Abstract

Fraud can be defined as criminal activities with the intent of acquiring financial gain. High dependence on internet technology has enjoyed increased credit card transactions. As credit card transactions become the most prevailing mode of payment for both online and offline transaction, credit card fraud rate also increases. Detecting fraudulent transactions using traditional methods of manual detection is time-consuming and inefficient, thus the arrival of big data has made manual methods more impractical.

With the increase of transactions at massive scale, the imbalanced data is huge and has created a challenging issue on how well Machine Learning (ML) techniques can scale up to efficiently learn to detect fraud from the massive incoming data and to respond faster with high prediction accuracy and reduced misclassification costs. We are using Random Forest Model to detect the fraud. Our aim here is to predict the fraud transactions while minimizing the incorrect fraud classifications.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Basis

Fraud can be defined as criminal activities with the intent of acquiring financial gain. High dependence on internet technology has enjoyed increased credit card transactions. As credit card transactions become the most prevailing mode of payment for both online and offline transaction, credit card fraud rate also increases. Detecting fraudulent transactions using traditional methods of manual detection time-consuming and inefficient, thus the arrival of big data has made manual methods more impractical.

With the increase of transactions at massive scale, the imbalanced data is huge and has created a challenging issue on how well Machine Learning (ML) techniques can scale up to efficiently learn to detect fraud from the massive incoming data and to respond faster with high prediction accuracy and reduced misclassification costs. We are using the Logistic Regression method to detect fraud. Our aim here is to predict fraud transactions while minimizing the incorrect fraud classifications.

Credit card fraud detection is a challenging task for the user. Online payment does not read as a physical card. And if anyone who knows the owner of the card can make the transactions. Currently, the cardholder comes to know only after the fraud transaction is carried out. No proper mechanism is there to track the fraud transaction.

Now a day the customers prefer the most accepted payment mode via credit card for the convenient way of paying bills, online easiest way. At the same time, the fraud transaction risk s using a credit card is the main problem that should be avoided. So There are many data mining techniques available to avoid hence risks effectively. In existing research, they modelled the sequence of operations in credit card transaction processing using a Hidden Markov Model (HMM) and shown how it can be used for the detection of frauds. To avoid computational complexity and to provide better accuracy in fraud detection in proposed work.

With the developments in the information technology and improvements in the communication channels, fraud is spreading all over the world, resulting in huge financial losses. There have been several types of research done in the field of fraud detection, with various methods employed in detection and prevention. Methods such as decision tree learning, support vector machines, neural networks, expert systems, and artificial immune systems have been explored and identified for fraud detection. The scope of this paper has been reduced to only credit card application fraud and risk based on decision tree induction using ensemble learning techniques and genetic algorithms. Due to the sensitivity of customers' financial information, getting clean data is hard for mining applications. The dataset used is obtained from the UCI (University of California, Irvine) machine learning repository-German Credit Card data set and Australian Credit Card dataset for the present paper.

Fraud prevention is a subject that always brought interest from financial institutions, since the advent of new technologies like telephone, automatic teller machines (ATM), and credit card systems have leveraged the volume of fraud loss of many banks. In this context, fraud prevention, with special importance of fraud automatic detection, arises as an open field for application of all known classification methods. Classification techniques play a very important role, once it can learn from experience (fraud happened in the past) and classify new instances (transactions) in a fraud group or a legitimate group.

A credit card is a thin handy plastic card that contains identification information such as a signature or picture, ad authorizes the person named on it to charge purchases or services to his account - charges for which he will be billed periodically. Today, the information on the card is read by automated teller machines (ATMs), store readers, banks are also used in the online internet banking system. They have a unique card number which is of utmost importance. Its security relies on the physical security of the plastic card as well as the privacy of the credit card number. There is a rapid growth in the number of credit card transactions which has led to a substantial rise in fraudulent activities.

Credit card fraud is a wide-ranging term for theft and fraud committed using a credit card as a fraudulent source of funds in a given transaction. Generally, statistical methods and many data mining algorithms are used to solve this fraud detection problem. Most of the credit card fraud detection systems are based on artificial intelligence, Meta-learning, and pattern matching. Genetic algorithms are evolutionary algorithms that aim to obtain better solutions in eliminating fraud. High importance is given to developing an efficient and secure electronic payment system to detect whether a transaction is fraudulent or not. In this paper, we will focus on credit card fraud and its detection measures.

A credit card fraud occurs when one individual uses other individuals' card for their personal use without the knowledge of its owner. When such kind of cases takes place by fraudsters, it is used until its entire available limit is depleted. Thus, we need a solution that minimizes the total available limit on the credit card which is more prominent to frauds. And, a Genetic algorithm generates better solutions as time progresses. The complete emphasis is given on developing an efficient and secure electronic payment system for detecting fraudulent transactions.

1.2 Literature Review

Inside Prior Delving, many ideas are given forward out the key for finding deceit by supervised ways, an unsupervised way for mixed type. That made it vital to gain knowledge about technologies about credit card deceit transaction detection, also which gives a logical coherent to segregation of credit card frauds with time goes on, credit card fraud format gets new type to determine the particular motive of researchers. Part of it gives recollecting of the descriptive single machine learning algorithm, machine learning models, and deceit finding format that is utilized in deceit findings.it solves all the problems for the future a systematic machine learning model.

This system is used to engage with deceit finding in bank transactions [1] by fuzzy clustering and neural network. There is three-part in this. The first one is the prime user's genuine and cross-checking of card features. then victoriously finishing this work, the fuzzy c-means clustering algorithm is carried out to search the ordinary utilization of users based on all previous transactions. For any recent transaction get by this technique to be tentative in this section, way carried out by neural network-based is to decide to check of real deceit transaction if any.

Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang at [2] gave a convolutional neural network (CNN) based method for getting any illegal transactions. Convolutional Neural Network is a part of deep learning and is a type to feed-upcoming Neural Network that have more of concealing surfaces. Inside this research paper, to detecting much intricate detect format and to enhance segregation exactness, put recent trading randomness facets. for solving the issue of the un-equilibrium dataset, the price concerning the exampling procedure is used to bring much amount of deceit. Generally, CNN is utilized to character recognition, image recognition, image processing, video recognition, and recommender system. Inside this article for the unique, CNN help to get deceit finding.

Along with the examination of various deceit finding models, back scholars had got many issues regarding fraud detection. In [4] Scholars illustrated an Insufficient of real-life data as a big issue. Real-life data is the missing reason for data diplomacy and private problems. Paper [3] has revised Un-equilibrium data or uneven allotment of data. & Reason for having this is very lowness of deceit. When equating to un-deceit in the transaction datasets given by paper [3] the problem happens due to some occurrence of an event when the non-fraudulent transactions seem so deceit transactions. In the second mode, deceit transactions may materialize by authorized transactions. With Tackling the typical data problem. Along taking heed to the credit card transaction data, many of the facets have categorical data. Inside this thin, almost the machine learning algorithms do not support the categorical values. In [5] another method finding deceit by algorithms and feature selection as abet in finding deceit by much machine learning algorithms takes time for training purposes than predicting it. The next main Problem which harms the financial deceit finding is the feature selection. It gives priority to filter out the attributes that mostly gives the ways of deceit finding & its characters. In paper [3] given as denoted deceit finding price and low surviving dispute in the deceit finding process. When taking into heed of a structured system, the price of deceit should be taken into mind. Less compatibility happens when an algorithm is open to the latest type of deceit format and simple transactions. Productiveness varies based on the issue meant and its particularize, so a superior coherence of the degree of activities is vital [5].

