Big Data & AI Course:
Fraud Detection - PySpark MLib

Presented By: Meriem MEKKI

26/01/2026

Introduction

This mini project is part of the Big Data and AI course. The objective of this work is to build a fraud flag classification model for loan applications using Apache Spark to handle large-scale data processing.

The project is based on a financial fraud dataset composed of two linked sources: loan applications data and customer transaction records. The workflow began with Exploratory Data Analysis (EDA). Followed by data preprocessing and feature engineering, including cleaning, encoding categorical variables, scaling numerical features, and joining multiple data sources using a common customer identifier.

Finally, several machine learning classification models were trained and evaluated in a distributed Spark environment to predict fraudulent loan applications.

Dataset Description

This project uses a financial fraud dataset composed of two complementary CSV files:

- **loan_applications.csv**
- Contains information related to loan requests, including:
  - Applicant demographics
  - Financial and employment details
  - Loan characteristics (amount, term, purpose, etc.)
  - A binary fraud_flag indicating whether the loan application is fraudulent
- **transactions.csv**
- Provides historical transaction data for customers, including:
  - Transaction type and amount
  - Merchant and location details
  - Transaction-level fraud_flag

Both datasets share a common customer_id, which allows them to be joined to enrich loan applications with customer transactional behavior.
This structure enables more robust fraud detection by combining static applicant information with dynamic financial activity.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis was conducted using Python libraries (Pandas, Matplotlib) to gain an initial understanding of the data before modeling.

The main EDA steps included:

- Dataset inspection
- Fraud label distribution
- Feature exploration
- Correlation analysis
- Outlier detection
- Data quality checks

Exploratory Data Analysis (EDA)

