# Exploratory Analysis

Since the documentation shared on Kaggle about the dataset does not provide specific details about the overall dataset structure, my first goal was learning more about its content and characteristics. I used my first project Jupyter Notebook to tackle the following tasks:

1. Understand what each column in the dataset represents:

Graphical user interface, text, application

Description automatically generated

Engaging with the column names and content allows me to identify a number of columns that will not be relevant in future analysis related to fraud prediction:

* 'cc\_num' (unique credit card number),
* ‘merchant’ (unique merchant first and last name),
* 'first' (credit card owner first name),
* 'last' (credit card owner last name),
* 'trans\_num' (unique transaction number).

We do not want the model to develop bias related to specific names or unique card/transaction numbers so it would be actively harmful to take forward this data. In addition, the 'trans\_date\_trans\_time' column might not be necessary since the same data is also recorded in the 'unix\_time' column. For the time being, I will treat 'trans\_date\_trans\_time' as duplicate data.

1. Graphical user interface, application, Word

   Description automatically generatedUnderstand the datatypes comprising the dataset:

The data comprising the following columns is already stored as numeric values and therefore does not require further manipulation to be used:

* ‘amt’,
* ‘zip’,
* ‘city\_pop’,
* ‘unix\_time’
* ‘is\_fraud’

However, all other columns are storing object data, meaning they will require pre-processing before they can be fed into a model.

1. Checking whether there are missing values to handle:

I carried out checks to ensure there were no invalid or missing data instances to handle, however the dataset did not present any challenges in this context. One of the advantages of using synthetic data is that the dataset does not include any anomalies that are normally generated due to human error.

1. Identify which features I want to take forward to the pre-processing phase and which challenges they present:

Graphical user interface, text, application

Description automatically generated

Analysing the distribution of the ‘is\_fraud’ variable reveals that the dataset is extremely unbalanced. While we have many instances of genuine transactions, there are very few examples of fraudulent ones. This is one of the key challenges we will need to tackle to build a solid predictive model that can identify fraudulent transactions: since only a very small percentage of all transactions taking place daily are fraudulent, we need to devise strategies to deal with the unbalanced nature of the dataset.

Graphical user interface, text, application

Description automatically generated

The ’dob’ column representing the date of birth of credit card holders needs to be converted into a numeric variable. The suggested approach for the time being is only keeping the digits expressing the year of birth and automatically converting that into age given the current year. When presenting results using ‘age’ rather than ‘year of birth’ as a variable will improve results readability.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

I have identified the columns ‘category’, ‘state’, ‘gender’ and ‘job’ as potential input values to pre-process for future use. They all store categorical data that is limited to a relatively small pool of unique values, however since it is not ordinal it raises the challenge of encoding it without introducing an irrelevant order that might bias the machine learning model.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

The columns ‘street’ and ‘city’ represent interesting geographical data, however the pool of unique values present in the dataset is very large. For the time being, I am excluding these potential input variables as they are more complex to process and with the limited resources available I prefer to focus on more manageable relevant features.

Graphical user interface, text, application

Description automatically generatedWhile the column ‘zip’ is one of the few input variables available that is already stored as a numeric value, I have decided to exclude it as irrelevant. Since it is not an ordinal variable I believe that introducing it might bias the model.

Graphical user interface, text, application

Description automatically generatedFinally, the columns ‘long’, ‘lat’, ‘merch\_long’, ‘merch\_lat’ store information about the specific coordinates of the credit card owner and merchant involved in a transaction. I am not considering these potential input variables for the time being, as being data about geographical locations they require a different pre-processing protocol and given the resources available it is not feasible at this time to harvest them .

In conclusion, there are two main challenges that emerged from initial analysis of the dataset:

1. How to convert categorical data into numeric variables considering that they are not ordinal.
2. How to deal with the unbalanced nature of the dataset.

# Data Pre-Processing

After researching available solutions to transform categorical data into numeric variables, I decided to employ OneHotEncoder to encode data stored in the columns ‘category’, ‘state’, ‘gender’ and ‘job’. I have chosen this encoding method since our categorical data does not have a pre-established order. For example, if we were to assign 1 to a specific individual credit card owner's job title and 2 to a different job title, this would train the model to consider these values as somehow relevant to the analysis. In other words, the fact that the latter is a larger quantity than the former would be interpreted as significative, when the numeric values assigned to each job title are in fact random. Since I am not able to convert this categorical data into numeric values that express a meaninfgul ordinal relationship, using OneHotEncoder is well suited to the task.

