FRAUDOLENT TRANSACTION CLASSIFICATION

RICCARDO LA MARCA, MATR. 1795030

BIG DATA COMPUTING COURSE A.A. 2021/2022

OVERVIEW

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

I 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 fraudolent 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 fraudolent and not fraudolent transactions, is capable of classifying whether a new transaction is fraudolent or not.

~709 MiB

Divided into identity.csv and transaction.csv

590540 entries

434 features

Target feature is «isFraud». 0 means not fraudolent, 1 otherwise

2 THE DATASET

The Dataset is available on Kaggle

3 EXPLORE AND FEATURE ENGINEERING OUTLINE

Merging

- Transaction.csv and identity.csv have been merged together

Label Distributrion

- The dataset is highly unbalanced

Missing Values

- The dataset has a huge number of missing values

Feature Engineering

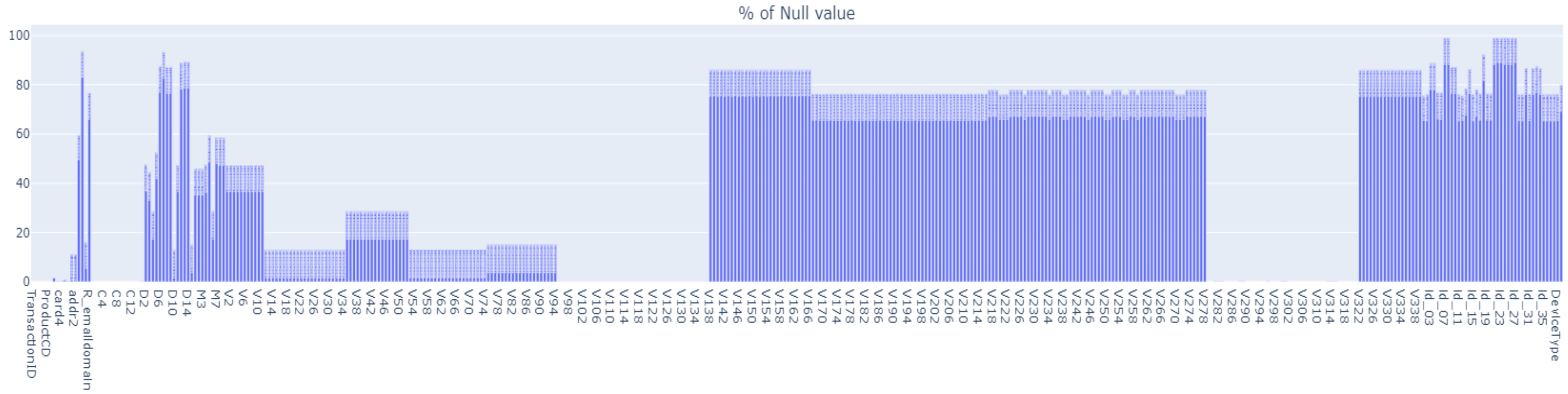
- Drop some features
- «Standardizing»
- Apply «Imputing»
- Use Pearson Correlation

3.1 - LABEL DISTRIBUTION

- With respect to the target label «isFraud» the dataset results highly unbalanced
- ~96.5 % are not-fraudolent transactions
- ~3.5 % are fraudolent 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



