DEVELOPMENT OF AN AUTOMATED REAL-TIME CREDIT CARD FRAUD DETECTION SYSTEM

ADEJUMO JOSEPH ADELEKE

17010301044

BEING A PROJECT SUBMITTED IN THE DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, COLLEGE OF BASIC AND APPLIED SCIENCES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE

MOUNTAIN TOP UNIVERSITY, IBAFO

OGUN STATE, NIGERIA

2022

CERTIFICATION

This project titled, **DEVELOPMENT OF A PREDICTIVE MODEL FOR DETECTING FRADULENT ACTIVITIES IN CREDIT CARD TRANSACTION USING MACHINE, LEARNING** in partial fulfilment of the requirement for the degree of **BACHELOR OF SCIENCE (Computer Science)** is hereby accepted.

----------(Signature and Date)

DR. Chinwe P. IGIRI

Supervisor

----------(Signature and Date)

Dr M. O. Odim

Acting Head of Department, Department of Computer Science and Mathematics

Accepted as partial fulfillment of the requirement for the degree of BACHELOR of SCIENCE (Computer Science)

DEDICATION

This endeavour is dedicated to God Almighty, who has been loyal and merciful in seeing me through to the end of this project. I also dedicate this work to my father, Mr. JAMES ADEDOTUN ADEJUMO and my mother, Mrs. FOLAKE ADEJUMO for being a major source of support in every way imaginable. I would also like to dedicate this project to my siblings ADEJUMO ADEBIMPE DEBORAH AND ADEJUMO JOHN ADEDEJI for their love and care for me.

ACKNOWLEDGEMENT

My sincere gratitude goes to the God who created all things and manifests himself in diverse ways than we can comprehend for his mercy, loving kindness and presence in the times I was in need. I also appreciate the entire staff and management of Mountain Top University for the immeasurable impact they have had on my life academically and spiritually from the person of the chancellor, Dr. D.K Olukoya down to every member of the senate and the university community. I specially recognize my supervisor, Dr. CHINWE PEACE IGIRI for her support and loving attitude. I also recognize all my lecturer who have made special contributions to the success of my academic pursuit.

Finally, I appreciate my siblings, friends and colleagues who made this journey a success. May the God of heaven water your lives.

Table of content

ABSTRACT

Assume you have a credit card in your possession. Your previous spending patterns will be discovered. For example, how much money you spend, where you spend it, how often you spend it, and what you buy. If your current credit card transaction deviates from your previous spending habits, it will be suspected of fraud; otherwise, it will be treated as a legitimate transaction and fraud transactions will be alerted in the dashboard. Millions of transactions will be used to make such predictions. Distributed frameworks that can scale as the number of transactions increases are therefore employed. Spark Kafka and Cassandra are used to create this system for real-time credit card fraud detection. Preprocessing is done using Spark Machine Learning Pipeline Stages such String Indexer, Vector Slicer, Standard Scaler, and Vector Assembler. Vector Slicer, Standard Scaler and Vector Assembler is used for Preprocessing. Utilizing the Random Forest Algorithm, a Machine Learning model is produced. K-means Algorithm is used for data balancing. Automation of both Spark Machine Learning and Spark Streaming with Kafka and Cassandra is done using Apache Airflow.

CHAPTER 1

1.1 INTRODUCTION

From the moment payment systems came to existence, there have always been people who will find new ways to access someone's finances illegally. This has become a major problem in the modern era, as all transactions can easily be completed online by only entering your credit card information. (Chuprina, 2021).

Fraud is as old as mankind itself and can take an unlimited variety of different forms (Lakshmi, Mettu ,& Hameed, 2020). As a result, fraud detection has become an important and urgent task for businesses. Necessary prevention measures can be taken to stop this abuse and the behavior of such fraudulent practices can be studied to minimize it and protect against similar occurrences in the future.

The solutions to the fraud can be categorized into prevention and detection. Fraud prevention involves preventing the fraud in the source itself. The technologies like the Address Verification System (AVS) and Card Verification System (CVM) are usually operated to prevent fraud. (Shakya, 2018)

When fraud cannot be prevented, it must be recognized as quickly as feasible and appropriate action taken. The action conducted following a fraudulent incident is known as fraud detection. It is the process of determining whether or not a transaction is valid. It entails tracking the behaviors of large groups of people in order to predict, detect, and avert unwanted conduct such as fraud, intrusion, and defaulting. These frauds are classified as:

- Credit Card Frauds: Online and Offline
- Card Theft
- Account Bankruptcy
- Device Intrusion
- Application Fraud
- Counterfeit Card
- Telecommunication Fraud

The current era's rapid technical growth has fueled a demand for more innovative payment options. Payment methods such as cheques and cash were previously used. Credit card usage has risen

dramatically over the world, and many people now believe in going cashless and rely solely on internet transactions. Thanks to credit cards, digital transactions have become easier and more accessible. A payment card, such as a credit card or debit card, is used to perpetrate fraud, and this practice is referred to as credit card fraud. The goal could be to pay into another account that is under criminal control or to receive goods or services. Credit card fraud can be authorized—where a legitimate customer uses their own credit card to make a payment to an account that is managed by a criminal—or unauthorized—where the account holder does not give consent for the payment to occur and a third party completes the transaction.

There are two kinds of card fraud:

- card-present fraud
- card-not-present fraud

The compromise can occur in several ways and can usually occur without the knowledge of the cardholder. A compromised account's credentials may be kept by a fraudster for months prior to any theft, making it challenging to pinpoint the point of penetration. Stolen cards can be reported swiftly by cardholders. Unauthorized use might not be discovered by the cardholder until they get a statement.

Until the cardholder contacts the issuing bank and the bank places a block on the account, the credit card can be used for unauthorized purchases. For early reporting, most banks offer free, 24-hour telephone numbers. Even so, before the card is cancelled, a thief may use the card to make illicit purchases.

There are various types of payment card fraud

- Application fraud: Application fraud takes place when a person uses stolen or fake documents to open an account in another person's name. To create a personal profile, criminals may steal or fabricate papers like utility bills and bank accounts. The fraudster could take money out of the account or apply for credit in the victim's name after opening the account with bogus or stolen documents.
- Account takeover: An account takeover refers to the act by which fraudsters will attempt to assume control of a customer's account (i.e. credit cards, email, banks, SIM card and more).
- Due to its varied characteristics, including class imbalance, this issue is particularly difficult to solve from the standpoint of learning. Fraudulent transactions are vastly outnumbered by legitimate ones, and transaction patterns frequently shift over time.

- Skimming: Skimming is the stealing of private data that was used in a typically routine transaction. Employing simple techniques like duplicating receipts or more sophisticated ones like using a small electrical gadget (skimmer) to swipe and save several victims' card numbers, the thief can obtain the victim's card number.
- Unexpected repeat billing: Repeat billing, also referred to as "repeated bank charges," is a result of online bill payment or online transactions made with a bank account. These are banker's orders or standing orders from customers to honor and pay the payee a specific sum each month.

In Big computing communities such as machine learning and data science the solution to this problem can be automated. Due to its varied characteristics, including class imbalance, this issue is particularly difficult to solve from the standpoint of learning. Fraudulent transactions are vastly outnumbered by legitimate ones, and transaction patterns frequently shift over time. stical properties over the course of time, and considering the huge traffic of transaction data, and it is not possible for humans to check manually every transaction one by one if it is fraudulent or not.

Algorithms for machine learning are used to analyze all permitted transactions and flag any that seem suspect. Professionals look into these reports and get in touch with the cardholders to confirm whether the transaction was legitimate or fraudulent.

1.2 BACKGROUND OF STUDY

With the growth of online business around Nigeria, the number of credit card frauds has also increased drastically. The fraudulent credit card transactions result in yearly losses of a significant sum of money. Unauthorized financial fraud losses using credit cards and online banking in the UK reached £844.8 million in 2018. While in 2018, banks and credit card companies stopped £1.66 billion in unlawful fraud. This translates to the prevention of £2 out of every £3 of attempted fraud. In 2015, fraud losses on credit, debit, and prepaid cards issued worldwide reached $21.84 billion, according to a Bloomberg report. By 2020, Bloomberg predicts that this could grow by a rate of 45 percent. (Jesus, 2019)

Interestingly credit card fraud affects card owners the least because their liability is limited to the transactions made. The existing legislations and cardholder protection policies as well as insurance schemes in most countries protect the interests of the cardholders. However, the most affected are the merchants, who, in most situations, do not have any evidence (e.g. digital signature) to dispute the

cardholders' claim of misused card information. Merchants end up bearing all the loses due to chargeback, shipping cost of goods, card issuer fees and charges as well as their own administrative costs. Numerous fraudulent incidents involving the same business can scare away customers, force banks that provide credit cards to stop accepting payments, and harm the business' brand and goodwill. (Amanze & Onukwugha, 2018)

Some famous credit card fraud attacks:

Between July 2005 and mid-January 2007, a breach of systems at TJX Companies exposed data from more than 45.6 million credit cards. Albert Gonzalez is accused of being the ringleader of the group responsible for the thefts. In August 2009 Gonzalez was also indicted for the biggest known credit card theft to date information from more than 130 million credit and debit cards was stolen at Heartland Payment Systems, retailers 7-Eleven and Hannaford Brothers, and two unidentified companies.

Approximately 40 million sets of credit card data were stolen from Adobe Systems in 2012 as a result of a cyberattack. According to Chief Security Officer Brad Arkin, the data exposed included client names, encrypted credit card numbers, expiration dates, and details on orders.

In July 2013, press reports indicated four Russians and a Ukrainian were indicted in the U.S. state of New Jersey for what was called "the largest hacking and data breach scheme ever prosecuted in the United States." Albert Gonzalez was named as a co-conspirator in the attack, which resulted in more than $300 million in losses and at least 160 million credit card losses. American and European businesses, such as Citigroup, Nasdaq OMX Group, and PNC Financial, were impacted by the attack financial Services Group, Visa licensee Visa Jordan, Carrefour, J. C. Penny and JetBlue Airways.

A Target Corporation system breach occurred between November 27 and December 15, 2013, exposing information from roughly 40 million payment cards. Names, account numbers, expiration dates, and card security codes were among the data taken.

A hacking attack that occurred between July 16 and October 30, 2013, exposed about a million sets of credit card data kept on Neiman-Marcus computers. Target's systems were compromised by malware that was intended to hook into cash registers and monitor the credit card authorisation process (RAM-scraping malware), exposing data from as many as 110 million consumers.

The Home Depot acknowledged that their payment systems had been breached on September 8th, 2014. Later, they issued a statement claiming that the incident led to the theft of 56 million credit card details by hackers.

In a planned theft on May 15, 2016, a gang of about 100 people stole $12,7 million from 1400 convenience stores in Tokyo over the course of three hours using the information from 1600 South African credit cards. They are thought to have gained enough time to escape Japan before the robbery was uncovered by operating on a Sunday and in a nation other than the bank that issued the cards.

1.3 STATEMENT OF PROBLEM

The ease of payment credit cards has brought to the marketing world is immeasurable. Transactions can now be made without any hassle. The world at large is taking advantage of this technology and now, almost all transaction are made with credit cards. This plight has brought about a need for more a secure way to handle these transactions. Fraudsters have been taking advantage of the technology to rob people of their money and they need to be stopped.

When a credit card is copied or stolen, the transactions made by them are labeled as fraudulent. These fraudulent transactions should be prevented or detected in a timely manner otherwise the resulting losses can be huge.

There has been a growing amount of financial losses due to credit card frauds as the usage of the credit cards become more and more common. As a result, numerous articles [2, 20] detailed significant losses in various nations. 64 I. Elikucuk and E. Duman There are numerous ways to commit credit card fraud, including straightforward theft, application fraud, using fake cards, never receiving issue (NRI), and online fraud (where the card holder is not present). In online fraud, the transaction is made remotely and only the card's details are needed. A manual signature, a PIN or a card imprint are not required at the time of purchase. Though prevention mechanisms like CHIP&PIN decrease the fraudulent activities through simple theft, counterfeit cards and NRI; online frauds (internet and mail order frauds) are still increasing in both amount and number of transactions. Online scams accounted for around 50% of all credit card fraud losses in 2008, according to Visa reports about European countries. When the fraudsters obtain a card, they usually use (spend) all of its available (unused) limit. Statistics show that people accomplish this in four to five transactions on average [18]. Thus, although the standard predictive modeling performance metrics are extremely essential for the fraud detection problem, as specified by bank authorities, a performance criterion, assessing the loss that can be avoided on the cards whose operations are identified as fraudulent, is more prominent. In other words, identifying fraud on a bigger available limit card is more beneficial than detecting fraud on a smaller available limit card.

1.4 AIMS AND OBJECTIVES

The aim of this project is to develop and automate a real-time credit card fraud detection system so as to prevent the continuity of credit card fraud by detecting fraudulent online transactions

The specific objectives are:

1. to create a credit card fraud detection system
2. to enable the credit card fraud detection system to detect fraud in real-time transaction
3. to automate both Machine Learning and Real-time Streaming Job using Apache Airflow
4. to create a fraud alert dashboard and display all the predicted fraudulent transactions

1.5 SIGNIFICANCE OF THE STUDY

The process of searching for fraud is lengthy due to the amount of data involved. The credit card dataset is classified using the random forest method in the suggested system. An approach for classification and regression is called Random Forest. In a nutshell, it is a group of decision tree classifiers. Random forest has advantage over decision tree as it corrects the habit of overfitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others. The Random Forest algorithm has been found to provide a good estimate of the generalization error and to be resistant to overfitting. (Lakshmi, Mettu ,& Hameed, 2020)

1.6 LIMITATIONS OF STUDY

Several challenges are associated with credit card fraud detection, some of these challenges are: determining which learning strategy to use (e.g., supervised learning or unsupervised learning), which algorithms to use (e.g., Logistic regression, decision trees, etc.), which features to use, how to deal with the class imbalance problem (fraudulent cases are extremely sparse as compared to the legitimate cases). (Shakya, 2018) The profile of fraudulent conduct is dynamic, meaning that fraudulent transactions frequently resemble genuine ones; credit card transaction databases are infrequent and severely biased; selection of features (variables) for the models that is optimal; appropriate metric to assess the effectiveness of strategies on distorted credit card fraud data.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter provides information on numerous studies carried out by top experts and clarifies technical terms related to creating a model for identifying fraudulent behaviour in credit card transactions. It will focus on supervised learning, and the model will be trained with previously labeled data. The input is a record of the credit card transaction, and the output is whether it is likely to be fraudulent or not.

2.2 RELATED WORKS

Over the years, credit card fraud detection has drawn a lot of by researchers' interest and several techniques have been suggested.

(Aman, 2021)Aman, Arpit Mishra, Ashish Kumar and Naveen Pandey from Inderprastha Engineering College in their project "Credit Card Fraud Detection using Machine Learning and Data Science" illustrated the modelling of a data set using machine learning. Their objective was to detect 100% of fraudulent transaction while minimizing the incorrect fraud classification. They achieved this by concentrating on the analysis and pre-processing of data sets, as well as the application of numerous anomaly detection methods, including Local Outlier Factor and Isolation Forest algorithm on the PCA converted Credit Card Transaction data. In their results, the code prints out the number of false positives transactions it detected and compares it with the actual values. This is done to determine how accurate and precise the algorithms are. While the algorithm does reach over 99.6% accuracy, its precision remains only at 28% when a tenth of the data set is taken into consideration. However, when the entire dataset is fed into the algorithm, the precision rises to 33%. The large percentage of correctness was anticipated due to the large disparity between the number of legitimate and authentic transactions. As a result, they encountered an issue with the dataset's imbalance.

