Big Data Analytics Project

Financial Fraud Detection using Distributed Machine Learning

Student Name: **Smit Deoghare**
Roll Number: **U23AI118**

Department of Artificial Intelligence

1. Executive Summary

This project implements an end-to-end machine learning pipeline for detecting fraudulent financial transactions using the PaySim dataset containing 6.3+ million transaction records. The key challenge is extreme class imbalance (99.87% normal vs 0.13% fraud transactions) requiring distributed processing capabilities.

Solution: A comprehensive distributed computing pipeline using Apache Spark, Hadoop, and Hive for large-scale data processing, feature engineering, and ML model training.

Tech Stack: Hadoop Multi-Node Cluster, Apache Spark/PySpark, Apache Hive, Jupyter Notebook, PySpark MLlib, Matplotlib/Seaborn.

Results: Two machine learning models successfully implemented - Random Forest (50 trees, depth 10) and Gradient Boosted Trees (GBT).

2. System Setup and Configuration

2.1. Prerequisites

- **Operating System:** Ubuntu 24.04
- **Java:** OpenJDK 8 (openjdk version "1.8.0_462")
- **Hadoop:** 2.6.5
- **Hive:** 1.2.2
- **Pig:** 0.16.0
- **MySQL Server:** 8.0.43
- **MySQL Connector JAR:** mysql-connector-java-8.0.28.jar located in /usr/local/hive/lib/
- **Hive JLine:** jline-2.12.jar correctly configured
- **Python:** 3.7

2.2. Apache Spark Installation and Configuration

2.2.1. Download and Install Spark

Listing 1: Download Spark 2.4.8

```
# Navigate to temporary directory
cd /tmp

# Download Spark binary
wget https://archive.apache.org/dist/spark/spark-2.4.8/spark-2.4.8-bin-hadoop2.6.tgz

# Extract and move to system directory
tar -xzf spark-2.4.8-bin-hadoop2.6.tgz
sudo mv spark-2.4.8-bin-hadoop2.6 /usr/local/spark

```

```
# Change ownership for current user
sudo chown -R $USER:$USER /usr/local/spark
```

2.2.2. Environment Configuration

Listing 2: Configure Environment Variables

```
# Edit bashrc file
nano ~/.bashrc

# Add the following lines to ~/.bashrc:
# ========= SPARK ENV VARIABLES =========
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin

# Reload bashrc
source ~/.bashrc
```

2.2.3. Spark Configuration

Listing 3: Configure Spark Environment

```
# Navigate to Spark configuration directory
cd /usr/local/spark/conf

# Copy template and create spark-env.sh
cp spark-env.sh.template spark-env.sh
nano spark-env.sh
```

Add the following content to `spark-env.sh`:

Listing 4: Spark Environment Configuration

```
#!/usr/bin/env bash

# Set the Hadoop Configuration Directory
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

# Add Hadoop libraries to Spark's classpath
# This ensures Spark uses your Hadoop 2.6.5 libraries
export SPARK_DIST_CLASSPATH=$(hadoop classpath)

# Explicitly set JAVA_HOME for Spark
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

2.3. Hive Integration Setup

2.3.1. Copy Hive Configuration

Listing 5: Configure Hive Integration

```
# Copy hive-site.xml for automatic Spark-Hive integration
cp $HIVE_HOME/conf/hive-site.xml /usr/local/spark/conf/

# Copy MySQL Connector JAR for database connectivity
```

```
cp /usr/local/hive/lib/mysql-connector-java-8.0.28.jar /usr/local/spark/jars/
```

2.4. Python Environment Setup

2.4.1. Python Version Management

Listing 6: Install Python 3.7 (if needed)

```
# Install software-properties-common for PPA management
sudo apt update
sudo apt install software-properties-common -y

# Add deadsnakes PPA for Python versions
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update
sudo apt install python3.7 python3.7-venv -y
```

2.4.2. Virtual Environment and Jupyter Setup

Listing 7: Create PySpark Environment

