



# FRAUDOLENT TRANSACTION CLASSIFICATION

RICCARDO LA MARCA, MATR. 1795030

BIG DATA COMPUTING COURSE A.A. 2021/2022

1

## **Introduction**

A brief presentation of the addressed problem

2

## **Dataset**

A brief description of the dataset used in the project

3

## **Explore and Feature Engineering**

How the dataset was modified

4

## **Machine Learning Models**

The ML models and Pipelines applied for the task

5

## **Results**

A description of the results obtained from the previous step

# OVERVIEW



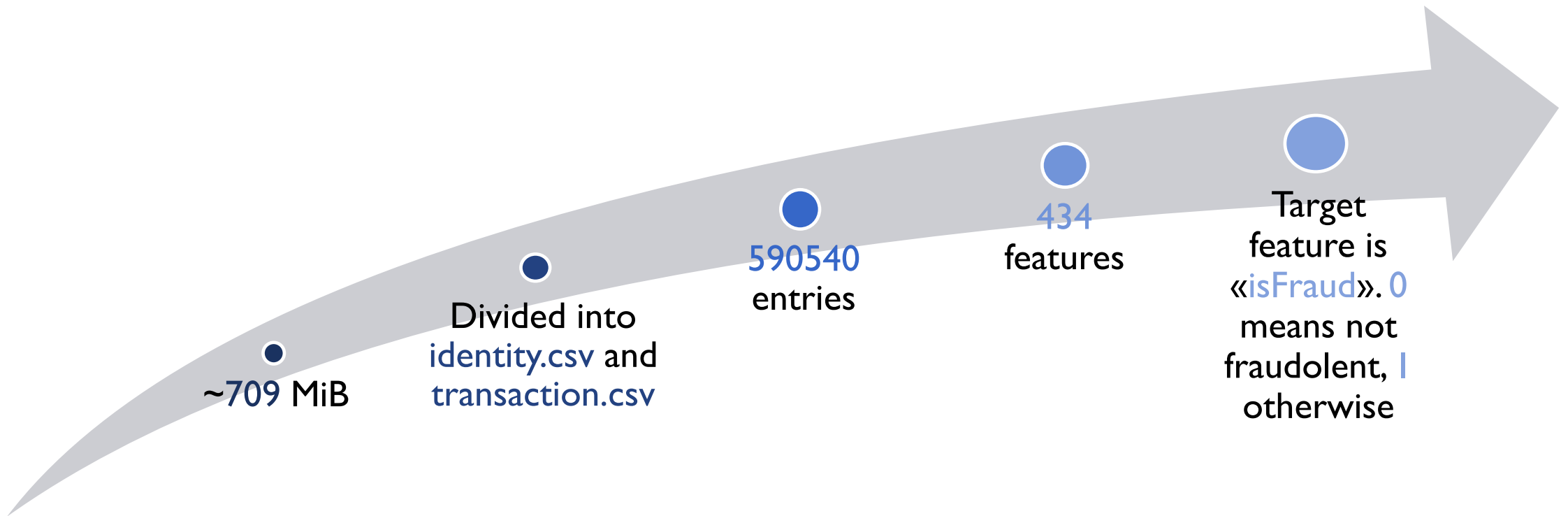
## ADDRESSED PROBLEM

Financial fraud is a problem that has a huge impact on the financial industry

Credit card fraud detection is a challenge mainly due to 2 problems that it poses

- Both profiles of fraudulent and normal behaviours change
- Usually used datasets are highly skewed

The goal of the task is to create a Machine Learning model that, given a set of samples of fraudulent and not fraudulent transactions, is capable of classifying whether a new transaction is fraudulent or not.