(Vaishnavi Nath Dornadulaa, 2019) Vaishnavi Nath Dornadula and Geetha from the Vellore Institute of Technology in India created a novel fraud detection method for Streaming Transaction Data in their study on Credit Card Fraud Detection Using Machine Learning Algorithms with the aim of examining past transaction details of the customers and extracting behavioral patterns. Based on the value of their transactions, cardholders were divided into various groups. Then, they combined the transactions performed by cards from various groups in order to extract the appropriate behavioral

patterns for each group using the sliding window technique. The groups are then used to train various classifiers individually in the future. The classifier with the highest rating score was then picked as one of the most effective ways to detect fraud. They observed that the Matthews Correlation Coefficient was the better parameter to deal with imbalance dataset. Although it was not the only solution. By applying the SMOTE, they tried balancing the dataset and found that the classifiers were performing better than before. They also stated another way of handling imbalance dataset which is to use one-class classifiers like one-class SVM. Finally, they stated that Logistic regression, decision tree and random forest are the algorithms that gave best results.

In July 2018, Amanze, B.C and Onukwugha, C. G (Amanze & Onukwugha, 2018) stated that every cardholder has a certain shopping behavior, which establishes an activity profile for him. They discussed the ineffectiveness of existing fraud detection systems which try to capture behavioral patterns as static rules, when cardholder develops new patterns these rules become ineffective. Utilizing adaptive data mining and intelligent agents, they created a system for Nigerian financial industry that could incorporate evidence from both past and present behavior to assess the level of suspicion associated with each incoming transaction. The project focused on the predictive model using data mining that scores each transaction with high or low risk of fraud and those with high risk generate alerts. Predictive data mining perform interference on the current data to make predictions. The intelligent agents check those alerts and provide a feedback for each alert i.e. true positive (genuine) or false positive (fraud).

(Vaishnave Jonnalagadda, 2019)VaishnaveJonnalagadda, Priya Gupta and Eesita Sen from the SRM Institute of Science and Technology, Chennai, Tamil Nadu, worked on a project that focused on credit card fraud detection in real-world scenarios. In this project they designed a model to detect the fraud activity in credit card transactions. Initially, they collected the credit card usage data-set by users and classified it as trained and testing dataset using a random forest algorithm and decision trees. They examined the broader data set and user-provided current data set using this workable technique. The accuracy of the outcome data was then improved. Applied processing to some of the provided properties that may have an impact on fraud detection when viewing the graphical data visualization model. Accuracy, sensitivity, and specificity were used to evaluate how well the procedures performed. They obtained the outcome of an accurate credit card fraud detection value. i.e. 0.9994802867383512 (99.93%) using a random forest algorithm with new enhancements.The Random forest algorithm will provide better performance with many training data, but speed during testing and application will still suffer. Usage of more pre-processing techniques would also assist.

They proposed that their future work will try to represent this into a software application and provide a solution for credit card fraud.

2.3 CONCEPTUAL REVIEW

In the early days of 'intelligent' applications, many systems used hand coded rules of 'if" and 'else' decisions to process data or adjust to user input.Manually crafting decision rules is feasible for some applications, particularly those in which humans have a good understanding of the process to model. Using machine learning, however, simply presenting a program with a large collection of data is enough for an algorithm to determine what characteristics are needed to identify the solution to a problem. (Sarah & Muller, 2017)

2.4 WHAT IS MACHINE LEARNING?

Machine learning is about extracting knowledge from data. It is a research field at the intersection of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning. it has had a tremendous influence on the way data-driven research is done today (Sarah & Muller, 2017)

Machine learning is also defined as a field in artificial intelligence that provides the system the capability to learn from experience automatically without human intervention and aims to predict the future outcomes as accurate as possible utilizing various algorithmic models. (Shakya, 2018)

2.4.1 Types of machine learning algorithms

- Supervised learning algorithm
- Unsupervised learning algorithm
- Semi-supervised learning algorithm
- Reinforcement learning algorithm

However, the most commonly used ones are supervised and unsupervised learning

2.5 WHAT IS SUPERVISED LEARNING

Supervised learning algorithms are Machine learning algorithms that learn from input/output pairs. The algorithm receives the required solutions, or labels, as part of the training set. When an input/output pair exists and we want to forecast a specific outcome from that input, we use it. We use these input/output pairs as the training set for our machine learning model. Our objective is to accurately predict fresh, unforeseen facts. Building the training set is frequently labor-intensive, but

once it is done, supervised learning automates and frequently accelerates a time-consuming or impossible operation. (Sarah & Muller, 2017)

Mathematically, supervised learning can be represented as Y = f(x). Where x represents the input variables, Y denotes an output variable and f(X) is a mapping function. The goal is to approximate mapping function such that when an unseen input is given to the mapping function, it can predict the output variable (Y) correctly. (Shakya, 2018)

Supervised learning is commonly used in real world applications, such as face and speech recognition, products or movie recommendations, and sales forecasting. Supervised learning can be further classified into two types: Regression and Classification. The objective of a classification task is to forecast a class label, which is a choice from a preset list of alternatives, where the output variable is a category (for example, fraud or real, rainy or sunny, etc.). The objective of a regression problem is to forecast a continuous number, or in programming terms, a floating-point number, where the output variable is a real value (e.g., the price of a house, temperature, etc). (or real number in mathematical terms). The classification issue at the center of this endeavor.

2.5.1 Classification

A machine learning technique called classification uses existing categories to determine how new data should be categorized into them.

A predicted class, which takes the form of a discrete category, is produced by classification models. A discrete category prediction is necessary in order to reach a conclusion for the majority of real-world applications. (ModelingAppliedPredictive, 2013)

In classification tasks, the program must learn to predict discrete values for the dependent or output variables from one or more independent or input variables. That is, the program must predict the most probable class, category or label for new observations.

For example, fraud detection can be identified as a classification problem. In this case, the goal is to predict if a given transaction is fraud or genuine.

Generally, there are three types of classification: binary classification, where there are two output labels (e.g., classifying a transaction which may be fraud or genuine), multi-class classification, where there are more than two output labels (e.g., classifying a set of images of flowers which may be Rose or Lilly or Sunflower) and multi-label classification, where the data samples are not mutually exclusive and each data samples are assigned a set of target labels (e.g., classifying a crab on the basis of the sex and color in which the output labels can be male/female and red/black).

(Shakya, 2018) This project deals with the binary classification problem where the output label is either normal or fraud.

2.5.2 Binary Classification

We can think of binary classification as trying to answer a yes/no question. In binary classification we often speak of one class being the positive class and the other class being the negative class. Here, positive doesn't represent having benefit or value, but rather what the object of the study is. Classifying emails as either spam or not spam is an example of a binary classification problem. In this binary classification task, the yes/no question being asked would be "Is this email spam?"

When working on classification predictive modeling problems, we must collect a training dataset.

2.5.3 Training Dataset

A training dataset is several examples from the domain that include both the input data (e.g. measurements) and the output data (e.g. class label).

Depending on the complexity of the problem and the types of models we may choose to use, we may need tens, hundreds, thousands, or even millions of examples from the domain to constitute a training dataset. In order to appropriately prepare the input data for modeling, the training dataset is used to better comprehend it. Additionally, it is used to assess a variety of modeling approaches. It is used to fine-tune a model's hyperparameters. The last step involves using the training dataset to create a model that can be applied to all data and used to predict future examples from the issue area. (Brownlee, 2019)

2.5.4 Imbalanced Classification Problems

The distribution of examples into classes is referred to as the class distribution. An imbalanced classification problem is one where there are not an equal number of examples in the training dataset for each class label. That is, when the distribution of students by class is biased or skewed rather than equal or nearly equal. (Brownlee, 2019)

The distribution of classes in a given training dataset identifies a problem's imbalance. The definition of class imbalance must consider a dataset or distribution. Class imbalance is often measured in relation to the training distribution since class labels are necessary to establish the level of class imbalance (Imbalanced Learning: Foundations, 2013). The class distribution in fraud detection is already unbalanced

2.5.5 Combating Imbalanced Training Data

Most machine learning algorithms work best when the number of samples in each class are about equal. This is so because algorithms are created to minimize error and increase accuracy. This section will cover potential solutions to the categorization imbalance issue that plagues the detection of credit card fraudThis is so because algorithms are created to minimize error and increase accuracy. This section will cover potential solutions to the categorization imbalance issue that plagues the detection of credit card fraud.

2.5.6 Resampling approach

To address the problem of imbalanced learning, many resampling techniques have been created. Resampling techniques include

- oversampling
- undersampling
- combining oversampling and under sampling techniques
- ensembling sampling.

Oversampling and undersampling both try to change the proportions between the majority and minority classes. Using oversampling and undersampling approaches together results in a more balanced new dataset. Resampling allows different classes to have about the same influence on the classification model's outputs by making the training data more balanced. (Bagui &Kunqi , 2021)

2.6 SELECTED MODEL

In this section, we will discuss the different models selected for the predictive analysis. Depending on the nature of the classification problem, we chose two very popular predictive model.

2.6.1 Random forest

Random forest is an ensemble learning algorithm, which can be used for both regression and classification task. In a random forest, the weak learners are decision trees. So, before going into the details of the random forest, we will try to understand the basics of decision trees. An approach for supervised learning that may be applied to both regression and classification is the decision tree. But classification issues are where it's most frequently used. It is made up of a number of internal nodes, where each node stands for a test of a characteristic (such as whether it will be sunny, cloudy, or rainy tomorrow). The tree's leaf nodes reflect the end result, while each branch shows the results of

the test (class label). It involves breaking down a training set into several subsamples. A decision tree involves a series of if-then conditions that are used to make the final decision.

The splitting of the total training data into subsets, which is done in each internal node according to some condition, is necessary for the construction of a decision tree. The Gini impurity and information gain metrics, among others, are used by the decision tree method to find the appropriate split of each node. Gini impurity is an indicator of how frequently a randomly selected element from the set would be mislabeled if it were randomly classified in accordance with the distribution of labels in the subset. At each stage of the tree-building process, the decision on which feature to split on is made using information gained. The process of splitting continues until the internal node has the class label value. Although decision trees are easy to understand and perform well in some datasets, they tend to have a high variance because of the greedy approach of the algorithm where the tree tends to always select the best split at each level and it cannot see far behind the current level. Due to this reason, there may be the possibility of overfitting, where the model only performs better in the training set and fails to perform well in test sets.

Random forest algorithm mitigates the overfitting problem well by using the bootstrap concept. In simple language, the random forest builds multiple decision trees and combines them to improve the performance of the model. Bootstrapping is the process of sampling the training data randomly with replacement.

Random forest utilizes bootstrapping such that each decision tree will be trained with different

subsamples of data.

Moreover, the random forest uses random subsets of features. For example, if there are 50 features in the data, random forest will only choose a certain number of them, let's say 10, to train on each tree. Thus, each tree will have 10 random features that will be used for training including finding the best split of each node of the tree. Once we have the collection of decision trees, the results of each tree will be aggregated to get the final result (vote). The model trained in such a way will ensure generalization since not one, but multiple decision trees are used for making the decision, and moreover, each tree is trained with different subsections of data.

2.7 EVALUATION METRICS

Often when predicting a model, we might think that the one with the highest accuracy should be our ideal selection. However, sometimes it is desirable to select a model with low accuracy since it

provides us with greater prediction power on the problem. This is known as the Accuracy Paradox. (Lamba, 2020)

In machine learning, we train the model with the training data, and then we check the generalization capability of the model. In simple terms, we examine how the model performs when tested on data that was unseen. So how do we measure the performance of the model? We use evaluation metrics for evaluating the performance of the model depending on the nature of the problem (whether it is a regression or classification). In this section, we will only discuss the evaluation metrics related to the classification problem. (Shakya, 2018)

2.7.1 Confusion matrix

It is the most commonly used evaluation metrics in predictive analysis mainly because it is very easy to understand, and it can be used to compute other essential metrics such as accuracy, recall, precision, etc. It is an NxN matrix that describes the overall performance of a model when used on some dataset, where N is the number of class labels in the classification problem.

Statistics like True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), which are computed using the combination of actual and expected values, make up a confusion matrix.

- True Positive (TP) refers to a situation when both the actual and expected values are positive (e.g., fraud).
- When the projected value is positive but the actual value is negative (such as normal), this is known as a false positive (FP).
- True Negative (TN) is a situation when both the predicted and actual values are negative (e.g., normal).
- False Negative (FN) cases occur when the anticipated value is negative but the actual value was positive (for example, fraud).

2.7.2 Recall

Recall and sensitivity are synonyms. It is the proportion of observations in the actual class that were correctly predicted as positive to all other observations. It is the ratio of actual positive instances to true positives. Recall can be defined as the proportion of all true positive cases that had true positives discovered and recalled. Recall $= \frac{TP}{\mathrm{T\,P + F\,N}}$

2.7.3 Precision

It is the proportion of real positives to both real and fake positives. It is the proportion of all positively expected observations to those that were successfully predicted. Simply said, precision measures the proportion of found cases that were true positives.

$$\text{Precision} = \frac{TP}{\text{T P} + \text{F P}}$$

2.7.4 F1 Score

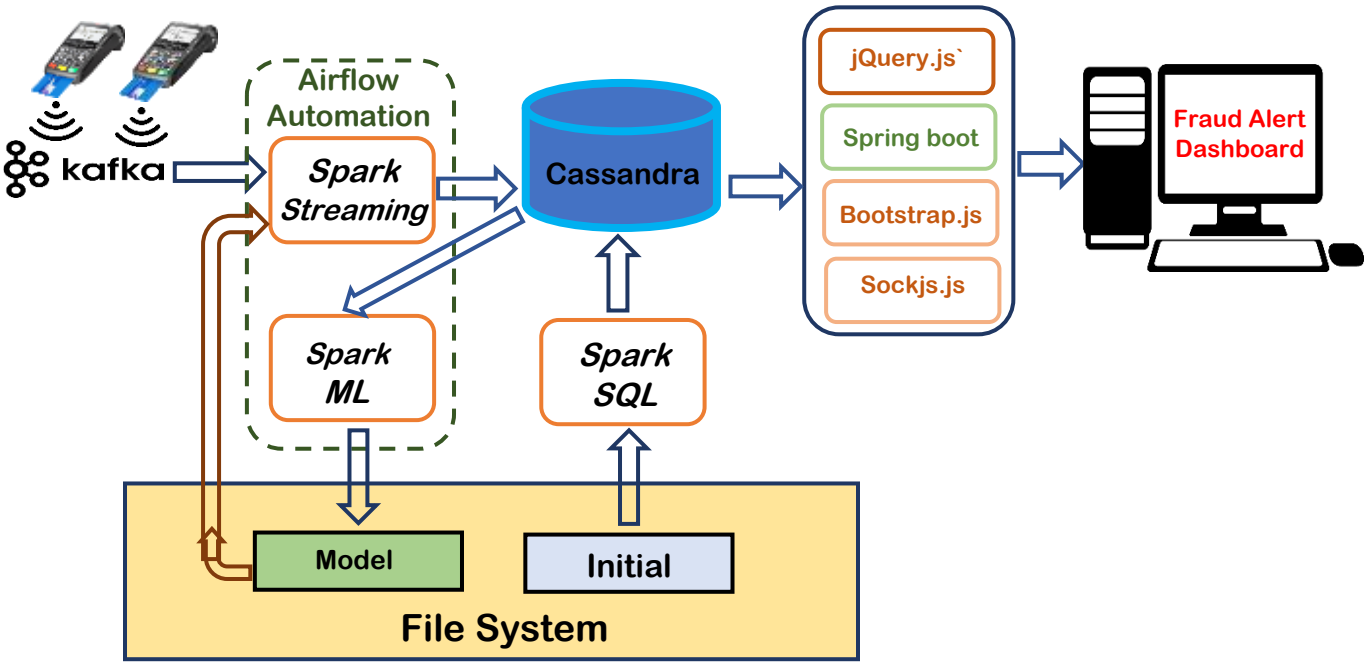
The harmonic mean of recall and precision is known as the F1 Score, sometimes spelled F Score or F-measure. It represents the weighted average of recall and precision. Its value is between 0 and 1, with 1 being the best and 0 being the worst. The calculation is as follows.

$$\text{F1} = \frac{2*(\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}}$$

CHAPTER 3

IMPLEMANTATION AND SYSTEM DESIGN

A real-time system for detecting credit card fraud will be created and automated as part of this project. Spark, Kafka, and Cassandra are used in its implementation.