This is a summary of the pre-processing phase with OneHotEncoder:

1. I successfully employed OneHotEncoder to encode 'gender', 'state', 'category', 'job'. However, I have executed this process manually. One of my next tasks will be experimenting with creating a pipeline that is automatically applied to the full dataset.
2. I have encountered some issues while merging the encoded dataframes. While I can join the original dataset on the encoded dataframes of ‘gender’, ‘state’ and ‘category’, I am not managing to also add the encoded ‘job’ dataframe. My hypothesis is that the number of columns is causing this (encoded ‘job’ dataframe has over 400 columns, the others have less than 100 between all of them). At the moment, I am considering removing ‘job’ from the potential input variables.
3. My next challenge will be creating a pre-processing pipeline that is automatically applied to the full dataset.

## Predictive Modelling

In my second Jupyter Notebook, I successfully created a pre-processing pipeline that employs OneHotEncoder to convert categorical data into numerical data. I have additionally applied the code developed by Juliana to convert the date of birth feature into a numerical value showing the age of the credit card holders. This allowed me to consider as input variables or features the following columns:

* 'dob',
* 'category',
* 'amt',
* 'gender',
* 'state',
* 'city\_pop',
* 'unix\_time'

After performing a literature review, I identified classification trees and logistic regression as two potential machine learning algorithms to train to predict fraud. I selected them according taking into account their wide popularity in the literature for fraud prediction, consistency in good performance metrics in relation to fraud prediction and explainability.

First, I used the dataset pre-processed through the OneHotEncoder pipeline to train a logistic regression model. As demonstrated in the graphs below, the results very clearly show that the model is currently prone to overfitting. Although its accuracy is extremely high, it can only correctly label genuine transactions, while it has labelled all fraudulent transactions incorrectly. Our aim is producing a model that is perhaps less accurate overall, however more often correctly identifies fraudulent transactions. The preferred outcome is sometimes flagging genuine transactions as suspicious, rather than labelling fraudulent transactions as legitimate. The overfitting issue is very likely due to the unbalanced nature of the dataset. Currently, the model has an abundance of examples for genuine transactions, however it has very few opportunities to learn about the characteristics of fraudulent transactions. I need to test several techniques to see if there is a way to improve the performance of this model despite the unbalanced nature of the dataset.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

I then tested a decision tree model with the same type of encoding. The performance is significantly better than the logistic regression model, with good metrics in accuracy, precision and recall scores. Specific accuracy in correctly labelling fraudulent transactions is 70%. While these results are very encouraging, decision tree models’ performance is often negatively impacted by OneHotEncoding, due to the added complications in dealing with sparse matrices. Juliana has been testing this model with a different type of encoding to see if it produces better results. Here I am including the confusion matrix, metrics and decision tree graph.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Diagram

Description automatically generated

While Juliana has proceeded with testing the decision tree model, I have kept focussing on logistic regression. I tried experimenting with upsampling and downsampling to deal with the unbalanced nature of the dataset and overcome the overfitting issue.

These are the observations I made after experimenting with different parameters:

1. I attempted to use upsampling to artificially inflate the instances of fraud to match those of genuine transactions. This however creates duplicate data of fraudulent transactions, which results in a model prone to overfitting that can still only recognise genuine transactions and never correctly labels instances of fraud.

Graphical user interface, text, application

Description automatically generated

1. I applied downsampling to only select a smaller pool of genuine transactions to match the number of fraud instances. This initially showed very encouraging metrics, with a decrease in performance for the accuracy score but a significant increase in recall and precision scores. The confusion matrix showed that the logistic regression model was correctly labelling fraudulent transactions in about 70% of cases. However, after restarting the runtime multiple times to test whether different cuts of the downsampled majority class would produce consistent results, it became clear that performance metrics are very inconsistent when using downsampling techniques on this dataset. This method produced models that could only correctly label fraudulent instances, as well as models that could only correctly label genuine transactions and varying combinations of accuracy, recall and precision metrics across the spectrum. It appears that logistic regression cannot successfully overcome overfitting issues through downsampling only in this instance. Here follow two confusion matrices exemplifying the logistic regression models that can be obtained through downsampling, with their associated metrics

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

1. I experimented with a combination of upsampling and downsampling, however these all produced either overfitted models that could only predict genuine transactions results or underfitted models with accuracy, recall and precision scores of about 50%.