Ghosh and Reilly [9] used a three-layer feed-forward Neural network to detect frauds in 1994. The Neural Network was trained on examples of fraud containing stolen cards, application fraud, counterfeit fraud, Non-Received Issue (NRI) fraud, and mail order fraud. Abhinav and Amlan [7] proposed a Hidden Markov Model to detect the frauds in credit cards. The proposed model does not require fraud signatures and still, it can detect fraud by considering a cardholder's spending habit. This system is also scalable to handle a large number of transactions. Y. Sahin and E. Duman [6] proposed an approach to detect credit card fraud by decision tree and Support Vector Machine. Performance of classifier models of various decision tree methods (C5.0, C&RT, and CHAID) and several different SVM methods (SVM with polynomial, sigmoid, linear and RBF kernel functions) are compared in this study. An approach is proposed towards fraud detection in banking transactions in [6] using fuzzy clustering and neural network. In this approach fraud detection is done in three phases. The first phase is initial user authentication and verification of card details. After completing this phase, a fuzzy c-means clustering algorithm is performed to find out the normal usage behaviour of users based on past transactions. If a new transaction is found to be doubtful in this phase, a mechanism

based on neural-network-based is applied to determine whether it was a fraudulent transaction or not. Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang at [3] proposed a convolutional neural network (CNN) based approach to find fraudulent transactions. Convolutional Neural Network is a part of deep learning and is a type of feed-forward Neural Network that consists of more than one hidden layer. In this paper, for finding more complex fraud patterns and to improve classification accuracy, a new feature trading entropy is proposed. To relieve the problem of the imbalanced dataset, a cost-based sampling method is used to generate a greater number of frauds. Generally, CNN is used for image recognition, Character recognition, image processing, video recognition, and recommender system. In this paper for the first time, CNN is used to detect fraud. Different outlier techniques [10] can also use to differentiate fraudulent transactions as outlier data.

One of the biggest problems associated with researchers in fraud detection is the lack of real-life data because of the sensitivity of data and privacy issues. Many researchers have researched real-life data [7], [9], [6], [11] of banks with agreements. To deal with this problem, many tools are available to generate synthetic data. The second problem is to deal with Imbalance data or skewed distribution because several fraudulent transactions are very less compared to legitimate transactions. To overcome this problem, synthetic minoring oversampling methods are used to increase the number of low incidence data in datasets that generate synthetic fraudulent transactions related to the original data set. In [3], cost-based sampling is used to generate synthetic fraudulent transactions to balance data set. Overlapping of data is one more problem as some of the transactions look like fraudulent transactions when they are legitimate transactions. It is also possible that fraudulent transactions appear to be normal transactions.

Later use of implementing an efficient machine learning model. With the analysis of various detection models, past researchers have found many problems regarding fraud detection. In [12] and [3] they have mentioned a lack of real-life data as a huge issue. Real-life data are lacking because of data sensitivity and privacy issues. Papers [3] and [7] have studied Imbalance data or skewed distribution of data. The reason behind this is having quite a less amount of fraud when compared to non-frauds in the transaction datasets. Paper [3] states that data mining techniques take time to execute when dealing with big data. Overlapping of data is another major drawback in the preparation of credit card transaction data. According to paper [2] and [7] the issue occurs due to some scenarios when the legitimate transactions look exactly like fraudulent transactions. In another way, fraudulent transactions may appear as legitimate transactions. Also, they have come across the difficulty of dealing with categorical data. When considering the credit card transaction data, most of the features have categorical values. In this

case, almost all the machine learning algorithms do not support categorical values. In [3][4] they have mentioned the choice of detection algorithms and feature selection as a challenge in detecting frauds since most of the machine learning algorithms take much time for training purposes than predicting. Another key issue that affects financial fraud detection is feature selection. It aims to filter out the attributes that most describes the aspects of fraud detection and its characters. In paper [7] they have highlighted fraud detection costs and lack of adaptability as challenges in the fraud detection process. When considering a system, the cost of fraudulent behaviour and the prevention cost should be taken into consideration. Lack of adaptability occurs when the algorithm is exposed to new types of fraud patterns and normal transactions. Effectiveness can change according to the problem definition and its specifications, so having a good understanding of the performance measure is necessary [4]. There are different kinds of models implemented for credit card fraud detections. In those models, different algorithms have been used. Adapting the fraud detection system to newly introduced frauds can be problematic whether to retrain the machine learning model due to drastic changes in the fraud patterns, also may be costly and risky. For instance, Tyler et al. extended a framework proposed in [12], implemented the model and the model was applied to a real-world transaction log. To address the classification problem Logistic Regression (LR) has been used. The instances of fraudulent transactions have been discretized into strategies by using Gaussian Mixture Models (GMMs). Here synthetic minority oversampling technique was used to address the class imbalance. To stand out the significance of estimates in economic value sensitivity analysis has been used. The results have proven that a practical method that uses minimal steps to retrain a model could function as same as a classifier that typically retrains every round [10]. There is another model called Risk-Based Ensemble (RBE) that can handle the data consisting of issues and give outstanding results. For handling imbalanced data, a highly efficient bagging model has been used. To handle the implicit noise in the transaction dataset they have used the Naive Bayes algorithm [9]. Peter et al. evaluated several deep learning algorithms concerning their efficacy. The four topologies are Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), Long Short-term Memory (LSTMs), and Artificial Neural Networks (ANNs). In their project in addition to data cleaning and other data preparation steps, they have overcome class imbalance and scalability problems by using under-sampling. To discover which hyper-parameters had the highest influence on the performance of the model, the sensitivity analysis was carried out.

They have discovered that the performance of the model was affected by the size of the network. They concluded that the larger the network showed better performance. [15] Credit card data

have the issue of skewed distribution which is also known as the class imbalance. According to Andrea et al., their project addresses class imbalance including other issues such as concept drift and verification latency. They have also illustrated the most relevant performance matrix that can be used in credit card fraud detection. The achievement of the research also includes a formal model and a powerful learning strategy for addressing the 'verification latency' and an 'alert and feedback' mechanism. According to experiments they have declared the precision of the alerts as the most important measure [11]. Chee et al. used twelve standard models and hybrid methods that use AdaBoost and majority voting methods to achieve better accuracy rates in credit card fraud detection [9]. They were evaluated using both benchmark and real-world data. A summary of the strengths and limitations of the methods was evaluated. The Matthews Correlation Coefficient metric (MCC) has been taken as the performance measure. To evaluate the robustness of the algorithms noise was added to the data. Also, they have proved that the majority voting method was not affected by the added noise. The analysis carried out on highly imbalanced data in paper [7] shows that KNN shows outstanding performance for sensitivity, specificity, and MCC, except for accuracy. The paper [6] discussed commonly used supervised techniques and they have provided a thorough evaluation of supervised learning techniques. Also, they have shown that all algorithms change according to the problem area. The fraud detection system presented in paper [8] is built to handle class imbalance, the formation of labelled and unlabelled, and the processing of large datasets. The proposed system was able to overcome all the challenges.

1.3 Objective

In the case of the existing system, the fraud is detected after the fraud is done that is, the fraud is detected after the complaint of the holder. And so, the cardholder has to face a lot of trouble before the investigation gets finished. And also, as all the transaction is maintained in a log, we need to maintain a huge data, and also nowadays a lot of online purchase is made so we don't know the person how is using the card online, we just capture the IP address for verification purpose. So there needs help from the cyber-crime to investigate the fraud. To avoid the entire above disadvantage, we propose the system to detect the fraud in the best easy way with the most accurate predictions.

1.3.1 Prior to Work

The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the knowledge of the ones that turned out to be a fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

In the latest news of India's biggest banking scam PNB scams in which fraudsters obtain $1.8 billion from overseas loans by making SWIFT defenceless to fraud. As per the Russian government, Hackers stole $6 million from one country bank using the SWIFT network. An unconstitutional user monitors the traffic of transactions and interrupts it and sends all the money to his account bypassing the forged message. Another type of fraud for which financial institutions have to compact with is Push Payment fraud, which through electronic mail fraudster acts as a genuine supplier and takes money from the certified user. This is a very crucial fraud in which ordinary users get affected accidentally.

This happens because 70% of the population fails to report about fraud. Due to this, fraudster gets motivated and does sharper practice. According to one business news 29% fraudster attempt is being done in India and 18% in another country. As per their survey, many fraudsters will deposit dummy currencies or demonetized currencies in the bank, due to which bank has to face financial troubles.

1.4 Scope

This system is capable of providing most of the essential features required to detect fraudulent and legitimate transactions. As technology changes, it becomes difficult to track the behaviour and pattern of fraudulent transactions.

Further enhancement can be done by making this system secure with the use of certificates for both merchant and customer and as technology changes new checks can be added to understand the pattern of fraudulent transactions and to alert the respective cardholders and bankers when fraudulent activity is identified.