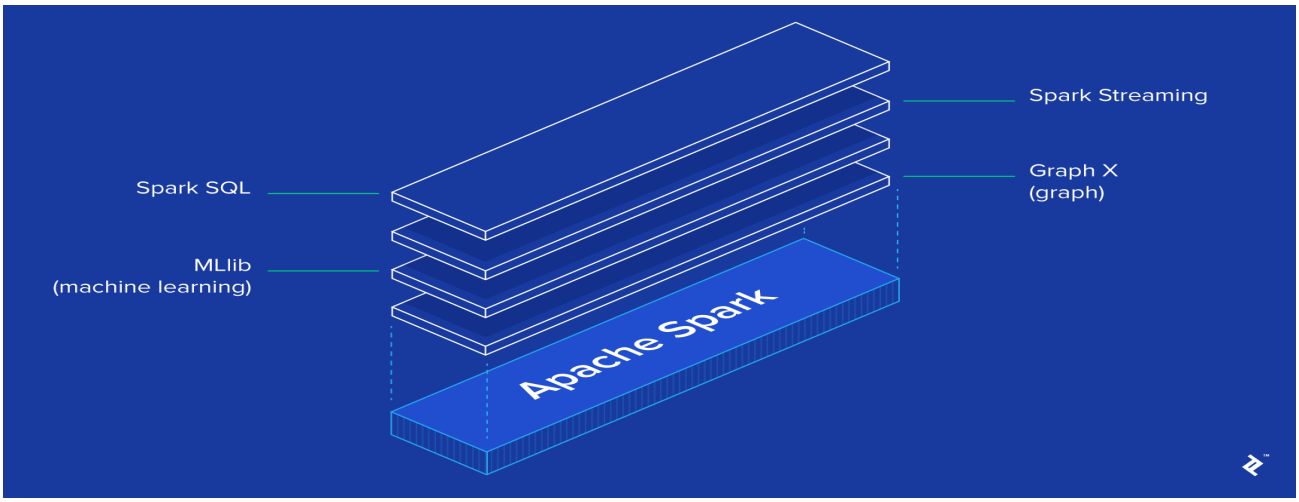
.

3.1 Apache Spark

The Apache project Spark is marketed as offering "lightning-fast cluster computing." It is a multilingual engine that may be used to run data engineering, data science, and machine learning tasks on single-node computers or clusters. . (Spark, Apache Spark™ - Unified Engine for large-scale data analytics, n.d.)

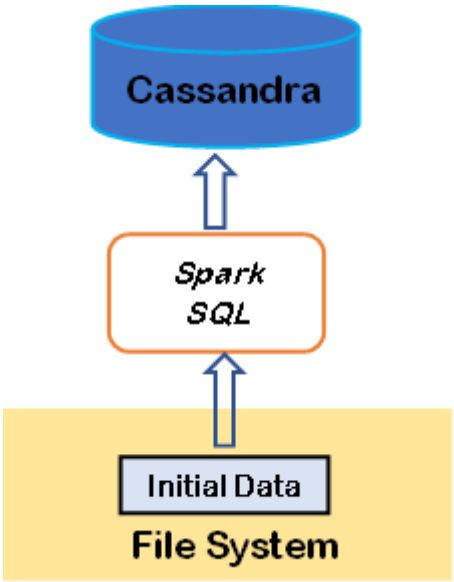
It is currently the most active Apache project and has a vibrant open-source community. Spark offers a framework for quicker and more versatile data processing. Compared to its rival "Hadoop," Spark enables you to run applications up to 100 times quicker in memory or 10 times faster on disk.

The foundational engine for massively parallel and distributed data processing is called the Spark core. A group of potent, higher-level libraries that may be utilized seamlessly in the same application serve as a complement to it. Currently, these libraries consist of SparkSQL, Spark Streaming, and



MLlib. (for machine learning), and GraphX.

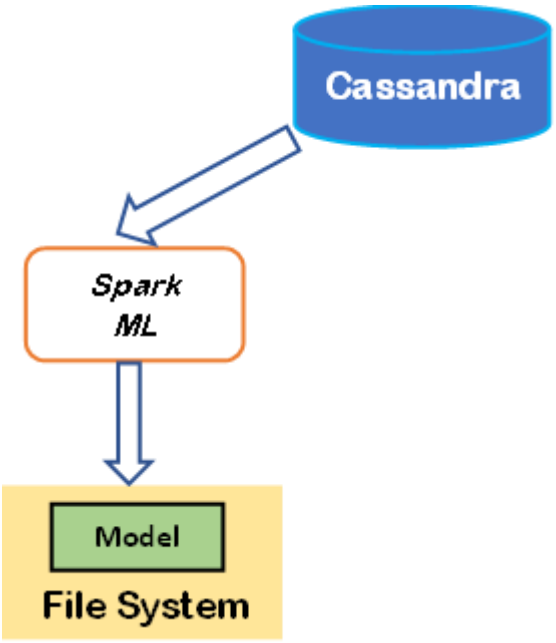
3.2 Spark SQL Job



Spark SQL is a Spark module for handling structured data. Thanks to the interfaces provided by Spark SQL, Spark has access to additional information about the data's structure and the computation being performed. Using this extra data, Spark SQL makes improvements internally. (Spark, Sql Programming Guide, n.d.)

The Cassandra database in this project will get all transaction data from the file system via the Spark SQL Job. This task imports non-fraudulent transactions to a non-fraudulent table and fraudulent transactions to a fraudulent table.

3.3 Spark Machine Learning Job



In this project, the Spark ML Job will read fraud and non-fraud transactions from Cassandra. It will train on this data and it will create a model. This model will be saved to the file system.

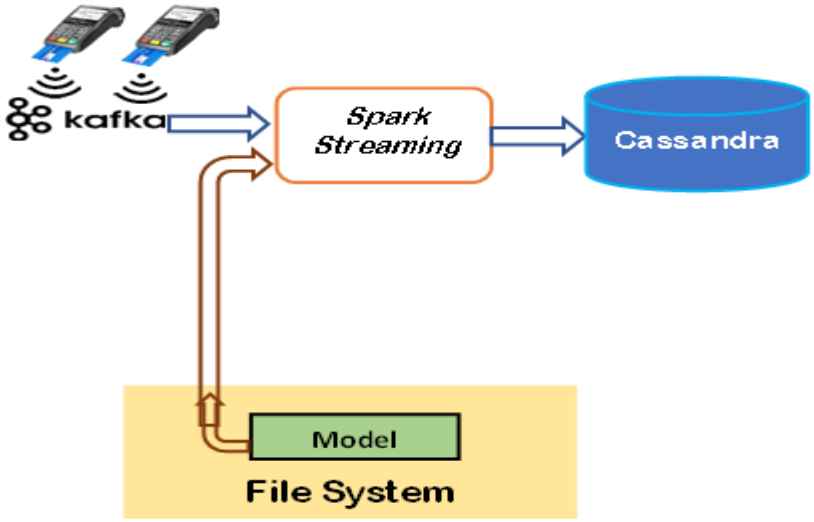
A new package called spark.ml was added to Spark 1.2 with the intention of offering a standardized set of high-level APIs to assist users in building and fine-tuning useful machine learning pipelines. . (Apache, Spark ML Programming Guide - Spark 1.2.2 Documentation (apache.org), n.d.)

To make it simpler to incorporate many algorithms into a single pipeline, or workflow, Spark ML standardizes machine learning APIs. The Spark ML API's primary principles are covered in this section. . (Apache, Spark ML Programming Guide - Spark 1.2.2 Documentation (apache.org), n.d.)

- **ML Dataset:** Spark ML uses the SchemaRDD dataset from Spark SQL, which accommodates a variety of data formats. For instance, a dataset could include several columns for text, feature vectors, actual labels, and predictions.
- **Transformer**: An algorithm called a Transformer can change one SchemaRDD into another. An RDD with features can be transformed into an RDD with predictions using an ML model, for instance.
- **Estimator**: A program called an Estimator can be used to generate a Transformer by fitting it to a SchemaRDD. An Estimator, for instance, is a learning algorithm that trains on data and builds models.
- **Pipeline**: To specify an ML workflow, a pipeline links numerous transformers and estimators together.

- **Param**: The specification of parameters is now standardized across all Transformers and Estimators.

3.4 Spark Streaming Job



The model will be loaded from the file system via the Spark Streaming Job. After that, it will begin consuming Kafka messages about credit card transactions. This model can be used to determine whether a transaction is fraudulent or not. The forecasts will then be saved to the Cassandra database. Non-fraudulent transactions are saved to a separate table from fraudulent transactions, which are saved to the fraud table.

Based on the Spark SQL engine, Spark Structured Streaming is a fault-tolerant and scalable stream processing engine. The same syntax that is used for batch computations on static data can also be used to define streaming computations. The Spark SQL engine will take care of running it continuously and incrementally, updating the outcome as fresh streaming data is received. (Apache, Structured Streaming Programming Guide - Spark 3.2.1 Documentation (apache.org), n.d.)

Internally, by default, Structured Streaming queries are handled by a micro-batch processing engine. This engine breaks up large data streams into smaller batches to process them, resulting in end-to-end latencies as low as 100 milliseconds and exact-once fault-tolerance guarantees. With at least-once guarantees, our Continuous Processing low-latency processing option, introduced with Spark 2.3, may reach end-to-end latencies as low as 1 millisecond. Without changing the Dataset/DataFrame operations in your queries, you will be able to choose the mode based on your application requirements. (Apache, Structured Streaming Programming Guide - Spark 3.2.1 Documentation (apache.org), n.d.)

With the same and well-known Spark APIs, Spark Structured Streaming makes it simple to create streaming applications and pipelines.

3.4.1 Apache Kafka

For mission-critical applications, high-performance data pipelines, streaming analytics, and data integration, many companies employ Apache Kafka, an open-source distributed event streaming platform.

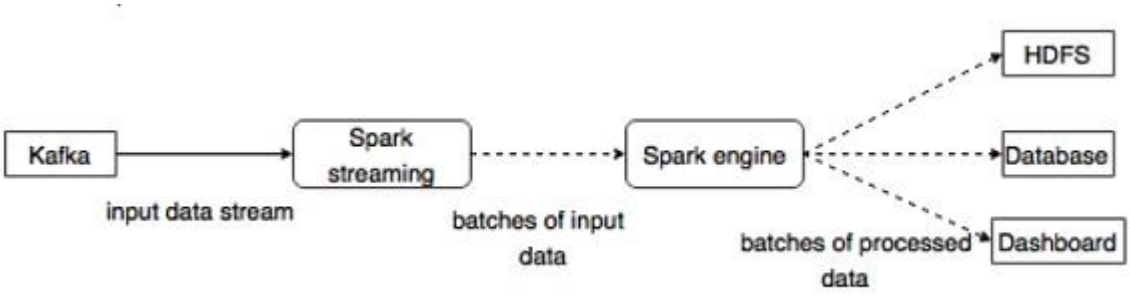
3.4.2 What is event streaming?

Event streaming serves as a digital representation of the human body's central nervous system. It offers the technological foundation for a "always-on" future in which companies are increasingly automated, software-defined, and reliant on software.

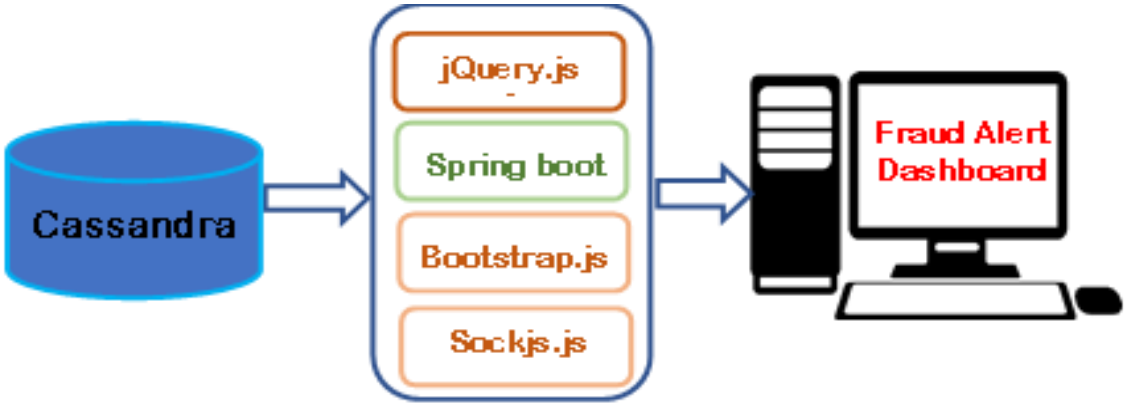
In terms of technology, event streaming refers to the practice of capturing data in real-time from event sources like databases, sensors, mobile devices, cloud services, and software applications in the form of streams of events; storing these event streams durably for later retrieval; manipulating, processing, and reacting to the event streams in real-time as well as retrospectively; and routing the event streams to various destination technologies as necessary. Thus, event streaming ensures that data is continuously interpreted and flowed, ensuring that the appropriate information is available at the appropriate time and location.

3.4.3 Integration of Kafka with Spark

Kafka could be used by Spark streaming as a communications and integration platform. Spark Streaming, which uses advanced algorithms and uses Kafka as its main hub, handles real-time streams of data. Following data processing, Spark Streaming can store the outcomes in HDFS, databases, dashboards, or publish them into a new Kafka topic. The graphic below depicts the conceptual flow.



3.5 Fraud-alert dashboard



The Fraud alert dashboard will internally query the Cassandra for the latest fraud transactions. If there are any fraud transactions, it will be displayed on the dashboard.

To create the fraud alert dashboard web application with spring boot, we will add the dependencies for bootstrap and jQuery to web jars (webjars are client-side web libraries (e.g., jQuery & Bootstrap) packaged into JAR (Java Archive) files that explicitly and easily manage the client-side dependencies in JVM-based web applications).

What is jQuery?

The "write less, do more" JavaScript library jQuery is small and efficient. JQuery aims to simplify the use of JavaScript on your website. By calling methods instead, jQuery enables you to carry out a variety of fundamental tasks that would otherwise need extensive JavaScript code. Only a few of the numerous intricate JavaScript features that jQuery dramatically simplifies include AJAX queries and DOM manipulation.

The jQuery library contains the following features: HTML/DOM manipulation, CSS manipulation, HTML event methods, Effects and animations, AJAX, Utilities.

Why jQuery?

Other JavaScript libraries abound, but jQuery is probably the most popular and versatile. jQuery helps traversing HTML pages, event management, animation, and Ajax interactions easy for speedy web development. Comparatively speaking, JavaScript and its other JavaScript libraries are more user-friendly than jQuery. jQuery requires less lines of code to be written than JavaScript does.

Spring Boot

A Java-based open-source framework called Spring Boot is used to build micro-Services (architecture that allows the developers to develop and deploy services independently). Pivotal Team is the one who creates it. Java programmers can utilize Spring Boot to build production-ready, stand-alone spring apps that are simple to use.

Why Spring Boot?

Spring Boot provides a flexible way to configure Java Beans, XML configurations, and database transactions. Strong batch processing is provided, and REST endpoints are maintained.