In conclusion, I could not find a way forward to harvest resampling techniques to overcome overfitting for this logistic regression model.

I then proceeded to test the logistic regression model performance when pre-processed through the encoding protocol developed by Juliana. Here follow the resulting confusion matrix and performance metrics.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

After applying Juliana's pre-processing protocol I determined that the performance metrics of the logistic regression models continue to show overfitting. As observed in tests carried out so far, the model only correctly recognises genuine transactions since these comprise almost the entirety of the dataset. I will now proceed to test cross-validation as a method to reduce overfitting.

I first experimented with KFold, but obtained poor results. The logistic regression model continued to show issues with overfitting despite multiple experiments with the n\_splits parameter. I kept increasing the number of splits up to 200, but the performance metrics were not affected by these changes and the processing time kept getting longer.

After consulting the documentation further, I realised that StratifiedKFold is more appropriate for our fraud dataset. This is because it creates subsets of the original dataset that preserve the same minority class:majority class ratio. This is extremely important in our case, since we are hoping to use cross-validation to handle overfitting issues caused by the unbalanced nature of the dataset. After experimenting with StratifiedKFold I continued to obtain consistently poor performance metrics, with the logistic regression model kept showing signs of overfitting. Finally, I decided to attempt RepeatedStratifiedKFold to introduce the n\_repeats parameter. This method produced the same performance metrics showing that logistic regression continues to be prone to overfitting. I am including here the metrics for this final attempt, however they are representative of all the cross-validation tests detailed in this paragraph.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

After obtaining poor results in applying cross-validation to solve the overfitting issue, I decided to test it again within a cost-sensitive learning framework. This technique attempts to prevent overfitting by associating different costs to incorrectly predicting the majority class and the minority class. More specifically, it establishes mislabelling the minority class as a more costly action compared to mislabelling the majority class. This is exactly what we want, since an effective anti-fraud predictive model needs to be extremely accurate in identifying potentially fraudulent transactions, or the minority class, over the genuine ones.

I defined manually how the model should weight the different classes by calculating the minority minority:majority class ratio:

Graphical user interface, text, application, email

Description automatically generated

The resulting performance metrics do not show any signs of the logistic regression model overcoming its overfitting issues.

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

In conclusion, employing cross validation techniques has not revealed any promising path forward to successfully prevent the logistic regression model from overfitting. It appears that proceeding with decision trees is the best course of action.

## What would I do better?

with more resources available I would have liked to tackle the following aspects of the project:

1. I would have harvested the geographical data present in the dataset better. I would have liked to include more detailed information about the location of credit card owners and merchants. I think perhaps calculating distance between customer and merchant and creating classes for different ranges could have revealed whether distance is a relevant variable in fraud prediction. I also noticed that while all credit card owners billing addresses were in the US, not all merchants operated within the US. Introducing a feature that coded whether the transaction was domestic or not could also have reveal something meaningful.
2. I would have harvested the categorical data related to the credit card owners’ job titles better. Perhaps to reduce the number of unique values, we could have performed a cluster analysis to group job titles by industry/management level/average income. This might have revealed interesting insights.
3. I would further experiment with cross-validation techniques and cost-effective learning. I do not think I have exhausted the potential of these techniques, however given the limited resources and time I could not perform any further tests.
4. I would have experimented with seeding for the decision tree. We have verified that restarting the runtime continues to produce consistent performance metrics, however it would have been interesting to record more accurate information about the trajectory of the performance metrics across different train/test splits.
5. I would have experimented with creating a decision tree ensemble to test how performance is affected by employing a more complex version of our most successful model.
6. I would have experimented with training a naïve Bayes model. Employing this machine learning algorithm when creating anti-fraud predictive models is a very popular approach in relevant literature. The presented performance metrics often appear consistent, so it would have been interesting to test whether we could have successfully employed it.