Software Requirements Specification
Version 1.0

Risk Sentinel

Theme: Fraud Shield

Category: Big Data Processing Using Data Science Technologies

Contents

1.1 Background and Necessity for the Application

The occurrence of cybercrime encompasses a wide range of criminal activities conducted through, within, or against computer applications and networks. As the threats of online fraud and unethical hacking continue to grow, instances of cybercrime are on the rise. To counter these escalating information and cyber safety risks, organizations must be prepared to predict and prevent cybercrime.

Cybercrime experts are leveraging Big Data tools to identify potential threats and detect incidents such as credit card fraud. Big Data Analytics empowers companies to analyze large volumes of data accumulated during financial transactions and other locale-specific information. Combating cybercrime has become crucial in today's landscape due to the serious risk of cyber theft. Big Data tools are employed to defend against cyber-attacks, facilitating the detection of fraud, identification theft, and aiding in digital forensic analysis.

1.2 Proposed Solution

The prevalence of credit card usage in today's society is undeniable. However, despite the worldwide integration of chip cards and existing protection applications, the number of credit card fraud cases continues to rise. This emphasizes the criticality of fraud detection in the present day. Credit card fraud

detection is a pervasive issue, particularly with the surge in online transactions and
e-commerce platforms. Fraudulent activity often occurs when a credit card is stolen or when unauthorized individuals exploit credit card information for personal gain. Given the widespread credit card issues, the introduction of a credit card fraud detection application has become imperative. This project aims to primarily validate various business rules to determine whether a transaction is fraudulent or genuine, promptly reporting the findings accordingly.

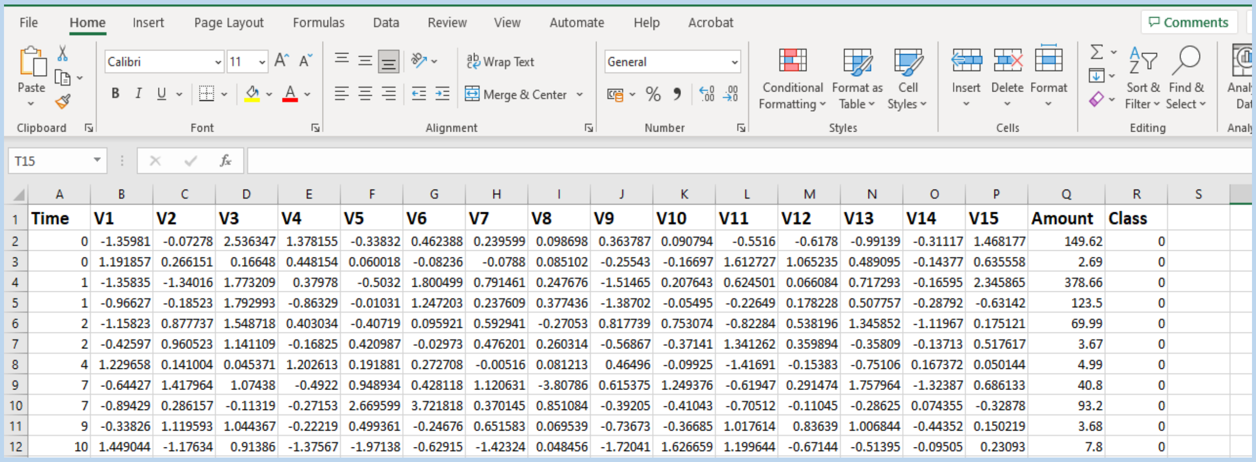
The implementation of a credit card fraud detection application should incorporate Big Data to enhance its capabilities and strive for improved performance. This can be achieved by considering three key rules:

1. **Upper Control Limit:** This rule sets a threshold or limit beyond which transactions are flagged as potentially fraudulent. By monitoring transaction amounts and comparing them to the upper control limit, suspicious activities can be identified.