With Spring Boot, everything is automatically configured; nothing needs to be done by hand. There are spring applications that use annotations, and dependency management has been simplified. .

What is SockJS?

The JavaScript package SockJS (for browsers) offers an object that resembles a WebSocket.

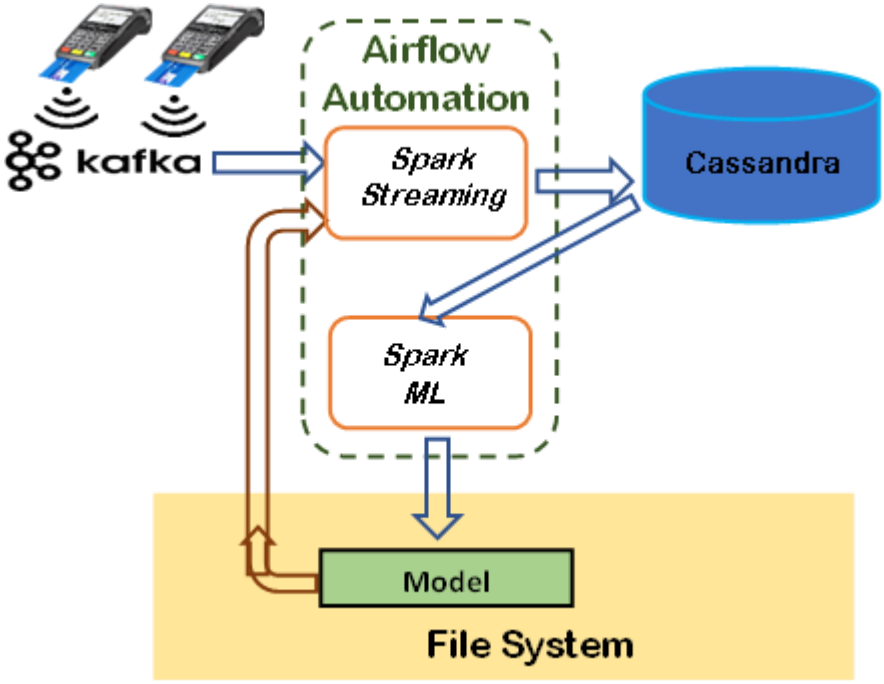
The WebSocket API is a cutting-edge technology that enables a user's browser and a server to begin an interactive two-way communication session. You can use this API to send messages to a server and receive event-driven responses without having to poll the server for a response.

SockJS gives you a uniform, cross-browser JavaScript API that creates a low-latency, full-duplex, cross-domain channel of communication between the browser and the web server, whether or not you use Web Sockets.

Bootstrap.JS, jQuery.JS and Spring boot

Bootstrap is a robust, feature-rich frontend toolkit, while jQuery is a well-known JavaScript library. Bootstrap.js provides the JavaScript code required to execute Bootstrap. Both are accepted client-side web development technologies. Additionally, Spring is a well-liked framework for creating enterprise Java applications, and Spring Boot is a collection of tools and frameworks that significantly simplify Spring development. .

3.6 Airflow Automation



Every time a new model is built, Airflow automation is utilized to stop and start a new Spark Streaming job. The Data Science Team will run the Spark ML Job manually in a production environment. To develop a new model, it will be manually ran once a week or once a month. The effectiveness of the new model will be evaluated. Its effectiveness will also be contrasted with that of the preceding model. The new model will be implemented if it is superior than the current model. In order to choose the new Model, the Spark Streaming Job that is currently running will be halted and a new one will be launched.

A tool for task automation is Airflow. Scheduling jobs helps businesses ensure that they are completed at the appropriate time. The employees are relieved of repetitive work because of this. Additionally, Apache Airflow is utilized to plan and coordinate data pipelines or operations.

Users can automate scripts to perform tasks with an application called Airflow. It has a scheduler that manages a variety of employees while sticking to a set of predetermined dependencies. Users can easily interact with directed acyclic graphs because to Airflow's comprehensive command-line tools (DAGs). The DAGs simplify task ordering and management for enterprises.

Key Features of Airflow

Dynamic Integration: Python is the backend programming language used by Airflow to create dynamic pipelines. Workflows are created by connecting DAG using a variety of operators, hooks, and connectors.

Elegant User Interface: Since Airflow builds its pipelines using Jinja templates, the pipelines are concise and straightforward. In Airflow, parameterizing your scripts is a simple process.

Airflow also has a rich user interface that makes it easy to monitor progress, visualize pipelines running in production, and troubleshoot issues when necessary.

3.7 SYSTEM DESIGN

INTRODUCTION

As we aforementioned Spark is a cluster computing framework. In Spark, applications run as independent sets of processes on a cluster. Spark gives control over resource allocation both across applications (at the level of the cluster manager) and within applications (if multiple computations are happening on the same Spark Context)

Spark has several facilities for scheduling resources between computations. Within each Spark application, multiple "jobs" (Spark actions) may be running concurrently if they were submitted by different threads.

3.8 Initial Spark Job

This Spark Job will read the customer data and transaction data from the filesystem. It will do some ETL transformations (a three-phase process where data is extracted, transformed and loaded into an output data container) on customer and transaction data, like computing age of the customer, distance between customer and merchant place, etc. It will save these transformed data into Cassandra database. It will save the customers data to customer table; Fraud transactions will be saved to fraud transaction table and non-fraud transactions to non_fraud_transaction.

3.8.1 ETL Transformations on data

Compute Age

Customer.csv has a column with date of birth of the customer. From the date of birth, we will get the age of the customer. Age will be used as a feature in the Spark ML Job.

Compute Distance

Customer.csv has customer's home location (latitude and longitude). Transactions.csv has Merchant Location (latitude and longitude). Using this information, we will compute the distance between Customer and Merchant. Again, distance will be used as a feature in Spark ML Job

3.9 Spark Training Job

This Job will read fraud and non-fraud transaction data from Cassandra. It will train on this data and it will create a model. This model will be saved to the file system.

3.9.1 Spark Training Steps

3.9.2 Read data from Cassandra

The features will be used to train this model are "category", "merchant", "distance", "amt" and "age".

```
val fraudTransactionDF =
DataReader.readFromCassandra(CassandraConfig.keyspace,
CassandraConfig.fraudTransactionTable)
      .select("cc_num" , "category", "merchant", "distance", "amt", "age",
"is_fraud")

val nonFraudTransactionDF =
DataReader.readFromCassandra(CassandraConfig.keyspace,
CassandraConfig.nonFraudTransactionTable)
      .select("cc_num" , "category", "merchant", "distance", "amt", "age",
"is_fraud")

val transactionDF = nonFraudTransactionDF.union(fraudTransactionDF)
    transactionDF.cache()
```

```
+----------------+------------+---------------------------+--------+-----+---+--------+
|cc_num          |category    |merchant                   |distance|amt  |age|is_fraud|
+----------------+------------+---------------------------+--------+-----+---+--------+
|5157595343543285|gas_transport|Schmitt Inc                |0.73    |111.0|46 |0.0     |
|5157595343543285|kids_pets    |Beer-Jast                  |1.73    |66.0 |46 |0.0     |
|5157595343543285|grocery_pos  |Hudson-Ratke               |1.23    |220.0|46 |0.0     |
|5157595343543285|entertainment|Kassulke Inc               |2.17    |100.0|46 |0.0     |
|5157595343543285|shopping_net |Boyer PLC                  |1.33    |80.0 |46 |0.0     |
|5157595343543285|entertainment|Spencer PLC                |1.29    |141.0|46 |0.0     |
|5157595343543285|entertainment|Brown-Greenholt            |0.8     |221.0|46 |0.0     |
|5157595343543285|travel       |Kovacek Ltd                |1.5     |91.0 |46 |0.0     |
|5157595343543285|gas_transport|Kutch LLC                  |0.71    |57.0 |46 |0.0     |
|5157595343543285|kids_pets    |Lowe, Dietrich and Erdman  |1.41    |158.0|46 |0.0     |
|5157595343543285|kids_pets    |Bednar PLC                 |1.99    |211.0|46 |0.0     |
|5157595343543285|kids_pets    |Hammes-Beatty              |0.88    |42.0 |46 |0.0     |
|5157595343543285|kids_pets    |Hilpert-Conroy             |0.97    |103.0|46 |0.0     |
|5157595343543285|gas_transport|Robel, Cummerata and Prosacco|1.26    |100.0|46 |0.0     |
|5157595343543285|grocery_pos  |Strosin-Cruickshank        |1.02    |75.0 |46 |0.0     |
|5157595343543285|grocery_pos  |Schultz, Simonis and Little|1.8     |65.0 |46 |0.0     |
|5157595343543285|home         |Beier and Sons             |1.66    |112.0|46 |0.0     |
|5157595343543285|misc_pos     |Turner, Ruecker and Parisian|1.81    |27.0 |46 |0.0     |
|5157595343543285|entertainment|Howe PLC                   |1.56    |151.0|46 |0.0     |
|5157595343543285|entertainment|Bauch-Blanda               |1.88    |126.0|46 |0.0     |
+----------------+------------+---------------------------+--------+-----+---+--------+
only showing top 20 rows
```

3.9.3 String Values to Numeric Values

String Indexer transforms string values to double values. Because, ML Algorithms does not understand String values, it understands only numeric values. Input to String Indexer is “category” and “merchant” column. Output is “category_indexed” and “merchant_indexed” column. StringIndexer will append these 2 columns to the result data frame. StringIndexer will convert the category and merchant names to double values.

```
/*String Indexer for the category and merchant columns*/
    val categoryIndexer = new
StringIndexer().setInputCol("category").setOutputCol("category_indexed")
    val merchantIndexer = new
StringIndexer().setInputCol("merchant").setOutputCol("merchant_indexed")
```

```
|cc_num         |category     |merchant                    |distance|amt  |age|is_fraud|category_indexed|merchant_indexed|
+---------------+-------------+----------------------------+--------+-----+---+--------+----------------+----------------+
|5157595343543285|gas_transport|Schmitt Inc                 |0.73    |111.0|46 |0.0     |0.0             |28.0            |
|5157595343543285|kids_pets    |Beer-Jast                   |1.73    |66.0 |46 |0.0     |4.0             |291.0           |
|5157595343543285|grocery_pos  |Hudson-Ratke                |1.23    |220.0|46 |0.0     |1.0             |310.0           |
|5157595343543285|entertainment|Kassulke Inc                |2.17    |100.0|46 |0.0     |9.0             |470.0           |
|5157595343543285|shopping_net |Boyer PLC                   |1.33    |80.0 |46 |0.0     |5.0             |78.0            |
|5157595343543285|entertainment|Spencer PLC                 |1.29    |141.0|46 |0.0     |9.0             |304.0           |
|5157595343543285|entertainment|Brown-Greenholt             |0.8     |221.0|46 |0.0     |9.0             |527.0           |
|5157595343543285|travel       |Kovacek Ltd                 |1.5     |91.0 |46 |0.0     |13.0            |663.0           |
|5157595343543285|gas_transport|Kutch LLC                   |0.71    |57.0 |46 |0.0     |0.0             |2.0             |
|5157595343543285|kids_pets    |Lowe, Dietrich and Erdman   |1.41    |158.0|46 |0.0     |4.0             |230.0           |
|5157595343543285|kids_pets    |Bednar PLC                  |1.99    |211.0|46 |0.0     |4.0             |101.0           |
|5157595343543285|kids_pets    |Hammes-Beatty               |0.88    |42.0 |46 |0.0     |4.0             |242.0           |
|5157595343543285|kids_pets    |Wilpert-Conroy              |0.97    |103.0|46 |0.0     |4.0             |223.0           |
|5157595343543285|gas_transport|Robel, Cummerata and Prosacco|1.26    |100.0|46 |0.0     |0.0             |32.0            |
|5157595343543285|grocery_pos  |Strosin-Cruickshank         |1.02    |75.0 |46 |0.0     |1.0             |181.0           |
|5157595343543285|grocery_pos  |Schultz, Simonis and Little |1.8     |65.0 |46 |0.0     |1.0             |133.0           |
|5157595343543285|home         |Beier and Sons              |1.66    |112.0|46 |0.0     |3.0             |81.0            |
|5157595343543285|misc_pos     |Turner, Ruecker and Parisian|1.81    |27.0 |46 |0.0     |7.0             |229.0           |
|5157595343543285|entertainment|Howe PLC                    |1.56    |151.0|46 |0.0     |9.0             |201.0           |
|5157595343543285|entertainment|Bauch-Blanda                |1.88    |126.0|46 |0.0     |9.0             |509.0           |
+---------------+-------------+----------------------------+--------+-----+---+--------+----------------+----------------+
only showing top 20 rows
```

3.9.4 Scaling Numeric Columns

The main idea is to Normalize/Standardize (mean = 0 and standard deviation = 1) the features before applying machine learning techniques. Standard Scaler does not work on double values. Hence, first we need to convert all 3 columns into vector, then sliced it and finally scale it.

Vector Assembler

Vector Assembler is used to transform multiple column values into a single vector column. Input to vector assembler is “distance”, “amt” and “age” columns. The output is the “rawfeature” column. It is a vector column.

```
val numericAssembler = new VectorAssembler().setInputCols(Array("distance",
"amt", "age" )).setOutputCol("rawfeature")
```

distance	amt	age	is_fraud	category_indexed	merchant_indexed	rawfeature
0.73	111.0	46	0.0	0.0	28.0	[0.73,111.0,46.0]
1.73	66.0	46	0.0	4.0	291.0	[1.73,66.0,46.0]
1.23	220.0	46	0.0	1.0	310.0	[1.23,220.0,46.0]
2.17	100.0	46	0.0	9.0	470.0	[2.17,100.0,46.0]
1.33	80.0	46	0.0	5.0	78.0	[1.33,80.0,46.0]
1.29	141.0	46	0.0	9.0	304.0	[1.29,141.0,46.0]
0.8	221.0	46	0.0	9.0	527.0	[0.8,221.0,46.0]
1.5	91.0	46	0.0	13.0	663.0	[1.5,91.0,46.0]
0.71	57.0	46	0.0	0.0	2.0	[0.71,57.0,46.0]
1.41	158.0	46	0.0	4.0	230.0	[1.41,158.0,46.0]
1.99	211.0	46	0.0	4.0	101.0	[1.99,211.0,46.0]
0.88	42.0	46	0.0	4.0	242.0	[0.88,42.0,46.0]
0.97	103.0	46	0.0	4.0	223.0	[0.97,103.0,46.0]
1.26	100.0	46	0.0	0.0	32.0	[1.26,100.0,46.0]
1.02	75.0	46	0.0	1.0	181.0	[1.02,75.0,46.0]
1.8	65.0	46	0.0	1.0	133.0	[1.8,65.0,46.0]
1.66	112.0	46	0.0	3.0	81.0	[1.66,112.0,46.0]
1.81	27.0	46	0.0	7.0	229.0	[1.81,27.0,46.0]
1.56	151.0	46	0.0	9.0	201.0	[1.56,151.0,46.0]
1.88	126.0	46	0.0	9.0	509.0	[1.88,126.0,46.0]

Vector Slicer

Vector Slicer will slice a single vector into multiple vectors. Input to Vector Slicer is the output of VectorAssebleri.e “rawfeature” column. The output of VectorSlicer is the “slicedfeatures” column. This column contains “distance”, “amt” and “age” as individual vectors.

