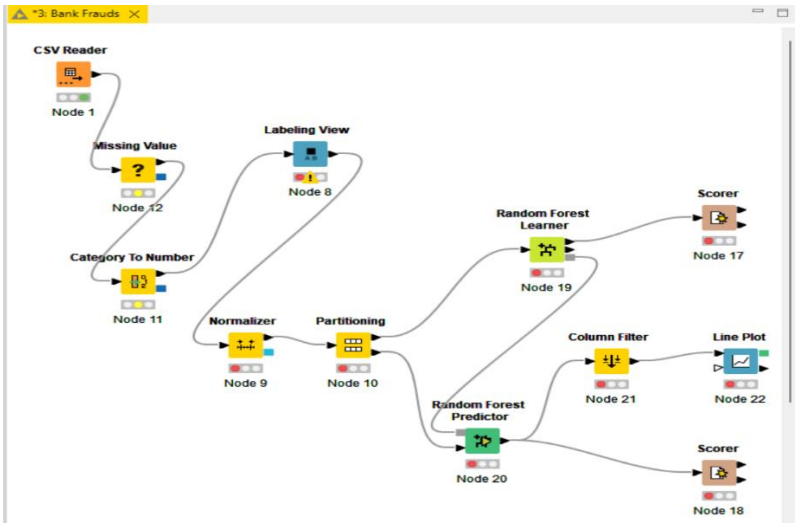
With the help of the GROUP BY operation, I had the chance to look at some stats. I got lucky with this dataset because no missing values are in it, except in the attribute AGE. However, it can be easily fixed with a MISSING VALUE operator. Once dragged and linked to the CSV READ icon, using the right button of the mouse, the user can easily configure the attribute in the column configure window. I set it as a fixed value of 0. Now, as regards the fraud column, on 600000 transactions only 7200 are fraudulent, which makes it highly unbalanced. To avoid it, the user might perform over or under-sampling. However, both operations have risks: oversampling will create copies or similar data points which sometimes would not be helpful in the case of fraud detection because fraudulent transactions may vary. Whereas under-sampling means losing data points, thus precious pieces of information. I've decided to keep the dataset as is.

The next step I've taken is to transform categorical data into numerical one. In RapidMiner, the user can transform different categories such as nominal to binomial, or numeric to polynomial. Whereas in KNIME, there is only one transform operator, categorical into numeric – the process is automatic, and there's no need to configure it. Then I terminated the Data prep by normalizing the dataset (NORMALIZING), selecting a label for my prediction (LABELING VIEW), and splitting my data for training and testing the model (PARTITIONING).

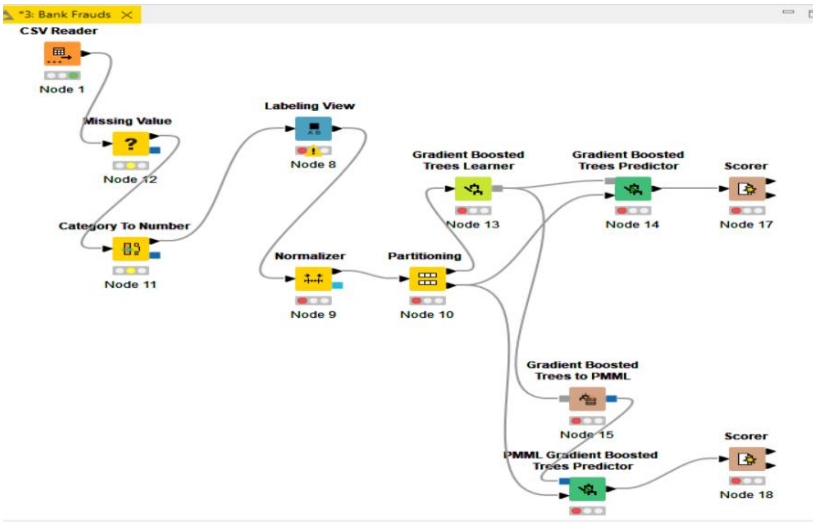
MODELING & RESULTS

For this section, I've decided to post some screenshots because each model requires a different setup for each model.

