ABSTRACT

Detecting frauds in credit card transactions is perhaps one of the best test-beds for computational intelligence algorithms. In fact, this problem involves a number of relevant challenges, namely: concept drift (customers’ habits evolve and fraudsters change their strategies over time), class imbalance (genuine transactions far outnumber frauds), and verification latency (only a small set of transactions are timely checked by investigators). However, the vast majority of learning algorithms that have been proposed for fraud detection rely on assumptions that hardly hold in a real-world fraud-detection system (FDS). This lack of realism concerns two main aspects: 1) the way and timing with which supervised information is provided and 2) the measures used to assess fraud-detection performance. This paper has three major contributions. First, we propose, with the help of our industrial partner, a formalization of the fraud-detection problem that realistically describes the operating conditions of FDSs that everyday analyzes massive streams of credit card transactions. We also illustrate the most appropriate performance measures to be used for fraud-detection purposes. Second, we design and assess a novel learning strategy that effectively addresses class imbalance, concept drift, and verification latency. Third, in our experiments, we demonstrate the impact of class unbalance and concept drift in a real-world data stream containing more than 75 million transactions, authorized over a time window of three years.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Chapter. No** | **Title** | **Page No.** |
|  | **Abstract** | **i** |
|  | **Table of Contents** | **ii** |
|  | **List of Figures** | **iv** |
|  | **Abbreviation** | **v** |
| **1.** | **INTRODUCTION** | 1-4 |
|  | 1.1 Introduction | 1 |
|  | 1.2 Existing System | 2 |
|  | 1.3 Proposed System | 4 |
|  | 1.4 Advantages of proposed system | 4 |
| **2.** | **REQUIREMENT ANALYSIS** | **5-6** |
|  | 2.1 Functional requirement | 5 |
|  | 2.2 Non Functional requirements | 6 |
|  | 2.3 Computational resource requirements  2.3.1 Hardware requirements  2.3.2 Software requirements | 6  6  6 |
| **3** | **DESIGN** | **7-10** |
|  | 3.1 Design | 7 |
|  | 3.2 Architecture | 7 |
|  | 3.3 Class Diagram | 8 |
|  | 3.4 Sequence Diagram | 9 |
|  | 3.5 State chart Diagram | 10 |
| **4** | **MODULES** | **11-12** |
|  | 4.1 Modules | 11 |
| **5** | **IMPLEMENTATION** | 13-14 |
|  | 5.1 Source Code | 13 |
|  | 5.2 Sample Dataset | 14 |
| **6** | **SCREENSHOTS** | **15-25** |
|  | 6.1 Screenshots  6.1.1 Importing libraries  6.1.2 Normal Transaction  6.1.3 Fraud x Normal Transaction by hours  6.1.4 Fraud x Normal Transaction by minutes  6.1.5 Class x Amount  6.1.6 Amount by minutes of Fraud and Normal Transactions  6.1.7 Amount by Hours of Fraud and Normal Transactions  6.1.8 V’s Features  6.1.9 Correlation Matrix  6.1.10 Feature Importance Plot  6.1.11 ROC Curve – Random Forest | 15  15  16  17  18  19  20  21  22  23  24  25 |
| **7** | **TESTING** | **26-29** |
|  | 7.1 Overview of Testing | 26 |
|  | 7.2 Types of Testing  7.2.1 Unit Testing.  7.2.2 Integration Testing.  7.2.3 Functional Testing. | 26  26  26  27 |
|  | 7.3 Unit Testing | 28 |
|  | 7.4 Integration Testing | 28 |
|  | 7.5 Acceptance Testing. | 28 |
| **8** | **CONCLUSION AND FUTURE WORK** | **30-31** |
|  | 8.1 Conclusion | 30 |
|  | 8.2 Scope for future work | 31 |
| **9** | **BIBILOGRAPHY** | 32 |

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Sl No** | **Title** | **Page No.** |
| 3.1 | Architecture | 7 |
| 3.2 | Class Diagram | 8 |
| 3.3 | Sequence Diagram | 9 |
| 3.4 | State Chart Diagram | 10 |
| 6.1 | Importing libraries | 15 |
| 6.2 | Normal Transaction | 16 |
| 6.3 | Fraud x Normal Transaction by hours | 17 |
| 6.4 | Fraud x Normal Transaction by minutes | 18 |
| 6.5 | Class x Amount | 19 |
| 6.6 | Amount by minutes of Fraud and Normal Transactions | 20 |
| 6.7 | Amount by Hours of Fraud and Normal Transactions | 21 |
| 6.8 | V’s Features | 22 |
| 6.9 | Correlation Matrix | 23 |
| 6.10 | Feature Importance Plot | 24 |
| 6.11 | ROC Curve – Random Forest | 25 |

ABBREVIATION

|  |  |
| --- | --- |
| GUI | Graphical User Interface |
| PY | Python |
| OPP | Object Oriented Programming |
| DRY | Don't Repeat Yourself |
| PIP | Package Installer for Python |
| MRO | Method Resolution Order |
| PEP | Python Enhancement Proposals |
| BDFL | Benevolent Dictator For Life |
| REPL | Read-Eval-Print Loop |