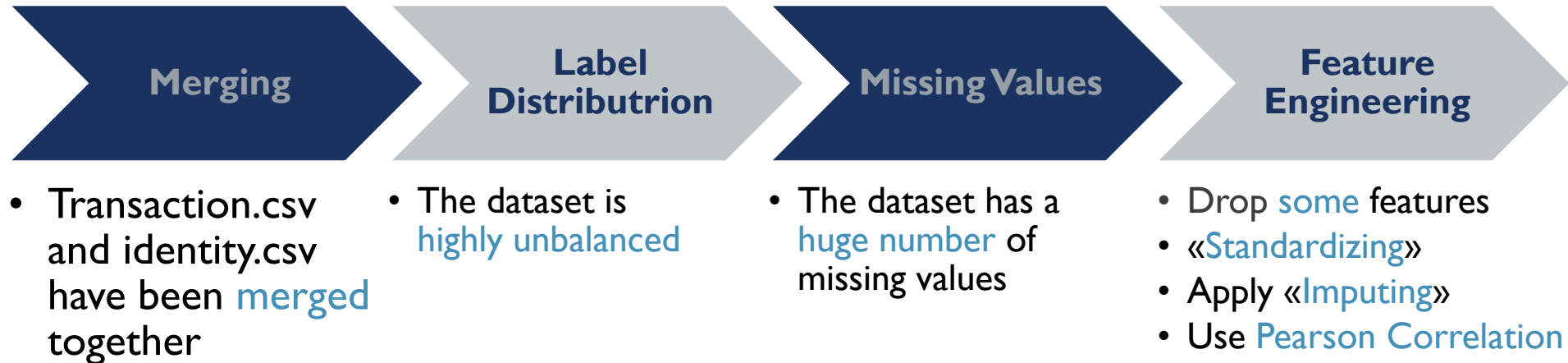
## 2

## THE DATASET

The Dataset is available on [Kaggle](#)

## 3

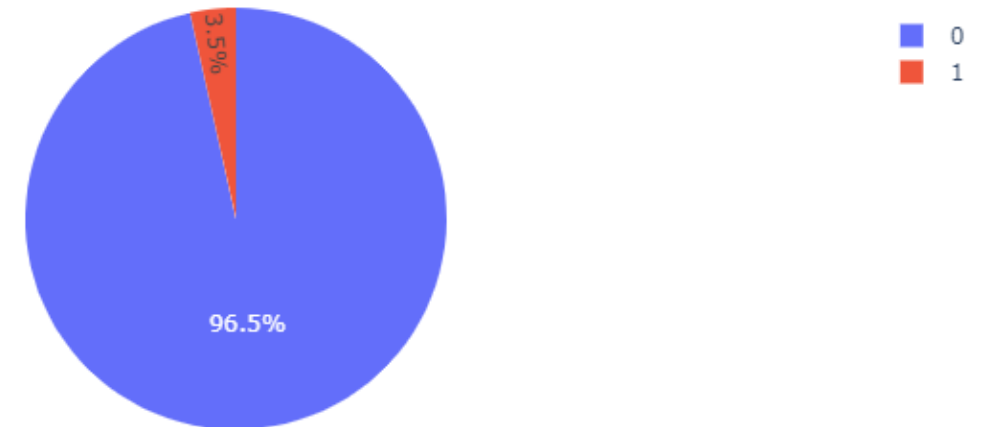
## EXPLORE AND FEATURE ENGINEERING OUTLINE



## 3

## .1 - LABEL DISTRIBUTION

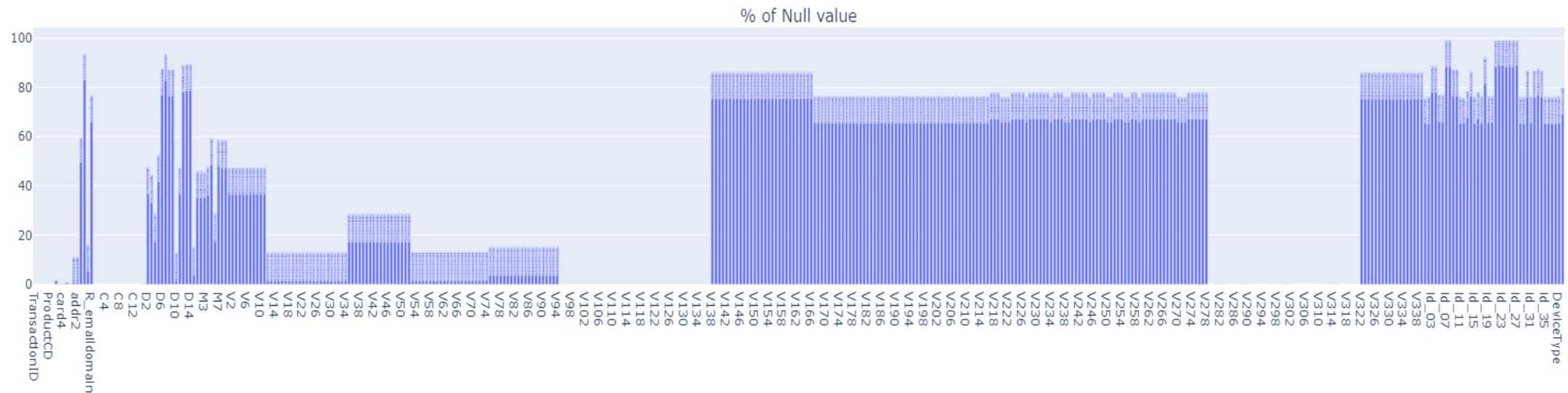
- With respect to the target label «isFraud» the dataset results **highly unbalanced**
- ~**96.5 %** are not-fraudulent transactions
- ~**3.5 %** are fraudulent transactions
- We have to handle this problem when splitting the dataset for training and testing the various ML models



## 3

## .2 – MISSING VALUES

- The dataset has a **high number** of features with a huge percentage **of missing values**
- The average range of percentages is **~70-90%**
- I handled this during the Feature Engineering step