2. **Credit Score:** The credit score of a customer is an essential factor in assessing the likelihood of fraudulent transactions. By considering credit scores as part of the fraud detection process, suspicious activities can be identified based on deviations or anomalies in credit scores.

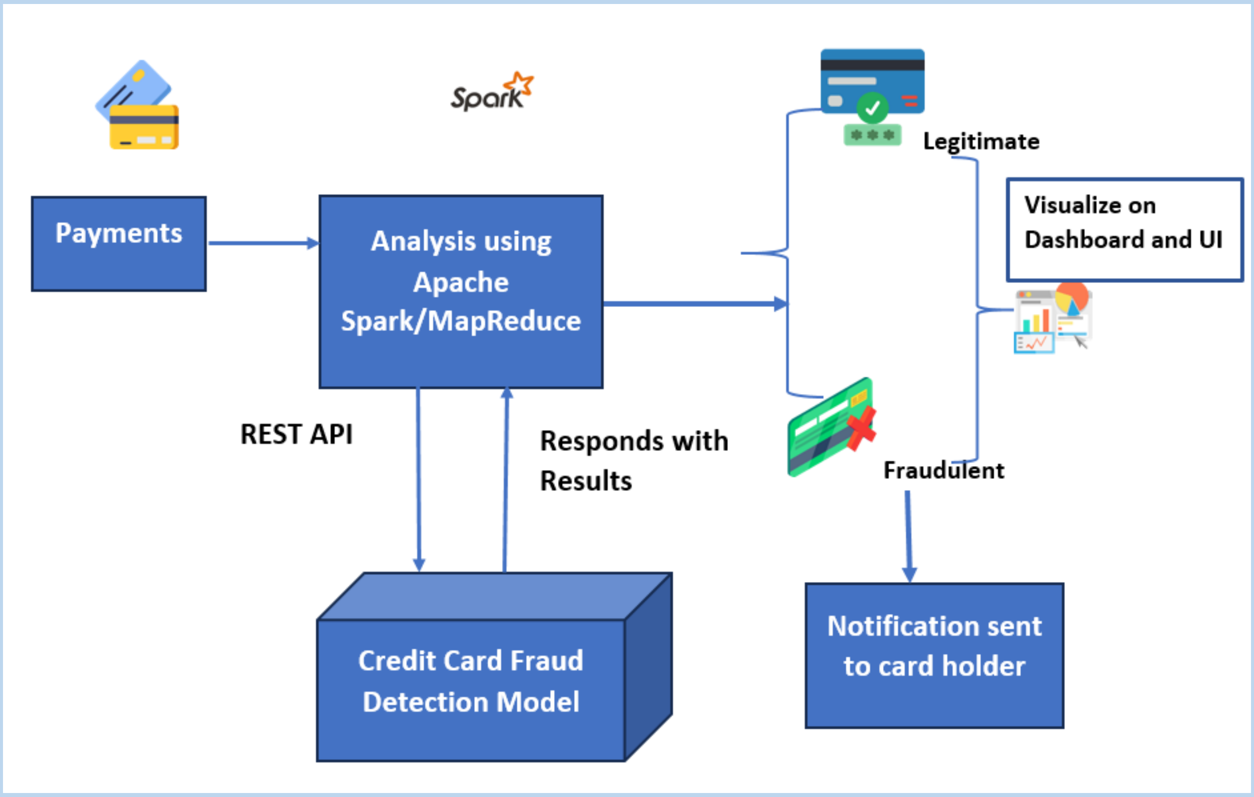
3. **Zip Code Distance:** This rule takes into account the distance between the customer's registered zip code and the location of the transaction. Unusually large distances or discrepancies between the two can raise suspicion and trigger further investigation into potential fraud.