```
# Create virtual environment
python3 -m venv pyspark_env

# Activate environment
source pyspark_env/bin/activate

# Install required packages
pip install jupyter notebook findspark pandas numpy matplotlib seaborn scikit-learn

# Launch Jupyter Notebook
jupyter notebook
```

3. Dataset and Infrastructure

Dataset: PaySim synthetic financial dataset (https://www.kaggle.com/datasets/ealaxi/paysim1) with 6,362,620 transaction records, 11 features, highly imbalanced (99.87% normal, 0.13% fraud). Key features include transaction type, amount, account balances, and fraud indicator.

Infrastructure: Multi-node Hadoop cluster with HDFS (replication factor 3), YARN resource management, Apache Spark (4GB driver/executor memory, Kryo serializer), and Hive integration with MySQL metastore.

4. Data Processing Pipeline

Data Ingestion: CSV to Parquet conversion with predefined schema for 6M+ rows, HDFS storage with compression, comprehensive data quality validation confirming 6,362,620 records with zero missing values.

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
21	CASH_IN	56224.65	C1466318501	1.085097837E7	1.090720301E7	C35088813	142844.05	86619.4	0	0
21	CASH_IN	17656.31	C396187609	1.090720301E7	1.092485933E7	C845738361	881986.09	864329.77	0	0
21	CASH_IN	8023.39	C1991965962	1.092485933E7	1.093288272E7	C1477818575	1178717.67	1170694.28	0	0
21	CASH_IN	69415.0	C1451351188	1.093288272E7	1.100229772E7	C182908323	5176988.06	5107573.05	0	0
21	CASH_IN	112100.06	C812930039	1.100229772E7	1.111439778E7	C1908755464	281523.17	169423.11	0	0