Dataset inspection

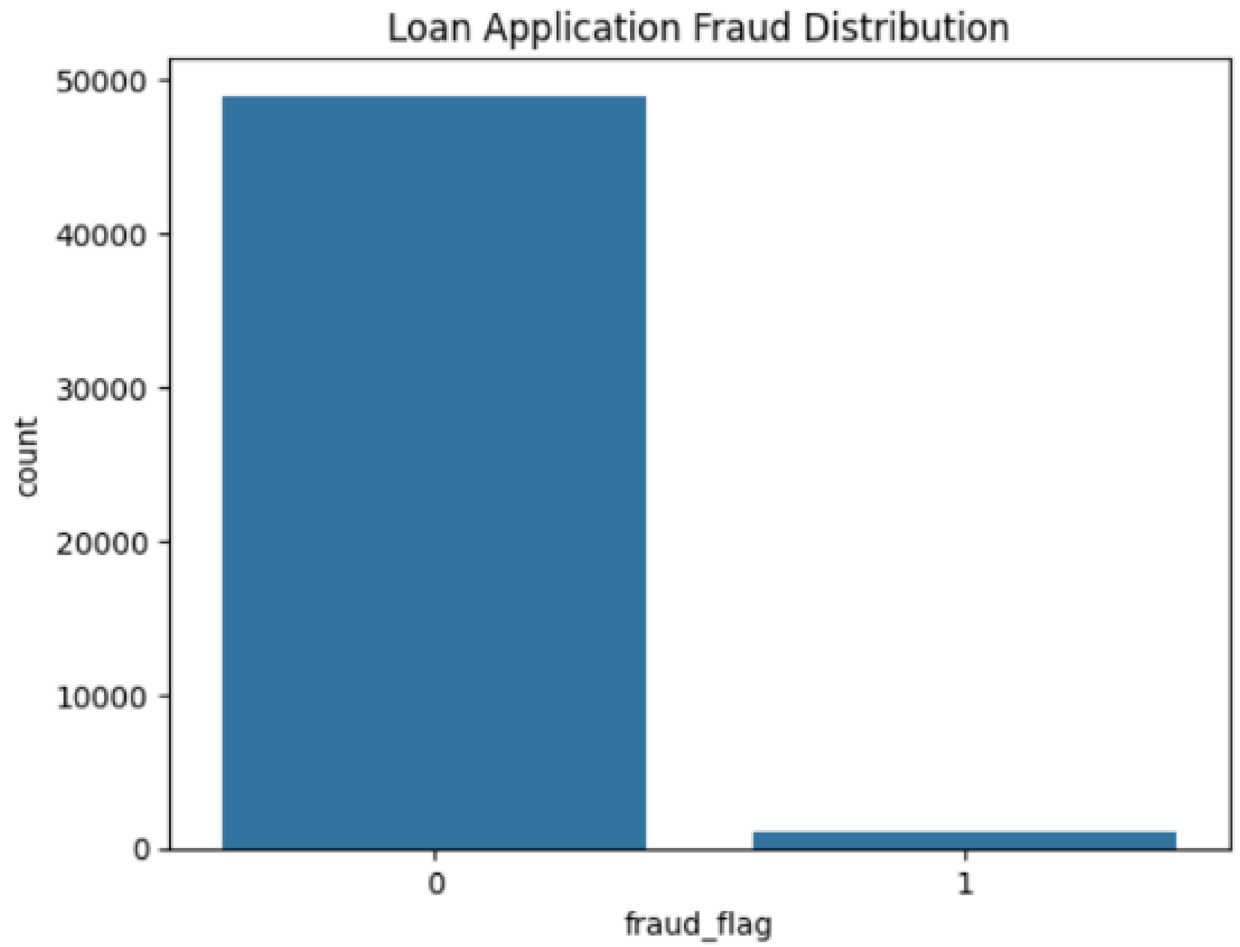
- Loaded and examined both loan_applications and transactions datasets
- Analyzed data types, structure, and dataset dimensions
- Verified consistency of the shared customer_id between datasets

```
:  loans.dtypes

:  application_id              object
   customer_id                 object
   application_date            object
   loan_type                   object
   loan_amount_requested      float64
   loan_tenure_months           int64
   interest_rate_offered      float64
   purpose_of_loan             object
   employment_status           object
   monthly_income             float64
   cibil_score                  int64
   existing_emis_monthly      float64
   debt_to_income_ratio       float64
   property_ownership_status   object
   residential_address         object
   applicant_age                int64
   gender                      object
   number_of_dependents         int64
   loan_status                 object
   fraud_flag                   int64
   fraud_type                  object
   dtype: object
```

```
[48]:  transactions.dtypes

[48]:  transaction_id                       object
       customer_id                          object
       transaction_date                     object
       transaction_type                     object
       transaction_amount                  float64
       merchant_category                    object
       merchant_name                        object
       transaction_location                 object
       account_balance_after_transaction   float64
       is_international_transaction          int64
       device_used                          object
       ip_address                           object
       transaction_status                   object
       transaction_source_destination       object
       transaction_notes                    object
       fraud_flag                            int64
       dtype: object
```

Fraud label distribution

- Analyzed the distribution of the fraud_flag in both datasets
- Visualized class imbalance using count plots
- Observed that fraudulent cases represent a minority class