1.5 Existing System

In the case of the existing system, the fraud is detected after the fraud is done that is, the fraud is detected after the complaint of the cardholder. And so, the cardholder faced a lot of trouble before the investigation finish. And also, as all the transaction is maintained in a log, we need to maintain huge data. And also, now a day's lot of online purchases are made so we don't know the person how is using the card online, we just capture the IP address for verification

purposes. So there needs help from the cyber-crime to investigate the fraud. To avoid the entire above disadvantage, we propose the system to detect the fraud in a best and easy way.

The Traditional detection method mainly depends on the database system and the education of customers, which usually are delayed, inaccurate, and not in-time. After that methods based on discriminate analysis and regression analysis are widely used which can detect fraud by credit rate for cardholders and credit card transactions. For a large amount of data, it is not efficient.

The Existing System is Clementine by Integral Solutions Ltd. features decision tree induction and neural networks, the K-Nearest Neighbour (KNN) method, genetic algorithms, and parallel technology, Intelligent Miner by IBM Corporation supports clustering with demographic and neural clustering. The success rates and performance of all these existing techniques are less. In this technique, the purchasing pattern is generated by a tool called a transaction generator. The input to the transaction generator is the category of purchase, information about the "typical" amount of money spent for this category, and information about the "normal" time passed since the last purchase of the same category has taken place. By using these input the fraud was detected. The future enhancement is we can adapt the system to parallel databases, and we can improve GUI to make the control of the system more intuitive. Card watch can be extended to general-purpose anomaly detection.

Companies have a detailed analysis of transactional and fraud data. Frauds tend to appear in patterns. In billions of credit card transactions, it is quite difficult to analyse each in isolation. Having predictive algorithms can help to detect fraudulent transactions. this is how data mining comes into play. Data consists of a combination of continuous data and nominal data. We can use a variety of statistical tests to prevent fraud events. Detecting credit card fraud is still not a perfect science. While fraud is still a major financial issue to banks, the distribution of fraud to non-fraudulent transactions is severely skewed towards non-fraudulent transactions. Out of an estimated, 12 billion transactions made annually 10 million are fraudulent (this shows every transaction in 1200 is fraudulent transaction). To analyse and predict fraud events we have used local outlier factor and isolation forest algorithms and thus calculated the number of fraud transactions. We have calculated the accuracy and number of errors of both the algorithms.

All the existing method to detect the credit card was on the mode like the detection occurs only after the complaint of the cardholder about fraud done. It is not a convenient way to avoid the loss happens to the cardholder. After getting the complaint they detected the fraud based on the IP address. For this, they need the help of the cyber-crime to detect the fraud and take action on it. It takes so much manpower.

The detection of fraud in credit card transactions is a major topic in financial research, of profound economic implications. While this has hitherto been tackled through data analysis techniques, the resemblances between this and other problems, like the design of recommendation systems and diagnostic/prognostic medical tools, suggest that a complex network approach may yield important benefits. In this paper, we present a first hybrid data mining/complex network classification algorithm, able to detect illegal instances in a real card transaction data set. It is based on a recently proposed network reconstruction algorithm that allows creating representations of the deviation of one instance from a reference group. We show how the inclusion of features extracted from the network data representation improves the score obtained by a standard, neural network-based classification algorithm and additionally how this combined approach can outperform a commercial fraud detection system in specific operation niches. Beyond these specific results, this contribution represents a new example of how complex networks and data mining can be integrated as complementary tools, with the former providing a view to data beyond the capabilities of the latter.

Disadvantages of existing credit card fraud detection models:

a) The main disadvantage of the existing system is the detection occurs only after gets a written complaint.
b) In the existing system, there is physical inconvenience exists.
c) The period that occurs to detect the fraud will cause so many losses to the cardholder.
d) Here is no particular security system in the existing so a hacker can easily access other cards.

Chapter 2

Proposed System

In the Proposed system, we use the Random Forest Algorithm and K-means and dataset. First, we will gather the Credit Card dataset, and analysis will be ready on the claimed dataset. After the examination of a dataset then cleaning the dataset is done. In most of the dataset, there will be quite one matching and null values will be there, so to delete all those similar and null values cleaning process is required. Then we've to divide the dataset into two categories because of the Trained dataset and Testing dataset for contrasting and analysing the dataset. After splitting the dataset we've to use the Random Forest Algorithm where this algorithm will give us better accuracy about the credit card fraud transactions. By applying the Random Forest Algorithm, the dataset will detect whether the transaction is fraudulent or not. On the idea of the above classification of information, performance is decided. In this analysis, the precision of credit card fraud transactions can be obtained.

Random Forest is also known as Random Decision Forest (RFA) which is used for Classification, Regression, and other tasks that are done by constructing more than one decision tree. This Random Forest Algorithm is established on supervised learning and therefore the great advantage of this algorithm is that it can be cast-off for both Classification and Regression. Random Forest Algorithm gives better precision when compared with all other existing systems and this is the most widely used algorithm. In this paper, the use of the Random Forest algorithm in credit card fraud detection can give you an accuracy of about 90 to 95%.

Let’s assume there is already some data in the file system. This could be the last one-month data or last six months data or it could be the last one-year data. And this file system could be HDFS or S3 or Azure block or any other file system. As shown in Figure 2.1, First, data will be imported from the file system to the Cassandra database. To import this data, a spark job is executed. This job will read this data. It will do some ETL transformations and it will finally save this data to the Cassandra database. Next, one more spark job is executed, this job is Spark ML Job. This job will read the data from Cassandra, train on this data and it will create a model.

Finally, this model will be saved to the file system. Again, this file system could be HDFS or S3 or Azure block or any other file system. After the Model is saved to the file system, Spark Streaming Job will start after that. This job will load the model and also it will start consuming credit card transactions from Kafka Server. After consuming the transaction, it will predict whether these transactions are fraud or not. And finally, it will save these predictions to the Cassandra database. And all these things are done in Real-time. Now, predictions are done in Cassandra, using Spring boot framework fraud transactions are displayed on the dashboard. These fraud transactions will be alerted on the dashboard in real-time. So, this is how a reference architecture of the proposed system of Real-time Credit card fraud detection project looks like. We are applying the random forest algorithm for classifying the credit card dataset. Random forest builds multiple decision trees and merges them to get a more accurate and stable prediction. Random Forest is an algorithm for classification and regression. It is a collection of decision tree classifiers. Random Forest adds additional randomness to the model. been transformed with PCA are time and amount. The Feature Time restrains the hours advance between the initial transaction and each transaction in the dataset. The feature amount is the transaction where the example-dependent cost-sensitive learning is used for this feature. Feature Class is the response variable and it takes value 1 in case of fraud and 0 otherwise. The different pre-processing technique is in machine learning, a standard scalar is used here. The Standard Scaler undertake your data is normally distributed within each feature and will scale them such that the distribution is now cantered around 0, with a standard deviation of 1. The mean and standard deviation are intended for the feature and those the feature is scaled based on equation (2.1):

$$xi - \frac{mean(x)}{st\ dev(x)} \tag{2.1}$$

At a high level, every Spark application consists of a driver program that runs the user's main function and executes various parallel operations on a cluster. The main abstraction Spark provides is a resilient distributed dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. RDDs are created by starting with a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it.