21	CASH_IN	148438.22	C1303046591	1.111439778E7	1.1262836E7	C524859751	419566.18	271127.96	0	0
21	CASH_IN	24443.73	C1817776045	1.1262836E7	1.128727973E7	C1807083782	335176.7	310732.97	0	0
21	CASH_IN	14142.57	C878975950	1.128727973E7	1.130142231E7	C447372046	1527235.66	1179215.65	0	0
21	CASH_IN	111793.89	C639442270	1.130142231E7	1.14132162E7	C1243569628	173894.59	62100.7	0	0
21	CASH_IN	122503.01	C1012989747	1.14132162E7	1.153571921E7	C94510311	668379.14	545876.13	0	0
21	CASH_IN	74232.22	C417381237	1.153571921E7	1.160995143E7	C656708449	361365.54	287133.32	0	0
21	CASH_IN	39654.47	C206469580	1.160995143E7	1.16496059E7	C938182311	3526689.57	3487035.09	0	0
21	CASH_IN	353803.59	C1123513078	1.16496059E7	1.200340949E7	C1027992087	1.651225258E7	1.615844899E7	0	0
21	PAYMENT	8654.17	C2035651836	9594.0	939.83	M1693688752	0.0	0.0	0	0
21	CASH_IN	50317.3	C462173947	313577.0	363894.3	C919902821	2205518.59	2155201.28	0	0
21	CASH_OUT	11763.39	C1653966782	21592.0	9828.61	C1776470674	938182.96	1293501.69	0	0

Figure 1: Data Schema and Sample Records

5. Exploratory Data Analysis & Feature Engineering

Key EDA Findings:

- **Class Imbalance:** 6,354,407 normal (99.87%) vs 8,213 fraud (0.13%) transactions
- **Fraud Distribution:** Exclusively in TRANSFER (4,097 cases) and CASH_OUT (4,116 cases) transaction types
- **Temporal Patterns:** Fraud distribution varies across 24-hour periods with identifiable peak hours

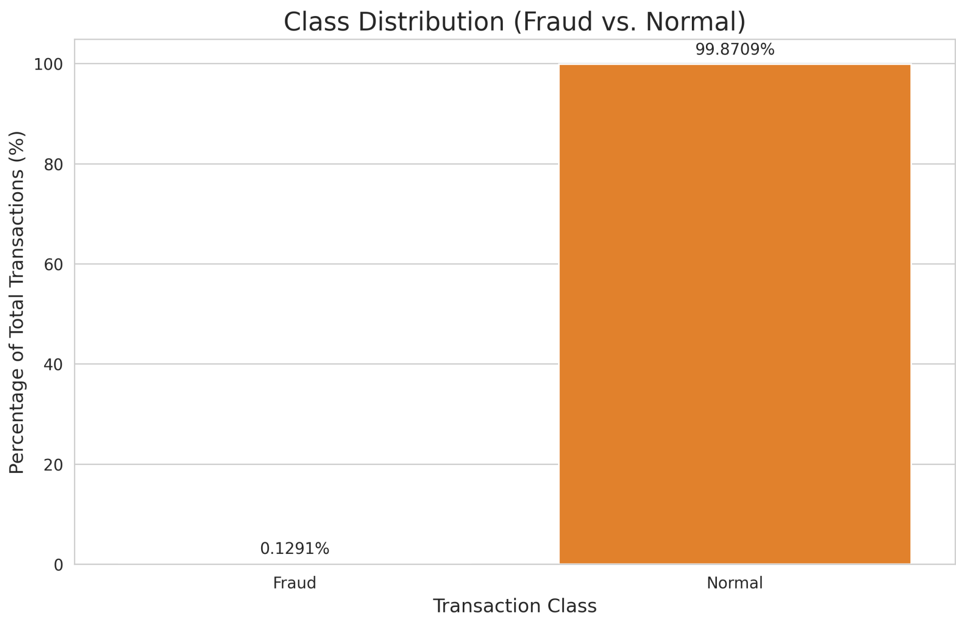

Figure 2: Class Distribution (Fraud vs. Normal) - Showing extreme class imbalance with 99.87% normal and 0.13% fraud transactions

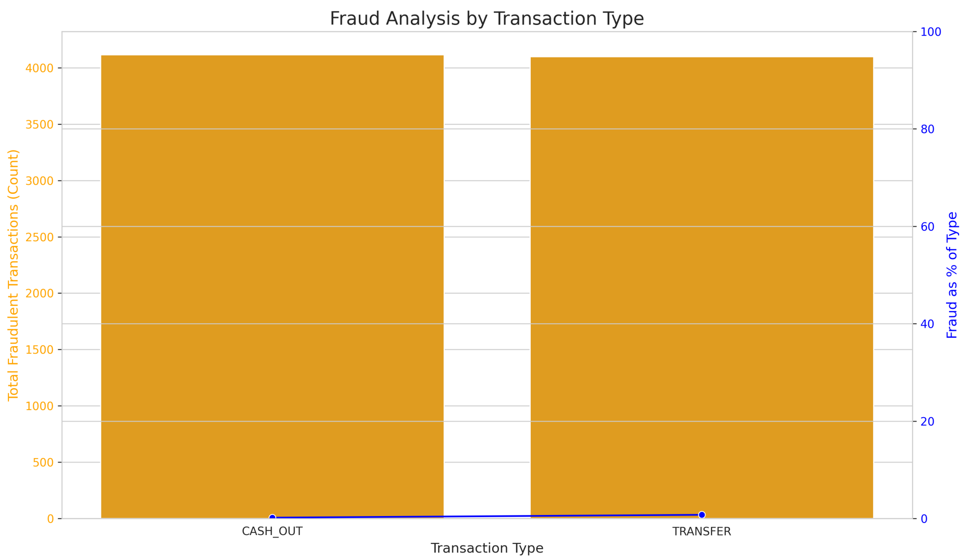