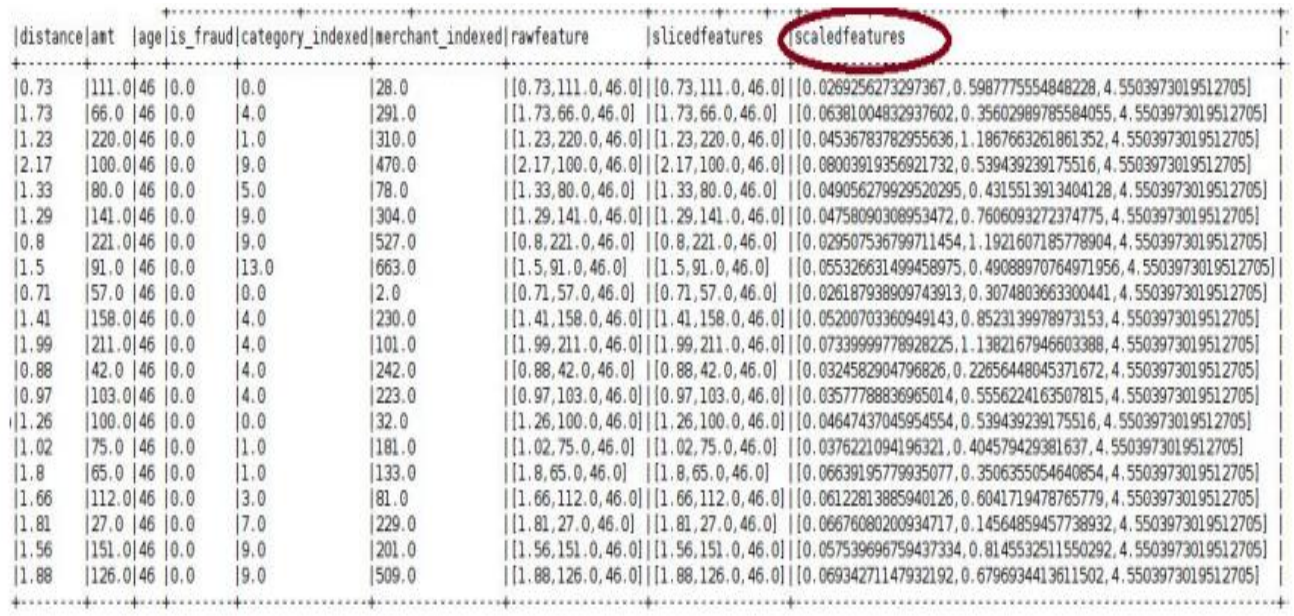
```
val slicer = new
VectorSlicer().setInputCol("rawfeature").setOutputCol("slicedfeatures").setNa
mes(Array("distance", "amt", "age"))
```

distance	amt	age	is_fraud	category_indexed	merchant_indexed	rawfeature	slicedfeatures
0.73	111.0	46	0.0	0.0	28.0	[0.73,111.0,46.0]	[0.73,111.0,46.0]
1.73	66.0	46	0.0	4.0	291.0	[1.73,66.0,46.0]	[1.73,66.0,46.0]
1.23	220.0	46	0.0	1.0	310.0	[1.23,220.0,46.0]	[1.23,220.0,46.0]
2.17	100.0	46	0.0	9.0	470.0	[2.17,100.0,46.0]	[2.17,100.0,46.0]
1.33	80.0	46	0.0	5.0	78.0	[1.33,80.0,46.0]	[1.33,80.0,46.0]
1.29	141.0	46	0.0	9.0	304.0	[1.29,141.0,46.0]	[1.29,141.0,46.0]
0.8	221.0	46	0.0	9.0	527.0	[0.8,221.0,46.0]	[0.8,221.0,46.0]
1.5	91.0	46	0.0	13.0	663.0	[1.5,91.0,46.0]	[1.5,91.0,46.0]
0.71	57.0	46	0.0	0.0	2.0	[0.71,57.0,46.0]	[0.71,57.0,46.0]
1.41	158.0	46	0.0	4.0	230.0	[1.41,158.0,46.0]	[1.41,158.0,46.0]
1.99	211.0	46	0.0	4.0	101.0	[1.99,211.0,46.0]	[1.99,211.0,46.0]
0.88	42.0	46	0.0	4.0	242.0	[0.88,42.0,46.0]	[0.88,42.0,46.0]
0.97	103.0	46	0.0	4.0	223.0	[0.97,103.0,46.0]	[0.97,103.0,46.0]
co 1.26	100.0	46	0.0	0.0	32.0	[1.26,100.0,46.0]	[1.26,100.0,46.0]
1.02	75.0	46	0.0	1.0	181.0	[1.02,75.0,46.0]	[1.02,75.0,46.0]
1.8	65.0	46	0.0	1.0	133.0	[1.8,65.0,46.0]	[1.8,65.0,46.0]
1.66	112.0	46	0.0	3.0	81.0	[1.66,112.0,46.0]	[1.66,112.0,46.0]
n 1.81	27.0	46	0.0	7.0	229.0	[1.81,27.0,46.0]	[1.81,27.0,46.0]
1.56	151.0	46	0.0	9.0	201.0	[1.56,151.0,46.0]	[1.56,151.0,46.0]
1.88	126.0	46	0.0	9.0	509.0	[1.88,126.0,46.0]	[1.88,126.0,46.0]

Standard Scaler

Now that we have converted all the 3 columns into vector, and then sliced it and finally scaled it, We can then use the StandardScaler. Input to StandardScaler is the output of the VectorSliceri.e "slicedfeatures" column. The output of StandardScaler is the "scaledfeatures" column. Now this column contains all normalized values.

```
val scaler = new
StandardScaler().setInputCol("slicedfeatures").setOutputCol("scaledfeatures")
```

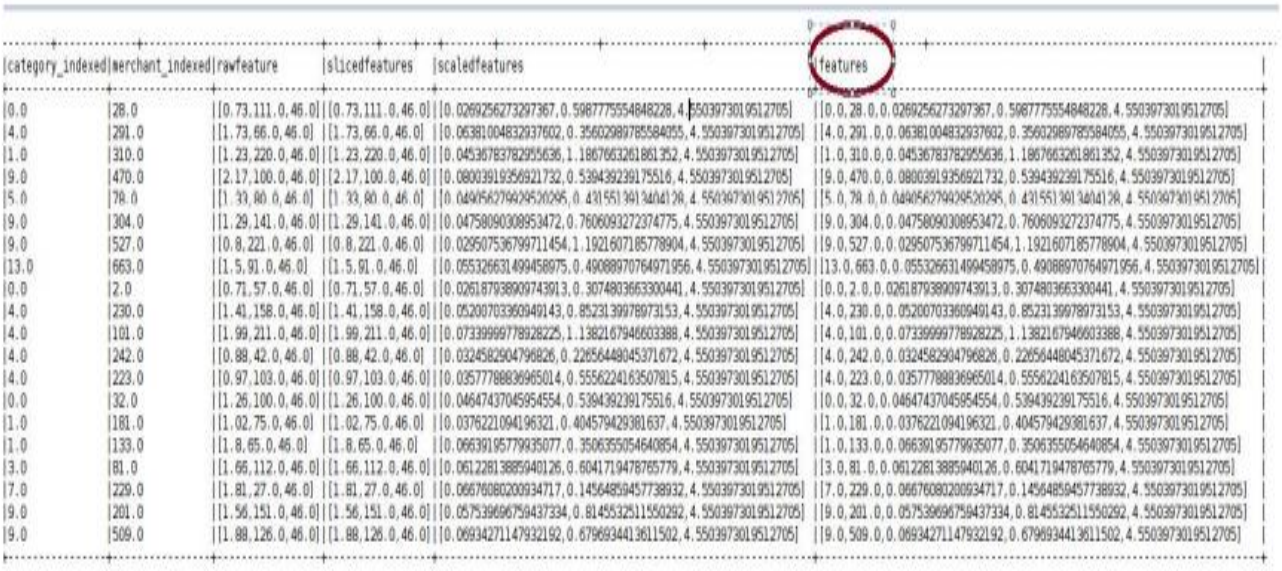


distance	amt	age	is_fraud	category_indexed	merchant_indexed	rawfeature	slicedfeatures	scaledfeatures
0.73	111.0	46	0.0	0.0	28.0	[0.73,111.0,46.0]	[0.73,111.0,46.0]	[0.0269256273297367,0.598777554848228,4.5503973019512705]
1.73	66.0	46	0.0	4.0	291.0	[1.73,66.0,46.0]	[1.73,66.0,46.0]	[0.06381004832937602,0.35602989785584055,4.5503973019512705]
1.23	220.0	46	0.0	1.0	310.0	[1.23,220.0,46.0]	[1.23,220.0,46.0]	[0.04536783782955636,1.1867663261861352,4.5503973019512705]
2.17	100.0	46	0.0	9.0	470.0	[2.17,100.0,46.0]	[2.17,100.0,46.0]	[0.08003919356921732,0.5394392391755l6,4.5503973019512705]
1.33	80.0	46	0.0	5.0	78.0	[1.33,80.0,46.0]	[1.33,80.0,46.0]	[0.049056279929520295,0.4315513913404128,4.5503973019512705]
1.29	141.0	46	0.0	9.0	304.0	[1.29,141.0,46.0]	[1.29,141.0,46.0]	[0.04758090308953472,0.7606093272374775,4.5503973019512705]
0.8	221.0	46	0.0	9.0	527.0	[0.8,221.0,46.0]	[0.8,221.0,46.0]	[0.029507536799711454,1.1921607185778904,4.5503973019512705]
1.5	91.0	46	0.0	13.0	663.0	[1.5,91.0,46.0]	[1.5,91.0,46.0]	[0.055326631499458975,0.49088970764971956,4.5503973019512705]
0.71	57.0	46	0.0	0.0	2.0	[0.71,57.0,46.0]	[0.71,57.0,46.0]	[0.026187938909743913,0.3074803663300441,4.5503973019512705]
1.41	158.0	46	0.0	4.0	230.0	[1.41,158.0,46.0]	[1.41,158.0,46.0]	[0.05200703360949143,0.8523139978973153,4.5503973019512705]
1.99	211.0	46	0.0	4.0	101.0	[1.99,211.0,46.0]	[1.99,211.0,46.0]	[0.07339999778928225,1.1382167946603388,4.5503973019512705]
0.88	42.0	46	0.0	4.0	242.0	[0.88,42.0,46.0]	[0.88,42.0,46.0]	[0.0324582904796826,0.22656448045371672,4.5503973019512705]
0.97	103.0	46	0.0	4.0	223.0	[0.97,103.0,46.0]	[0.97,103.0,46.0]	[0.03577788836965014,0.5556224163507815,4.5503973019512705]
1.26	100.0	46	0.0	0.0	32.0	[1.26,100.0,46.0]	[1.26,100.0,46.0]	[0.04647437045954554,0.539439239175l6,4.5503973019512705]
1.02	75.0	46	0.0	1.0	181.0	[1.02,75.0,46.0]	[1.02,75.0,46.0]	[0.0376221094196321,0.404579429381637,4.5503973019512705]
1.8	65.0	46	0.0	1.0	133.0	[1.8,65.0,46.0]	[1.8,65.0,46.0]	[0.06639195779935077,0.3506355054640854,4.5503973019512705]
1.66	112.0	46	0.0	3.0	81.0	[1.66,112.0,46.0]	[1.66,112.0,46.0]	[0.06122813885940126,0.6041719478765779,4.5503973019512705]
1.81	27.0	46	0.0	7.0	229.0	[1.81,27.0,46.0]	[1.81,27.0,46.0]	[0.06676080200934717,0.14564859457738932,4.5503973019512705]
1.56	151.0	46	0.0	9.0	201.0	[1.56,151.0,46.0]	[1.56,151.0,46.0]	[0.057539696759437334,0.8145532511550292,4.5503973019512705]
1.88	126.0	46	0.0	9.0	509.0	[1.88,126.0,46.0]	[1.88,126.0,46.0]	[0.06934271147932192,0.6796934413611502,4.5503973019512705]

3.9.5 Feature Column

Now all the feature columns have to be transformed into single feature column. i.e"category_index", "merchant_index" and "scaledfeatures" columns into single "feature" column. One more VectorAssembler is created. Input to this VectorAssembler is the output of previous transformations i.e. "category_indexed", "merchant_indexed" and "scaledfeatures" columns. Output is "feature column". All the columns are combined as a single column of vector data type.

```
val vectorAssember = new
VectorAssembler().setInputCols(Array("category_indexed",
"merchant_indexed", "scaledfeatures")).setOutputCol("features")
```



