**19bce245 Aayush Shah**

**19bce246 Akshat Shah**

**19bce247 Devansh Shah**

**Big Data Analytics**

**Innovative Assignment**

**Topic:**

**CREDIT CARD FRAUD DETECTION USING APACHE SPARK**



* **Introduction:**
  + **Credit Card**

Credit card is a tiny rectangular piece of plastic or metal supplied by a bank or financial services company that allows cardholders to borrow funds and use them to pay for goods and services at businesses who accept credit cards. Your credit card is accepted in millions of locations throughout the world, so you may use it anywhere. Using your credit card overseas, however, is usually subject to penalties or charges. When you apply for a credit card, you will be given a credit limit. This is the total amount of money you have available to spend on your credit card. Your statement will include information on all of your transactions, as well as: 1) The minimum amount to be paid. 2) The Last Date of paying at least the minimum amount. If you do not pay off your debt in full each month, you will be charged interest on the amount left on your account.

* + **Fraudulent Transactions**

The purpose might be to get things without paying for them or to remove money from an account without permission. Fraudulent transactions are orders and purchases made with a credit card or bank account that does not belong to the buyer. One of the most popular kinds of identity fraud, these transactions can hurt both businesses and identity fraud victims. Avoiding fraudulent transactions benefits both businesses and buyers, thus appropriate protections must be implemented while handling money accounts.

* + **Fraud Detection**

Fraud detection is monitoring user behaviour in order to anticipate, recognise, and avoid unfavourable behaviour. It is critical to understand the technologies involved in detecting and recognising various sorts of credit card thefts in order to prevent credit card fraud. Credit card transactions have increased significantly as a result of globalisation and increased internet use for online purchasing. As a result of the fast rise of credit card transactions, fraudulent activity has increased significantly. Credit card fraud refers to theft and fraud involving a credit card as a fraudulent source of funds in a transaction. Credit card thieves utilise a range of tactics to commit fraud. To successfully prevent credit card theft, it is vital to first understand how credit card fraud is detected. Credit card fraud has mostly stabilised over the years as a result of various credit card fraud detection and prevention technologies. In recent years, many have been concerned about credit card fraud detection methods based on data mining. Because our problem is addressed as a classification task, traditional strategies aren’t immediately relevant. As a result, a new technique is employed, which involves GBT Classifier, Local Outlier Factor and Isolation Forest Algorithms The purpose of this study is to identify an eclectic existing algorithms for a credit card fraud detection system based on machine learning algorithms and create system design for that algorithm. Iterative algorithms, aim to improve solutions over time. When a credit card is cloned, stolen, or lost, fraudsters usually use it until the allowable limit is surpassed. As a result, rather than measuring the number of correctly categorised transactions, a mechanism is being developed to restrict the overall allowable limit for fraud-prone cards. The purpose of this algorithm is to minimize the incorrect number of deceitful detection and detect maximum number of fraud transactions. Parameters with interval values have been fine-tuned