### Features Dropping

- Drop features with percentage value of missing values greater or equal to 90%

### Standardization

- Standardize certain features
- Given different values for the same feature but with equal meaning, replace with a single more general value
- Take *yahoo.co.jp*, *yahoo.co.uk* and *yahoo.net*, I replace it with *yahoo*

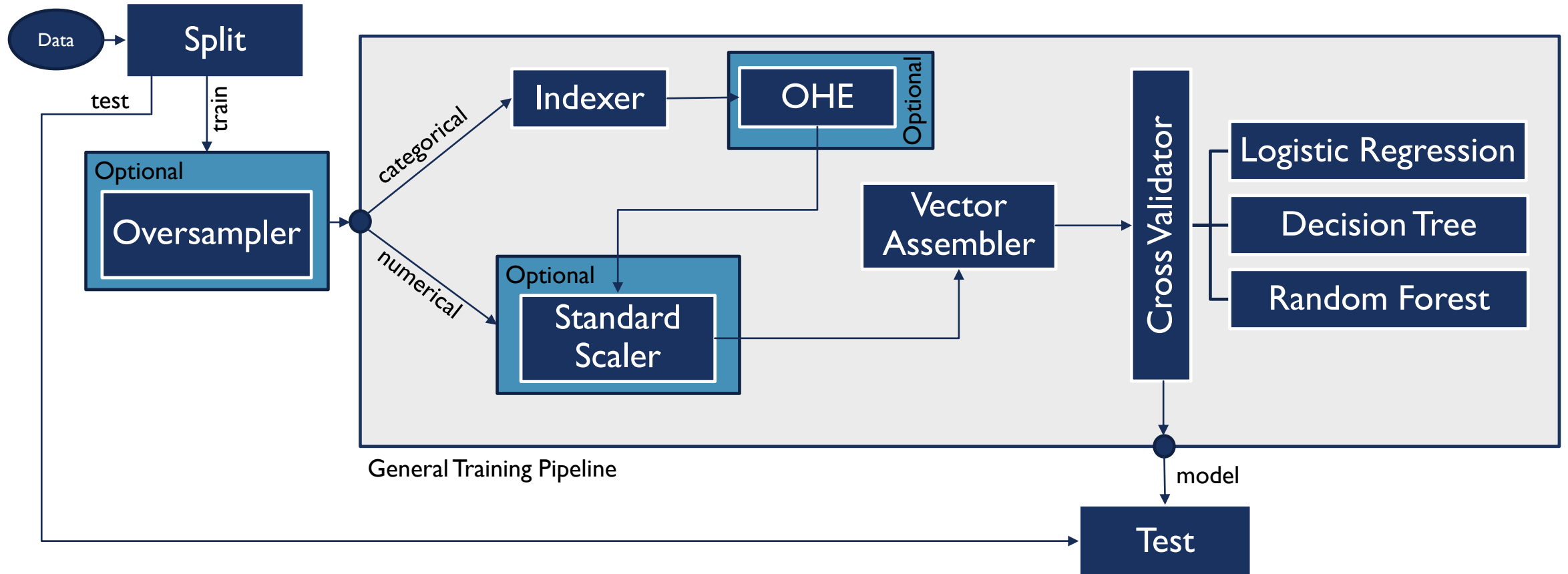
### Imputing

- Use the imputer to replace null values in the dataset according to a specific strategy
- Discrete values use strategy *mean*
- Nulls in categorical values have been replaced with «N»