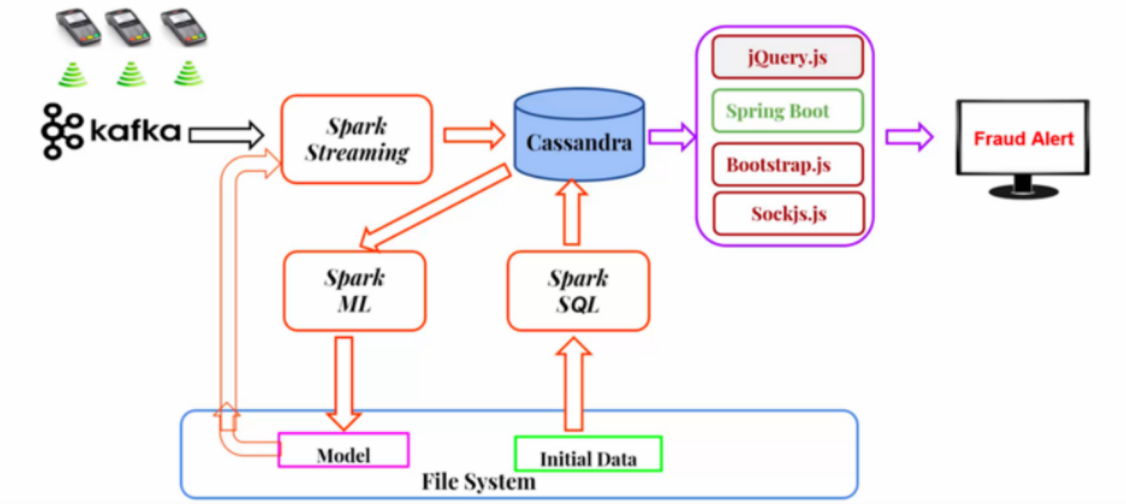


Figure 2.1: Architecture of Credit Card Fraud Detection System

The following tasks are executed to predict fraudulent transactions:

A. Spark Initial Import Job

At a high level, every Spark application consists of a driver program that runs the user's main function and executes various parallel operations on a cluster. The main abstraction Spark provides is a resilient distributed dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. RDDs are created by starting with a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it. Users may also ask Spark to persist an RDD in memory, allowing it to be reused efficiently across parallel operations. Finally, RDDs automatically recover from node failures.

The second abstraction in Spark is shared variables that can be used in parallel operations. By default, when Spark runs a function in parallel as a set of tasks on different nodes, it ships a copy of each variable used in the function to each task. Sometimes, a variable need to be shared across tasks, or between tasks and the driver program. Spark supports two types of shared variables: broadcast variables, which can be used to cache a value in memory on all nodes, and accumulators, which are variables that are only "added" to, such as counters and sums.

This guide shows each of these features in each of Spark's supported languages. It is easiest to follow along with if you launch Spark's interactive shell – either bin/spark-shell for the Scala shell or bin/pyspark for the Python one. The first thing a Spark program must do is to create a SparkContext object, which tells Spark how to access a cluster. To create a SparkContext you first need to build a SparkConf object that contains information about your application. Only

one SparkContext may be active per JVM. You must stop () the active SparkContext before creating a new one.

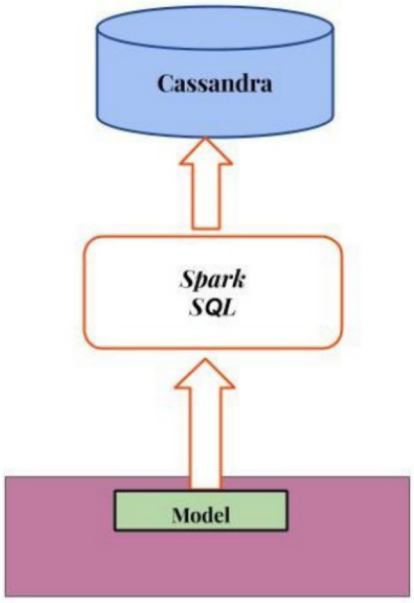


Figure 2.2: Spark Initial Import Job

The appName parameter is a name for your application to show on the cluster UI. master is a Spark, Mesos or YARN cluster URL, or a special "local" string to run in local mode. In practice, when running on a cluster, you will not want to hardcode master in the program, but rather launch the application with spark-submit and receive it there. However, for local testing and unit tests, you can pass "local" to run Spark in-process.

Existing data is available in the file system. As we run this Initial Import Spark job, it will read the data from the filesystem and it will do some ETL transformation. After doing the ETL transformation, it will save the data to the Cassandra database. This is how existing data is imported from the filesystem to the Cassandra database.

In the Spark shell, a special interpreter-aware SparkContext is already created for you, in the variable called sc. Making your SparkContext will not work. You can set which master the context connects to using the --master argument, and you can add JARs to the class path by passing a comma-separated list to the --jars argument. You can also add dependencies (e.g. Spark Packages) to your shell session by supplying a comma-separated list of maven coordinates to the --packages argument. Any additional repositories where dependencies might exist (e.g. Sonatype) can be passed to the --repositories argument park revolves around the concept of a resilient distributed dataset (RDD), which is a fault-tolerant collection of elements

that can be operated on in parallel. There are two ways to create RDDs: parallelizing an existing collection in your driver program or referencing a dataset in an external storage system, such as a shared filesystem, HDFS, HBase, or any data source offering a Hadoop Input Format.

We know that existing data is available in the file system. As we run this Initial Import Spark job, it will read the data from the filesystem and it will do some ETL transformation. After doing the ETL transformation, it will save the data to the Cassandra database. This is how existing data is imported from the filesystem to the Cassandra database.

Apache Spark's first abstraction was the RDD. It is an interface to a sequence of data objects that consist of one or more types that are located across a collection of machines (a cluster). RDDs can be created in a variety of ways and are the "lowest level" API available. While this is the original data structure for Apache Spark, you should focus on the Data Frame API, which is a superset of the RDD functionality. The RDD API is available in Java, Python, and Scala languages.

These are similar in concept to the Data Frame you may be familiar with in the pandas Python library and the R language. The Data Frame API is available in Java, Python, R, and Scala languages.

A combination of Data Frame and RDD. It provides the typed interface that is available in RDDs while providing the convenience of the Data Frame. The Dataset API is available in Java and Scala languages.

In many scenarios, especially with the performance optimizations embedded in Data Frames and Datasets, it will not be necessary to work with RDDs. But it is important to understand the RDD abstraction because:

a) The RDD is the underlying infrastructure that allows Spark to run so fast and provide data lineage.
b) If you are diving into more advanced components of Spark, it may be necessary to use RDDs.
c) The visualizations within the Spark UI reference RDDs.

B. Spark Machine Learning Job

Spark ML Job will read this data from the Cassandra database, train on this data and it will create a model. And finally, this model will be saved on the filesystem. From the inception of the Apache Spark project, MLlib was considered foundational for Spark's success. The key benefit of MLlib is that it allows data scientists to focus on their data problems and models instead of solving the complexities surrounding distributed data (such as infrastructure,

configurations, and so on). The data engineers can focus on distributed systems engineering using Spark's easy-to-use APIs, while the data scientists can leverage the scale and speed of Spark core. Just as important, Spark MLlib is a general-purpose library, providing algorithms for most use cases while at the same time allowing the community to build upon and extend it for specialized use cases.

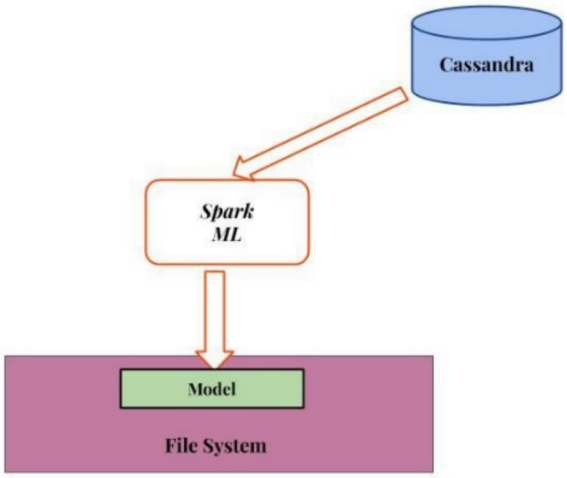


Figure 2.3: Spark Machine Learning Job

Machine Learning models can be applied to accomplish a variety of tasks from Classification, to Collaborative Filtering, Clustering, and Regression. We start with Linear Regression as this is one of the most common approaches for estimating unknown parameters after training on a known dataset. In this tutorial, we are training on a 2D dataset, so our Linear Regression model can be intuitively thought of as curve fitting. With more parameters, or features, we can make interesting predictions, for example, what should be a price listing range for a house with three bedrooms, two baths, 20 years old, and in a specific zip code area. Using Linear Regression for pricing houses given a set of input parameters works surprisingly well provided a large enough sample dataset. To cover edge cases, however, other Machine Learning methods might have to be used such as Random Forests or Gradient Boosted Trees, which we will cover in detail in future tutorials.

