**Analyzing Aadhaar Dataset using MapReduce & Spark**

**Problem Statement:**

To analyze the Aadhaar data using Hadoop and Spark to extract meaningful knowledge for the purpose of better decision-making by the Central and State Governments of India.

**Background:**

The proposed system concentrates on analyzing Aadhaar related data using Hadoop and Spark for the purpose of better decision making by the Government of India via a comparison between MapReduce and Spark in terms of execution and processing rate. This project work proposes the inconsistencies and fluctuations in enrollments based on demographics, ages, time, and the reaction of State and Central Governments and habitants of states and country.

**Methodolgy:**

**Step 1: Data Collection and Dataset Preparation**

The required data set is collected from the government web portal. The collected data set is loaded into Hadoop Distributed File System (HDFS) environment. The collected data set might consist of missing values and noisy data. If analysis is performed on this data, it may lead to wrong results. So to avoid this, data pre-processing is done on the data set.

**Step 2: Data Processing and Deployment**

Once the dataset is handy, the Hive query has been looked-in to count the number of Unique Identities generated in each state. As Hive runs MapReduce jobs internally, it has been walkthrough the MapReduce code that is required to compute the state-wise identities that are generated. To get an idea on how Hive executes queries internally using MapReduce, two MapReduce jobs have been created similar to Hive execution. The first MapReduce job should count the number of Aadhaar identities generated for each state. To sort the output in descending order, another MapReduce job has been required. So, Mapper, Reducer, Comparator and other classes have been deployed on Java and MapReduce job has been executed using .jar file exported from Java Eclipse. Coming to the Scala part, dataset is read using the databricks spark csv library, data is then registered as a temp table. Now, using HiveContext, the Hive metastore has been accessed so that Hive tables could be read, created and inserted from Spark. Dataframe provides its own domain specific language and also supports SQL queries. All queries on a dataset return a dataframe type. This dataframe can then be saved as table into Hive. From this, comparison can be easily inferred between MapReduce and Spark in terms of execution and processing rate.

**Step 3: Data Analysis and Visualization**

Now, the collected data set is ready for data analysis. Descriptive statistics like mean, median, mode, and percentile are applied. After the data analysis, the analyzed results need to be visualized. Tableau can be used for this purpose. Bar charts, Line charts and Pie charts are generated along with the table format.

**Project Outcomes:**

With the expansion of Aadhaar schemes, thousands of cases reported by the people themselves related to the UID duplicacy as there were no strict regulations and guidelines imposed to the 3rd party biometric-tech providers which were then focused on making profits at-par leading to the inconsistencies and fluctuations in the schemes. Therefore:

* Governments can immediately take corrective measures for the issues found in the Aadhar card related data analysis like Aadhaar duplicacy in UIDAI database.
* Central and State Governments can take necessary precaution measurements to avoid the issues related to the Aadhaar enrollments in the future.