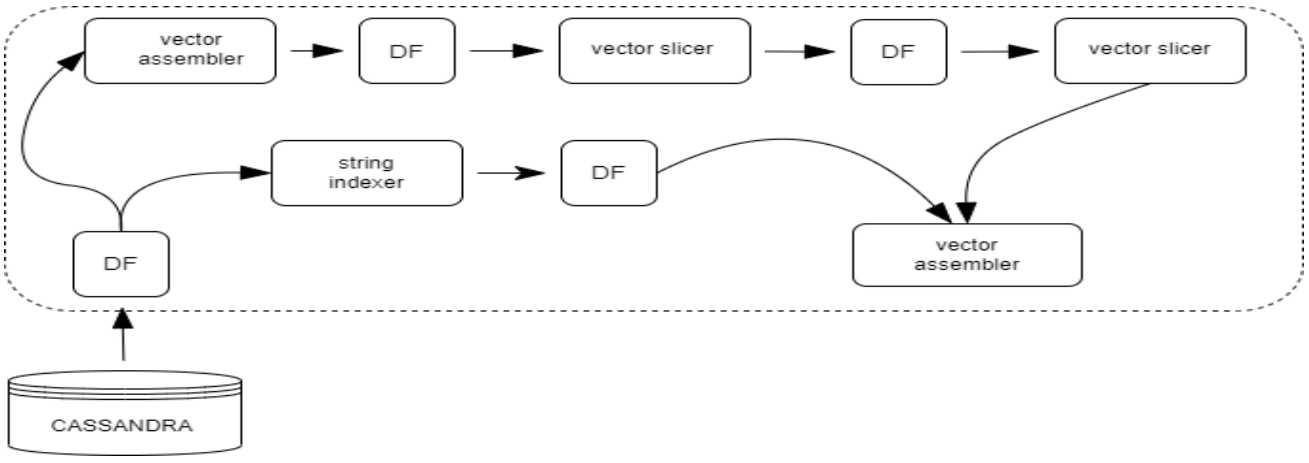
category_indexed	merchant_indexed	rawfeature	slicedfeatures	scaledfeatures	features
0.0	28.0	[0.73,111.0,46.0]	[0.73,111.0,46.0]	[0.0269256273297367,0.5987775554848228,4.5503973019512705]	[0.0,28.0,0.0269256273297367,0.5987775554848228,4.5503973019512705]
4.0	291.0	[1.73,66.0,46.0]	[1.73,66.0,46.0]	[0.06381004832937602,0.3560298978558405,4.5503973019512705]	[4.0,291.0,0.06381004832937602,0.3560298978558405,4.5503973019512705]
1.0	310.0	[1.23,220.0,46.0]	[1.23,220.0,46.0]	[0.045367837829555636,1.1867663261861352,4.5503973019512705]	[1.0,310.0,0.045367837829555636,1.1867663261861352,4.5503973019512705]
9.0	470.0	[2.17,100.0,46.0]	[2.17,100.0,46.0]	[0.08003919356921732,0.5394392391755516,4.5503973019512705]	[9.0,470.0,0.08003919356921732,0.5394392391755516,4.5503973019512705]
5.0	78.0	[1.33,80.0,46.0]	[1.33,80.0,46.0]	[0.04905627949295209,0.4315513913404128,4.5503973019512705]	[5.0,78.0,0.04905627949295209,0.4315513913404128,4.5503973019512705]
9.0	304.0	[1.29,141.0,46.0]	[1.29,141.0,46.0]	[0.04758090308953472,0.7606093272374775,4.5503973019512705]	[9.0,304.0,0.04758090308953472,0.7606093272374775,4.5503973019512705]
9.0	527.0	[0.8,221.0,46.0]	[0.8,221.0,46.0]	[0.029507536799711454,1.1921607185778904,4.5503973019512705]	[9.0,527.0,0.029507536799711454,1.1921607185778904,4.5503973019512705]
13.0	663.0	[1.5,91.0,46.0]	[1.5,91.0,46.0]	[0.05532663149945895,0.4908897076497956,4.5503973019512705]	[13.0,663.0,0.05532663149945895,0.4908897076497956,4.5503973019512705]
0.0	2.0	[0.71,57.0,46.0]	[0.71,57.0,46.0]	[0.026187938909743913,0.3074803663300441,4.5503973019512705]	[0.0,2.0,0.026187938909743913,0.3074803663300441,4.5503973019512705]
4.0	230.0	[1.41,158.0,46.0]	[1.41,158.0,46.0]	[0.05200703360949143,0.8523139978973153,4.5503973019512705]	[4.0,230.0,0.05200703360949143,0.8523139978973153,4.5503973019512705]
4.0	101.0	[1.99,211.0,46.0]	[1.99,211.0,46.0]	[0.07339999778928225,1.1382167946603388,4.5503973019512705]	[4.0,101.0,0.07339999778928225,1.1382167946603388,4.5503973019512705]
4.0	242.0	[0.88,42.0,46.0]	[0.88,42.0,46.0]	[0.03245829047968260,0.2265644804537167,2,4.5503973019512705]	[4.0,242.0,0.03245829047968260,0.2265644804537167,2,4.5503973019512705]
4.0	223.0	[0.97,103.0,46.0]	[0.97,103.0,46.0]	[0.03577788836965014,0.5556224163507815,4.5503973019512705]	[4.0,223.0,0.03577788836965014,0.5556224163507815,4.5503973019512705]
0.0	32.0	[1.26,100.0,46.0]	[1.26,100.0,46.0]	[0.04647437045954554,0.5394392391755164,4.5503973019512705]	[0.0,32.0,0.04647437045954554,0.5394392391755164,4.5503973019512705]
1.0	181.0	[1.02,75.0,46.0]	[1.02,75.0,46.0]	[0.03762210941963210,0.4045794293816374,4.5503973019512705]	[1.0,181.0,0.03762210941963210,0.4045794293816374,4.5503973019512705]
1.0	133.0	[1.8,65.0,46.0]	[1.8,65.0,46.0]	[0.06639195779935077,0.3506355054640854,4.5503973019512705]	[1.0,133.0,0.06639195779935077,0.3506355054640854,4.5503973019512705]
3.0	81.0	[1.66,112.0,46.0]	[1.66,112.0,46.0]	[0.06122813885940126,0.6041719478765779,4.5503973019512705]	[3.0,81.0,0.06122813885940126,0.6041719478765779,4.5503973019512705]
7.0	229.0	[1.81,27.0,46.0]	[1.81,27.0,46.0]	[0.06676080200934717,0.1456485945773893,2,4.5503973019512705]	[7.0,229.0,0.06676080200934717,0.1456485945773893,2,4.5503973019512705]
9.0	201.0	[1.56,151.0,46.0]	[1.56,151.0,46.0]	[0.05753969675943733,0.8145532511550292,4.5503973019512705]	[9.0,201.0,0.05753969675943733,0.8145532511550292,4.5503973019512705]
9.0	509.0	[1.88,126.0,46.0]	[1.88,126.0,46.0]	[0.06934271147932192,0.6796934413611502,4.5503973019512705]	[9.0,509.0,0.06934271147932192,0.6796934413611502,4.5503973019512705]

3.10 Pipeline Stages

Each level in a Spark Pipeline is either an Estimator or a Transformer, depending on the specification. The phases are carried out sequentially, and each one transforms the input DataFrame. A Transformer is an algorithm that applies the transform() method to transform one DataFrame into another. A feature transformer might, for instance, read one column from a DataFrame, map it to another column, and then produce a new DataFrame with the mapped column appended to it.

A Transformer is created by fitting or training on a dataset with an Estimator, an abstraction of learning techniques. A method called fit() that takes a DataFrame and outputs a Model, a Transformer, is implemented by an Estimator.

StringIndexer is an Estimator. VectorAssembler is a Transformer VectorSlicer is a Transformer StandarScaler is an Estimator.



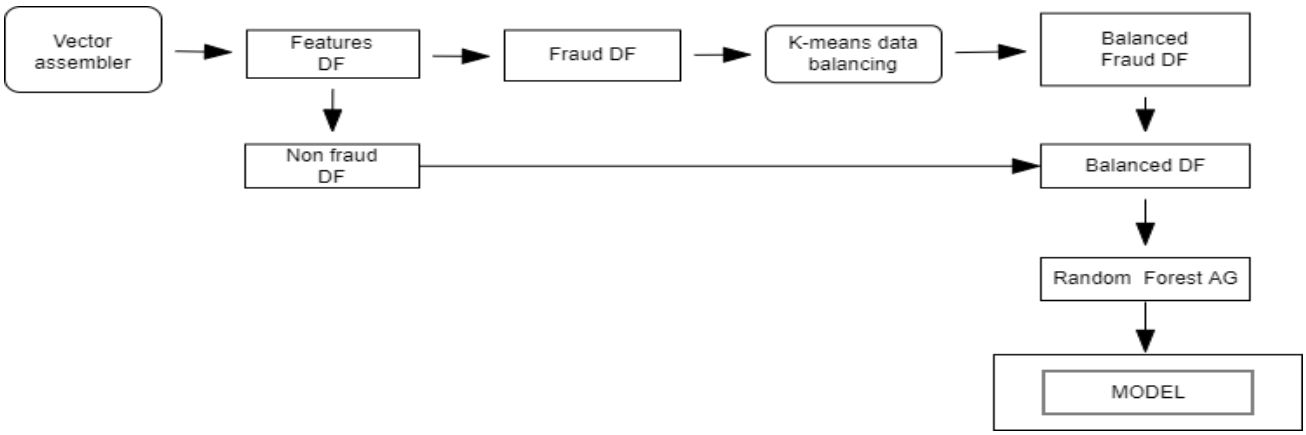
3.10.1 Split data

Split data into training and testing. Training data is 80% and Test data is 20%

```
val Array(trainData, testData) = featureDF.randomSplit(Array(0.8, 0.2))
```

3.10.2 Data Balancing

Typically, there will be millions more legitimate transactions than fraudulent ones (a few hundred). Such information is unbalanced and needs to be corrected. Real transactions must therefore be balanced. The number of legitimate transactions equals or decreases the number of fraudulent transactions. In this case, the K Means Algorithm is being used to balance real transactions. Other strategies for data balancing exist.



3.10.3 Training

A classification issue exists here. It can be either fraud or not. We are employing Random Forest in this project's context. We would experiment with several Classification algorithms in a production setting. Utilize the confusion matrix, AUC, ROC, etc. to evaluate these algorithms. But in this case, we are not comparing several algorithms. Random Forest is what we're using.

Estimator Radom Forest is. Therefore, we must call the fit() method. The Random Forest algorithm actually begins training on the data when the fit() method is called, and it eventually returns a model, which is a transformer. Our Random Forest Model is seen here. By using the convert() technique, we are able to forecast the test data using this model.

```
def randomForestClassifier(df: org.apache.spark.sql.DataFrame)(implicit
sparkSession:SparkSession) = {
    import sparkSession.implicits._
    val randomForestEstimator = new
RandomForestClassifier().setLabelCol("is_fraud").setFeaturesCol("features").setMax
Bins(700)
    val model = randomForestEstimator.fit(df)
    model
  }
```

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Confusion Matrix

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

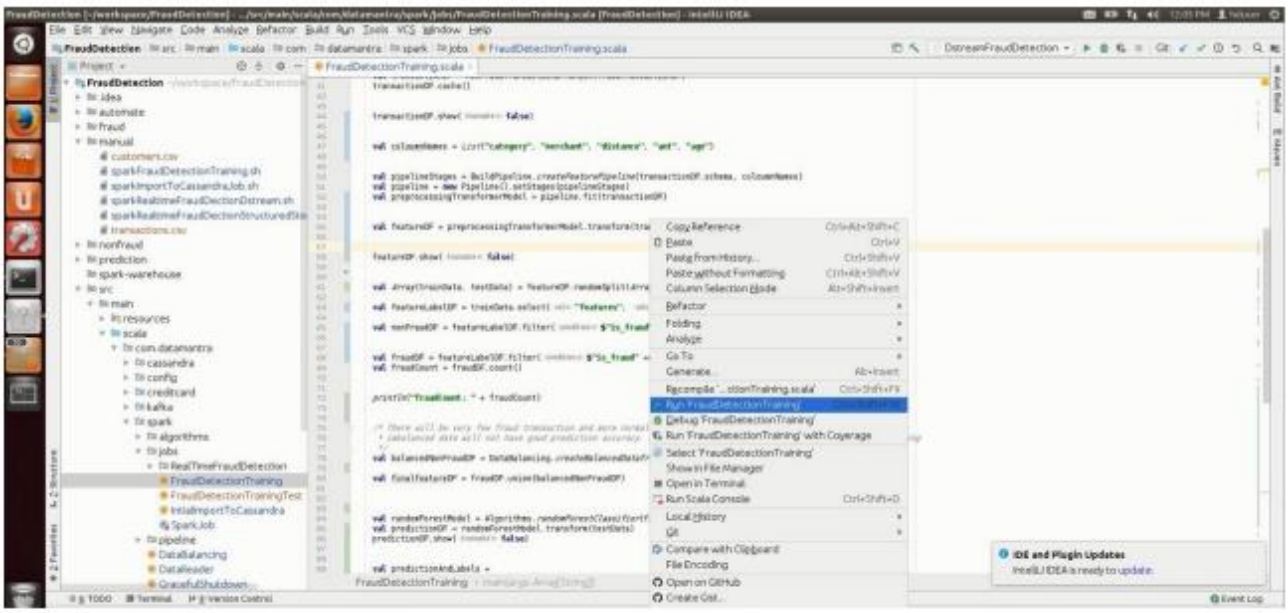
False Positive (FP) = Model predicting genuine transactions as fraud = 83

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

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

Out of all 91 fraud transactions, the model has correctly predicted all 91 fraud transaction as fraud.

Out of 2631 genuine transactions, the model has wrongly predicted 83 as fraud transactions and correctly predicted 2548 as genuine transactions.



Output

```
================== Confusion matrix ============================
#############| Actual = 1                          Actual = 0
------------+-------------------------------------------------
Predicted = 1| 91.000000                          83.000000
Predicted = 0| 0.000000                           2548.000000
================================================================

True Positive Rate: 1.0
False Positive Rate: 0.03154694
Precision: 0.5229885
```

True Positive Rate = TP/TP + FN = 91.0/(91.0 + 0) = 1.0

Out of all the fraud transactions, how much we predicted correctly, It should be high as possible.

False Positive Rate (FPR) = FP/FP + TN

Out of all the genuine transactions (not fraud), how much we predicted wrong (predicted as fraud). It should be low as possible

Precision = TP/(TP + FP) = 91/(91.0 + 83.0) = 0.5229885

Out of all the fraud transactions that we have predicted correctly, how many are actually fraud.

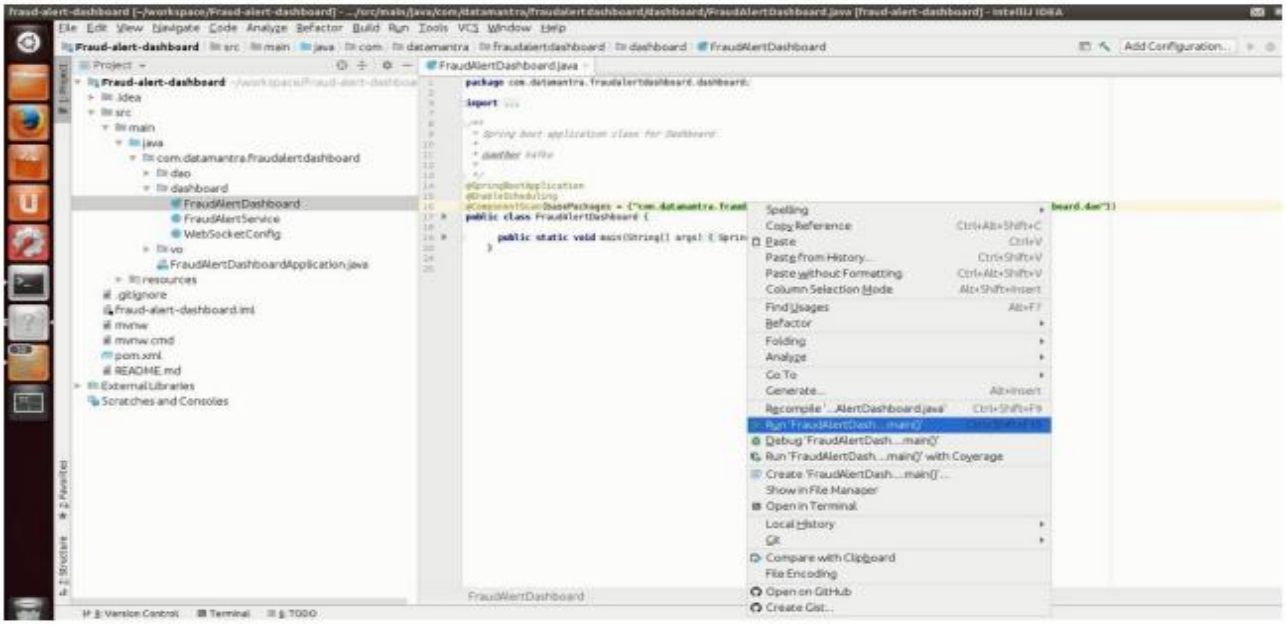
4.2 Spark Streaming

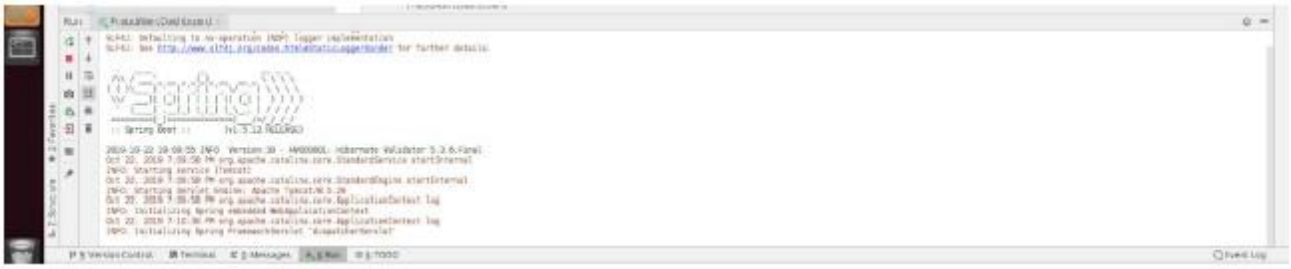
Both the Preprocessing Model and the Random Forest model that were produced by the Spark ML Job will be loaded by the Spark Streaming Job. It will start consuming Kafka messages about credit card transactions.

All the necessary columns (merchant, category, distance, amt, age) will be converted to feature columns using the preprocessing model. It will determine using the Random Forest Model if a transaction is fraudulent or not. Along with saving projected transactions to Cassandra, we are also saving the Kafka offset. Predicted transactions will be stored to Cassandra Key space. Fraud detection requires exactly-once semantics, which means that transactions cannot be lost and cannot be duplicated. By manually committing offset, we are able to achieve exactly-once semantics in this case.

