Credit Card Fraud

Project II Checkpoint I - Artificial Inteligence

Grupo 04_1A

up201906086@up.pt Marcelo Henriques Couto

up201907361@up.pt Francisco Pinto de Oliveira

Faculdade de Engenharia da Universidade do Porto

June 6, 2022

Problem Specification

Credit card fraud is a crime easily stopable if detected. However, this detection must be quick to occur, as transactions are supposed to be fast. The **objective** of this project is to **develop a machine learning model** based on supervised learning classification algorithms able to distinguish, based on some input data, if a given transaction related to a bank account is fraudulent or not.

In order to come up with the ideal model, we must experiment with at least three different classification algorithms, as well as carry out an **Exploratory Data Analasys**.

Problem Specification

The dataset contains the following data:

- **Distance from home** - continuous
- **Distance from last transaction** - continuous
- **Ratio to median purchase price:** Ratio of the value of the transaction to the median transaction value - continuous
- **Repeat retailer:** Whether the retailer where the transaction happened had other transactions registered for the same person
- **Used chip:** Transcation via credit card - discrete, binary
- **Used pin number** - discrete, binary
- **Online order** - discrete, binary
- **Fraud** - discrete, binary - **label**

We considered that all distances are expressed in kilometers.

Related Work

Code

- Imbalanced Learn - over sampling and undersampling tools
- Scikit Learn - machine learning algorithms

Websites

- https://machinelearningmastery.com/what-is-imbalanced-classification/
- https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

Articles

- Bekkar, M., Djemaa, H. K., Alitouche, T. A. (2013). Evaluation measures for models assessment over imbalanced data sets. J Inf Eng Appl, 3(10).
- Tharwat, A. (2021), "Classification assessment methods", Applied Computing and Informatics, Vol. 17 No. 1, pp. 168-192.

Tools and Algorithms

Tools

We will use **Python** as programming language, programming in a **Jupyter Notebook** environment. For machine learning algorithms, we will resource to **Scikit-Learn** and **Imbalanced Learn** libraries, as well as **Pandas** to read and handle the data and **Seaborn and Matplotlib** to visualize it.

Algorithms

Because our dataset is imbalanced, we plan to explore **Synthetic Minority Oversampling** and **Undersampling** techniques to handle this issue.

For the classification we plan on using **Logistic Regression**, **Random Forest**, **Decision Trees**, **Neural Networks** and possibly others.

Implementation Already Carried Out

Language: Python - Anaconda

Environment: Visual Studio Code using JupyterNotebooks

Data Structures: The dataset is represented as a pandas.DataFrame **Data**

Preprocessing and **Exploratory Data Analysis**:

- Analysis of dataset's validity and integrity
- Analysis of dataset balance
- Analysis of outliers, missing values and other errors
- Analysis of correlations between the multiple variables

Model Training

- Logistic Regression with cross validation using Area Under ROC Curve as evaluation measure

Data Pre-Processing

Our data set presented to be ready to use as:

- There were no missing values
- There were no apparent measurement errors
- There were many outliers but they did not represent any errors and were, in all likelihood, the exact object of our analysis

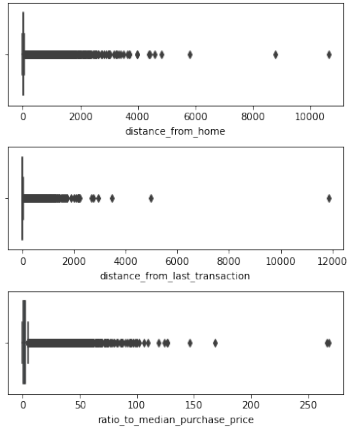

Figure: Boxplots

Exploratory Data Analysis

Performing an **Exploratory Data Analysis** we concluded that:

- As previously mentioned, our data set was moderately imbalanced (classification label is binary and 8.7% of the entries represent fraudulent transactions).
- Many of the features had little relation with the label feature, as their values did not vary much between the two label classes and their correlation was very low.

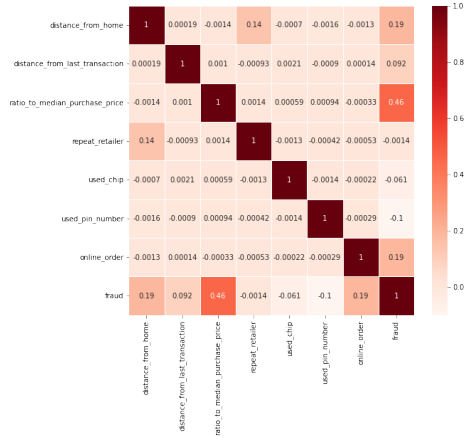

Figure: Correlation Matrix

Machine Learning Models

The algorithms we chose to train our models were:

- **Logistic Regression(LR):** Good with binary data, easy and simple to understand
- **Decision Tree(DT):** Decently robust to imbalanced datasets and irrelevant features
- **K Nearest Neighbours(KNN):** Overall simple
- **Random Forest(RF):** Handles imbalanced datasets well and is impervious to overfitting
- **Multilayer Perceptron(MLP):** Very powerful all arround

Notes:

- KNN was only tested with smaller data sets due to elevated training and testing times (lack of processing power)
- Logistic Regression was tested with a dataset where a column had been removed, in an attempt to fight the algorithm's sensitivity to irrelevant features
- All algorithms but MLP were tested with Grid Search with portions of the data set, in order to find the best parameters for the final model

Model Evaluation and Comparison

Model	Time to train (s)	Time to test (s)	Recall	Roc Auc	Precision	Accuracy	Balanced Accuracy
lr_control_full	19.497249	0.005409	0.599163	0.796077	0.890613	0.95874	0.796077
lr_smoted_full	55.498035	0.015769	0.980637	0.943067	0.497086	0.912032	0.943067
lr_weight_full	120.202866	0.016395	0.97291	0.94429	0.523563	0.920648	0.94429
dt_control_full	12.073779	0.008223	0.999908	0.999952	0.999954	0.999988	0.999952
dt_smoted_full	26.15653	0.016174	0.999862	0.999922	0.999816	0.999972	0.999922
dt_weight_full	6.897408	0.015349	0.999908	0.999952	0.999954	0.999988	0.999952
mlp_normal_full	960.845785	0.208947	0.892052	0.945036	0.977226	0.988804	0.945036
rf_control_full	361.549095	1.325936	0.999908	0.999954	1	0.999992	0.999954
rf_smoted_full	132.806744	0.117373	0.999908	0.99995	0.999908	0.999984	0.99995
rf_weight_full	303.098518	0.771723	0.999908	0.999952	0.999954	0.999988	0.999952
knn_control_10p	19.848561	47.906985	0.828354	0.904025	0.79605	0.966481	0.904025
knn_smoted_10p	233.760107	181.564677	0.989553	0.993826	0.980322	0.997353	0.993826
knn_none_10p	119.013859	121.487901	0.983827	0.991484	0.990951	0.997803	0.991484

Note: control are models trained with default parameters, **weight** are models trained on normal data sets but best parameters in grid search (focusing fixing the imbalance with class weights), **smote** are models trained with best parameters from grid search and using **SMOTE** in training data set, **none** is the same as weight but without class weights (for parameter details, check notebook)

Model Evaluation and Comparison

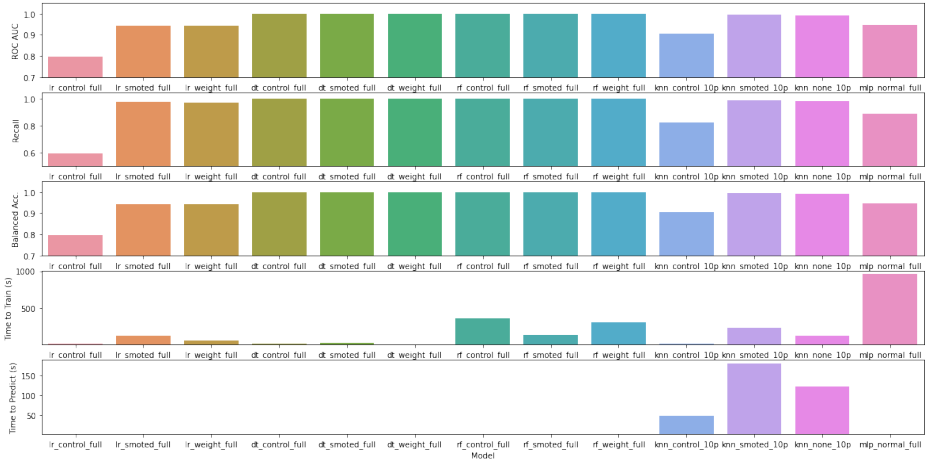

Figure: Model Scores and Times

Conclusions

From the graphs and table we are able to conclude that:

- The best models were the ones based in Random Forest and Decision Tree algorithms, which goes accordingly to their resistance against imbalance. Random Forest did not present much advantage as the Decision Tree models already featured perfect scores and clearly did not suffer from overfitting
- The worse models were the ones based in Logistic Regression algorithms, most likely because of their sensitivity to imbalance and overall lack of robustness
- K Nearest Neighbours models were way too slow to be a viable option, so much so that we only trained and tested with 10% of the data set
- Multi-layer Perceptron based models were not worth the complexity, as their results were not their great and the training time was big
- SMOTE and class weights had a decent impact in LR, yet they did not show mutch effects on the other algorithms