3.3 – FEATURE ENGINEERING

Features Dropping

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

Standardization

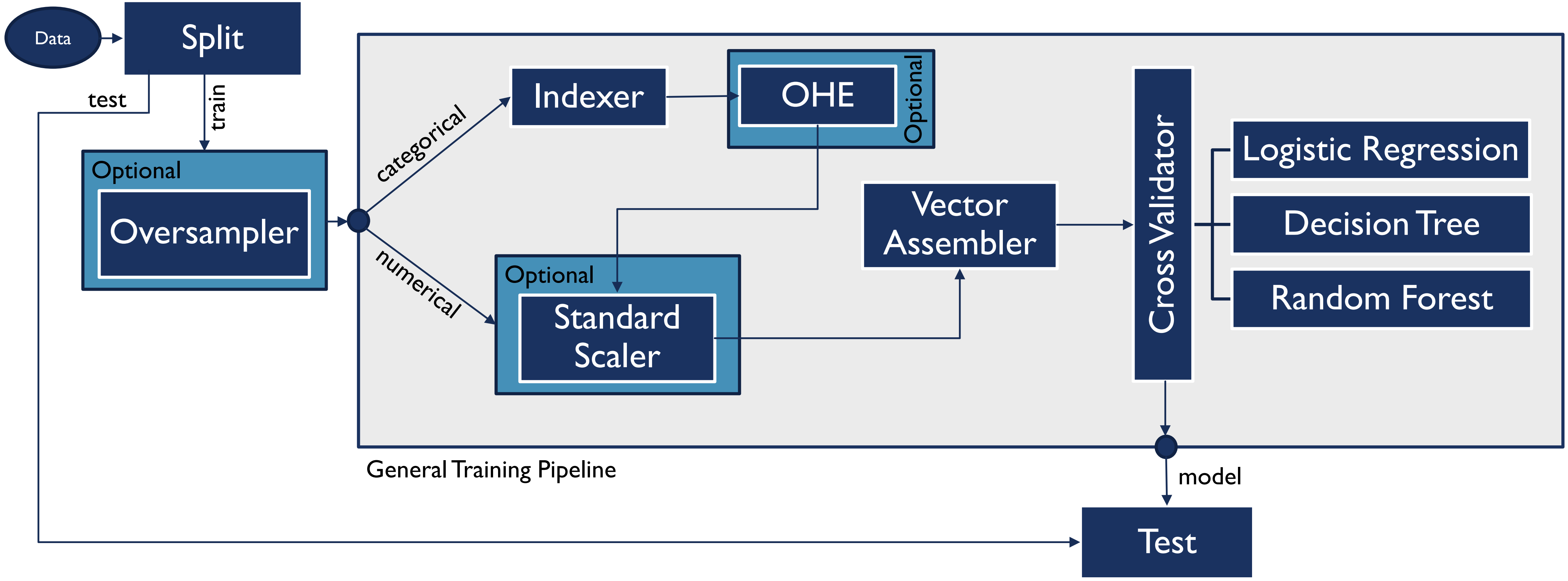
- Standardize certain features
- Given different values for the same feature but with equal meaning, replace with a single more general value
- Take *yahoo.co.jp*, *yahoo.co.uk* and *yahoo.net*, I replace it with *yahoo*