Hint: The sample of the dataset ccFraud downloaded from Kaggle for implementation purpose is as follows:



Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	Amount	Class
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	149.62	0
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	2.69	0
1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465	0.207643	0.624501	0.066084	0.717293	-0.16595	2.345865	378.66	0
1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495	-0.22649	0.178228	0.507757	-0.28792	-0.63142	123.5	0
2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074	-0.82284	0.538196	1.345852	-1.11967	0.175121	69.99	0
2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141	1.341262	0.359894	-0.35809	-0.13713	0.517617	3.67	0
4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.167372	0.050144	4.99	0
7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375	1.249376	-0.61947	0.291474	1.757964	-1.32387	0.686133	40.8	0
7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.074355	-0.32878	93.2	0
9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685	1.017614	0.83639	1.006844	-0.44352	0.150219	3.68	0
10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	1.626659	1.199644	-0.67144	-0.51395	-0.09505	0.23093	7.8	0

The sample architecture can be as follows:


Spark
Legitimate
Payments
Analysis using Apache Spark/MapReduce
Visualize on Dashboard and UI
REST API
Responds with Results
Fraudulent
Credit Card Fraud Detection Model
Notification sent to card holder

Sample Architecture of the Application

The development phase of the application includes following steps:

1. Table creation
2. Batch processing
3. Streaming
4. Visualizing the results
5. Sending alert notification to card holder for fraudulent transactions

1.3 Purpose of the Document

The purpose of this document is to present a detailed description of **‘Risk Sentinel’** to check the authenticity of transactions.

This document explains the purpose and features of **Risk Sentinel** and the constraints under which it must operate. This document is intended for both stakeholders and developers of the application.

1.4 Scope of Project

The application's primary objective is to accurately identify credit card transactions that are likely to be fraudulent. By leveraging Big Data analytics and advanced algorithms, the application can detect patterns, anomalies, and indicators of fraudulent behavior.

Detecting fraudulent transactions in real-time is crucial to minimize losses. Processing and analyzing data in near real-time to identify suspicious patterns and anomalies requires sophisticated algorithms and efficient processing techniques which is beyond the scope of this project. Furthermore, it should be noted that notifying bank authorities regarding fraudulent transactions falls outside the scope of this project.

1.5 Constraints

Inadequate resources, infrastructure limitations, or scalability challenges can hinder the application's ability to handle increasing data volumes and process transactions in real-time. Application failures, delays, or high latency can compromise the effectiveness of fraud detection.

1.6 Functional Requirements

Employing Big Data analytics in various aspects of fraud detection offers numerous advantages. Following functional requirements with Hadoop and data mining should be implemented:

i. **Load data into Hadoop cluster –** The data should be loaded into the Hadoop cluster making it ready for analytics.

ii. **Use Modelling Strategies -** Data mining strategies can be classified into two main categories: supervised learning and unsupervised learning. Supervised learning methods are utilized when there is a target variable with known values, and the goal is to make predictions using other input variables. In contrast, unsupervised learning methods are employed when there is no target variable with known values, but input variables are available for analysis.

iii. **Use Big Data Technique -** Hadoop facilitates the robust and distributed processing of vast unstructured data sets across clusters of commodity computers, where each node in the cluster has its own storage. Credit card transactions are analyzed using MapReduce logic in Hadoop Distributed File Application (HDFS), which offers an efficient approach for data analysis and rapid response. The MapReduce framework plays a crucial role by distributing tasks to different nodes within the cluster (known as mapping) and aggregating and consolidating the results from each node to provide a unified response to a query. The process involves importing data into HDFS using Flume/Sqoop, followed by running MapReduce (MR) and using PigLatin for data operations. After the operations are completed, the data is exported to an RDBMS, and visualization is created using Tableau. You can utilize HDFS and HBase for data storage and leverage Impala, Hive, or HBase for performing queries based on the project requirements and preferences.

iv. **Train, Validate, and Test -** The process of developing a model starts with dividing the data sets into three distinct parts: a training set, a validation set, and a test set. The training set is used to train the model, the validation set is used to assess and fine-tune the model's performance, and the test set is used to evaluate the final trained and validated model. By splitting the data in this manner, it ensures that the model does not simply memorize specific subsets of data, but instead learns generalized patterns and can be assessed for its performance on unseen data.