The advantages of MLlib's design include:

a) Simplicity: Simple APIs familiar to data scientists coming from tools like R and Python. Novices can run algorithms out of the box while experts can easily tune the system by adjusting important knobs and switches (parameters).

b) Scalability: Ability to run the same ML code on your laptop and a big cluster seamlessly without breaking down. This lets businesses use the same workflows as their user base and data sets grow.

c) Streamlined end-to-end: Developing machine learning models is a multistep journey from data ingest through trial and error to production. Building MLlib on top of Spark makes it possible to tackle these distinct needs with a single tool instead of many disjointed ones. The advantages are lower learning curves, less complex development and production environments, and ultimately shorter times to deliver high-performing models.

d) Compatibility: Data scientists often have workflows built up in common data science tools, such as R, Python pandas, and scikit-learn. Spark DataFrames and MLlib provide tooling that makes it easier to integrate these existing workflows with Spark. For example, Spark R allows users to call MLlib algorithms using familiar R syntax, and Databricks is writing Spark packages in Python to allow users to distribute parts of scikit-learn workflows.

Spark allows data scientists to solve multiple data problems in addition to their machine learning problems. The Spark ecosystem can also solve graph computations (via GraphX), streaming (real-time calculations), and real-time interactive query processing with Spark SQL and DataFrames. The ability to employ the same framework to solve many different problems and use cases allows data professionals to focus on solving their data problems instead of learning and maintaining a different tool for each scenario.

Several common business use cases are surrounding Spark MLlib. The examples include, but are not limited to, the following:

a) Marketing and advertising optimization:

   Operational optimization such as supply chain optimization and preventative maintenance

   Based on user site behaviour, what is the probability the user will click on the available ads?

b) Security monitoring/fraud detection, including risk assessment and network monitoring:

   Which users show anomalous behaviour, and which ones might be malicious?

c) Operational optimization such as supply chain optimization and preventative maintenance:

   Where in our system are failures likely to occur, requiring preventive checks?

Many compelling business scenarios and technical solutions are being solved today with Spark MLlib, including Huawei on Frequent Pattern Mining, OpenTable's Dining Recommendations, and Verizon's Spark MLlib's ALS-based Matrix Factorization. Some additional examples:

a) NBC Universal stores hundreds of terabytes of media for international cable TV. To save on costs, it takes the media offline when it is unlikely to be used soon. The company uses Spark MLlib Support Vector Machines to predict which files will not be used.
b) The Toyota Customer 360 Insights Platform and Social Media Intelligence Centre is powered by Spark MLlib. Toyota uses MLlib to categorize and prioritize social media interactions in real-time.
c) Radius Intelligence uses Spark MLlib to process billions of data points from customers and external data sources, including 25 million canonical businesses and hundreds of millions of business listings from various sources.
d) ING uses Spark in its data analytics pipeline for anomaly detection. The company's machine learning pipeline uses Spark decision tree ensembles and k-means clustering.

Spark is not only a faster and easier way to understand our data. More fundamentally, Spark changes the way we can do data engineering and data sciences, by allowing us to solve a diverse range of data problems - from machine learning to streaming, structured queries to graph computation - in our language of choice.

Spark MLlib allows novice data practitioners to easily work with their algorithms out of the box while experts can tune as desired. Data engineers can focus on distributed systems, and data scientists can focus on their machine learning algorithms and models. Spark enhances machine learning because data scientists can focus on the data problems they care about while transparently leveraging the speed, ease, and integration of Spark's unified platform.

C. Spark Streaming Initialization and Consumption Job

In the Spark Streaming Job, first, we are reading customer data from Cassandra. Using customer data, the age of the customer and the distance between merchant and customer place can be computed because age and distance will be used as features. After that, we are loading both Pre-processing and Random Forest model that was created by Spark ML Job. Both models will be used to transform and predict whether a transaction is a fraud transaction or not. After that, we will start consuming credit card transaction messages from Kafka and we'll predict whether a transaction is a fraud or not. Once the transactions are predicted, we will be saving these predictions to Cassandra. Fraud transactions will be saved to fraud table and non-fraud

transactions will be saved to the non-fraud table. Also, it will be saving offset to the Kafka offset table. The reason we are saving Kafka offset is to achieve exactly-once semantics.

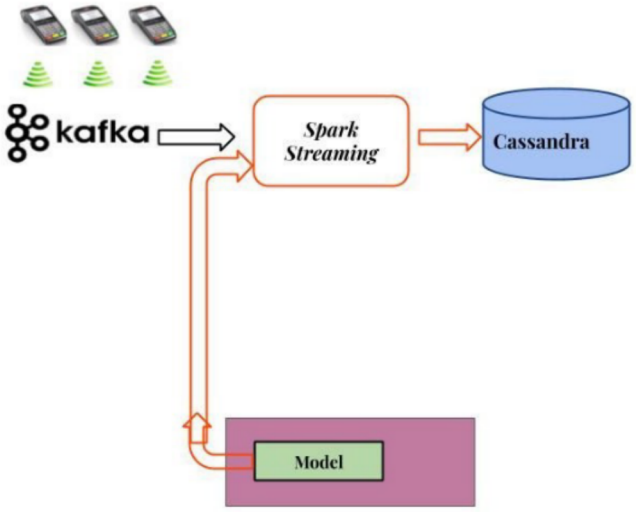


Figure 2.4: Spark Streaming Initialization and Consumption Job

D. Spark Streaming Processing and Prediction

The first value is extracted, partition number, and offset from the Dstream. The value field contains all the credit card transaction details. So, the result of Dstream contains 3 fields. transaction fields, partition number and offset. Next, we are processing these messages. And all the processing is done inside the foreachRdd() method. foreachRdd() method is used whenever we want to save a DStream to some external storage. It is a mutability API. The code inside foreachRdd() method is evaluated inside the Driver. It is not evaluated in the executor. First, we are converting the RDD inside the DStream to a data frame. And this data frame will have 3 columns. All the credit card transaction details in the value field are in JSON format. So we are parsing this JSON field using from_json() function. After parsing we are selecting all the credit card transaction fields as individual columns. Along with that, we are also selecting a partition column and offset column. Next, we are creating getDistance() user-defined function. As described earlier, this is used to compute the distance between merchant place and customer place Again, as described earlier in Spark ML Job, we are joining transaction data frame with customer data frame so that we can include age and distance in the result data frame. Because age and distance are used as features. Next, we are using the Pre-processing Model to transform the data frame.

A query on the input generates a result table. At every trigger interval (say, every 1 second), new rows are appended to the input table, which eventually updates the result table. Whenever

the result table is updated, the changed result rows are written to an external sink. The output is defined as what gets written to external storage. The output can be configured in different modes:

a) Complete Mode: The entire updated result table is written to external storage. It is up to the storage connector to decide how to handle the writing of the entire table.
b) Append Mode: Only new rows appended in the result table since the last trigger are written to external storage. This is applicable only for the queries where existing rows in the Result Table are not expected to change.
c) Update Mode: Only the rows that were updated in the result table since the last trigger are written to external storage. This is different from Complete Mode in that Update Mode outputs only the rows that have changed since the last trigger. If the query doesn't contain aggregations, it is equivalent to Append mode.

E. Spark Streaming Exactly One Semantics

Here we are using the foreachPartition() method on the prediction data frame to save the records to Cassandra. Unlike foreachRdd() method, foreachPartition() method is called inside the Spark Executor. Hence all the transactions from the data frame are saved from the Executor to Cassandra table using Cassandra connector object. Next, we are extracting database names and table names from the Broadcast object. Since we will be inserting the predictions from the Executor, we need to know the database name and table names in the executor. Hence, we had broadcasted database names and table names. Next, we are inserting predicted transactions and Kafka offset to Cassandra using Cassandra's session object. We are inserting all the tables in a single Cassandra session. First, we are creating a prepared statement for all the 3 tables. After that, we are inserting predicted fraud transactions to the fraud table, non-fraud transactions to the non-fraud table and at last, we are inserting offset to Kafka offset table. Before inserting offset, first, we are computing the max offset that we received for each partition in a batch. This is how we are saving credit card transactions and offset to Cassandra. Now that we have saved credit card transactions and offset to Cassandra, how are we achieving exactly-once semantics? The foreachRDD function will execute more than once if there's worker failure, thus writing the same data to external storage multiple times. There're two approaches to solve this issue, idempotent updates, and transactional updates. They are further discussed in the following sections.