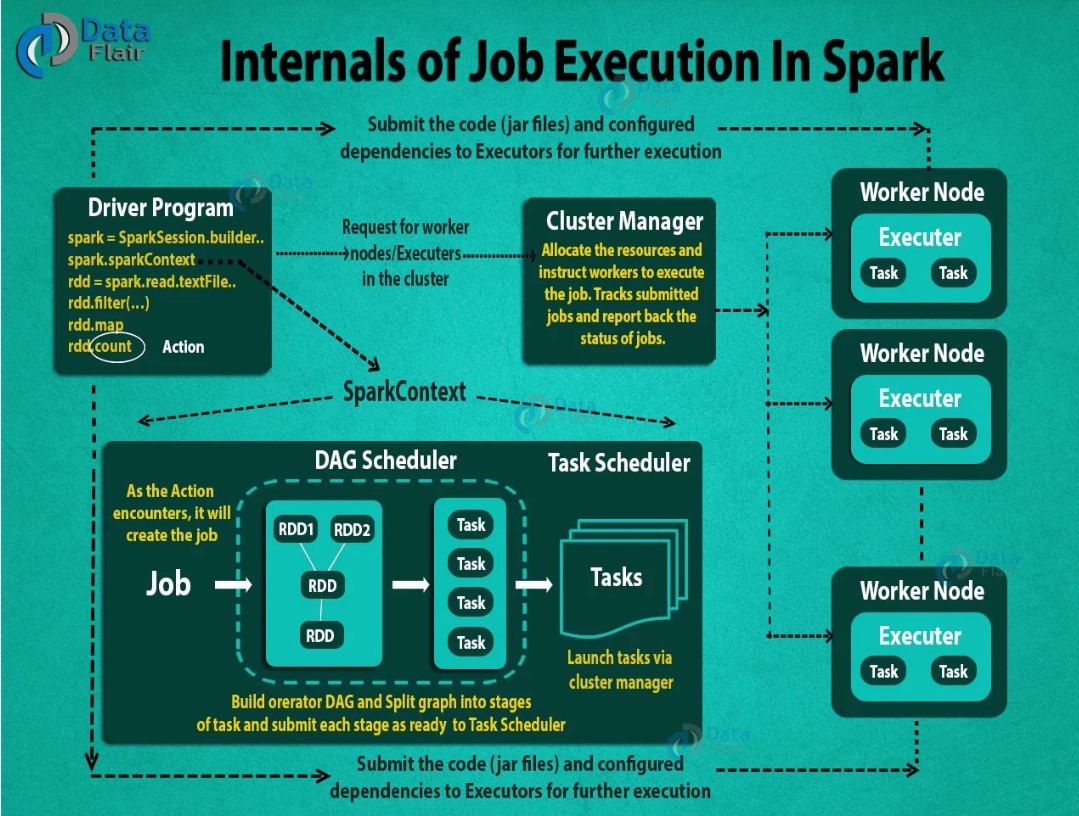
* **Existing Work**

With the help of one of the big data tool i.e. Apache Spark, people are taking the benefit of it and research on this topic has been done with the help of linear regression, logistic regression, Decision tree, Random Forest Classifier and all the other models that can be used in machine learning and people are trying to get as accurate results as possible in shorter span of time thanks to spark which makes our work faster.

* **Problem Statement**

We know that in day-to-day life there is myriad increase in transactions of credit card and henceforth this issue involves customizing former transactions of credit card with the comprehension of the ones that came out to be fraudulent. Now, the best selected model is then utilized to recognize whether a new transaction occurred is deceitful or not. So, our main aim here is to detect maximum number of deceptory transactions. Here we will see few methods with different approach and will try to achieve the goal of detecting maximum number of treachery transactions and try to find with the help of model regardless the transaction took place is untrustworthy or not.

* **Apache spark working:**

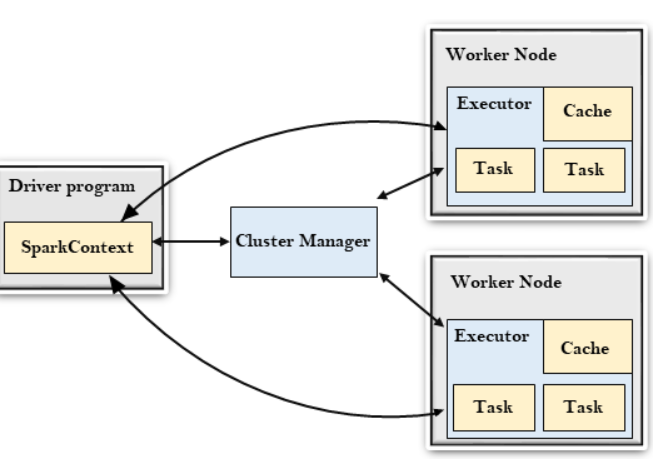


Apache Spark is a general-purpose, open-source distributed computing engine used for processing and data analysis. It works with the system to distribute data across the cluster and process the data in parallel, much as Hadoop MapReduce. One central coordinator and numerous dispersed workers make up the master/slave architecture used by Spark. The driver is the term used here for the central coordinator.