Feature exploration

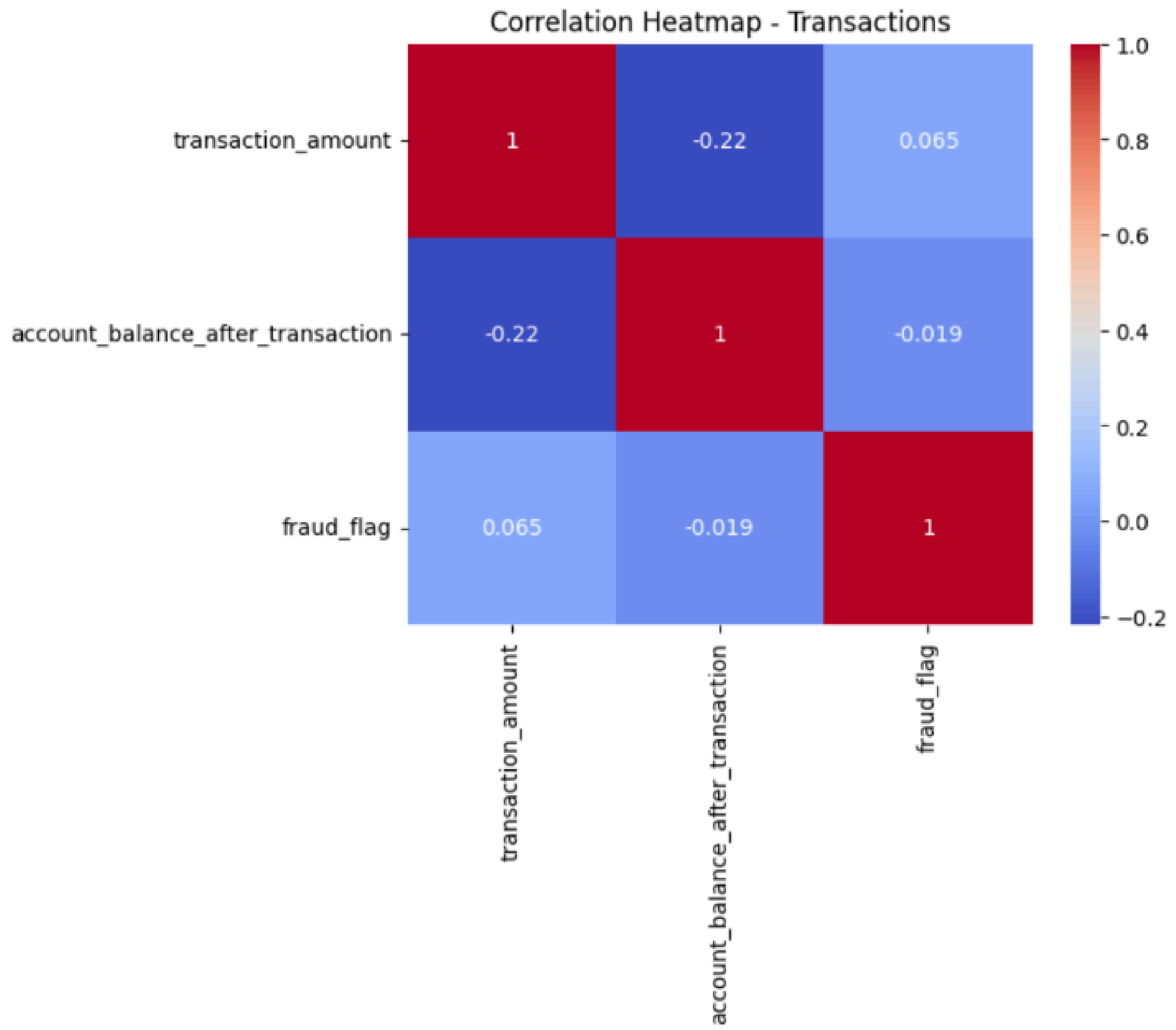
- Examined unique values of categorical variables (e.g., loan type, purpose of loan, transaction type, device used, merchant)
- Explored numerical features such as transaction amount, account balance, and loan amount

```
Loan Applications DataFrame Unique Value Counts:
application_id              50000
customer_id                 18314
application_date             1096
loan_type                       5
loan_amount_requested        1312
loan_tenure_months              7
interest_rate_offered         983
purpose_of_loan                 7
employment_status               6
monthly_income               1101
cibil_score                   354
existing_emis_monthly         108
debt_to_income_ratio         3288
property_ownership_status       3
residential_address         18314
applicant_age                  45
gender                          3
number_of_dependents            5
loan_status                     4
fraud_flag                      2
fraud_type                      4
dtype: int64
```

```
Transactions DataFrame Unique Value Counts:
transaction_id                      50000
customer_id                         18318
transaction_date                    30012
transaction_type                       10
transaction_amount                    639
merchant_category                      12
merchant_name                       35312
transaction_location                 8823
account_balance_after_transaction   30267
is_international_transaction            2
device_used                             4
ip_address                          50000
transaction_status                      2
transaction_source_destination      39993
transaction_notes                      12
fraud_flag                              2
dtype: int64
```

Correlation analysis

- Computed correlation matrices between numerical features and the fraud label
- Visualized correlations using heatmaps to identify potentially relevant predictors



Outlier detection

- Identified numerical columns in both datasets
- Used boxplots to detect outliers in financial variables

Data quality checks

- Checked for missing values in both datasets
- Analyzed feature cardinality using unique value counts

Preprocessing

Why preprocessing is critical

- Raw data comes from multiple sources (customer + transactions)
- Data exists at different granularities
- ML models require:
  - Numerical representations
  - Aggregated behavioral signals
  - Balanced class distributions