### Pearson Correlation

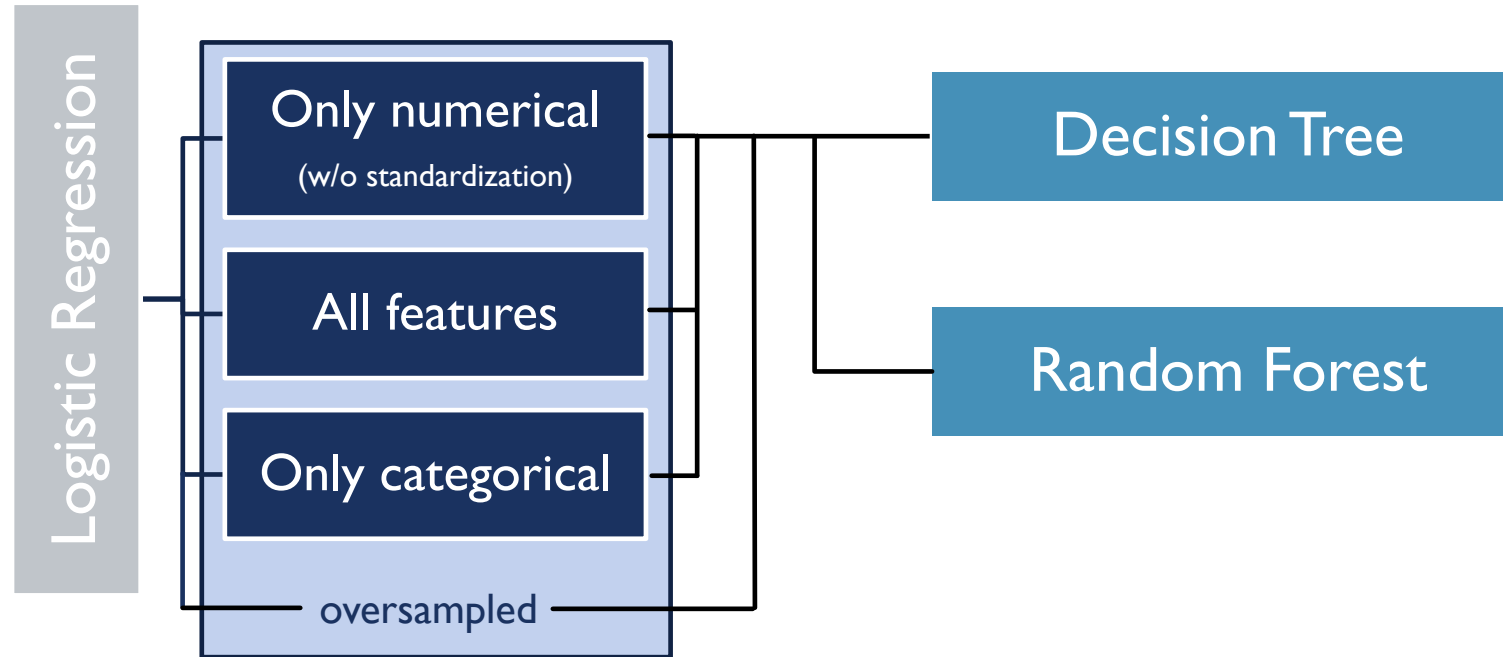
- Drop more features using the Pearson Correlation
- If the PC > .95, then drop that feature
- Avoiding duplicate features





- The dataset is highly unbalanced, thus we cannot apply a **simple random splitting**
- This might lead to a **poor splitting strategy**
  - For instance the test set ends up containing only examples that are labeled with the most representative class
  - In this case such a class is the one for *non-fraudulent transactions*
- For this reason I used the so-called **Stratified Random Sampling**
  - It guarantees that both the training and the test split follow the same class distribution of the original dataset
  - For the experiments I selected **60%** of 0's and **70%** of 1's
- After splitting we last with: **357041 × 232** (train set) and **233499 × 232** (test set)

- After the stratified sampling, the train dataset was **still highly unbalanced**
- We have **342530** of 0s and **14511** of 1s.
- I decided to apply **oversampling** on the train set
  - After this I had 342530 of equal entries for fraudulent transactions
  - I decided to keep only the **60%** of them
  - That because, keeping all of them, I obtained a high number of False Positive
  - A high number of non fraudulent have been classified as fraudulent
- Finally, the train set contains **63.3%** of non fraud and **36.7%** of fraudulent



Accuracy	Numerical (oversample)		All Features (oversample)	Categorical (oversample)
	with standardization	w/out standardize		
Logistic Regression	0.9772 (?)	0.97725 (?)	0.9777 (?)	0.9733 (?)
Decision Tree	0.9773 (?)	0.9773 (?)	(?)	0.9734 (?)
Random Forest	(?)	<b>0.9789</b> (?)	(?)	(?)

AUC ROC	Numerical (oversample)		All Features (oversample)	Categorical (oversample)
	with standardization	w/out standardize		
Logistic Regression	0.832 (?)	0.834 (?)	0.857 (?)	0.800 (?)
Decision Tree	0.428 (?)	0.428 (?)	(?)	0.7074 (?)
Random Forest	(?)	<b>0.845</b> (?)	(?)	(?)

F1-Score	Numerical (oversample)		All Features (oversample)	Categorical (oversample)
	with standardization	w/out standardize		
Logistic Regression	0.7137 (?)	0.7139 (?)	0.7227 (?)	0.5897 (?)
Decision Tree	0.7138 (?)	0.7138 (?)	(?)	0.5961 (?)
Random Forest	(?)	<b>0.7438</b> (?)	(?)	(?)

P/R	Numerical (oversample)		All Features (oversample)	Categorical (oversample)
	with standardization	w/out standardize		
Logistic Regression	0.853/0.613 (?)	0.852/0.614 (?)	0.858/0.624 (?)	0.706/0.505 (?)
Decision Tree	0.878/0.601 (?)	0.878/0.601 (?)	(?)	0.732/0.502 (?)
Random Forest	(?)	<b>0.937/0.616</b> (?)	(?)	(?)