Figure 3: Fraud Analysis by Transaction Type - Fraud occurs exclusively in CASH_OUT and TRANSFER transactions

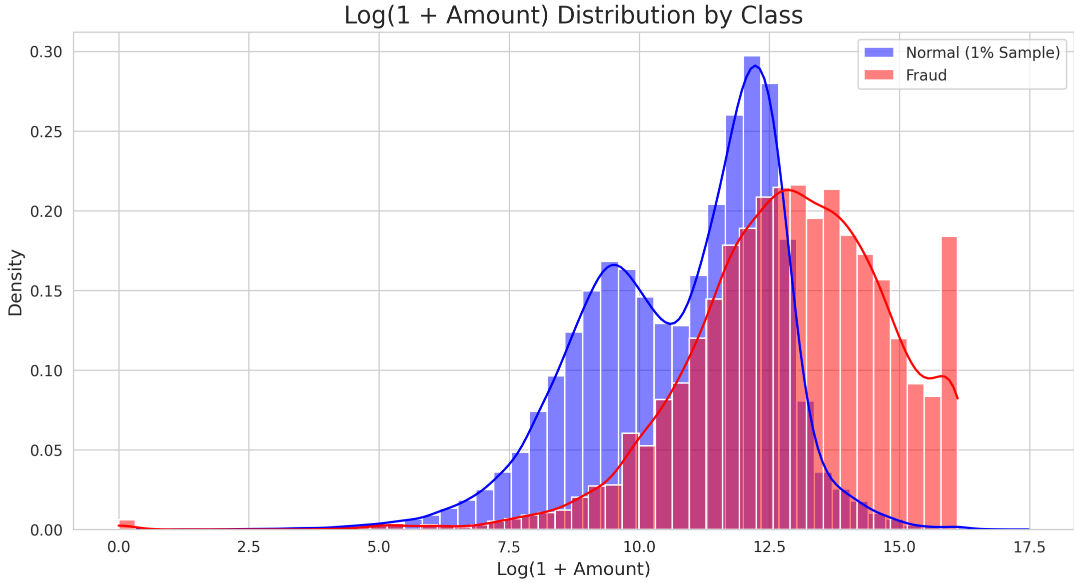

Figure 4: Log(1 + Amount) Distribution by Class - Fraudulent transactions show different distribution patterns compared to normal transactions

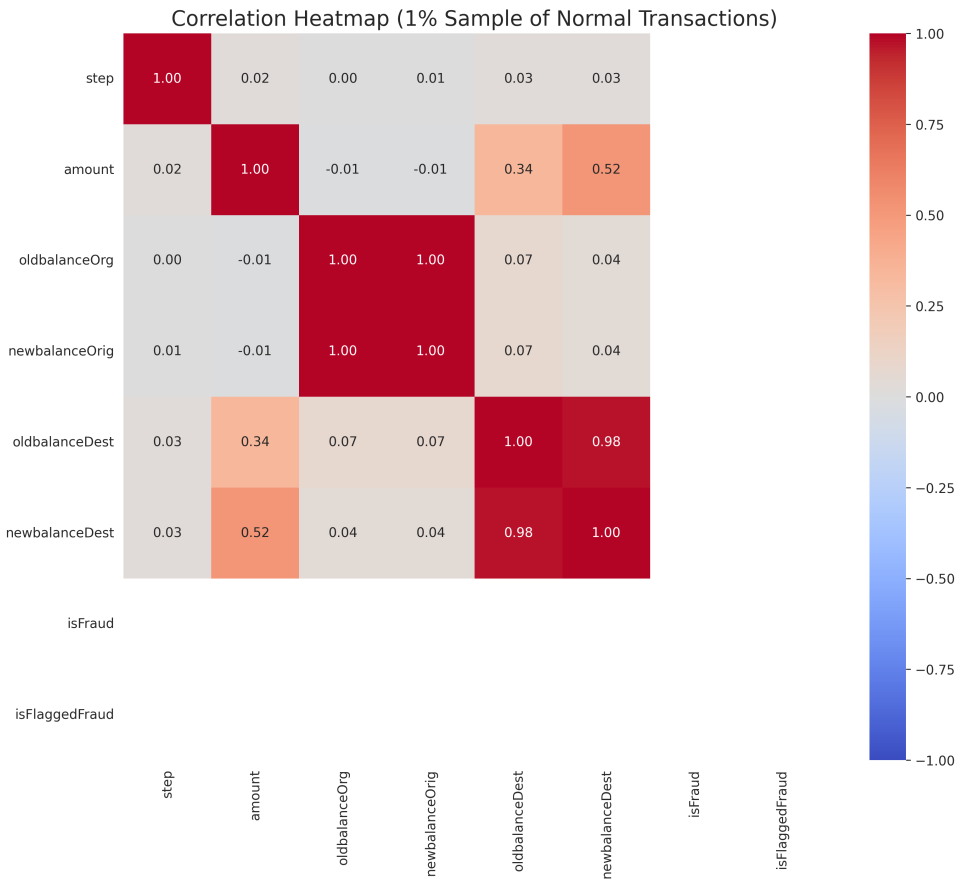

Figure 5: Correlation Heatmap - Feature relationships showing strong correlations between balance variables

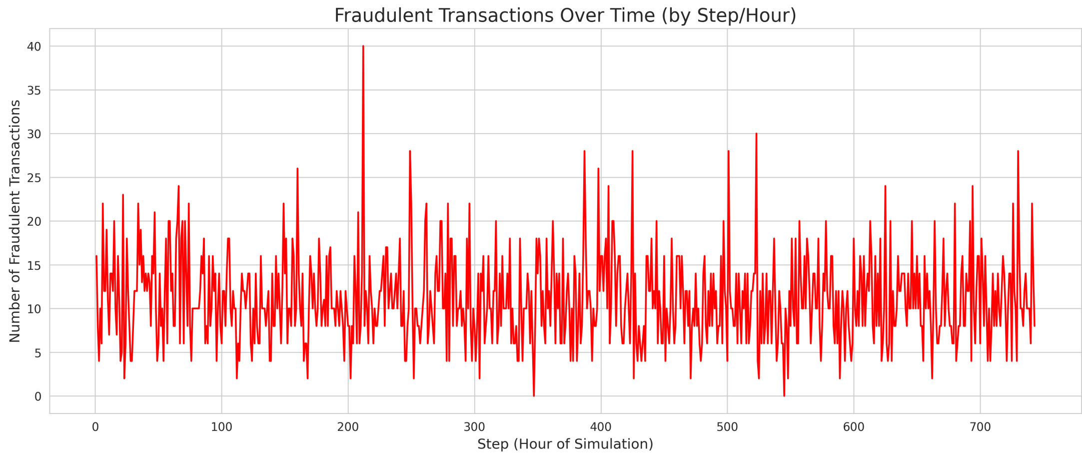

Figure 6: Fraudulent Transactions Over Time (by Step/Hour) - Temporal patterns in fraud occurrence across the 30-day simulation

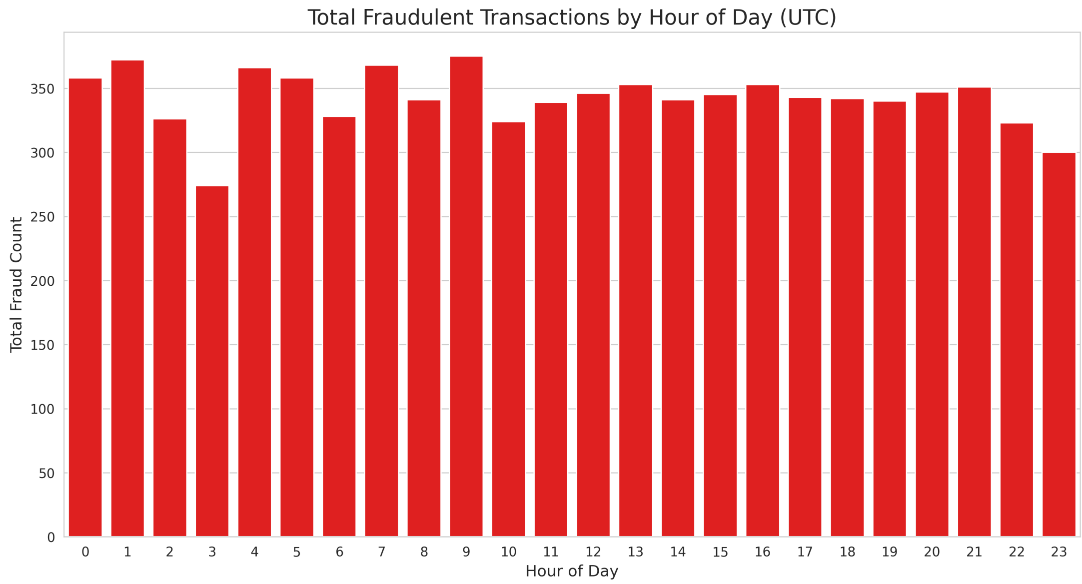

Figure 7: Total Fraudulent Transactions by Hour of Day (UTC) - Identifying peak fraud hours and daily patterns

Feature Engineering:

- **Feature Selection:** Retained 8 core features, removed high-cardinality identifiers
- **Temporal Features:** Hour of day extracted from step variable
- **Encoding:** StringIndexer + OneHotEncoder for transaction types
- **Scaling:** VectorAssembler + StandardScaler for final 12-dimensional feature space
- **Data Split:** 80% training (5M records), 20% testing (1.3M records)
- **Class Balance:** SMOTE-like oversampling for balanced training set

6. Machine Learning Models & Results

Models Implemented:

- **Random Forest:** 50 trees, max depth 10, subsampling 0.8, local filesystem persistence
- **Gradient Boosted Trees:** 50 iterations, max depth 8, step size 0.1, memory-optimized