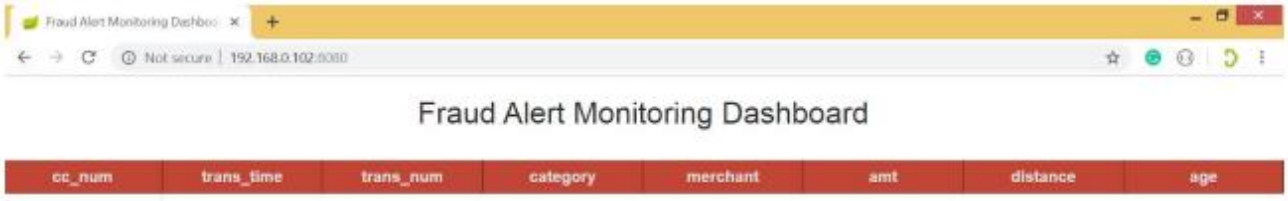
Run Fraud-alert-dashboard project from IntelliJ

The fraud alert dashboard web server is up and running. We verify this by accessing through the web browser

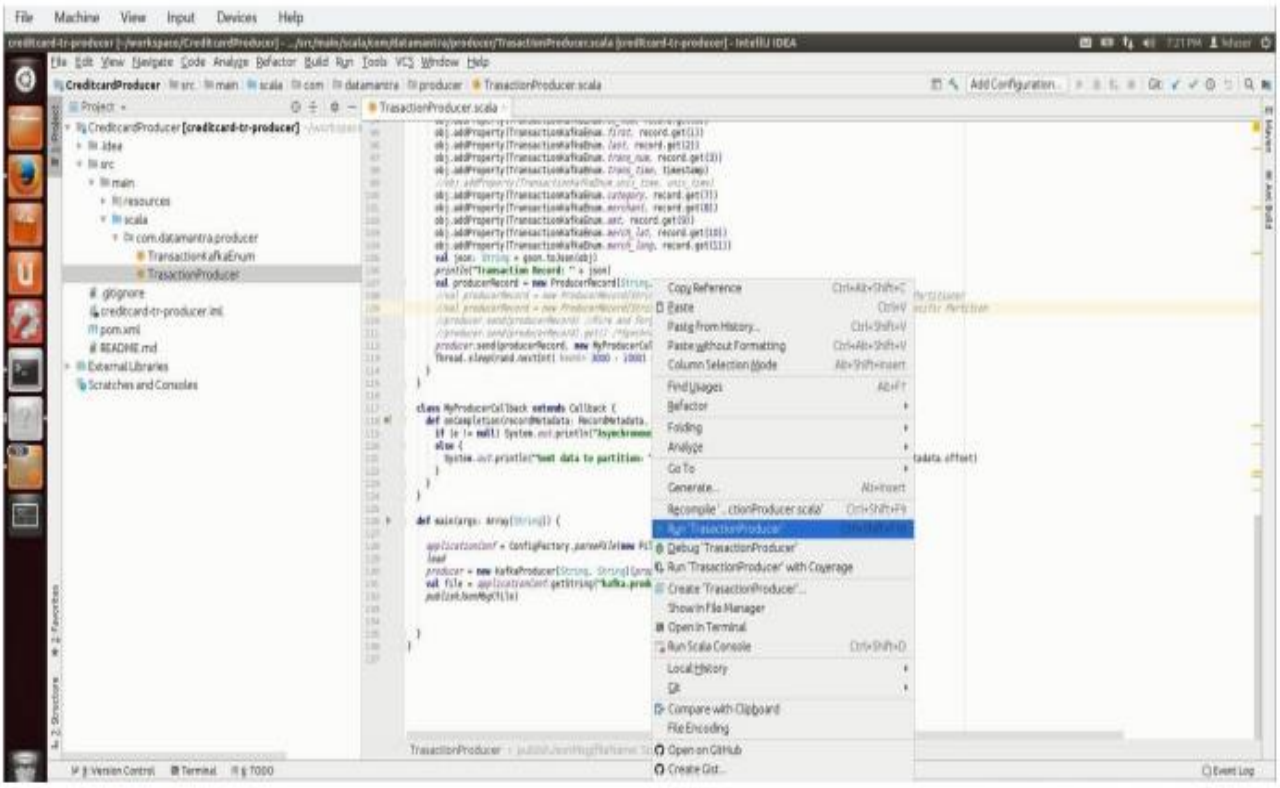




http://<ubuntu-ipadd>:8080



4.3 Run Spark Streaming Job from Intellij



Output

Whenever Spark Streaming Job predicts a fraud transaction it will be displayed on Fraud Alert Dashboard.

4.4 Airflow Automation

A framework for automation is Apache Airflow. Jobs are automated using it. In place of XML workflows, Python can be used to write workflows. Here, both the Spark Streaming and Spark ML jobs are automated. However, a Spark ML Job is manually executed to construct a Model in a production context. The model's effectiveness will be assessed. The Model will be implemented if it is effective. We need to rebuild the model utilizing both the old data and the new data as fresh transaction data is recorded. Therefore, a new model will often be developed once a week or once a month. However, we execute the Spark ML Job every 20 minutes for the sake of demonstration. Upon creation of the new model, Spark Streaming Job must select

4.4.1 Create Airflow database and exit

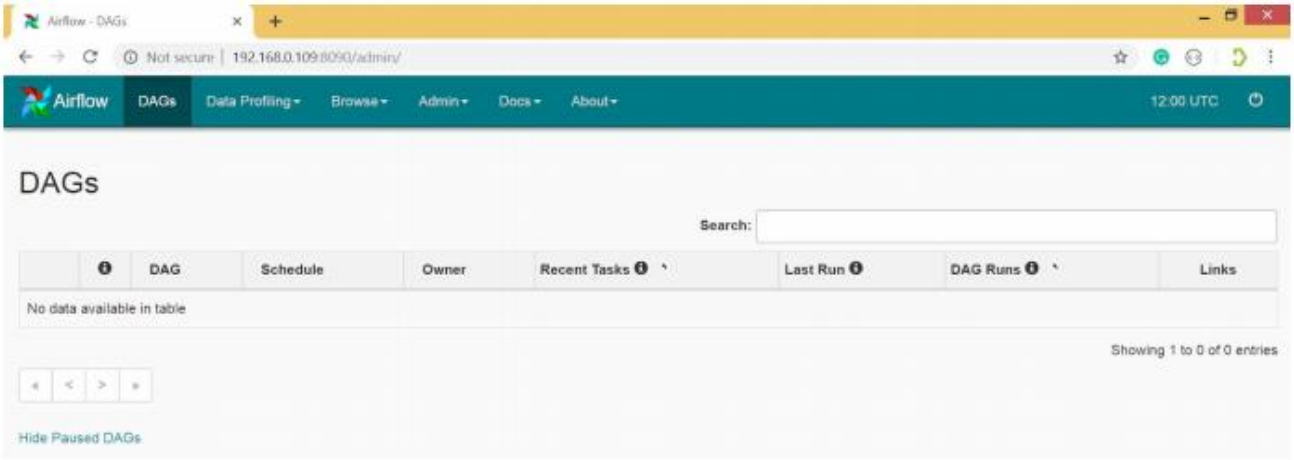
```
CREATE DATABASE airflow;
EXIT;
```

4.4.2 Start Airflow Webserver

```
airflow webserver --port 8090
```

4.4.3 Access Airflow Web Server

Access `https://<my_ipaddress>:8080`to verify whether the airflow web server is up and running



Airflow is configured successfully

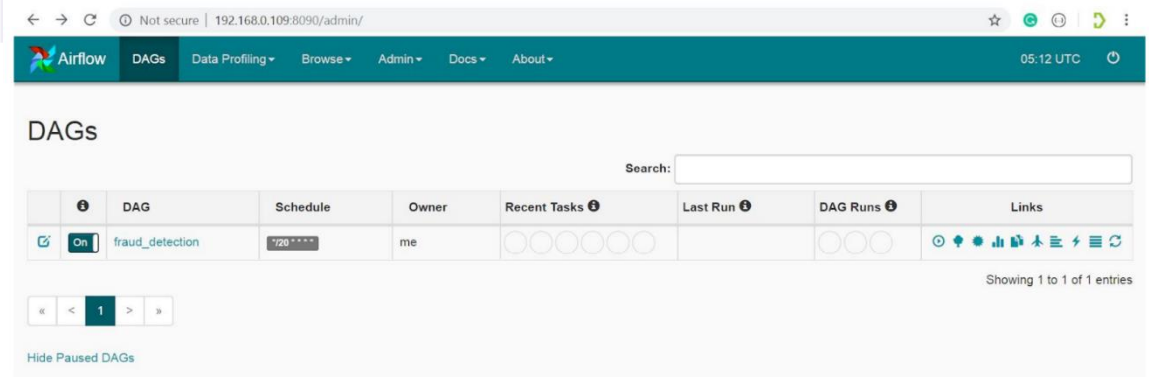
4.4.4 Automation Steps

1. **Remove Old Model:** This training folder contains Preprocessing Model and Random Forest Model. We are just deleting the folder.
2. **Create New Model:**Here we are creating a new model by running Spark ML Job through spark-submit command.
3. **Sleep for 5 seconds**
4. **Stop and Start Spark Streaming Job**: Here we are call stopStartStreamingJob function. In this function, first, we are stopping the currently running Streaming Job by creating a dummy file /tmp/shutdownmarker. A thread running in the Streaming Job will be continuously checking for this file. As soon as this file is created, Streaming Job will be shutdown. Next, we are waiting for the Streaming job to shutdown. As soon as it is shut down, we are deleting the dummy file so that newly started Streaming job must not get shutdown. Finally, we are starting the new Streaming job using the spark-submit command.

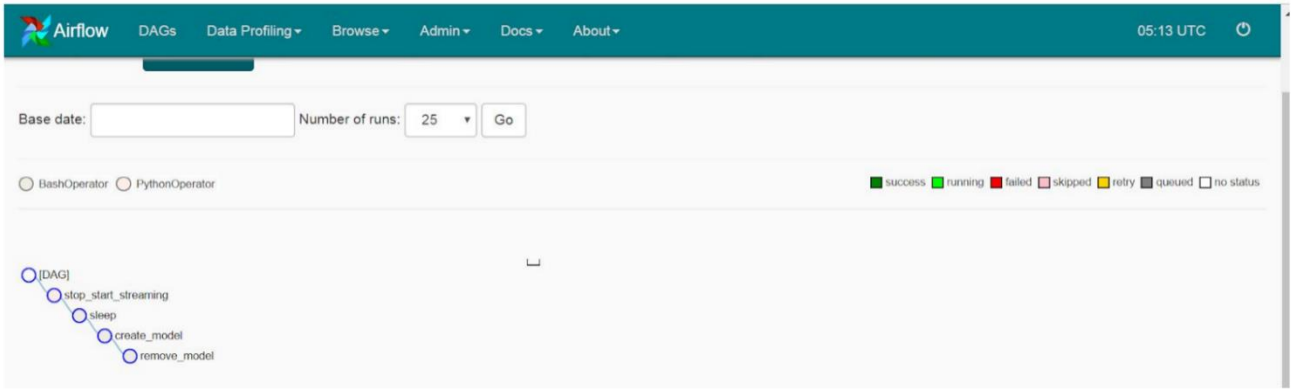
4.4.5 Create an Airflow DAG directory

Automation scripts would be kept in this directory. Airflow will look for scripts in this directory and it will load the scripts to the web server. So we copy frauddetection.py file from FraudDetectionProject to airflow dags directory.

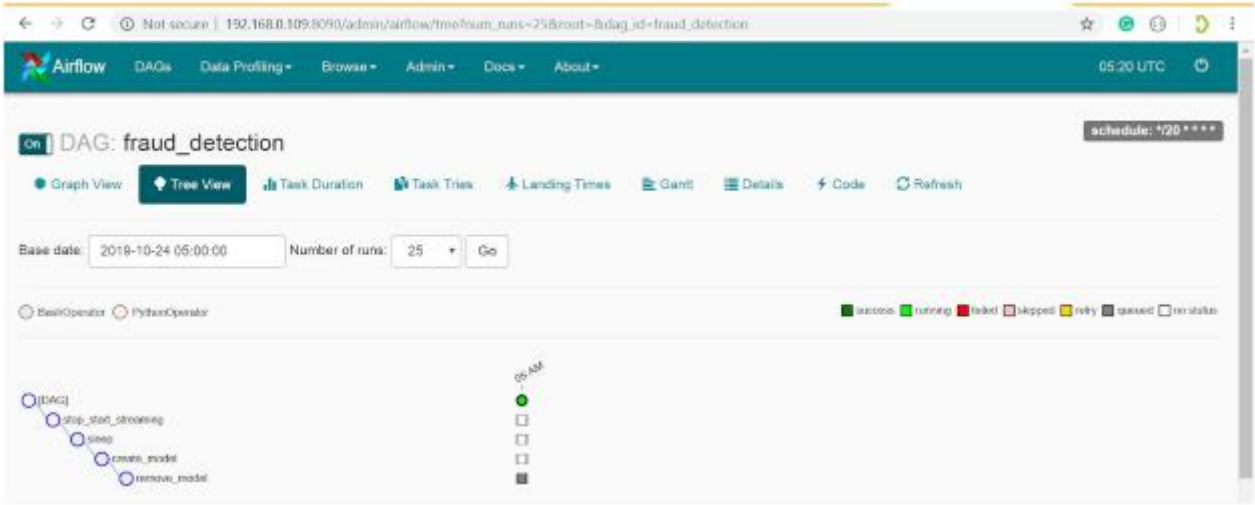
```
mkdir ~/airflow/dags
cp ~/workspace/FraudDetection/automate/airflow/frauddetection.py
```



When I click on fraud_detection we see the automation steps.



Airflow Scheduler is the one which will actually kick start the automation.