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



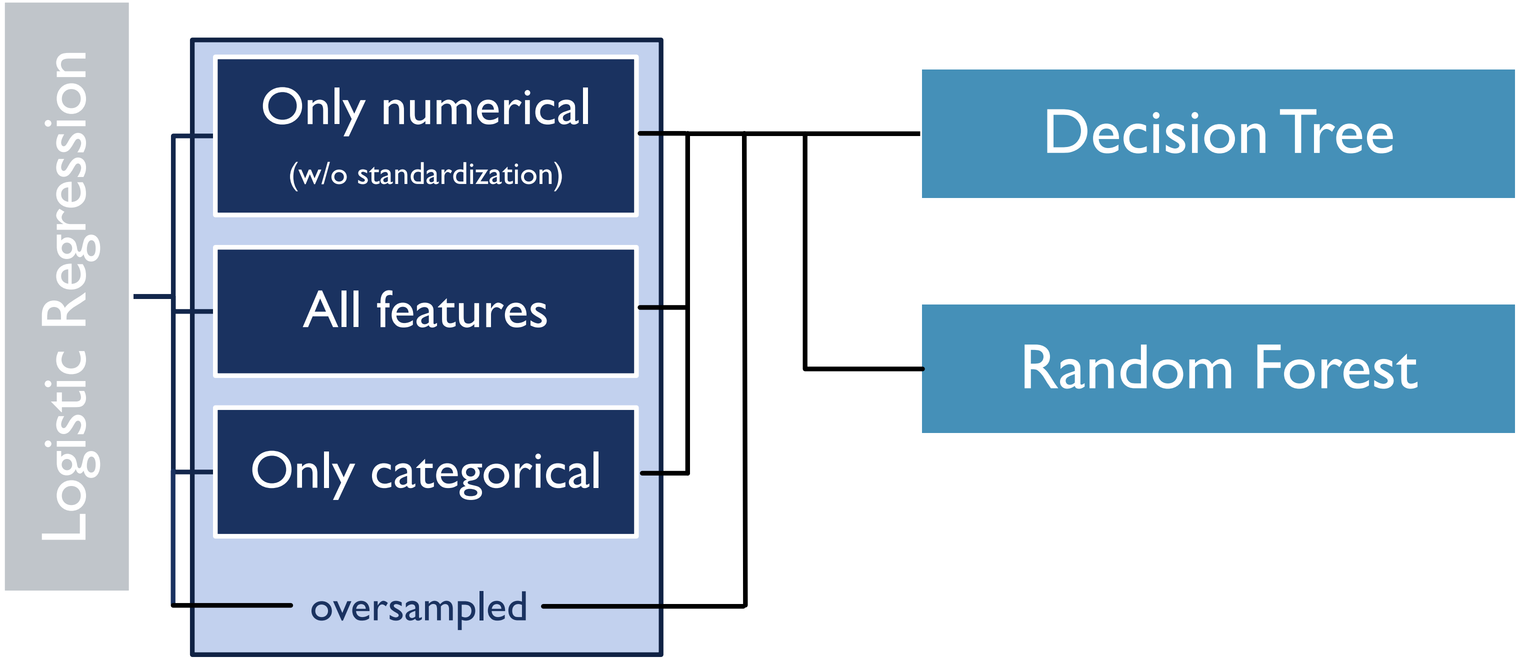
4 MACHINE LEARNING PIPELINE

- 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-fraudolent 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 x 232 (train set) and 233499 x 232 (test set)

4.1 – DATASET SPLITTING

- 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 fraudolent 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 fraduolent have been classified as fraduolent
- Finally, the train set contains 63.3% of non fraud and 36.7% of fraudolent

4 .2 – TRAIN OVERSAMPLER



4.3 - EXPERIMENTS

5.1 - EXPERIMENTAL RESULTS

Accuracy	Numerical (oversample)		All Features (oversample)	Categorical (oversample)
	with standardization	w/out standardize		
Logistic Regression	0.9772 (0.9099)	0.97725 (0.9099)	0.9777 (0.9033)	0.9733 (0.8604)
Decision Tree	0.9773 (0.8586)	0.9773 (0.8586)	**0.9790** (0.8586)	0.9734 (0.7862)
Random Forest	0.9787 (0.9390)	0.9789 (0.9397)	0.9786 (0.9410)	0.9734 (0.8829)

AUC ROC	Numerical (oversample)		All Features (oversample)	Categorical (oversample)
	with standardization	w/out standardize		
Logistic Regression	0.832 (0.840)	0.834 (0.840)	0.857 (0.862)	0.800 (0.801)
Decision Tree	0.428 (0.535)	0.428 (0.535)	**0.858** (0.535)	0.707 (0.679)
Random Forest	0.844 (0.864)	0.845 (0.866)	0.848 (0.872)	0.784 (0.810)

5.2 - EXPERIMENTAL RESULTS

F1-Score	Numerical (oversample)		All Features (oversample)	Categorical (oversample)
	with standardization	w/out standardize		
Logistic Regression	0.7137 (0.6446)	0.7139 (0.6446)	0.7227 (0.6556)	0.5897 (0.6109)
Decision Tree	0.7138 (0.6101)	0.7138 (0.6101)	**0.7472** (0.6101)	0.5961 (0.5999)
Random Forest	0.7406 (0.6787)	0.7438 (0.6787)	0.7390 (0.6842)	0.4932 (0.6165)

P/R	Numerical (oversample)		All Features (oversample)	Categorical (oversample)
	with standardization	w/out standardize		
Logistic Regression	0.853/0.613 (0.572/0.737)	0.852/0.614 (0.572/0.737)	0.858/0.624 (0.574/0.764)	0.706/0.505 (0.542/0.699)
Decision Tree	0.878/0.601 (0.541/0.698)	0.878/0.601 (0.541/0.698)	**0.869/0.654** (0.541/0.698)	0.732/0.502 (0.529/0.692)
Random Forest	0.935/0.613 (0.611/0.762)	0.937/0.616 (0.612/0.761)	0.930/0.613 (0.616/0.768)	0.486/0.5 (0.550/0.701)