Running in its own Java process is the driver. These drivers interact with the executors, a class of potentially numerous distributed workers. A separate Java process represents each executor. A driver and its own executors are combined to form a Spark Application. A Spark Application is started on a number of machines with the assistance of a cluster manager. The built-in cluster manager by default for Spark is called Standalone Cluster Manager. Spark works with several open source cluster managers, including Hadoop Yarn, Apache Mesos, and others, in addition to its built-in cluster manager.

1. **Driver**: The program's main() method is executed by the driver. The process that executes user code to produce RDDs, carry out transformation and action, and also produce SparkContext is known as the driver. We have developed a driver programme when the Spark Shell starts to run. The application is completed when the driver is terminated.

The executor will run the tasks that the driver programme divides up from the Spark application. The driver is home to the job scheduler, which assigns tasks to the workforce. There are two primary roles for drivers:



user programme into the task conversion.

task planning for the executor. At a higher level, the structure of the Spark programme is as follows: RDDs are made up of some input data, and new RDDs are created from existing ones using a variety of transformations. The DAG (directed acyclic graph) of operations is created implicitly in the Spark program. The driver then transforms that Spark DAG into a physical execution plan when it is running.

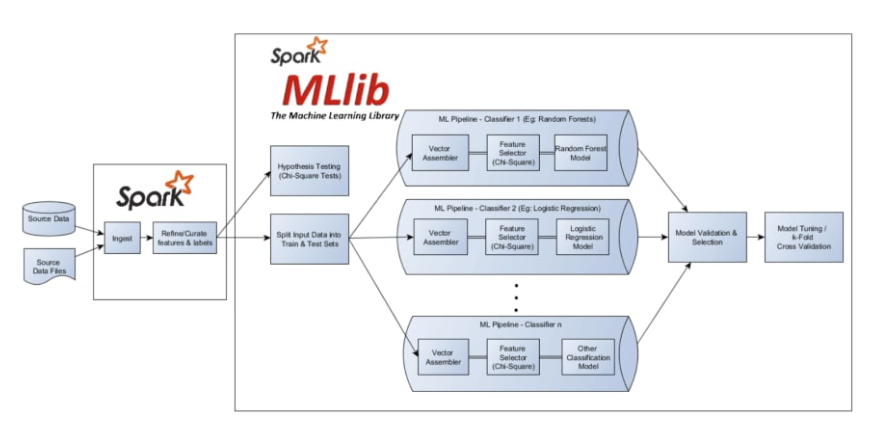
1. **Cluster Manager:**

Spark depends on cluster management for the execution of executors, and in some circumstances, drivers as well. It can be plugged into Spark as a component. Jobs and activity within a Spark application are FIFO-scheduled by Spark Scheduler on the cluster manager. As an alternative, round robin schedule can also be used. Depending on the workload, a Spark application's resource usage may change dynamically. As a result, the application can release resources that aren't in use and then request them once more when necessary. This is accessible on all coarse-grained cluster managers, including Mesos coarse-grained mode, YARN mode, and standalone mode.

1. **Executors**: Each task in the specified Spark job is executed by a Spark executor. Executors launch once at the start of a Spark application and continue to operate throughout the duration of the programme. The Spark application can continue even if the Spark executor crashes. The executors' two primary functions are as follows:

runs the application's component tasks and provides the driver with the results.

Give the user-cached RDDs access to in-memory storage.



* **Learning Outcomes**:

Instead of addressing the challenges posed by dispersed data, data scientists may concentrate on their data problems and models thanks to the Apache Spark machine learning library (MLlib). Using in memory computing and other improvements, Spark can process massive amounts of data 100 times quicker than Hadoop. When data is kept on a disc, Spark is also quick. Spark employs random access memory (RAM) rather than reading and writing intermediate data to discs, making it faster.