4.5 Spark Cluster Setup

4.5.1 Start Spark Master

```
start-master.sh
```

4.5.2 Start Spark Slave

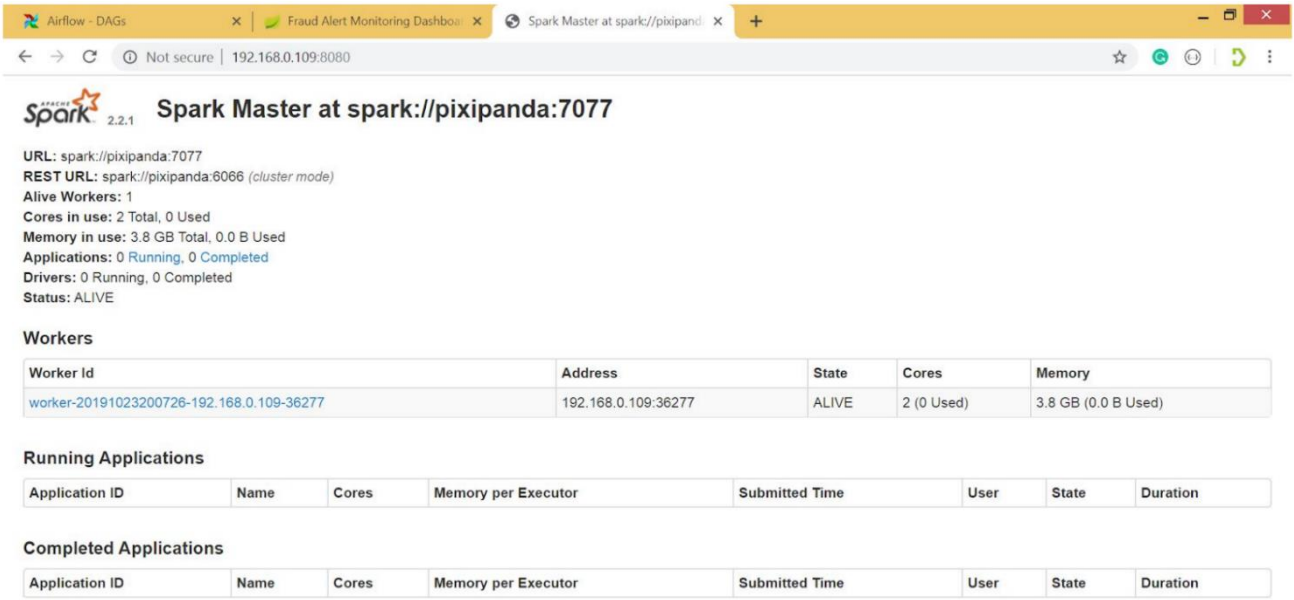
```
start-slave.sh spark://datamantra:7077
```

4.5.3 Access Spark Web UI

Check whether Spark Cluster is up and running

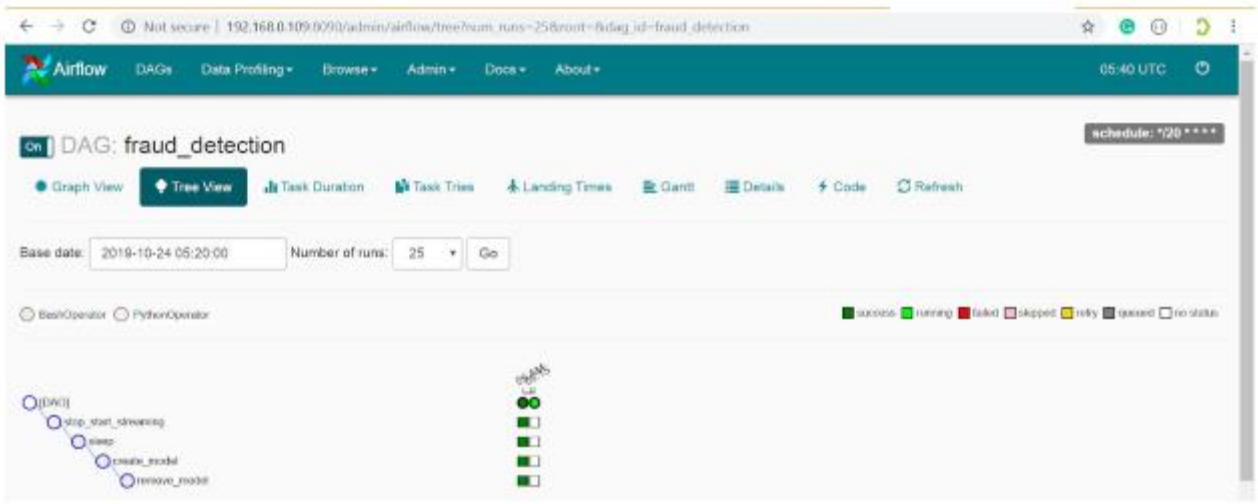
```
http://ubuntu_ip:8080/
```

Output

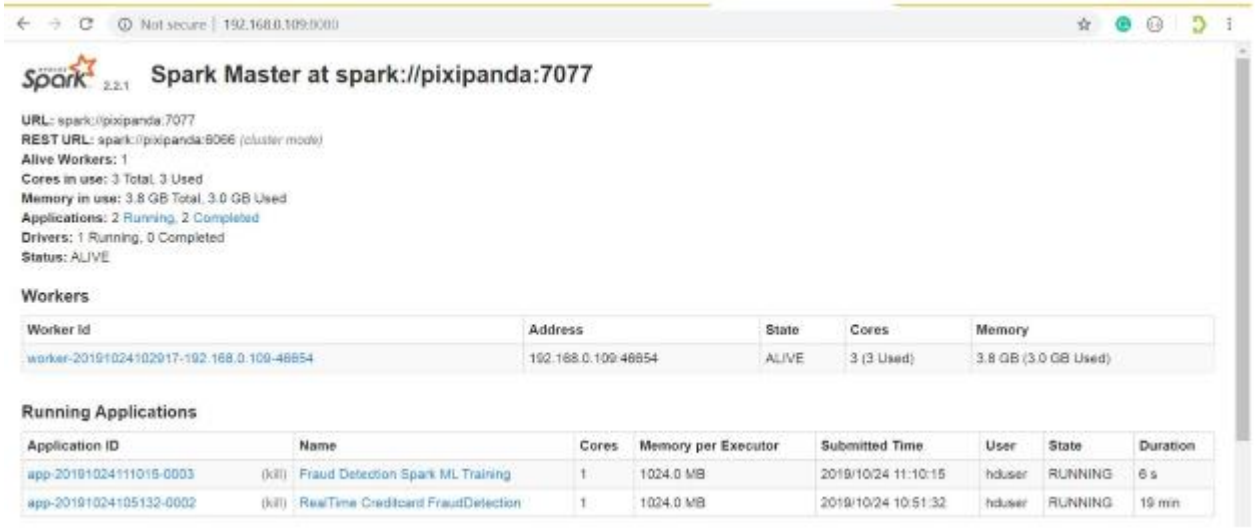

Airflow - DAGs
Fraud Alert Monitoring Dashboa
Spark Master at spark://pixipand
Not secure | 192.168.0.109:8080
Spark 2.2.1
Spark Master at spark://pixipanda:7077
URL: spark://pixipanda:7077
REST URL: spark://pixipanda:6066 (cluster mode)
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 3.8 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE
Workers
Worker Id
Address
State
Cores
Memory
worker-20191023200726-192.168.0.109-36277
192.168.0.109:36277
ALIVE
2 (0 Used)
3.8 GB (0.0 B Used)
Running Applications
Application ID
Name
Cores
Memory per Executor
Submitted Time
User
State
Duration
Completed Applications
Application ID
Name
Cores
Memory per Executor
Submitted Time
User
State
Duration

4.6 Next set of automation

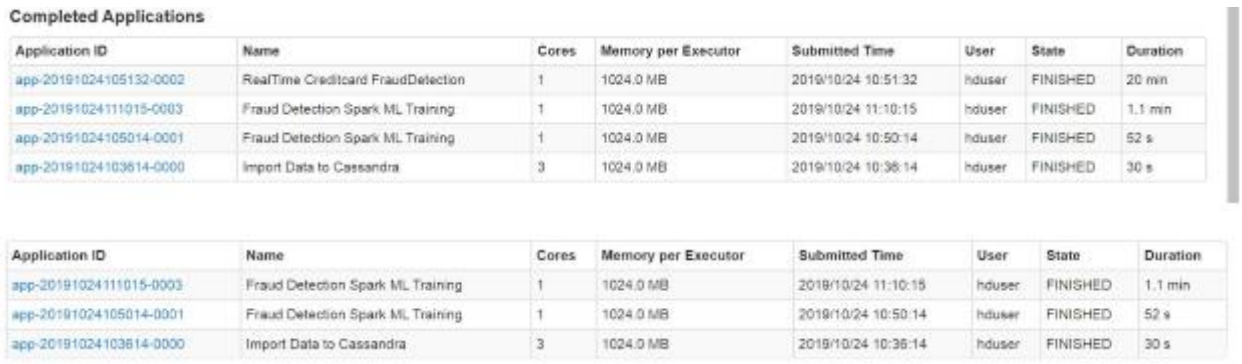
Next set of automation steps scheduled to start every 20 mins



Spark ML Job is running This will create new model

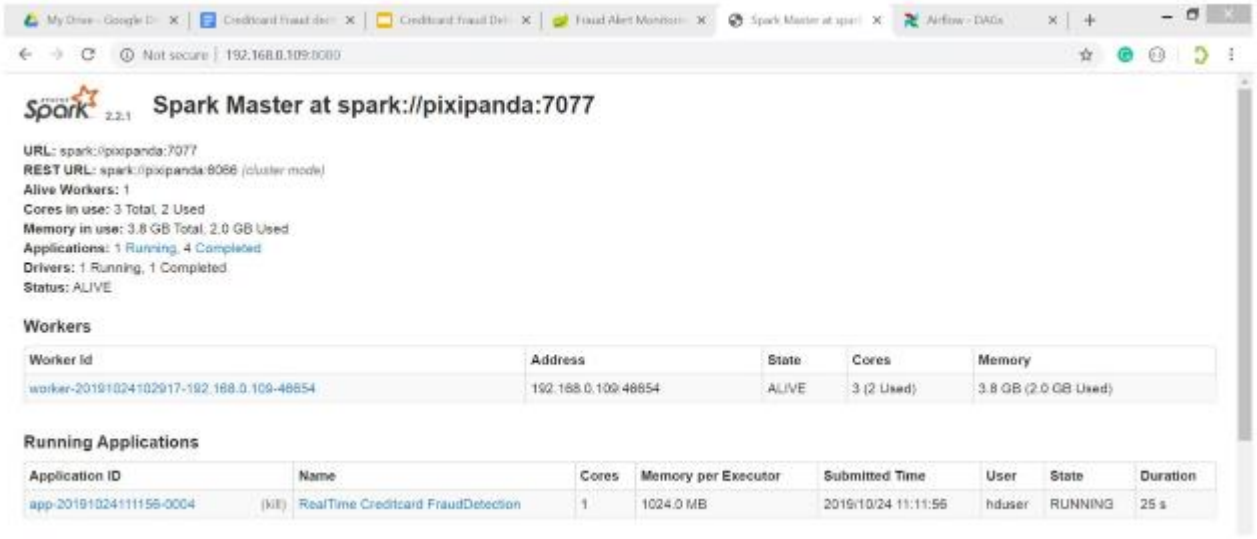


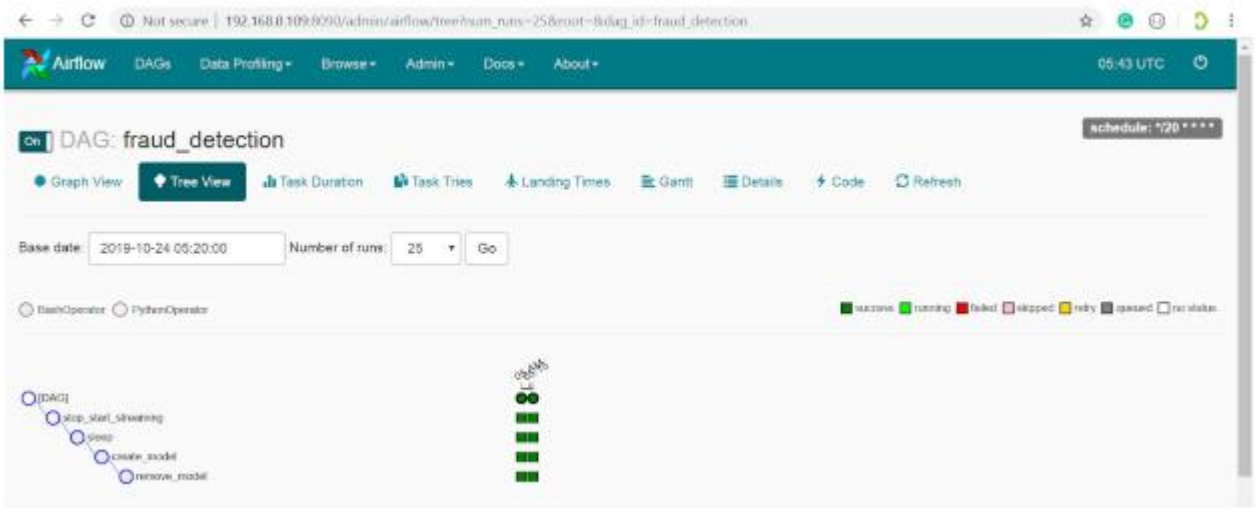
Spark Streaming Job Stopped



New Streaming Job Started

This will load the new model created by the previous Spark ML Job. Transactions will be predicted using this new model





CHAPTER 5 SUMMARY AND CONCLUSION

A framework for automation is Apache Airflow. Jobs are automated using it. In place of XML workflows, Python can be used to write workflows. Here, both the Spark Streaming and Spark ML jobs are automated. However, a Spark ML Job is manually executed to construct a Model in a production context. The model's effectiveness will be assessed. The Model will be implemented if it is effective. We need to rebuild the model utilizing both the old data and the new data as fresh transaction data is recorded. Therefore, a new model will often be developed once a week or once a month. However, we execute the Spark ML Job every 20 minutes for the sake of demonstration. Upon creation of the new model, Spark Streaming Job must select

REFERENCES

Aman, A. M. (2021, july). Credit Card Fraud Detection using Machine Learning and Data Science. *International Journal for Research in Applied Science & Engineering Technology (IJRASET), 9*(vii).

Amanze, B. C., & Onukwugha, C. G. (2018, DECEMBER). AN ENHANCED MODEL FOR BANK FRAUD DETECTION IN NIGERIAN. *INTERNATIONAL EDUCATIONAL JOURNAL OF SCIENCE AND ENGINEERING (IEJSE), 1*(5).

Apache. (n.d.). *Spark ML Programming Guide - Spark 1.2.2 Documentation (apache.org)*. Retrieved from https://spark.apache.org/docs/1.2.2/ml-guide.html

Apache. (n.d.). *Structured Streaming Programming Guide - Spark 3.2.1 Documentation (apache.org)*. Retrieved from https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

Bagui, S., & Kunqi , L. (2021). Resampling imbalanced data for network intrusion detection datasets. *Journal of Big Data*.

Brownlee, J. (2019, December 23). A Gentle Introduction to Imbalanced Classification. *Imbalanced Classification*.

Chuprina, R. (2021, feburary 25). Credit Card Fraud Detection: Top ML Solutions in 2021.

Imbalanced Learning: Foundations, A. a. (2013). Imbalanced Learning: Foundations, Algorithms, and Applications. In *Imbalanced Learning: Foundations, Algorithms, and Applications* (p. 16).

Jesus, A. d. (2019, November 22). Machine Learning for Credit Card Fraud – 7 Applications for Detection and Prevention. *Discover the critical AI trends and applications that separate winners from losers in the future of business.*

Lakshmi, M. G., Mettu , V., & Hameed, K. (2020). Credit Card Fraud Detection Using Random Forest. *Journal of Information and Computational Science, 10*(3). Retrieved from www/joics.org

Lamba, H. (2020). *CREDIT CARD FRAUD DETECTION IN REAL TIME.* thesis, CALIFORNIA STATE UNIVERSITY SAN MARCO.

ModelingAppliedPredictive. (2013). *Applied Predictive Modeling.*

PROSHARE. (2020, AUGUST 31). Fraudulent E-transactions Involving Credit and Debit Cards: Who is Liable? - A Case Study.

Sarah , G., & Muller, A. C. (2017). *Introduction to Machine Learning.* United States of America: O'Reilly Media, Inc.

Shakya, R. (2018). *Application of Machine Learning Techniques in Credit Card Fraud detection.* University of Nevada, Las Vegas, Department of Computer Science.

Spark, A. (n.d.). *Apache Spark™ - Unified Engine for large-scale data analytics*. Retrieved from https://spark.apache.org/

Spark, A. (n.d.). *Sql Programming Guide*. Retrieved from https://spark.apache.org/docs/latest/sql-programming-guide.html

TutorialsPoint. (2015). *Machine learning with python.* Tutorials Point (I) Pvt. Ltd.

Vaishnave Jonnalagadda, P. G. (2019). Credit card fraud detection using Random Forest Algorithm. *International Journal of Advance Research, Ideas and Innovations in Technology*, *5*(2). Retrieved from www.ijariit.com

Vaishnavi Nath Dornadulaa, G. S. (2019). Credit Card Fraud Detection using Machine Learning Algorithms . *INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ADVANCED COMPUTING 2019, ICRTAC 2019* .

Wikipedia. (2021, June 23). Credit card fraud. *the free encyclopedia*.