Objective

- Transform raw data into a clean, ML-ready dataset
- Preserve fraud-related patterns
- Ensure scalability using PySpark

Tools & Frameworks

Technology stack used

- PySpark
  - Distributed processing
  - Scalable feature engineering
- Spark ML Pipeline
  - StringIndexer
  - VectorAssembler

Data Loading & Integration

Combining heterogeneous data sources

- Loaded:
  - Customer / loan application data
  - Transaction-level data
- Joined datasets using:
  - customer_id
- Created a unified DataFrame:
  - full_df

Why Transaction Aggregation is Needed

Problem: Transaction-level data is too granular

- Each customer can have:
  - Hundreds of transactions
- ML models expect:
  - Fixed-length feature vectors

Transaction Aggregation

Behavioral features extracted
For each customer:

- txn_count → total number of transactions
- total_txn_amount
- avg_txn_amount
- max_txn_amount
- distinct_merchant_count

Fraud-sensitive transaction signals

- international_txn_count
- failed_txn_count

Derived features:

- international_txn_ratio
- failed_txn_ratio

Why ratios?

- Normalize behavior across customers
- Avoid bias toward high-volume users

Financial Feature Engineering

Capturing financial stress & capacity
Created ratio-based features:

- Loan-to-Income Ratio
  - Measures debt pressure
- EMI Burden Ratio
  - Monthly repayment stress

Age-Based Risk Engineering

Domain-driven risk assumptions

- Created binary risk flag:
  - is_high_risk_age
  - True if age < 21 or age > 65

Age Bucketing

Transforming age into categorical groups
Age buckets created:

- very_young
- young
- early_career
- mid_career
- senior

Categorical Feature Handling

Identified categorical columns:

- Age group
- Employment / demographic features ...

Applied:

- StringIndexer
  - Converts categories → numeric indices
  - handleInvalid = "keep"

Feature Vector Assembly

Creating the ML input format
Combined:

- Numerical features (ratios, counts, amounts)
- Indexed categorical features (*_idx)

Used:

- VectorAssembler

Output:

- Single column: features

→ Required by all Spark ML classifiers

Handling Class Imbalance

Why this matters in fraud detection

- Fraud cases are rare
- Random splitting can:
    - Remove fraud cases from train or test

Stratified Train/Test Split

Ensuring fair evaluation

- Used stratified sampling
- Maintained class distribution:
    - 80% training
    - 20% testing
- Both sets contain:
    - Fraud & non-fraud samples

Final Preprocessing Output

What I achieved

- Customer-level dataset
- Aggregated transaction behavior
- Engineered financial & risk features
- Encoded categorical variables
- Balanced train/test datasets
- ML-ready feature vectors

Models Trained

The following spark ML models were trained:

- Logistic Regression
- Random Forest Classifier
- Gradient-Boosted Trees (GBT)
- SVM
- Naive Bayes

All models use the same:

- Feature vector
- Train/test split
- Label column (fraud_flag)

Training Process

- Used:
  - Stratified training dataset
- Same preprocessing pipeline applied to all models
- Each model:
  - Learns decision boundaries from customer-level features

This ensures fair comparison across models.

Why Accuracy Is Not Enough

Problem with accuracy in fraud detection

- Fraud cases are rare
- A model can achieve:
  - 95% accuracy
  - While missing most fraud cases

Evaluation Metrics Used

Metrics selected

- Precision
  - How many predicted frauds are truly fraud
- Recall
  - How many actual frauds are detected
- F1-score
  - Balance between precision & recall
- ROC-AUC
  - Overall discrimination ability

Evaluation Methodology

- Models evaluated on:
  - Held-out test dataset
- Used Spark's:
  - BinaryClassificationEvaluator
- Predictions include:
  - prediction
  - probability

This ensures unbiased performance estimation.

Key Observations

- Logistic Regression:
    - Serves as a strong baseline
- Random Forest:
    - Good balance between robustness and interpretability
- Gradient Boosted Trees (GBT)
    - Best overall performance
    - Highest fraud detection capability
    - Selected as the final model → Performed cross validation with grid search on this model