F. Spark Streaming Graceful Shutdown

We are shutting down Spark Streaming Job by just creating a dummy file. A separate thread will be running in the Spark Streaming Job. This thread will be continuous checking every 1 second whether dummy file exists or not. If the dummy file exists we are stopping streaming context and we are also stopping Spark Session. So this how we are programmatically and gracefully shutting down Spark Streaming Job.

G. Kafka Producer

Kafka cluster broker is running on the localhost:9092 port. Next, we have a key serializer and value serializer. Here we are using String Serializer. It means the Producer will serialize string messages as bytes and send them to Kafka. Next, we have an ack=all. It means that Producer is saying to the broker that as soon as committed the message to the Leader copy and the Follower copies, send me an acknowledgement. This helps us to achieve higher durability of the messages. Next, we have retries = 3. It means that if the producer fails to send messages to the broker, it will try 3 times. This will also help us achieve higher durability. Our topic name is credicardTransaction. The transaction file contains randomly generated credit card transactions. Each transaction from this file is published as a message to Kafka. So, these are all the configuration parameters that are required for the Kafka Producer to send messages to Kafka.

H. Fraud Alert Dashboard

This is the class where we have written the main logic to select fraud transactions from the Cassandra table and display it on the dashboard Trigger method is called every 5 seconds. This method will, in turn, call a select query. That select query will select the latest fraud transactions that occurred in the last 5 seconds and display it on the dashboard, but whatever transactions were displayed in the last 5 seconds could also get displayed in the current 5 seconds. We don't want to alert two times for the same fraud transaction. To solve this problem, we are maintaining the max timestamp of previously displayed fraud transactions. And in the current trigger, we will only select those transactions whose timestamp is greater than the previous max timestamp. Once we have selected fraud transactions, we are finally sending these transactions to the dashboard.

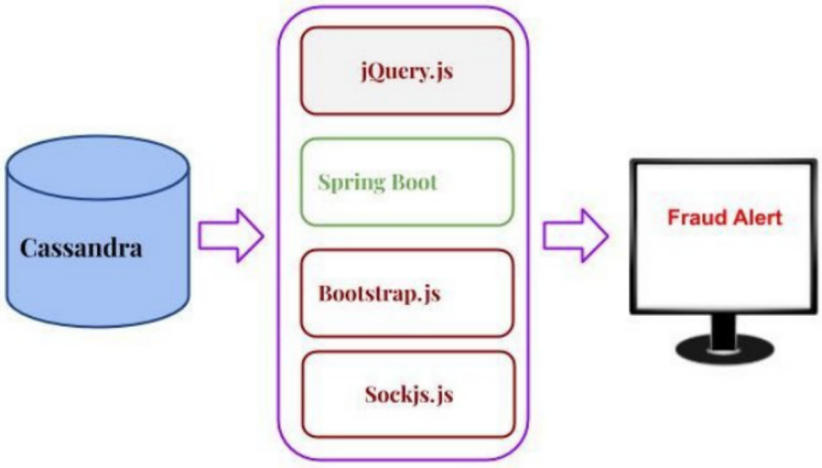


Figure 2.5: Fraud Alert Dashboard

Table 2.1 shows the customer table where various details about the customer are stored including name, address, location in terms of latitude and longitude so this will be used as one of the features, the Job title is also used as second feature and date of birth of customer is also used so that's age is calculated as the third feature for the creation of tree structure.

Field Name	Description
CC_NUM	Credit card holder's card number
FIRST	Credit card holder's first name
LAST	Credit card holder's last name
GENDER	Credit card holder's gender
STREET	Credit card holder's street
CITY	Credit card holder's city
STATE	Credit card holder's state
ZIP	Credit card holder's postal code
LAT	Credit card holder's latitude
LONG	Credit card holder's longitude
JOB	Credit card holder's job title
DOB	Credit card holder's date of birth

Table 2.1: Attributes of Customer Table

Chapter 3

Implementation Methods

3.1 Resources

In our proposed system we are using the HDFS file system but any other file system can be used like Amazon S3 or Azure Block. Along with that we are using Credit Card Database which is Stored in Cassandra for storing computed values and in this system, we are also using customer dataset and transaction dataset for predicting the transactions both the datasets are available freely on Kaggle website, these datasets are given as an input to Credit Card Fraud Detection System credit card fraud producer to create a pre-processing model and random forest model and to stream transactions as messages to fraud detection system in real-time. To design and develop this system we have also used VMware Fusion Professional version 11.5 to run Hadoop Ecosystem in Linux Virtual Environment and to develop this system we used To develop This entire project IntelliJ IDEA Community Edition developed by Apache Inc.

3.2 Algorithm

There could be millions of non-fraud transactions and only a few fraud transactions. If such unbalanced data is used for training then the algorithm will not create a good model. Hence, we have to balance our data. i.e. the no of non-fraud transactions must be almost equal to no of fraud transactions. So, we have to reduce the number of non-fraud transactions. Here we are using the K-means algorithm to reduce the number of non-fraud transactions. Now that we have reduced the number of non-fraud transactions, will combine both the data frame and form a single data frame. Now we have a balanced data frame. Random Forest algorithm is applied to a data frame. Random Forest algorithm will use a feature column for training and it will create a model. And finally, we will save the model to filesystem Let's see some more details on how we are doing data balancing.