v. **Analyze Results -** The evaluation diagnostics employed in model assessment differ between supervised and unsupervised learning. In the case of classification problems, analysts commonly analyze gain, lift, and profit charts, threshold charts, confusion matrices, and various fit statistics for both the training and validation sets, or alternatively, for the test set. Additionally, having business domain knowledge is highly valuable in interpreting the results of the model.

vi. **Implement Results -** Clustering models can be assessed in terms of their overall performance or the quality of specific data groupings. When evaluating the overall model, the diagnostics typically aim to measure the effectiveness of the model in categorizing the input data into distinct sets of similar cases.

The Big Data application should also be able to implement following functionalities:

- Notify users through email or SMS in the event of fraudulent transactions.
- Visualize the data on a dashboard or user interface for easy comprehension.
- Send users confirmation messages for legitimate transactions.

1.7 Non-Functional Requirements

There are several non-functional requirements that should be fulfilled by the application as follows:

1. **<u>Performance:</u>** The application should be able to process large volumes of data efficiently and provide real-time or near real-time fraud detection. It should handle complex data processing algorithms and analytics within acceptable response times.

2. **<u>Scalable</u>**: The application should be designed to handle increasing data volumes and growing computational demands. It should be able to scale horizontally by distributing the workload across multiple nodes or clusters as the data size and processing requirements increase.

3. **<u>Secure</u>**: Robust security measures should be in place to protect sensitive credit cards and customer information. This includes secure data transmission, encryption of data at rest, access controls, and compliance with data protection regulations.

4. **<u>Usable</u>**: The application should have a user-friendly interface that allows easy interaction with the data and fraud detection features. It should provide intuitive visualizations and meaningful insights to users, enabling effective decision-making and fraud investigation.

5. **<u>Reliable:</u>** The application should be highly reliable, minimizing downtime and ensuring continuous operation. It should have built-in fault tolerance mechanisms, such as data replication, backup and recovery procedures, and automated monitoring and alerting for potential failures.

These are the bare minimum expectations from the project. It is a must to implement the functional and non-functional requirements given in this SRS. Once they are complete, you can use your own creativity and imagination to add more features if required.

1.8 Interface Requirements

1.8.1 Hardware

Intel Core i5 Processor or higher
8 GB RAM or higher
Color SVGA
500 GB Hard Disk space
Mouse
Keyboard

1.8.2 Software

Technologies to be used:

1. **Frontend**: HTML5, JavaScript, CSS3, or any other frontend programming languages
2. **Data Store**: HDFS 3.x, Apache Spark 3.x, Apache Hive 3.x or higher, Apache HBase 2.x, MongoDB 5.x
3. **Backend**: Impala 4.x or higher
4. **Programming**: Python 3.x/R 4.x (or higher)
5. **Visualization**: Tableau Desktop 2023.2

1.9 Project Deliverables

You will design and build the project and submit it along with a complete project report that includes:

- Problem Definition
- Design specifications
- Diagrams such as user journey map
- Source Code
- Test Data Used in the Project
- Project Installation Instructions for configuring Tableau, Big Data technologies, and UI code
- Database Design
- Link of published blog
- Link of GitHub for accessing the uploaded project code (Link should have public access.)
- Tableau Dashboard with appropriate access rights or permission to access for testing

The consolidated project must be submitted on GitHub with a ReadMe.doc file listing assumptions (if any) made at your end. Ensure that you provide the GitHub URL where the project has been uploaded for sharing. The repository on GitHub should have public access. Documentation is a very important part of the project hence, all crucial aspects of the project must be documented properly. Ensure that documentation is complete and comprehensive.

You should publish a blog of minimum 2000 words on any free blogging Website such as Blogger, Tumblr, Ghost or any other blogging Website. The link of the published blog should be submitted along with the project documentation.

In addition, you must submit a video clip showing the actual working of the application. Over and above the given specifications, you can apply your creativity and logic to improve the application.

*~~~ End of Document ~~~*