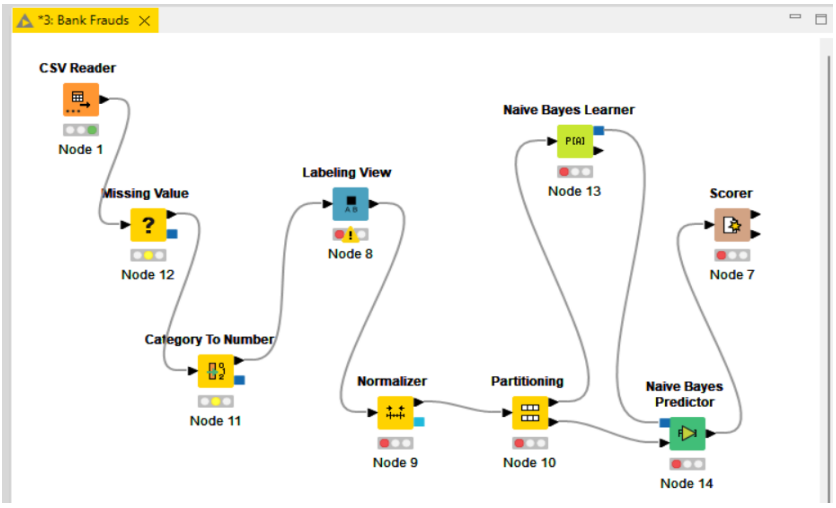
- Random Forest



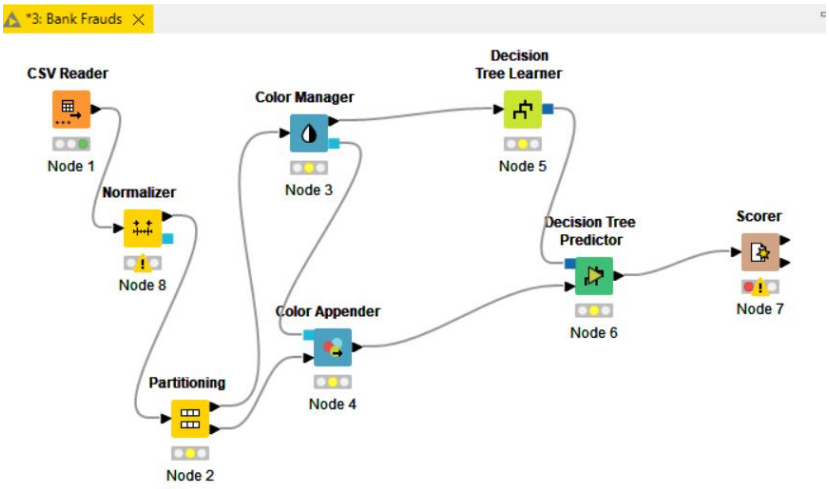
- Gradient Boosted Tree



- Naïve Bayes



- Decision Tree



Here's the toughest part. After spending a lot of time understanding the data and figuring out to get the software working, I had to turn down all the models except the naïve bayes one – more in the next paragraph. The model returned an accuracy rate of 99.5%. Even though at first impact everything looks great, it shows a precision of detecting fraud of just 65%. Such an experiment can be a good starting point in creating more powerful algorithms that could help save money for both enterprises and consumers.

3 W'S

- WHAT WENT WELL

  The dataset itself did not come with much contamination. Generally, a scientist spends most of his time cleaning, transforming, replacing, or slicing the data to reduce model confusion. Here, except for having some missing records in the attribute AGE, the data came well collected and categorized. It helped me cut time in Data preparation.

- WHAT DID NOT GO WELL

  First, my computer is not powerful enough to run machine learning algorithms when it comes to applying them to large datasets, despite being two months old with 16 GB RAM, an I7 intel processor, and a 1 TB hard disk of space. Most of the time, the computer would choke or take hours before running a result – Once I left it run for 6 hours without receiving concrete results. This is the reason why I ended up with the simplest model "naïve Bayes".

  Second, detecting fraud is challenging. Since 1% is considered a fraudulent transaction, accuracy itself is not an efficient way to evaluate a model. Besides, the model would need to be fed with a larger dataset to reduce the bias, which would require extremely powerful computers perhaps.

- WHAT WOULD YOU DO DIFFERENTLY NEXT TIME

  Buying a better-performing computer.

  Invest additional resources (time, money, effort) to master a software application.