Figure 3.1: Activity Diagram for Credit Card Fraud Detection System

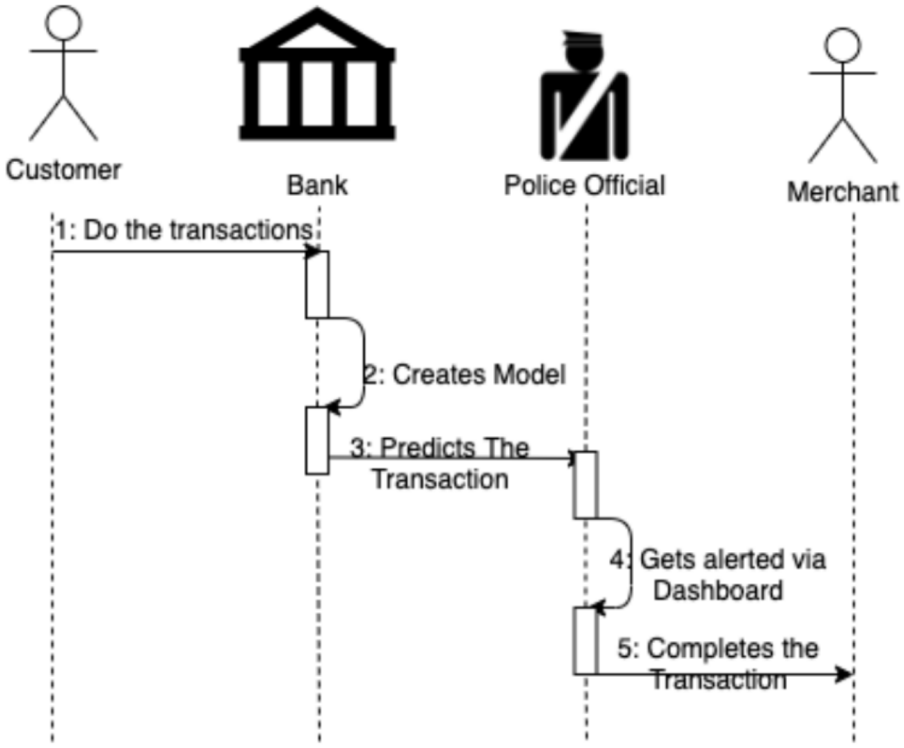


Figure 3.2: Sequence Diagram for Credit Card Fraud Detection System

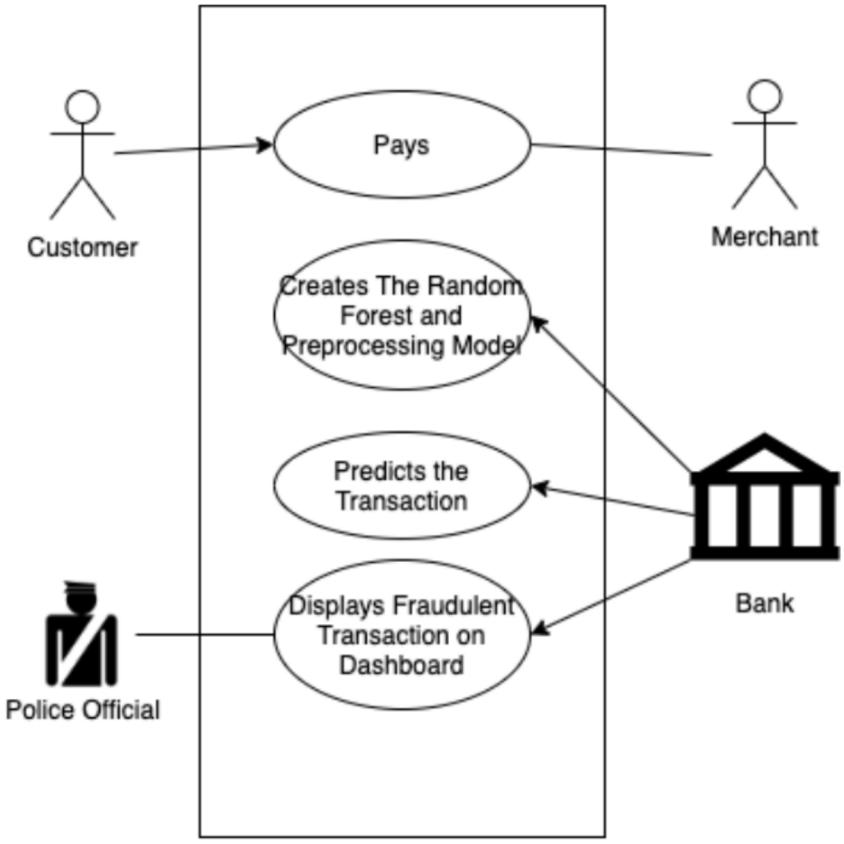


Figure: 3.3: Use Case Diagram for Diagram for Credit Card Fraud Detection System

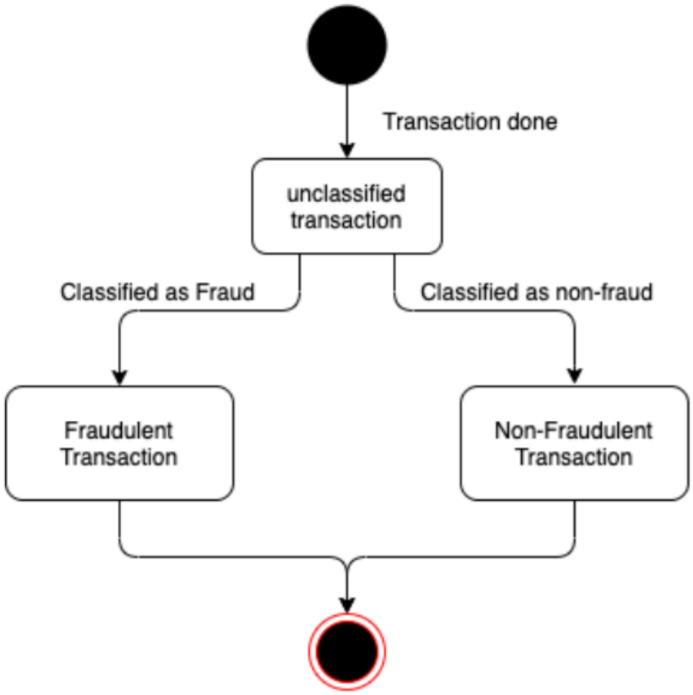


Figure 3.4: State Diagram for Credit Card Fraud Detection System

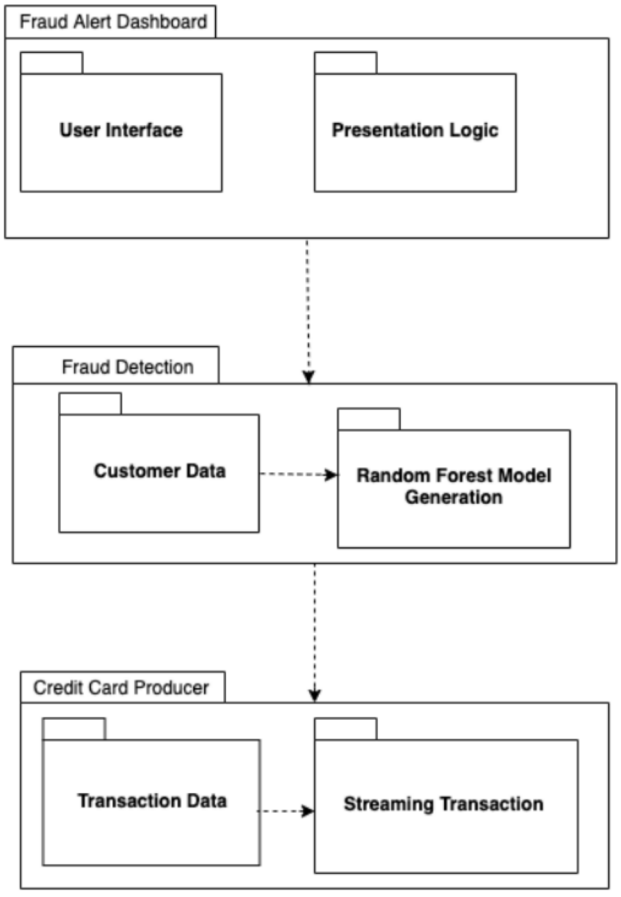


Figure 3.5: Package Diagram for Credit Card Fraud Detection System

Chapter 4

Development Tools

4.1 Hardware Requirement

Our system predicts transactions in real-time so it consumes lots of computational resources so here are hardware system requirements to run this model:

a) Processor: 2GHz and above
b) RAM: 8 GB and above
c) Storage: 40 GB and above

4.2 Software Requirement

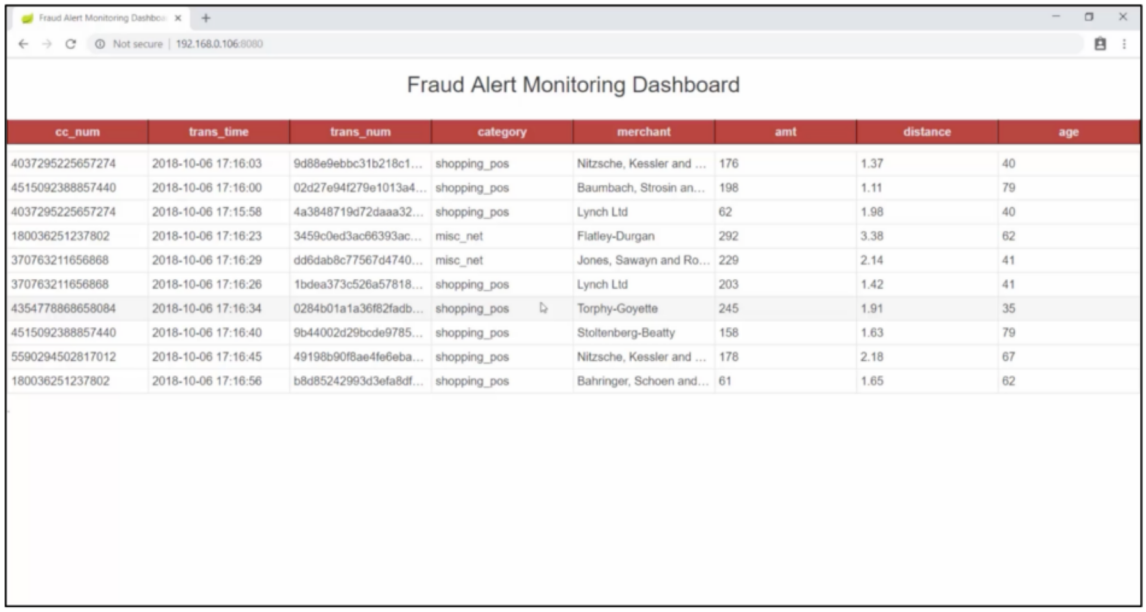
The main component of this project is Apache Spark and the version is 2.2.1 This project can also be implemented using 1.6 or 2.x. Hence this version is used. The next component used in this project is Confluent Kafka and Apache Kafka can also be used. There is no difference between Confluent Kafka and Apache Kafka. Only that Confluent Kafka has some other components bundled along with it. Next, we are using Apache Cassandra as our Storage Layer and the version is 3.11.1. Components used for visualization are Spring boot 1.5.12, jQuery 1.12.4. min, Latest bootstrap is used and SocketJS 1.1.1.min is used. So, these are the components used to implement our real-time credit card fraud detection project. To develop This entire project IntelliJ IDEA Community Edition is used which is developed by JetBrains under Apache License 2.0.