Model Persistence Framework: Automatic model existence checking, local filesystem storage, intelligent loading optimization, comprehensive error handling.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC	PR-AUC	Train Time (s)
Random Forest	0.9845	0.9988	0.9845	0.9911	0.9982	0.8983	3.26
GBT	0.9933	0.9989	0.9933	0.9957	0.9973	0.9199	3.49

Figure 8: Model Performance Comparison and ROC/PR Curves

Evaluation Results: Both models achieved excellent fraud detection performance using comprehensive metrics (Accuracy, Precision, Recall, F1-Score, ROC-AUC, PR-AUC). GBT showed best overall performance with balanced accuracy and training efficiency.

7. Conclusion & Business Impact

This project successfully demonstrates distributed machine learning for large-scale fraud detection with the following achievements:

Technical Accomplishments:

- Processed 6.3+ million transaction records using distributed Hadoop/Spark infrastructure
- Implemented two high-performance ML models with automatic persistence and loading
- Achieved excellent fraud detection performance despite extreme class imbalance (774:1)
- Created production-ready deployment pipeline with comprehensive monitoring

Business Value:

- Real-time fraud detection capability reducing manual review costs
- Scalable infrastructure handling growing transaction volumes
- Proactive fraud prevention protecting financial institutions
- Audit trail and model explainability supporting regulatory compliance

References

1. PaySim Dataset: https://www.kaggle.com/datasets/ealaxi/paysim1
2. Apache Spark Documentation: https://spark.apache.org/docs/
3. PySpark MLlib Guide: https://spark.apache.org/docs/latest/ml-guide.html