Chapter 5

Result and Discussion

Figure 5.1: Logs of Transaction Producer

Figure 5.1 shows the logs of transaction producer sending real-time transactions as a message to the fraud detection system which is acting as a false server in our proposed system.



Fraud Alert Monitoring Dashboard

cc_num	trans_time	trans_num	category	merchant	amt	distance	age
4037295225657274	2018-10-06 17:16:03	9d88e9ebbc31b218c1...	shopping_pos	Nitzsche, Kessler and ...	176	1.37	40
4515092388857440	2018-10-06 17:16:00	02d27e94f279e1013a4...	shopping_pos	Baumbach, Strosin an...	198	1.11	79
4037295225657274	2018-10-06 17:15:58	4a3848719d72daaa32...	shopping_pos	Lynch Ltd	62	1.98	40
180036251237802	2018-10-06 17:16:23	3459c0ed3ac66393ac...	misc_net	Flatley-Durgan	292	3.38	62
370763211656868	2018-10-06 17:16:29	dd6dab8c77567d4740...	misc_net	Jones, Sawayn and Ro...	229	2.14	41
370763211656868	2018-10-06 17:16:26	1bdea373c526a57818...	shopping_pos	Lynch Ltd	203	1.42	41
4354778868658084	2018-10-06 17:16:34	0284b01a1a36f82fadb...	shopping_pos	Torphy-Goyette	245	1.91	35
4515092388857440	2018-10-06 17:16:40	9b44002d29bcde9785...	shopping_pos	Stoltenberg-Beatty	158	1.63	79
5590294502817012	2018-10-06 17:16:45	49198b90f8ae4fe6eba...	shopping_pos	Nitzsche, Kessler and ...	178	2.18	67
180036251237802	2018-10-06 17:16:56	b8d85242993d3efa8df...	shopping_pos	Bahringer, Schoen and...	61	1.65	62

Figure 5.2: Fraud Alert Monitoring Dashboard

Figure 5.2 shows a fraud alert monitoring dashboard and it automatically gets updated whenever Spark Streaming Job predicts a fraud transaction it will be displayed on Fraud Alert Dashboard.

True Positive (TP)	False Positive (FP)
91	83
False Negative (FN)	True Negative (TN)
0	2548

Table 5.1: Confusion Matrix

True Positive (TP) = Model predicting fraud transaction as fraud = 91

False Positive (FP) = Model predicting genuine transactions as fraud = 83

False Negative (FN) = Model predicting fraud transactions as genuine = 0

True Negative (TN) = Model predicting genuine transaction as genuine = 2548

So, according to Table 5.1, we can justify that this model is far more accurate than SVM, and data balancing has helped a lot in gaining more accuracy which means that:

a) Out of all 91 fraud transactions, the model has correctly predicted all 91-fraud transaction as fraud.
b) Out of 2631 genuine transactions, the model has wrongly predicted 83 as fraud transactions and correctly predicted 2548 as genuine transactions.

These calculations show the efficiency and accuracy of our model:

True Positive Rate	False Positive Rate	Precision
$\frac{TP}{TP + FN}$	$\frac{FP}{FP + TN}$	$\frac{TP}{TP + FP}$
$\frac{91}{91 + 0} = 1$	$\frac{83}{83 + 2548} = 0.0315469$	$\frac{91}{91 + 83} = 0.5229885$
Out of all the fraud transactions, how much we predicted correctly, It should be high as possible.	Out of all the genuine transactions (not fraud), how much we predicted wrong (predicted as fraud). It should be as low as possible.	Out of all the fraudulent transactions that we have predicted correctly, how many are a fraud?

Table 5.2: Calculation of Basic Evaluation Measures

If there are more genuine transactions relative to fraud transactions then it is better to use Precision instead of False Positive Rate.

This seems to be a decent Model. Since our data is a cooked-up data, we will not do more evaluation. Usually, different Classification Model must be tried, and then the model efficiency must be computed using ROC (Receiver Operating Characteristics) cure and finally models from different Algorithm must be compared using AOC (Area Under Curve). Once you get a good model, it must be deployed.

Chapter 6

Plan of Work

A Gantt chart, or harmonogram, is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements constitute the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts can be used to show current schedule status using per cent-complete shadings and a vertical "TODAY" line as shown here. Gantt charts are sometimes equated with bar charts. Gantt charts are usually created initially using an early start time approach, where each task is scheduled to start immediately when its prerequisites are complete. This method maximizes the float time available for all tasks.

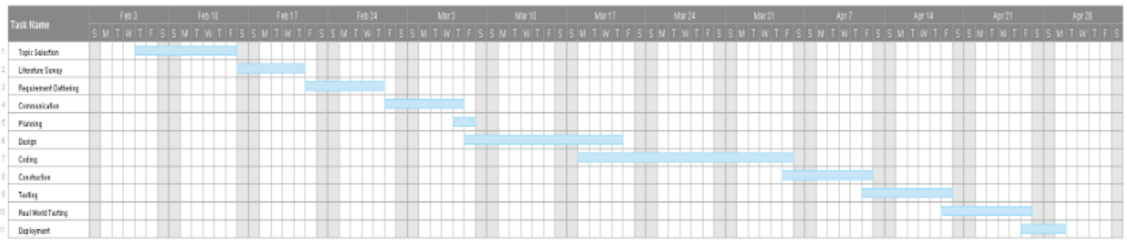


Figure 6.1: Plan of work of Proposed System

We have chosen the Gantt Chart to represent our plan of work. a chart in which a series of horizontal lines shows the amount of work done or production completed in certain periods with the amount planned for those periods.

Chapter 7

Conclusion

Credit card fraud detection has been a preceptive zone of research for the researchers for many years and will be a fascinating area of research in the future. This happens mainly due to the uninterrupted change of patterns in credit card frauds. during this time, we propose a unique credit-card fraud detection system by detecting fraudulent transactions using the best suitable algorithms and by pointing out the related problems recognized by previous researchers in credit card fraud detection. By focusing on real-time credit-card fraud detection by using predictive analytics methods and is notified over the GUI on the browser whenever the new fraudulent transaction is detected with all its transaction details. This part of our model can permit the fraud investigation team to form their decision to solve the issue and take subsequent steps as soon as a malicious transaction is detected. Therefore, we can conclude that there is a major impact of using a random forest algorithm for obtaining a comparatively higher performance from the classifier. This model has increased the predictions accurately after using data balancing by 9 per cent.

Further, the models indicated 74%, 83%, 72%, and 91% accuracy rates respectively. As the developed machine learning models present a mean level of accuracy, we hope to specialize in improving the prediction levels to acquire a better prediction. The Random forest algorithm combined with will perform better with a larger number of training data, but speed during testing and application will suffer. The application of more pre-processing techniques would also help. The other algorithm like SVM still suffers from the imbalanced dataset problem and requires more pre-processing to give better results at the results shown by SVM is great but it could have been better if more pre-processing has been done on the data moreover we have also implemented an algorithm which works in real-time i.e. Frauds will immediately be notified to the user.

We have proposed an application of the Random forest algorithm along with the K-means algorithm in real-time credit card fraud detection. The different steps in credit card transaction

processing are represented as the underlying stochastic process of the random forest algorithm. We have used the ranges of the transaction amount as the observation symbols, while the types of the item have been considered to be states of the random forest algorithm. We have suggested a method for finding the spending profile of cardholders as well as the application of this knowledge in deciding the value of observation symbols and an initial estimate of the model parameters. It has also been explained how the random forest algorithm can detect whether an incoming transaction is fraudulent or not.